Universidad de las Ciencias Informáticas

Facultad 15

"Diseño e implementación de un plugin de Eclipse que agilice el desarrollo de aplicaciones Web que utilicen la arquitectura única Dalas"

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autores:

Reynaldo Mariño Echemendia

Yandy González Hervis

Tutor:

Ing. Gueorgui Obregón Obregón

Ciudad de La Habana, junio de 2010

"Año 52 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaramos ser los autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste se firma la presente a los 25 días del mes de junio del año 2010.

Reynaldo Mariño Echemendia

Firma del Autor

Yandy González Hervis

Firma del Autor

Ing. Gueorgui Obregón Obregón

Firma del Tutor

DATOS DE CONTACTO

Ing. Gueorgui Obregón Obregón graduado en el año 2008 de la carrera Ingeniería de la Ciencias Informáticas. Ha trabajado en la Universidad de Ciencias Informáticas desde septiembre de 2008 participando en eventos como el Fórum de Ciencia y Técnica 2008 a nivel de base. Opta actualmente por la categoría Instructor Graduado. Se ha vinculado a los proyectos productivos en el rol de programador de Acceso a Datos y Diseñador, además ha impartido cursos de capacitación sobre el framework Hibernate a integrantes de proyectos productivos de la UCI.

AGRADECIMIENTOS

A la Revolución y a Fidel por haber creado esta magnífica universidad y en ella poder realizar el sueño de graduarme como ingeniero.

A todas las personas que hicieron posible la realización de este trabajo; en especial a nuestro tutor Gueorgui Obregón Obregón por apoyarnos y guiarnos durante el desarrollo del mismo, a Licet Gutiérrez Mompié e Ismary Díaz Rodríguez por ser incondicionales y ayudarnos en todo lo que pudieron.

A quienes siempre me han apoyado en cada momento y han hecho que mi vida universitaria quede para siempre en mi memoria como una de las etapas que nunca olvidaré. A mis compañeros de curso y amigos de la universidad, en especial a los que siempre tuvieron que aguantarme en los buenos y los malos momentos que son: Reiniel, Lisbel, Yoelnis, José Otilio, Elizabeth, Yadira, Ivette.

A tres personas que más que amigos, son como mis hermanos y han estado en todo momento apoyándome dentro y fuera de la universidad, ellos son: Adalberto, Fuoman y Félix.

A mi compañero de tesis Yandy que sin el esfuerzo y compromiso de él, no hubiera sido posible este trabajo, además de ser otro de los buenos amigos con que he contado en esta universidad.

Reynaldo

A todas las personas que de una forma u otra han contribuido a la realización de este trabajo, especialmente a nuestro tutor Gueorgui Obregón Obregón, del cual aprendimos muchísimo. A todos los amigos que he adquirido a lo largo de estos cinco años y junto a los que he pasado los mejores años de mi vida.

A mi novia Ivette por su apoyo y por su cariño todos estos años.

A mi compañero de tesis Reynaldo por su esfuerzo y dedicación para hacer posible la realización de este trabajo y además de ser uno de mis mejores amigos.

Yandy

DEDICATORIA

A mi madre por siempre estar conmigo dándome su apoyo y confianza, por guiarme y enseñarme a ser respetuoso, justo, solidario. Por todos los sacrificios y esfuerzos realizados para que nunca nos faltara nada ni a mí, ni a mis hermanos. Por todo su cariño, por noches de desvelo cuando algo de mí le preocupaba. Por todo eso y más, lo dedico este trabajo a ella.

A mi padre por estar conmigo en todo momento que lo necesite, por toda la sabiduría y confianza que en mí depositó, por enseñarme a ser la persona en la que me he convertido y al igual que mi madre, saber guiarme a lo largo de mi vida.

A mis hermanas Mirelys y Yeny y mi hermano Lester Enmanuel por quererme de la forma que lo hacen, por la sobrinita que cada una de mis hermanas me dio a las cuales quiero mucho, a Mirelys porque siempre ha estado presente en mi vida, por ser mi consejera en algunos momentos, por su confianza y por la paciencia que conmigo ha tenido en momentos que quizás otras personas no la tuvieran. A Yeny que a pesar de lo separada que hemos tenido nuestras vidas, me ha apoyado en todo lo que ha podido y no ha permitido que la distancia nos aleje. A mi hermano que lleva sus siete años de vida siendo una de mis razones de ser, por su cariño, su sonrisa, por desear que pase rápido el tiempo que estoy lejos de él. En fin, a los tres por quererme tanto.

A mi padrastro Luis Enrique y mi madrastra Evelyn por poco a poco convertirse los dos en padres para mí, por apoyarme, confiar en mí y sobre todo por cuidar a las personas que se encuentran al lado de ellos, por esto y más, gracias.

A mis abuelos, tíos, primos, de manera general, a toda mi familia que de una forma u otra estuvieron presentes en mi vida.

Reynaldo

Especialmente a mi mamá, mi hermana y mi abuela por haberme apoyado en todo momento, por haberse sacrificado tanto por mí y por haberme querido tanto. Hoy estoy donde estoy y puedo decir que soy ingeniero informático gracias a estas tres maravillosas mujeres.

A toda mi familia por haber creído en mí siempre.

Yandy

RESUMEN

En la Universidad de las Ciencias Informáticas se desarrollan un número significativo de proyectos, muchos de los cuales se basan en soluciones Web sobre Java Enterprise Edition. Por parte del Centro de Consultoría y Análisis de Sistemas a los proyectos productivos de la universidad, se ha definido una arquitectura única para el desarrollo de estos proyectos denominada Dalas. En la actualidad no se cuenta con herramientas que agilicen la configuración de Dalas en proyectos Web utilizando Eclipse.

En el presente trabajo se propone el desarrollo de un plugin para Eclipse, con el propósito de automatizar y agilizar la configuración de Dalas en proyectos Web, dándole solución a la problemática anterior. Como parte del mismo se construirán componentes que posibiliten: crear proyectos, automatizar la creación de estructuras de módulos, generar clases personalizadas y graficar la estructura de un proyecto para obtener una vista global del mismo. Además, se contará con una ayuda integrada a Eclipse.

Palabras claves: Arquitectura, Eclipse, Plugin.

TABLA DE CONTENIDOS

INTRODUCCIÓN

La Universidad de las Ciencias Informáticas (UCI), desde su concepción, se ha convertido en una universidad que prepara y gradúa ingenieros informáticos de alto nivel, así como en una productora de software; que poco a poco va alcanzando prestigio a nivel internacional. En ésta se desarrollan un gran número de proyectos de software, muchos de los cuales se basan en soluciones Web[1] sobre Java Enterprise Edition (JEE)[2] y hacen uso de herramientas libres tales como Spring IDE e Hibernate Tools[3]. A raíz de esto, en la universidad se crea una arquitectura de software de dominio específico denominada ArBaWeb, que brinda soporte a los requerimientos mencionados anteriormente. Con el objetivo de evitar las extensas configuraciones requeridas por ArBaWeb y para agilizar el tiempo de desarrollo del software, se desarrolló una herramienta denominada ArBaWeb Integrator Tools, que facilita la integración y configuración de ArBaWeb en proyectos Web sobre JEE, utilizando la plataforma de desarrollo Eclipse[4] IDE.

Con el continuo avance de las herramientas libres como Spring IDE e Hibernate Tools y debido a que ArBaWeb no brinda soporte para estas nuevas versiones, se crea una nueva arquitectura única establecida por el Centro de Consultoría y Análisis de Sistemas a los proyectos productivos de la UCI denominada Dalas (véase sección 1.3), que da solución a lo antes planteado. En la actualidad no se cuenta con herramientas que agilicen la configuración de Dalas en proyectos Web utilizando Eclipse, lo que trae como consecuencia que a los desarrolladores, que no tienen experiencia previa utilizando este tipo de arquitectura, en un comienzo les resulte un poco complicado tener que realizar de forma manual las extensas configuraciones requeridas por Dalas. El tiempo para configurar Dalas puede que no sea significativo en proyectos conformados por pocos módulos pero considerable para aquellos que no cumplan esa condición. Para lograr un mayor entendimiento, si tomamos que un proyecto de la universidad puede contener 35 módulos y además, si cada módulo estuviese conformado por aproximadamente 15 paquetes y al menos 4 ficheros, esto daría un aproximado de 525 paquetes y 140 ficheros que habría que crear de forma

[1] Web: Abreviatura de World Wide Web (WWW); interfaz de comunicación en la Internet, que hace uso de enlaces de hipertexto en el interior de una misma página, o entre distintas páginas.

[2] JEE: La versión empresarial de Java después de J2EE 1.4 es llamada Java EE 5.0; destacando así los cambios significantes de los framework de peso ligero traídos en los estándares empresariales de Java.

[3] Hibernate Tools: Plugin de Eclipse diseñado para trabajar en proyectos que hagan uso del framework de persistencia Hibernate.

[4] Eclipse: Plataforma de software de código. En el contexto que se utiliza, se ubica como IDE ya que la herramienta se utiliza con el plugin "Java Development Tooling (JDT)" integrado, que lo convierte en un IDE de java, más adelante en la sección 1.2 se aborda con mayor profundidad.

manual. Si a esto le añadimos todo el tiempo que se invierte en documentarse con lo referente a esta arquitectura, se puede observar claramente un aumento del tiempo de desarrollo del producto.

Dada la situación antes planteada surge el siguiente **problema a resolver**:

¿Cómo agilizar las funcionalidades de configuración de la arquitectura única, establecida por el Centro de Consultoría y Análisis de Sistemas a los proyectos productivos de la UCI, en un proyecto Web sobre JEE?

El **objeto de estudio** lo constituyen las herramientas que permiten la configuración de frameworks. El **campo de acción** quedaría enmarcado en las herramientas de configuración de la arquitectura en proyectos Web sobre JEE.

Se plantea la siguiente **pregunta científica** para guiar la investigación:

¿Qué procesos o áreas de Dalas pueden ser automatizados para lograr una mejor configuración de la misma en aplicaciones web sobre JEE?

Como **objetivo general** de esta investigación se plantea implementar un plugin[5] para Eclipse que agilice la configuración de Dalas en un proyecto Web.

Para dar solución a lo antes planteado se definen las siguientes **tareas de investigación**:

- Caracterizar las tecnologías que se usarán en el desarrollo.
- Identificar los requisitos del software.
- Elaborar el diseño de la solución del plugin.
- Implementar el diseño realizado.
- Realizar pruebas unitarias y de integración.

[5] Plugin: Plugin o plug-in -en inglés "enchufar"-, también conocido como addin, add-in, addon o add-on) es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente capítulo se dan a conocer algunos conceptos y definiciones con el objetivo de lograr un mayor entendimiento para llevar a cabo el desarrollo de un plugin de Eclipse. Se realiza un análisis en cuanto a las características y funcionalidades de algunos de los plugins más utilizados, los cuales facilitan el proceso de configuración de frameworks[6] tanto en el mundo como en la UCI. Además, se definen las tecnologías a utilizar en la construcción del plugin.

1.1 Ambiente de desarrollo integrado

Entorno de desarrollo integrado o Integrated Development Environment (IDE), es un programa informático compuesto por un conjunto de herramientas de programación, con el objetivo de facilitar el trabajo de los programadores y agilizar el desarrollo del software.

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código, un compilador, un depurador y un constructor de Interfaz Gráfica de Usuario o Graphical User Interface (GUI)[7]. Muchos de los IDEs modernos llevan integrados además, un navegador de clases, un inspector de objetos y diagramas de herencia de clases para ser usados en el desarrollo orientado a objetos.

Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes. Además proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación, tales como C++, Python, Java, C#, Visual Basic.

Un IDE puede utilizarse o bien para un lenguaje de programación en específico, como es el caso de Borland C++ o para varios como es el caso de Eclipse, al que mediante plugins se le puede añadir soporte de lenguajes adicionales.

1.2 Eclipse

En noviembre de 1998, una de las transnacionales más grandes del mundo IBM[8], otorgó la tarea a uno de sus grupos de desarrollo de software, denominado Object Technology International (OTI),

[6] Frameworks: Es un esquema (un esqueleto, un patrón) para el desarrollo y/o la implementación de una aplicación.

[7] GUI: Interfaz Gráfica de Usuario, actúa de interfaz de usuario, utilizando un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz.

[8] IBM: Empresa que fabrica y comercializa hardware, software y servicios relacionados con la informática.

de crear una plataforma de desarrollo común para confeccionar sus productos, basados en el lenguaje de programación Java, uno de los más populares en ese momento; surgiendo de esta manera el denominado proyecto Eclipse. (21)

Al inicio, Eclipse no era muy conocido en la comunidad internacional, por lo que el consorcio de la IBM se mostró renuente a invertir en la plataforma. A finales del 2001, IBM adoptó una licencia de código abierto para incrementar el desarrollo del mismo y acelerar la acogida por la comunidad de desarrolladores. Para ese entonces, puso el proyecto Eclipse en manos de un consorcio (Eclipse.org) de empresas fabricantes de herramientas de software. Originalmente, la junta directiva incluía a Borland, MERANT, IBM, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft y Webgain; pero con el tiempo el número de miembros fue aumentando y ya en el 2003 poseía 80 integrantes. (22)

El 2 de febrero del 2004, el consorcio anunció la reorganización de Eclipse en una corporación sin ánimo de lucro, independientemente de su fundadora original IBM, denominada Fundación Eclipse. Desde ese momento el código fuente y la tecnología desarrollada por la comunidad de Eclipse estuvieron disponibles gratuitamente a través de la Eclipse Public License. Eclipse se distribuye actualmente bajo la Licencia Pública Común o Common Public License (CPL) versión 1.0 de IBM, aprobada por la organización Open Source Initiative (OSI). (22)

Hasta aquí se ha hecho una breve introducción del surgimiento de Eclipse, pero en realidad: ¿Qué es Eclipse?

En el sitio oficial de Eclipse, el mismo se define como “una plataforma (IDE), abierta para todo y para nada en particular”. Eclipse es una plataforma debido a que no es una aplicación acabada de por sí, pero está diseñada para ser extendida indefinidamente con herramientas sofisticadas. Es un IDE, puesto que provee herramientas para administrar áreas de trabajo (o workspaces en inglés), construir, lanzar y depurar aplicaciones, así como compartir artefactos con un equipo y versionar el código fuente. Es abierto, debido a que su diseño le permite ser extendido fácilmente por terceras partes. Es apropiado para todo, ya que ha sido utilizado exitosamente para construir ambientes para un amplio rango de temas como el desarrollo en lenguaje de programación Java, Servicios Web, certámenes de programación de juegos, entre otros. Eclipse no es para nada en particular, pues no se enfoca en ningún dominio específico. (18)

1.2.1 Arquitectura

Cuando la plataforma Eclipse fue donada a la comunidad open source por IBM en el año 2001, la noticia no fue notable en el mundo por la enorme suma de dinero que IBM declaró que se había gastado en el desarrollo de la misma (40 millones de dólares); si no debido al resultado obtenido con esta millonaria inversión: un producto inigualable por su arquitectura madura, bien diseñada, extensible y basada en plugins. (17)

De forma general, la plataforma Eclipse está conformada por plugins y por un pequeño núcleo o kernel denominado Platform Runtime. Dentro de estos plugins se pueden encontrar cuatro fundamentales, que son Workbench, Workspace, Help y Team (Fig. 1).

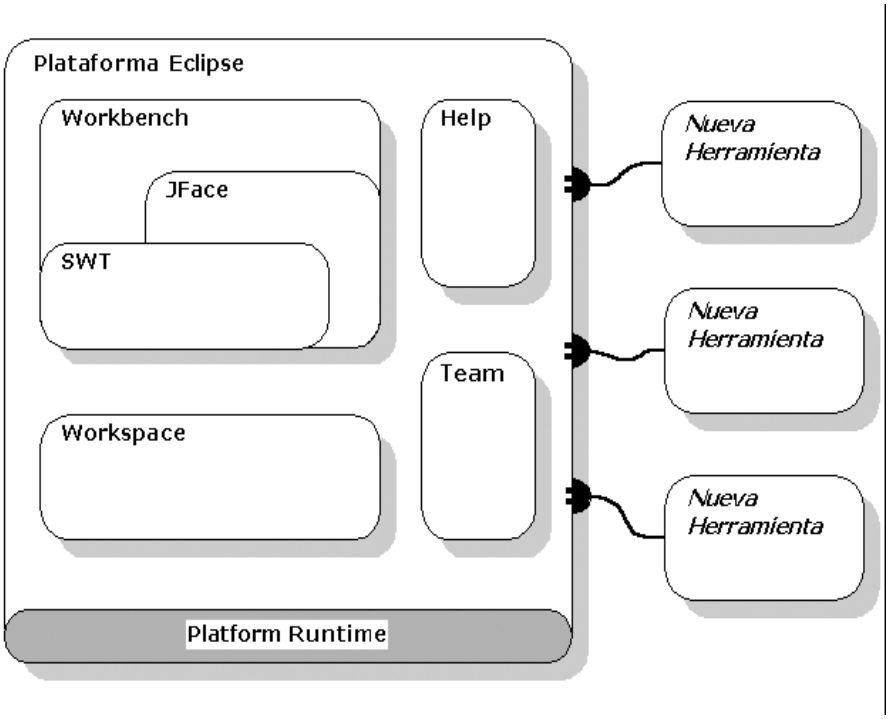

Fig. 1: Arquitectura basada en plugins

Workspace

Se caracteriza por ser el responsable de manejar los recursos de los usuarios, los cuales se organizan en uno o más proyectos a un nivel superior. Cada proyecto corresponde a un subdirectorio del directorio de trabajo de Eclipse y en él se pueden encontrar carpetas y ficheros. También puede proporcionar el código para configurar los recursos del proyecto según sea necesario y como en el caso del Workbench, contiene además los puntos de extensión para que otros plugins interactúen con los recursos de los usuarios. (5)

Workbench

Es la GUI del Eclipse, siendo el componente de la plataforma responsable de la presentación y de la coordinación de la misma. En adición a mostrar los menús y la barra de herramientas está organizado en perspectivas, conteniendo vistas y editores. También posee puntos de extensión para extender funcionalidades de la GUI para otros plugins. (5)

Team

Facilita el soporte del sistema de control de versiones (o manejo de configuración) para manejar los recursos de usuario en un proyecto y define el flujo de trabajo necesario para salvar y recuperar del repositorio. (5)

Help

La plataforma Eclipse proporciona mecanismos para definir y contribuir con documentación. El contenido está distribuido como HyperText Markup Language (HTML). Se le puede añadir documentación en formato HTML y usando Extensible Markup Language (XML) puede definir una estructura de navegación. (1)

Además de los plugins ya referidos, se pueden encontrar otros que aunque no son parte integral de la plataforma, se encuentran integrados en el Eclipse SDK[9], ellos son el Java Develop Tooling (JDT) y el Ambiente de Desarrollo de Plugin o Plugin Development Environment (PDE), que como su nombre describe, es un plugin que facilita el desarrollo de plugins con el fin de extender las funcionalidades de la misma plataforma.

1.2.2 Plugin

Es la unidad mínima de funcionalidad de Eclipse, que puede ser distribuida de manera separada. Herramientas pequeñas se escriben como un único plugin, mientras que en las complejas la funcionalidad está en varios de éstos. Excepto un pequeño núcleo de la plataforma Eclipse, el resto de las funcionalidades están implementadas como plugins.

Los plugins están escritos en Java y están formados por un Java Archive (JAR) de código Java, ficheros de lectura y otros recursos como imágenes, catálogos de mensajes y librerías de código nativo.

[9] Eclipse SDK: Se denomina también proyecto Eclipse. Es la forma entregable de la plataforma Eclipse, y que al descargarse contiene también además de la Plataforma Eclipse el plugin JDT y el PDE.

La plataforma Eclipse consiste aproximadamente, en 100 plugins trabajando juntos. A los límites entre ellos, que permiten conectar un plugin a otro, se le denomina: puntos de extensión. Los puntos de extensión son el mecanismo por el cual un plugin puede adicionarle funcionalidades a otro.

Estructura de un plugin

Cada plugin posee un manifiesto del plugin, en el cual se describe su interconexión con otros plugins, o sea, se declara un número determinado de puntos de extensión y a su vez, se exponen las extensiones que se han realizado de otros puntos de extensión definidos por otros plugins. El manifiesto del plugin está representado por un par de ficheros (Fig. 2): el MANIFEST.MF y el plugin.xml. A continuación se hace una breve descripción de los mismos:

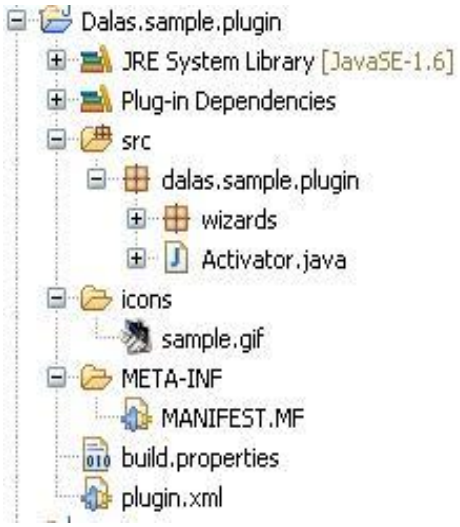

Fig. 2: Estructura simple de un plugin de Eclipse

- MANIFEST.MF: este fichero es generado dentro del directorio META-INF cuando es creado un proyecto plugin. Es un manifiesto del paquete OSGi[10] donde se describen las dependencias que posee el plugin con otros plugins en tiempo de ejecución y sus atributos, como son: el nombre que posee, el proveedor, la versión del mismo, entre otros.
- plugin.xml: se encuentra en la misma raíz del proyecto y consiste en un archivo de formato XML que describe todas las extensiones realizadas por el plugin y declara, además, los puntos de extensión que el propio plugin ha definido.

[10] OSGi: Se refiere a Open Services Gateway Initiative (Iniciativa de enlace de servicios abiertos) , es una corporación independiente, sin ánimo de lucro que trabaja para definir y promover especificaciones abiertas de software que permitan diseñar plataformas compatibles que puedan proporcionar múltiples servicios.

Además de estos elementos, en el directorio de un plugin se encuentran otros archivos y carpetas como son:

- icons: directorio en el cual se encuentran las imágenes o iconos que va a utilizar el plugin.
- plugin.properties: contiene cadenas de caracteres que son referenciadas en el plugin.xml.
- about.html: archivo en formato HTML que es utilizado para mostrar informaciones relacionadas con licencias.

1.3 Dalas Framework

La arquitectura de un sistema constituye la columna vertebral de toda aplicación. Con una correcta definición de una arquitectura se asegura soportar todas las necesidades que se presentarán en el desarrollo, así como la integración con otros sistemas y la reutilización de componentes. (20)

En la UCI se hizo un estudio por parte del Centro de Consultoría y Análisis de Sistemas a los proyectos productivos que desarrollaban sobre JEE. Se llegó a la conclusión de que los proyectos productivos, aunque desarrollan con tecnologías muy similares, no pueden reutilizar soluciones entre ellos pues están amarrados a características específicas de la arquitectura de las aplicaciones. El problema fundamental radica en que no existe una estandarización en el desarrollo de soluciones sobre JEE en la universidad. (20)

Dalas es un framework que surge a raíz de un conjunto de problemas que se presentaron al utilizar soluciones informáticas de gran tamaño, basadas en Spring Framework. El framework ha evolucionado hasta conformar una base arquitectónica para las aplicaciones que utilizan Spring Framework, actuando como integrador y manejador de los componentes de la aplicación. Uno de los principales retos en las empresas constructoras de software es la reutilización de soluciones. En Dalas la reutilización constituye su principal razón de ser, para lo cual se han definido estándares sobre cada tipo de componente, permitiendo mover las funcionalidades entre aplicaciones sin afectaciones. (20)

El framework Dalas divide la aplicación en la mayor cantidad de componentes posibles y minimiza las dependencias innecesarias. El framework define qué es un componente de negocio, qué estructura debe tener, así como mecanismos de colaboración entre los mismos. Define además qué es un componente reutilizable fuera del negocio, mecanismos de interacción entre sí y con los componentes del negocio. (20)

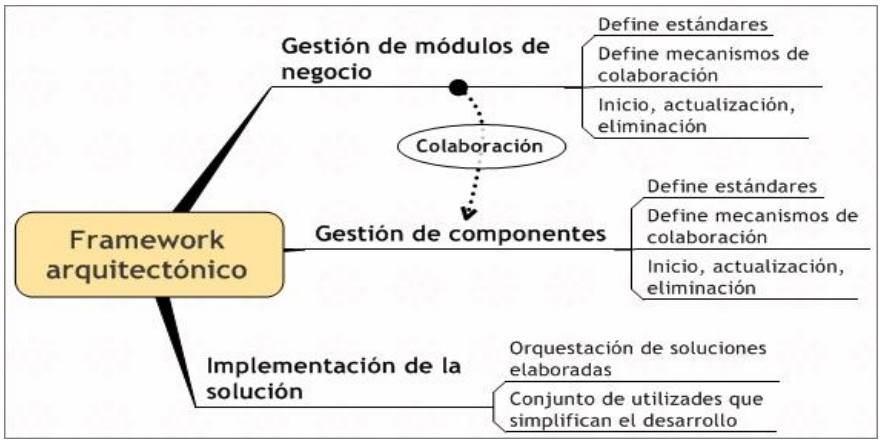

Fig. 3: Principales aspectos de Dalas.

Uno de los principales aportes del framework Dalas es la reducción del consumo de memoria. En la gráfica se muestra el consumo de memoria antes y después de implantado el framework en un proyecto de la universidad, que a mitad de desarrollo ya consumía todos los recursos que se destinaron a la aplicación completa.

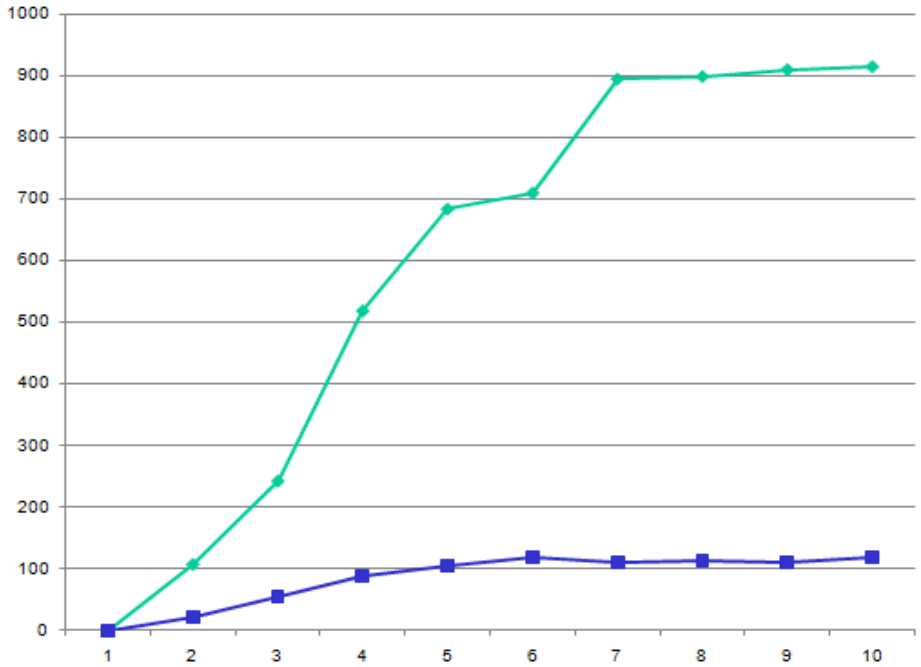

Fig. 4: Consumo de la aplicación completa antes (verde) y después (azul) de la utilización de framework.

1.4 Metodología a utilizar en la solución del plugin

La metodología de desarrollo es un aspecto de gran importancia para el proceso de desarrollo del software, manifiesta más trascendencia si el proyecto a realizar es complejo aunque todo

desarrollo de software es riesgoso y difícil de controlar. Es necesario llevar una metodología adecuada, con el propósito de evitar que tanto los clientes como los desarrolladores queden insatisfechos con el resultado obtenido.

1.4.1 Metodologías tradicionales

Las metodologías tradicionales centran su atención en llevar una documentación absoluta de todo el proyecto, están focalizadas en documentación, planificación y procesos. Se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que, como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil predecir desde el comienzo del proyecto cada resultado. Para el desarrollo de los productos generalmente existe un contrato prefijado y el cliente interactúa con el equipo de desarrollo mediante reuniones. Otra de las características importantes dentro de este enfoque, son los altos costos al implementar un cambio y la falta de flexibilidad en proyectos donde el entorno es volátil. (24)

La solución a algunos de los problemas presentados por las metodologías tradicionales se logra con una gran evolución del modelo espiral. El proceso unificado propone la elaboración de varios ciclos de desarrollo, donde cada uno finaliza con la entrega al cliente de un producto terminado. Este se enmarca entre los conocidos modelos iterativo-incremental. (24)

1.4.2 Metodologías Ágiles

Las metodologías ágiles están especialmente orientadas para entornos variables, proyectos pequeños donde los individuos y las interacciones entre ellos, son más importantes que las herramientas y los procesos empleados y donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad. Las metodologías ágiles dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas.

La prioridad de este enfoque es satisfacer al cliente mediante tempranas y continuas entregas que le aporte un valor, por lo que es más importante crear un producto software que funcione sin tener que escribir documentación exhaustiva, tributando con una elevada simplificación pero sin renunciar a las prácticas esenciales para asegurar la calidad del producto. La colaboración con el cliente debe prevalecer sobre la negociación de contratos por lo que se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Las metodologías ágiles están revolucionando la manera de producir software y a su vez, generando un amplio debate entre sus

seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales.

1.4.3 ¿Por qué usar Metodologías Ágiles?

Las metodologías ágiles están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes. Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas. Las metodologías ágiles presentan diversas ventajas, entre las que se pueden destacar:

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo.
- Entrega continua y en plazos breves de software funcional.
- Trabajo conjunto entre el cliente y el equipo de desarrollo.
- Importancia de la simplicidad, eliminado el trabajo innecesario.
- Atención continua a la excelencia técnica y al buen diseño.
- Mejora continua de los procesos y el equipo de desarrollo.

Para seleccionar una metodología ágil se analizaron dos metodologías ágiles que más fuerza están cobrando en nuestra universidad.

eXtreme Programing

La programación extrema o eXtreme Programing (XP) es una metodología reciente en el desarrollo de software. La filosofía de XP es satisfacer al completo las necesidades del cliente, por eso lo integra como una parte más del equipo de desarrollo. Fue inicialmente creada para el desarrollo de aplicaciones donde el cliente no sabe muy bien lo que quiere, lo que provoca un cambio constante en los requisitos que debe cumplir la aplicación.

XP está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, donde la comunicación sea más factible que en grupos grandes de desarrollo. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes. Las características esenciales de esta metodología son las siguientes:

- Comunicación: Los programadores están en constante comunicación con los clientes para satisfacer sus requisitos y responder rápidamente a los cambios de los mismos. Muchos problemas que surgen en los proyectos se deben a que después de concretar los requisitos que debe cumplir el programa no hay una revisión de los mismos, pudiendo dejar olvidados puntos importantes.

- Simplicidad: Codificación y diseños simples y claros. Muchos diseños son tan complicados que cuando se quieren ampliar resulta imposible hacerlo, por lo que se tienen que desechar y partir de cero.
- Realimentación (Feedback): Mediante la realimentación se ofrece al cliente la posibilidad de conseguir un sistema apto a sus necesidades, ya que se le va mostrando el proyecto a tiempo para poder ser cambiado y poder retroceder a una fase anterior para rediseñarlo a su gusto.

OpenUP

Es una versión más ágil de lo que es el Rational Unified Process (RUP), es un proceso unificado que aplica propuestas iterativas e incrementales dentro del ciclo de vida, tratando de ser manejable en relación con RUP. Plantea que se debe tener un software ya funcional, o lo que es lo mismo, un proyecto ejecutable en poco tiempo. Plantea que se debe utilizar sólo los procesos que sean necesarios, sin demasiados artefactos y sobre todo que el proyecto debe acoplarse a las necesidades del usuario, pudiendo ser éste modificado, mejorado y extendido.

OpenUP tiene dos ventajas importantes, este tipo de método disminuye los riesgos y además puede utilizarse tanto en proyectos pequeños como en proyectos grandes, aunque está concebida para proyectos pequeños. Si se maneja con cuidado y con profesionalismo se puede desarrollar un software de gran calidad, a pesar de que se le diseñe en poco tiempo y con poca documentación. Se utiliza preferentemente en proyectos pequeños, dígase proyectos de 3 a 6 personas. OpenUP contiene las características esenciales de RUP, que incluye el desarrollo iterativo de casos de uso, además de proporcionar un acercamiento a la arquitectura central del sistema. El resultado es un proceso mucho más simple que sigue fielmente los principios de RUP.

OpenUP se organiza en dos dimensiones diferentes: Método y Proceso.

- *Método*: Los roles, las tareas y los artefactos están definidos, sin tener en cuenta cómo son aplicados en el ciclo de vida del proyecto.
- *Proceso*: Es cuando los elementos del proceso son aplicados en el sentido conductual, donde el mismo brinde los roles, las tareas y los artefactos. Pueden crearse ciclos de vida diferentes para proyectos diferentes.

Después de analizar estas dos metodologías se ha decidido utilizar OpenUP.

1.4.4 ¿Por qué OpenUP?

Preserva la esencia de RUP y al igual que él, posee un modelo de desarrollo iterativo e incremental, pero con la diferencia de que al ser un proceso mucho más ligero. Permite micro incrementos en breves períodos de tiempo, lo que posibilita ir creando raleases del producto mientras se desarrolla y efectuar modificaciones a los requerimientos sin altos costos en cuanto a tiempo, precio e implementación. Es apropiado para proyectos pequeños y de bajos recursos, permite detectar errores tempranos a través de un ciclo iterativo. Además, evita la elaboración de documentación, diagramas e iteraciones innecesarias requeridas por RUP y puesto que es una metodología ágil, tiene un enfoque centrado en el cliente con iteraciones muy cortas.

1.5 Plugins existentes que permiten la configuración de frameworks

Desde la aparición de los primeros frameworks, siempre ha sido una tarea primordial de los programadores crear herramientas que faciliten el trabajo de configuración de los mismos. Actualmente, plataformas de desarrollo como Eclipse y NetBeans poseen cientos de estas herramientas o plugins que facilitan el trabajo con los más diversos tipos de frameworks.

Es objetivo de este epígrafe es realizar un estudio de un grupo de los plugins de Eclipse más utilizados, tanto en la universidad como en el mundo, que brindan soporte a frameworks de amplia difusión. Se han analizado de ellos las diferentes características y funcionalidades que brindan para dar soporte.

1.5.1 Spring IDE

Spring Framework es un framework de código abierto de desarrollo de aplicaciones para la plataforma Java; su diseño facilita la integración con los estándares JEE y herramientas comerciales existentes. Es el más popular y ambicioso de todos los framework de peso ligero. Spring IDE es un plugin de Eclipse que sirve como interfaz gráfica de usuario para configurar los archivos usados por Spring Framework. Provee las siguientes funcionalidades:

- *Naturaleza Spring:* adiciona naturaleza Spring a un proyecto para hacerlo un proyecto Spring. Con este tipo de naturaleza se adiciona a la lista de constructores de proyectos de Eclipse, un constructor incremental genérico denominado: "Spring Project Builder". Además, soporta una lista de archivos de configuración de beans[11] de Spring o Spring beans.

[11] Beans: Son simples clases de Java con sus métodos de acceso.

- *Constructor Incremental*: permite validar todo lo modificado en los archivos de configuración Spring beans definidos en un proyecto Spring.
- *Vista*: muestra un árbol con todos los proyectos Spring y sus archivos de configuración.
- *Editor Gráfico*: muestra gráficamente todos los beans definidos y sus relaciones en uno o varios archivos de configuración.
- *Editor de XML*: configura archivos de Spring beans. Permite el completamiento de etiquetas, valores de atributos y elementos en estos archivos de configuración. Soporta la creación de plantillas personalizadas para generar todas las etiquetas de un bean.
- *Extensión de Eclipse*: permite hacer búsquedas de beans definidos en los archivos de configuración.
- *Asistentes*: dispone de asistentes para crear nuevos proyectos integrados con Spring.

1.5.2 Hibernate Tools

Hibernate es un framework de Java que provee mecanismos de mapeo de objetos relacionales con el propósito de definir cómo los objetos de Java son almacenados, actualizados, eliminados y recuperados. Ofrece servicios de búsqueda y recuperación que pueden optimizar el esfuerzo de desarrollo en los ambientes de Structured Query Language (SQL) y Java Database Connectivity (JDBC). Hibernate Tools es un conjunto de herramientas enteramente para trabajar con el framework Hibernate, implementado como una suite integrada de plugins para Eclipse. Proporciona facilidades para trabajar con los archivos de mapeos de Hibernate y realizar diferentes tipos de consultas mediante el lenguaje de consultas de Hibernate (HQL[12]) y otras operaciones que soporta este plugin, de una forma más cómoda para los desarrolladores. Hibernate Tools presenta las siguientes funcionalidades:

- *Editor de mapeos*: es un editor para los archivos de mapeos XML de Hibernate. Soporta auto completamiento semántico para nombres de clases, propiedades, tablas y columnas, haciéndolo más versátil que un editor de XML normal.
- *Consola de Hibernate*: la consola es una nueva perspectiva en Eclipse. Provee la facilidad de visualizar las clases persistentes y sus relaciones. La consola permite ejecutar consultas HQL a una base de datos y buscar los resultados directamente en Eclipse.
- *Ingeniería inversa*: Es su característica más importante, ya que permite generar las clases del modelo de dominio y los archivos de mapeo de Hibernate, los EJB 3 entity beans anotados y documentación en formato HTML, a partir de un esquema de base de datos.

[12] HQL: lenguaje de consultas definido por Hibernate.

- *Asistentes*: provee un conjunto de asistentes que pueden generar rápidamente, entre otros artefactos, archivos de configuración de Hibernate (cfg.xml).
- *Integración con Eclipse JDT*: Hibernate Tools se integra dentro del completamiento de código Java de Eclipse. Esto permite completar código de HQL dentro de código Java y puede incluso señalar los errores dentro de la consulta, si la misma no es válida en la consola de configuración asociada al proyecto.

1.5.3 MyEclipse

MyEclipse es un IDE disponible comercialmente para la plataforma JEE, creado y mantenido por la compañía Genuitec, miembro fundador de la Fundación Eclipse. Está construido sobre la plataforma Eclipse e integra soluciones tanto propietarias como de código abierto en un mismo ambiente, proveyendo soporte para el ciclo de desarrollo completo: codificación, despliegue, prueba y depuración. Compite con el WTP[13], aunque extiende del mismo, es un proyecto separado y ofrece un conjunto de características y funcionalidades diferentes.

MyEclipse tiene dos versiones principales: una profesional y otra estándar. La edición estándar brinda herramientas para intercambiar con sistemas gestores de base de datos y diseñadores Web. Ofrece herramientas que brindan soporte a diferentes frameworks como Spring, Struts[14], JavaServer Faces (JSF) y la Application Programming Interface (API) de persistencia de Java (JPA[15]), que incorpora proyectos como Hibernate y TopLinks. A continuación se muestran un grupo de esas funcionalidades y herramientas para los diferentes frameworks:

Spring:

- Asistentes para agregar capacidad Spring.
- Importa todas las principales librerías de Spring y sus dependencias a un nuevo proyecto en el proceso de agregar capacidad Spring al mismo.
- Extiende y mejora el Spring IDE adicionando herramientas que permiten integrar un proyecto Spring con Hibernate.

[13] WTP: Plugin de Eclipse utilizado para desarrollar aplicaciones sobre la Web y JEE, véase sección 1.6.1.3

[14] Struts: es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma JEE.

[15] JPA: es la API de persistencia desarrollada para la plataforma JEE.

JPA:

- Provee soporte para configurar proyectos JPA.
- Genera las entidades persistentes desde la base de datos junto con sus clases Data Access Object (DAO), pudiendo generar clases DAOs integradas con el framework Spring.
- Define una perspectiva denominada “MyEclipse Java Persistence” y editores avanzados para trabajar con los diferentes elementos que define el API JPA.

Struts:

- Contiene editores multipáginas para los ficheros de configuración de Struts.
- Permite crear plantillas para generar páginas JSP pre-configuradas, para utilizar los elementos de Struts.
- Agrega capacidad Struts a cualquier proyecto Web de MyEclipse. El proceso agrega automáticamente un proyecto Web con las librerías y los XML de configuración de Struts.

En todos los casos anteriores se tratan de plugins desarrollados con el objetivo de agilizar el trabajo de los desarrolladores con distintos frameworks, sin embargo, ninguno representa una solución viable al problema en cuestión.

En el presente trabajo se propone una alternativa con el fin de solucionar los problemas planteados anteriormente, esta solución consiste en el desarrollo de un plugin para Eclipse denominado Dalas IDE que permitirá agilizar el desarrollo en proyectos que hagan uso de Dalas Framework.

1.6 Tecnologías a utilizar en la solución del plugin

Eclipse no sólo es una plataforma que permite desarrollar todo tipo de lenguajes, sino que viene equipada con todo para poder construir sus propias herramientas y extender sus funcionalidades. En este punto se abordan las principales herramientas utilizadas para lograr la construcción de un plugin de Eclipse, de forma eficiente y con calidad. Es objetivo de esta sección es definir herramientas que faciliten la administración de los recursos de un proyecto Plugin.

1.6.1 Plugins

JDT

El Java Develop Tooling (JDT) es uno de los principales subproyectos de Eclipse y es el que permite que la plataforma Eclipse se transforme en un potente IDE de Java, uno de los más reconocidos del mundo. Es el conjunto de plugin más avanzado y elaborado que posee Eclipse. El JDT asiste a los usuarios en los procesos de escribir, compilar, probar, depurar y editar programas escritos en el lenguaje de programación Java, incluyendo los plugins de Eclipse.

El JDT está estructurado en tres componentes principales que contienen los paquetes del API:

- JDT Core: define el núcleo de los elementos Java y provee clases útiles para manipular archivos de extensión .class y el modelo de elementos Java.
- JDT UI: contiene la interfaz de usuario que provee el IDE.
- JDT Debug: brinda un conjunto de clases que permiten correr y depurar código Java.

PDE

El Plugin Development Enviroment (PDE) es un ambiente de desarrollo para plugin, que provee Eclipse y uno de los principales subproyectos del mismo, junto con JDT. Está compuesto por un conjunto de herramientas que cubren todo el ciclo de vida completo del desarrollo de un plugin. Facilita todo el proceso de crear, desarrollar, probar, depurar, construir y desplegar plugins de Eclipse.

Web Tool Platform

La plataforma de herramientas Web de Eclipse o Eclipse Web Tool Platform (WTP) provee varias APIs para desarrollo de aplicaciones sobre la Web y J2EE. Éstas incluyen editores gráficos de código fuente para una variedad de lenguajes, asistentes y aplicaciones incorporadas para simplificar el desarrollo de Servicios Web, además de herramientas y API para soportar el despliegue, ejecución y prueba de aplicaciones.

WTP ofrece un sin número de herramientas que, de cierto modo, satisfacen como punto de partida a muchos desarrolladores, pero en muchos casos quedan con la solución incompleta, ya que el alcance del WTP se encuentra restringido solamente a los estándares de las aplicaciones sobre Web y JEE y no ofrece soporte a otras tecnologías populares como Struts, Spring e Hibernate. No obstante, WTP como proyecto de Eclipse, ofrece la posibilidad de expandir la plataforma,

agregándole los requerimientos específicos necesitados a través de los puntos de extensión definidos dentro del WTP; por ejemplo: el org.eclipse.wst.common.project.facet.core.facets que permite agregar acciones necesarias mientras se está creando un proyecto Web, o el org.eclipse.wst.validation.validator que ofrece la posibilidad de crear validadores para nuevos tipos de ficheros.

Subclipse

Subclipse es un plugin para Eclipse que permite la integración para el control de versiones (Subversion, específicamente), permitiendo operaciones de sincronización, actualización, entre otras. Permite bloqueos a recursos para que otros usuarios no puedan modificarlo. Dispone de una vista de comparación entre el recurso local y remoto, en caso que exista conflicto entre ellos.

Mylyn

Mylyn es uno de los plugins de gran utilidad que viene con Eclipse, permite integrar esta plataforma con gestores de tareas como Bugzilla o JIRA, enfocando el desarrollo a la resolución de tareas. Ofrece la posibilidad de almacenar tareas localmente en un espacio de trabajo o en uno o más repositorios de tarea.

Mylyn usa el contexto de las tareas para enfocar la GUI en la información de interés, oculta lo que es poco interesante y encuentra lo que está automáticamente relacionado, es decir, considerando la información que se necesita para trabajar, logra una mejora en la productividad al reducir el tiempo que se dedica en la búsqueda, desplazamiento y navegación de la misma. Haciendo el contexto de la tarea explícito, Mylyn también ayuda con la ejecución de tareas múltiples, planificando, rehusando esfuerzos anteriores y compartiendo técnicas. (19).

1.6.2 Herramientas para la gestión de proyecto

Gforge

Gforge fue desarrollado por la comunidad de software libre como un ambiente en el cual se almacenan proyectos, de una manera en la que el código, la documentación y los archivos son accesibles públicamente a todo el que desee verlos, los miembros del público pueden contribuir con opiniones, detección de errores, ideas y sugerencias, además de ayudar a desarrollar el código, módulos, documentación y recursos para el software. Provee de un completo sistema de desarrollo de software, incluido un generador de versiones, un sitio Web por proyecto y herramientas para la comunicación entre los miembros de un equipo de desarrollo. Sus

herramientas permiten a los miembros de un equipo de desarrollo una mejor organización del trabajo, así como crear un conocimiento base para futuros proyectos. Es precisamente una herramienta muy poderosa para el desarrollo colaborativo de la comunidad del software.

Una de las ventajas más importantes que ofrece es la posibilidad del desarrollo colaborativo del software y aunque en estos momentos en la facultad se posee una discreta documentación, ésta puede ser una de las potencialidades más explotadas del Gforge.

Gforge puede proveer, de manera centralizada, muchas utilidades y herramientas para la administración de proyectos. Estas herramientas son:

- Hospedaje virtual para proyectos.
- Foros de discusión.
- Seguimiento de errores.
- Solicitudes de soporte.
- Solicitudes de funcionalidades nuevas.
- Seguimientos de registros.
- Listas de correo.
- Administración de tareas.
- Administración de documentos.
- Encuestas para usuarios y administradores.
- Anuncios y noticias.
- Administración de versiones de ficheros.
- Repositorio de ficheros.
- Repositorio de código fuente.
- Mensajería instantánea.
- Interfaz Web para subversion.
- Estadísticas de uso.

1.6.3 Control de versiones

Se denomina Control de Versiones a la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. Los sistemas de control de versiones facilitan la administración de las distintas versiones de cada producto desarrollado, así como las posibles especializaciones realizadas. Un sistema de control de versiones debe proporcionar:

- Mecanismos de almacenaje para los elementos que debe gestionar (ej. archivos de texto, imágenes, documentación).
- Posibilidad de realizar cambios sobre los elementos almacenados (ej. modificaciones parciales, añadir, borrar, renombrar o mover elementos).
- Registro histórico de las acciones realizadas con cada elemento o conjunto de elementos (normalmente pudiendo volver o extraer un estado anterior del producto).

Todos los sistemas de control de versiones se basan en disponer de un repositorio, que es el conjunto de información gestionada por el sistema. Este repositorio contiene el historial de versiones de todos los elementos gestionados. Cada uno de los usuarios puede crearse una copia local duplicando el contenido del repositorio para permitir su uso. Es posible duplicar la última versión o cualquier versión almacenada en el historial. Este proceso suele conocerse como checkout o desproteger. Para modificar la copia local existen dos semánticas básicas:

- *Exclusivos*: para poder realizar un cambio es necesario marcar en el repositorio el elemento que se desea modificar y el sistema se encargará de impedir que otro usuario pueda modificar dicho elemento.
- *Colaborativos*: en el que cada usuario descarga la copia, la modifica y el sistema automáticamente mezcla las diversas modificaciones. El principal problema es la posible aparición de conflictos que deban ser solucionados manualmente o las posibles inconsistencias que surjan al modificar el mismo fichero por varias personas no coordinadas. Además, esta semántica no es apropiada para ficheros binarios.

Aunque un sistema de control de versiones puede realizarse de forma manual, es muy aconsejable disponer de herramientas que faciliten esta gestión y un ejemplo de ello podría ser el Subversion.

Subversion

Subversion es un sistema de control de versiones libre y de código fuente abierto, es uno de los reemplazos más populares de Concurrent Versions System (CVS), ofreciendo mejoras en muchas de las dificultades que presenta este último. Subversion maneja ficheros y directorios a través del tiempo, tiene un árbol de ficheros en un repositorio central, permitiendo recuperar versiones antiguas de los datos o examinar el historial de cambios de los mismos. En este aspecto muchas personas piensan en los sistemas de versiones como en una especie de “máquina del tiempo”. Subversion es un sistema general que puede ser usado para administrar cualquier conjunto de ficheros ya sea código fuente o de otro tipo.

Funcionalidades de Subversion:

- Versionado de directorios: implementa un sistema de ficheros versionado “virtual” que sigue los cambios sobre árboles de directorios completos a través del tiempo.
- Verdadero historial de versiones: permite añadir, borrar, copiar y renombrar ficheros y directorios. Los ficheros nuevos que son añadidos comienzan con un historial nuevo, limpio y completamente suyo.
- Envíos atómicos: permite a los desarrolladores construir y enviar los cambios como fragmentos lógicos e impide que ocurran problemas cuando sólo una parte de los cambios enviados lo hace con éxito.
- Versionado de metadatos: cada fichero y directorio tiene un conjunto de propiedades (claves y sus valores) asociados a él. Se puede crear y almacenar cualquier par arbitrario de clave/valor que desee. Las propiedades son versionadas a través del tiempo al igual que el contenido de los ficheros.
- Manipulación consistente de datos: Subversion expresa las diferencias del fichero usando un algoritmo de diferenciación binario, que funciona idénticamente con ficheros de texto (legibles para humanos) y ficheros binarios (ilegibles para humanos).
- Ramificación y etiquetado eficientes: se pueden crear ramas y etiquetas simplemente copiando el proyecto, usando un mecanismo similar al enlace duro. De este modo estas operaciones toman solamente una cantidad de tiempo pequeña y constante.
- Hackability: Subversion no tiene un equipaje histórico, está implementado como una colección de bibliotecas compartidas en C con APIs bien definidas; permitiendo que Subversion sea extremadamente fácil de mantener y reutilizable por otras aplicaciones y lenguajes.

1.6.4 Lenguaje Unificado de Modelado

El Lenguaje Unificado de Modelado o Unified Modeling Language (UML) es un estándar que tiene como objetivo principal proveer a los arquitectos del sistema, que trabajan con análisis y diseño, un lenguaje consistente para especificar, visualizar, construir y documentar los artefactos del sistema del software, como también para el modelado del negocio.

Este estándar representa la convergencia de las mejores prácticas en la industria de la tecnología de objetos. Es la unión de estos lenguajes de modelado e incluye expresiones adicionales para manejar los problemas de modelado que estos métodos no dirigieron totalmente.

1.6.5 Herramienta Case

Se seleccionó el Visual Paradigm como herramienta para el modelado por ser una herramienta multiplataforma, además de poseer un entorno de creación de diagramas para UML 2.1. Soporta una gama de lenguajes en la generación de código e ingeniería inversa como son: Java, C++, CORBA IDL, PHP, Esquema de XML, Ada, Python, C #, VB .NET, Ruby, Delphi y Perl, lo cual facilita el desarrollo del producto. Soporta el ciclo de vida completo de desarrollo del software, permitiendo crear todo tipo de diagramas UML. El modelado UML ayuda a una rápida construcción de una aplicación de calidad y a un menor costo. Es de fácil uso y permite generar código desde diagramas, así como generar documentación, agilizando el trabajo del desarrollador. Teniendo en cuenta todo lo anterior, el Visual Paradigm se convierte en una herramienta potente para el modelado y desarrollo del producto.

1.7 Conclusiones del capítulo

En la UCI no existe una estandarización en el desarrollo de soluciones sobre JEE, aunque muchos de los proyectos que utilizan tecnologías similares no pueden reutilizar soluciones entre ellos, debido a que están atados a la arquitectura propia de cada proyecto. La arquitectura Dalas representa una solución viable, sin embargo, hasta el momento no existe una herramienta que agilice la configuración de Dalas, por ello se ha decidido desarrollar un plugin para Eclipse que agilice este proceso. Para llevar a cabo el desarrollo del mismo se estudiaron algunos plugins existentes que se han desarrollado en esta plataforma y que hoy en día gozan de gran éxito.

Se seleccionó como metodología guía para el proceso de desarrollo de software OpenUP, debido a que es una metodología apropiada para proyectos pequeños y de bajos recursos. Como herramienta de control de versiones se hace uso de Subversion ya que es un sistema de control de versiones libre y de código abierto. Como lenguaje de modelado se seleccionó UML y como herramienta case el Visual Paradigm, puesto a que es una herramienta potente y multiplataforma.

CAPÍTULO 2: CARACTERÍSTICAS DEL PLUGIN

Introducción

En el presente capítulo se realiza el diseño e implementación del plugin Dalas IDE, se exponen los artefactos necesarios que fueron creados con la guía de la metodología OpenUP a lo largo del desarrollo del plugin. Además, se analizan las características de Dalas que serán automatizadas y finalmente se describen las principales funcionalidades del plugin.

2.1 Propuesta del sistema

El plugin a realizar tiene como objetivo automatizar el proceso de configuración de la arquitectura única establecida por el Centro de Consultoría y Análisis de Sistemas a los proyectos productivos de la UCI, además de agilizar sus procesos. Se pretende automatizar las funcionalidades: crear un proyecto Web integrado a la arquitectura Dalas, generar la estructura de módulos con los ficheros propios del módulo, además de generar clases e interfaces por cada módulo.

2.2 Artefactos generados

2.2.1 Visión Dalas IDE

Dentro de los artefactos que plantea generar OpenUP encontramos el documento visión. Este artefacto contiene la definición de la mirada de los stakeholders[16] del producto a desarrollar, especificado en términos de las necesidades y características claves de los stakeholders. Proporciona una visión completa del sistema de software en desarrollo y los soportes del contrato entre los clientes y la organización de desarrollo. Cada proyecto necesita una fuente para capturar todas las expectativas de los stakeholders y es escrito desde la perspectiva de los clientes, enfocado en las características esenciales del sistema y niveles aceptables de calidad.

Declaración del Problema

El problema	¿Cómo agilizar las funcionalidades de configuración de la arquitectura única establecida por el centro de consultoría UCI en un proyecto Web sobre JEE?
Afecta a	Desarrolladores de proyectos en cualquiera de las plataformas Java que utilicen el framework

[16] Stakeholders: Partes interesadas ya sea inversionistas, clientes, gobiernos, comunidades locales o empleados vinculados a una empresa u organización.

	arquitectónico Dalas.
Cuyo impacto	Provoca una pérdida de productividad por parte de los desarrolladores, principalmente de poca y mediana experiencia.
Una solución exitosa sería	Agilizar el desarrollo de software por parte de los programadores con la creación de un plugin para Eclipse IDE.

Tabla 1: Visión Dalas IDE Declaración del Problema

Posicionamiento del Producto

Para	Desarrolladores de Java.
Quien	Necesite de una herramienta que automatice la configuración de la arquitectura Dalas para agilizar el proceso de producción.
El nombre del producto	Dalas IDE.
Eso	Agiliza el desarrollo y rige a los desarrolladores a la arquitectura Dalas establecida por el Centro de Consultoría UCI.
Al contrario de	
El Producto	Permite la creación de elementos con ataduras a Dalas Framework.

Tabla 2: Visión Dalas IDE Posicionamiento del Producto

Ambiente de Usuario

En la Universidad de la Ciencias Informáticas el número de proyectos que usan Eclipse IDE es aproximadamente un 32 %, de la totalidad de proyectos en la universidad y dado el auge que está tomando la comunidad de software libre y las características de la universidad, estos datos deben mantenerse.

Las restricciones de este plugin es que tiene que ser instalado en el Eclipse IDE 3.5.

Hoy en día, en la universidad y en el mundo, la plataforma Java se utiliza muchísimo debido al auge que ha tomado la comunidad de software libre, además de todas las potencialidades que esta plataforma brinda. Entre ellas es que una aplicación realizada en la plataforma Java puede

correr sobre cualquier sistema operativo. Tanto el Hibernate Tools, como El Spring IDE son herramientas usadas frecuentemente por las potencialidades de los framework que representan. Dalas IDE no necesita integrarse con estas herramientas mencionadas; pero tampoco dificulta el trabajo con las mismas.

Necesidades y características

Necesidad	**Prioridad**	**Características**	**Liberación planeada**
Agilizar las funcionalidades de configuración de la arquitectura única establecida por el centro de consultoría UCI.	ALTA	Se brindarán asistentes y preferencias.	30/5/2010

Tabla.3: Visión Dalas IDE Necesidades y Características

2.2.2 Requerimientos no funcionales.

Usabilidad:

- Facilidad de uso y de comprensión.

Soporte:

- Fácil de instalar.
- Fácil de actualizar.
- Compatibilidad con Eclipse IDE 3.5.

Interfaz de Usuario:

- Legible.
- Simple de usar.

Software:

- JDK 1.5 o superior.
- Eclipse IDE 3.5.

2.2.3 Diagrama de Casos de Uso del plugin Dalas

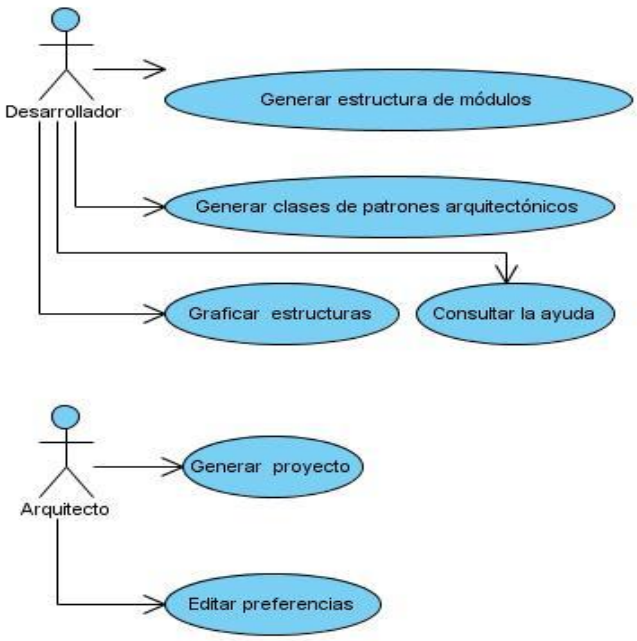

Fig. 6: Diagrama de Casos de Uso

Actores:

Desarrollador: Este actor representa al programador o desarrollador que interactúa con una parte del sistema.

Arquitecto: Este actor representa al arquitecto del proyecto que interactúa con una parte del sistema.

Casos de Uso:

Generar proyecto: En este Caso de Uso se describe cómo el arquitecto genera un proyecto con las librerías definidas, además de las librerías que utiliza Dalas y los ficheros definidos por la misma.

Editar preferencias: Este Caso de Uso describe cómo el arquitecto configura los diferentes elementos que se muestran en cada una de las pantallas, qué estructura tendrán los módulos, librerías que se adicionarán al crear un nuevo proyecto Dalas, así como las clases de patrones arquitectónicos que van a ser generadas y la configuración de asistentes, que define las clases de patrones arquitectónicos que tendrá cada asistente.

Generar clases de patrones arquitectónicos: En este Caso de Uso se describe cómo el desarrollador genera las clases de patrones arquitectónicos definidas en las preferencias.

Graficar estructuras: Este Caso de Uso describe cómo el desarrollador podrá graficar la estructura que posee el proyecto para tener una vista global de las relaciones entre los módulos del mismo.

Generar estructura de módulos: Este Caso de Uso describe cómo el desarrollador genera un módulo según la estructura definida en las preferencias, o en caso de no haber definido alguna entonces la que se propone por defecto.

Consultar ayuda: Este Caso de Uso describe cómo el desarrollador podrá consultar la ayuda para obtener una explicación detallada de cómo realizar cualquier Caso de Uso.

2.2.4 Especificación de Casos de Uso

Caso de Uso: Generar estructura de módulos.

Breve Descripción del Caso de Uso:

Este Caso de Uso describe cómo el desarrollador genera un módulo según la estructura definida en las preferencias, o en caso de no haber definido alguna entonces la que se propone por defecto.

*Actor***:**

Desarrollador.

Precondiciones:

- El Eclipse IDE 3.5 se encuentra abierto.
- El plugin Dalas IDE instalado.
- Se encuentra creado un proyecto.

Flujo Básico de Eventos:

1. El Caso de Uso comienza cuando el desarrollador, selecciona New->Other o Ctrl + N.
2. El sistema muestra la vista.
3. El desarrollador selecciona en la categoría Dalas el elemento Generar Módulos.
4. Haga clic en el botón Next.

5. El usuario llena los campos requeridos.
6. El sistema habilita el botón Next y Finish.
7. El usuario al dar clic en Next se muestra una vista de la estructura del módulo donde podrá agregar o eliminar paquetes, cambiar nombre de los elementos, agregar ficheros a la estructura así como salvar o cargar una estructura de lo contrario da clic en Finish.
8. El sistema genera la estructura de módulo correspondiente.
9. Finaliza el Caso de Uso exitosamente.

Flujos Alternos:

- Adicionar Paquete.

Si en el paso 7 del flujo básico el usuario selecciona la opción de adicionar un paquete, entonces:

1. El sistema solicita que el usuario entre el nombre del nuevo paquete.
2. El usuario llena el campo con el nombre del paquete y da clic en aceptar.
3. El sistema verifica que el nombre sea válido.
4. Se adiciona el nuevo paquete a la estructura propuesta.
5. Finaliza exitosamente el flujo alterno y se regresa al paso 7 del flujo básico.

- Eliminar Paquete.

Si en el paso 7 del flujo básico el usuario selecciona la opción de eliminar un paquete, entonces:

1. El sistema muestra un mensaje confirmando que desea eliminar el paquete seleccionado.
2. El usuario confirma que desea eliminar el paquete.
3. El sistema elimina el paquete de la estructura propuesta.
4. Finaliza exitosamente el flujo alterno y se regresa al paso 7 del flujo básico.

- Cambiar nombre.

Si en el paso 7 del flujo básico el usuario selecciona la opción de cambiar nombre, entonces:

1. El sistema solicita que el usuario entre el nombre nuevo.
2. El usuario llena el campo con el nombre nuevo y da clic en aceptar.
3. El sistema verifica que el nombre sea válido.
4. Se cambia el nombre en el elemento seleccionado.
5. Finaliza exitosamente el flujo alterno y se regresa al paso 7 del flujo básico.

- Agregar fichero.

Si en el paso 7 del flujo básico el usuario selecciona la opción de agregar ficheros, entonces:

1. El sistema muestra una vista con el campo del nombre que tendrá el fichero y una vista de la plantilla por la que se creará.
2. El usuario llena el campo con el nombre y selecciona la plantilla del fichero, luego da clic en Finish.
3. El sistema verifica los datos.
4. Se adiciona el fichero a la estructura.
5. Finaliza exitosamente el flujo alterno y se regresa al paso 7 del flujo básico.

- Eliminar fichero.

Si en el paso 7 del flujo básico el usuario selecciona la opción de eliminar ficheros, entonces:

1. El sistema muestra un mensaje confirmando que desea eliminar el fichero seleccionado.
2. El usuario confirma que desea eliminar el fichero.
3. El sistema elimina el fichero de la estructura propuesta.
4. Finaliza exitosamente el flujo alterno y se regresa al paso 7 del flujo básico.

- Salvar template.

Si en el paso 7 del flujo básico el usuario selecciona la opción de salvar template, entonces:

1. El sistema muestra una vista para que el usuario seleccione el directorio donde desea guardarlo.
2. El usuario selecciona el directorio y el nombre con el cual lo desea salvar y da clic en Guardar.
3. El sistema salva el template en el directorio seleccionado.
4. Finaliza exitosamente el flujo alterno y se regresa al paso 7 del flujo básico.

- Cargar template.

Si en el paso 7 del flujo básico el usuario selecciona la opción de cargar template, entonces:

1. El sistema muestra una vista para que el usuario seleccione el directorio de donde desea cargar el template.

2. El usuario selecciona el directorio y el template que desea cargar y da clic en Abrir.
3. El sistema carga el template del directorio seleccionado y cambia la estructura actual del módulo, por la que tiene el template seleccionado.
4. Finaliza exitosamente el flujo alterno y se regresa al paso 7 del flujo básico.

- Cancelar.

Si en el paso 7 del flujo básico el usuario desea cancelar, entonces:

1. Finaliza el Caso de Uso con condición de falla.

Post-condiciones:

Realización Exitosa:

Se genera la estructura de módulo deseada.

Condición de Falla:

No se genera la estructura de módulo deseada.

Requerimientos Especiales:

Caso de Uso: Consultar Ayuda.

Breve Descripción del Caso de Uso:

Este Caso de Uso describe cómo el desarrollador podrá consultar la ayuda para obtener una explicación detallada de cómo realizar cualquier Caso de Uso.

Actores:

Desarrollador.

Precondiciones:

- El Eclipse IDE 3.5 se encuentra abierto.
- El plugin Dalas IDE instalado.

Flujo Básico de Eventos:

1. El Caso de Uso comienza cuando el Usuario selecciona en el menú →Help→Help Contents.
2. El sistema muestra una página que contiene la ayuda de Eclipse IDE.
3. El usuario da clic en Dalas IDE.
4. El sistema muestra los temas referentes a la ayuda de Dalas.
5. El usuario cierra la página.
6. Finaliza el Caso de Uso exitosamente.

Caso de Uso: Generar clases de patrones arquitectónicos.

Breve Descripción del Caso de Uso:

En este Caso de Uso se describe como el desarrollador genera las clases de patrones arquitectónicos definidas en las preferencias.

Actores:

Desarrollador.

Precondiciones:

- El Eclipse IDE 3.5 se encuentra abierto.
- El plugin Dalas IDE instalado.

Flujo Básico de Eventos:

1. El Caso de Uso comienza cuando el Usuario, selecciona un paquete dentro de un subsistema y con clic derecho selecciona New->Other o Ctrl + N.
2. El sistema muestra la vista.
3. El desarrollador selecciona en la categoría Dalas la opción Generar Clases Patrones.
4. El desarrollador da clic en el botón Next.
5. El sistema muestra las clases de patrones que se generaran en correspondencia al asistente del paquete seleccionado.
6. El desarrollador puede modificar los campos que desee.
7. El usuario da clic en Finish.
8. El sistema guarda los valores de los campos.
9. El sistema genera las clases de patrones correspondientes.
10. Finaliza el Caso de Uso exitosamente

Flujos Alternos:

- Si no hay ningún paquete seleccionado.

Si en el paso 1 el usuario no ha seleccionado ningún elemento entonces:

1. El sistema muestra un mensaje informando que no existe ningún elemento seleccionado.
2. El usuario da clic en el botón OK.

Finaliza exitosamente el flujo alterno y se regresa al paso 1 del flujo básico.

- Si no existen asistentes.

Si en el paso 4 el paquete seleccionado por el usuario no tiene ningún asistente de las clases de patrones predeterminado entonces:

1. El sistema muestra una vista de los asistentes que existen para que el usuario pueda seleccionar el que desea generar.
2. El usuario da clic en el botón Finish.

Finaliza exitosamente el flujo alterno y se regresa al paso 5 del flujo básico.

Post-condiciones:

Realización Exitosa:

Se crean las clases de patrones arquitectónicos en el proyecto.

Requerimientos Especiales:

Caso de Uso: Generar proyecto.

Breve Descripción del Caso de Uso:

En este Caso de Uso se describe cómo el arquitecto genera un proyecto con las librerías definidas por el usuario, además de las librerías que utiliza Dalas y los ficheros definidos por la misma.

Actores:

Arquitecto.

Precondiciones:

- El Eclipse IDE 3.5 se encuentra abierto.
- El plugin Dalas IDE instalado.
- Se encuentra creado un proyecto.

Flujo Básico de Eventos:

1. El Caso de Uso se inicia cuando el arquitecto da clic derecho selecciona New->Other o Ctrl +N y selecciona la opción Dynamic Web Project.
2. El sistema muestra una vista correspondiente para crear un nuevo proyecto Web.
3. El usuario selecciona en Configuración la opción Dalas Web Project.
4. El usuario llena el campo Project name.
5. Se habilita el botón Finish.
6. El usuario da clic en Finish.
7. Finaliza el Caso de Uso exitosamente.

Flujos Alternos:

- Cancelar.

Si en el paso 2 del flujo básico el usuario desea cancelar, entonces:

1. Finaliza el Caso de Uso con condición de falla.

Post-condiciones:

Realización Exitosa:

Se genera el proyecto con todas las librerías definidas por el usuario además de las librerías y ficheros requeridos por Dalas.

Condición de Falla:

No se genera el proyecto.

Requerimientos Especiales:

Caso de Uso: Editar preferencias.

Breve Descripción del Caso de Uso:

Este Caso de Uso describe cómo el arquitecto configura los diferentes elementos que se muestran en cada una de las pantallas, qué estructura tendrán los módulos, librerías que se adicionarán al crear un nuevo proyecto Dalas, así como las clases de patrones arquitectónicos que van a ser generadas y la configuración de asistentes, que define las clases de patrones arquitectónicos que tendrá cada asistente.

Actores:

Arquitecto.

Precondiciones:

- El Eclipse IDE 3.5 se encuentra abierto.
- El plugin Dalas IDE instalado.

Flujo Básico de Eventos para configurar las clases de patrones:

1. Se inicia cuando el arquitecto selecciona Windows -> Preferences y en el elemento del árbol Dalas -> Clases patrones.
2. El sistema muestra una vista de la estructura predeterminada o si no existe alguna predeterminada muestra la que propone Dalas por defecto.
3. El usuario puede mediante el clic derecho agregar clase o agregar interfaz en cualquier paquete de la estructura, así como eliminar o editar alguna clase o interfaz existente.
4. El usuario da clic en el botón OK.
5. Finaliza exitosamente el flujo básico.

Flujo Básico de Eventos para configurar los asistentes:

1. Se inicia cuando el arquitecto selecciona Windows -> Preferences y en el elemento del árbol Dalas -> Generador de asistentes.
2. El sistema muestra una vista que contiene una tabla de asistentes.
3. El usuario puede crear un nuevo asistente dando clic en el botón nuevo, editar un asistente seleccionándolo y dando clic en el botón Editar o eliminar algún asistente existente seleccionándolo y dando clic en el botón Eliminar.
4. El usuario da clic en el botón OK.
5. Finaliza exitosamente el flujo básico.

Flujo Básico de Eventos para configurar las plantillas:

1. Se inicia cuando el arquitecto selecciona Windows -> Preferences y en el elemento del árbol Dalas -> Module templates.
2. El sistema muestra una vista que contiene una tabla con todas las estructuras de módulos creadas.
3. El usuario puede crear una nueva estructura dando clic en el botón Nuevo, editar una estructura seleccionándola y dando clic en el botón editar, eliminarla seleccionándola y dando clic en el botón Eliminar, importar una estructura existente en algún lugar físico de la computadora dando clic en el botón Importar, exportar una estructura seleccionando la estructura y dando clic en el botón Exportar o predeterminar alguna estructura seleccionando la estructura y dando clic en el botón Predeterminar.
4. El usuario da clic en el botón OK.
5. Finaliza exitosamente el flujo básico.

Flujo Básico de Eventos para configurar la creación de proyecto:

1. Se inicia cuando el arquitecto selecciona Windows -> Preferences y en el elemento del árbol Dalas -> Project.
2. El sistema muestra una vista que contiene dos pestañas.
3. En la pestaña web.xml class el usuario puede seleccionar en el checkbox “web.xml Registration class” si desea registrar las clases que se muestran y en caso de querer cambiar el nombre de las mismas, solo tiene que editar los campos y en la pestaña Libraries que contiene una tabla con las librerías creadas, puede crear nueva librería dando clic en el botón Nuevo, editar una librería seleccionándola y luego clic en el botón Editar y eliminar una librería seleccionándola y luego clic en el botón Eliminar.
4. El usuario da clic en el botón OK.
5. Finaliza exitosamente el flujo básico.

Caso de Uso: Graficar Estructura.

Breve Descripción del Caso de Uso:

Este Caso de Uso describe cómo el desarrollador podrá graficar la estructura que posee el proyecto para tener una vista global de las dependencias entre los módulos del mismo.

Actores:

Desarrollador.

Precondiciones:

- El Eclipse IDE 3.5 se encuentra abierto.
- El plugin Dalas IDE instalado.
- Se encuentra creado un proyecto.

Flujo Básico de Eventos:

1. El Caso de Uso comienza cuando el Usuario da clic derecho sobre un proyecto y selecciona la opción DalasTools -> Open graph.
2. El sistema muestra una vista gráfica de todos los módulos que conforman el proyecto así como las relaciones entre los mismos.
3. El usuario mediante clic derecho podrá abrir el fichero module.properties de cada módulo así como salvar la vista como una imagen.
4. El usuario cierra la vista.
5. Finaliza el Caso de Uso exitosamente.

Flujos Alternos:

- Abrir fichero.

Si en el paso 3 del flujo básico el usuario selecciona abrir el fichero entonces:

1. El sistema muestra el fichero module.properties.
2. El usuario hace los cambios en caso de que desea hacer alguno.
3. El sistema guarda los cambios.
4. Finaliza exitosamente el flujo alterno y se regresa al paso 3 del flujo básico.

Post-condiciones:

Realización Exitosa:

Se aplican los cambios realizados al proyecto.

Requerimientos Especiales:

2.2.5 Estructura del diseño

Este artefacto describe la realización de las funcionalidades requeridas del sistema y sirve como una abstracción del código fuente, es decir, describe los elementos del sistema para que puedan ser examinados y entendidos de una forma que no es posible leyendo el código fuente.

En la siguiente figura se muestra la estructura básica del plugin Dalas IDE, compuesto por cuatro componentes fundamentales: Modulegen, DalasGraphicalEditor, DalasFacet y Help.

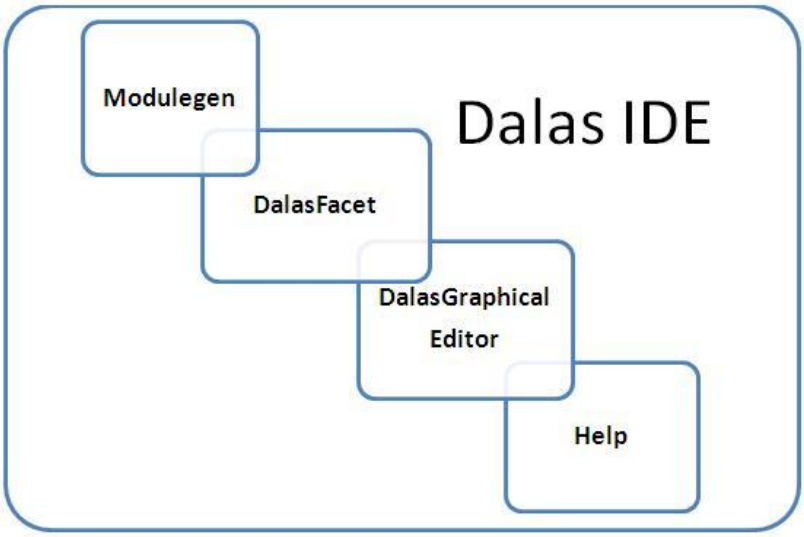

Fig. 7: Estructura básica de Dalas IDE

El *Modulegen* es el encargado de realizar todas las funcionalidades correspondientes para facilitar la creación de la estructura definida por la arquitectura Dalas para un módulo, así como todas las configuraciones.

El *Help* es el elemento que se responsabiliza de ofrecer la documentación necesaria para desarrollar utilizando Dalas IDE, muestra mediante un manual de usuario integrado a la ayuda del Eclipse todos los pasos e instrucciones a seguir.

El *DalasGraphicalEditor* es el encargado de mostrar gráficamente al usuario toda la estructura del proyecto, permitiendo visualizar de forma global las relaciones entre los módulos.

El *DalasFacet* es el encargado de posibilitar la creación de un proyecto Web integrado a Dalas, agregando automáticamente a dicho proyecto las librerías definidas por el usuario conjunto a las librerías propias de Dalas, así como añadir naturaleza Dalas a algún proyecto y registrar clases en el web.xml.

2.2.6 Diagramas de clases

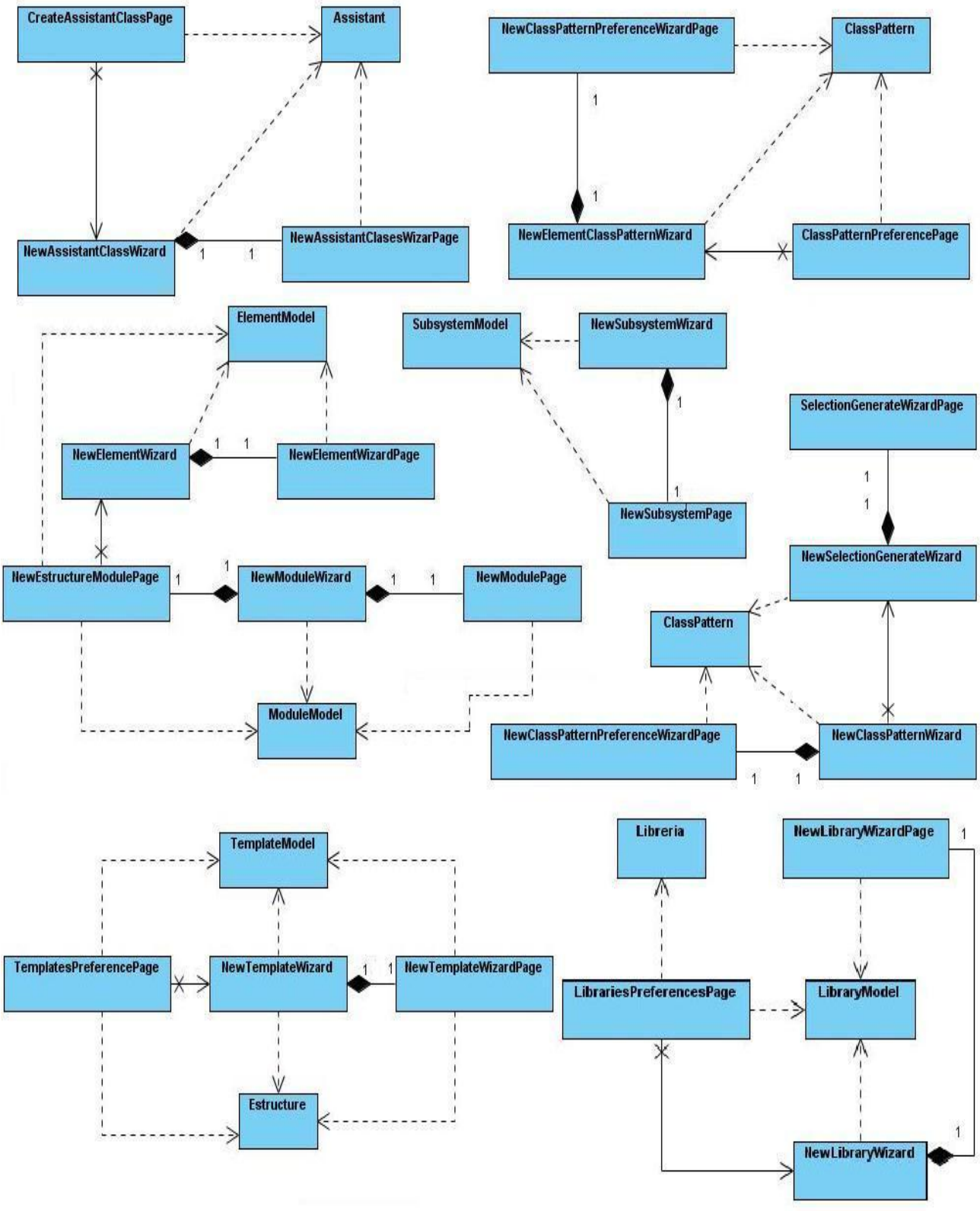

Fig. 8: Diagrama de clases del Modulegen

Descripción de las principales clases del Modulegen

TemplatesPreferencePage: Clase que brinda los controles necesarios para crear nuevas plantillas de módulos, editarlas, eliminarlas, exportarlas como un archivo XML e importar alguna plantilla existente.

NewTemplateWizard: Asistente que permite definir la estructura de un módulo. Contiene la clase NewTemplateWizardPage que posee todos los componentes gráficos que se muestran al usuario, además de la clase TemplateModel que almacena todas las configuraciones realizadas en la clase NewTemplateWizardPage.

LibraryPreferencesPage: Clase que brinda los controles gráficos para gestionar las librerías que el usuario defina, además de permitir que se agreguen o no cuando se crea un proyecto Web y definir si desea registrar clases en el web.xml.

NewLibraryWizard: Asistente que permite crear o editar librerías que el usuario desee. Contiene la clase NewLibraryWizardPage que posee todos los componentes gráficos que se muestran al usuario, además de la clase LibraryModel que almacena todas las configuraciones realizadas en la clase NewLibraryWizardPage.

CreateAssistantClassPage: Clase que brinda los controles gráficos para gestionar los asistentes, así como las clases que se deseen generar en cada asistente.

NewAssistantClassWizard: Pantalla que permite definir las clases que contendrá el asistente, así como el paquete donde se mostrará dicho asistente. Contiene la clase NewAssistantClassWizardPage que posee los componentes gráficos que se muestran al usuario.

NewClassPatterWizard: Pantalla que se encarga de generar las clases del asistente correspondiente. Contiene la clase SelectionGenerateWizardPage que se encarga de mostrar los asistentes definidos, además de la clase NewClassPatterPreferenceWizardPage que posee los componentes gráficos para que verifique los datos definidos para las clases.

ClassPatternPreferencePage: Clase que brinda los controles gráficos para gestionar la estructura de las clases que se desean generar. Contiene la clase NewClassPatterPreferenceWizardPage que posee los componentes gráficos para que el usuario defina o modifique los datos de la clase.

NewModuleWizard: Asistente que permite generar la estructura de un módulo según la estructura predeterminada. Contiene la clase NewModulePage que posee los componentes gráficos para que el usuario introduzca los datos del módulo, contiene además la clase ModuleModel que almacena todos los datos de la clase NewModulePage y contiene la clase NewStructureModulePage que posee los componentes gráficos para modificar, en caso de que se desee, la estructura del módulo que se va a generar.

NewSubsystemWizard: Asistente que permite generar un subsistema. Contiene la clase NewSubsystemPage que posee los componentes gráficos para definir el nombre del subsistema y el proyecto en que se va a generar.

Help

El *Help* es el elemento que se responsabiliza de ofrecer la documentación necesaria para desarrollar utilizando Dalas IDE, muestra mediante archivos en formato HTML y XML un manual de usuario integrado a la ayuda del Eclipse, que contiene todos los pasos a seguir por el desarrollador para hacer uso de cada una de las funcionalidades que brinda el plugin.

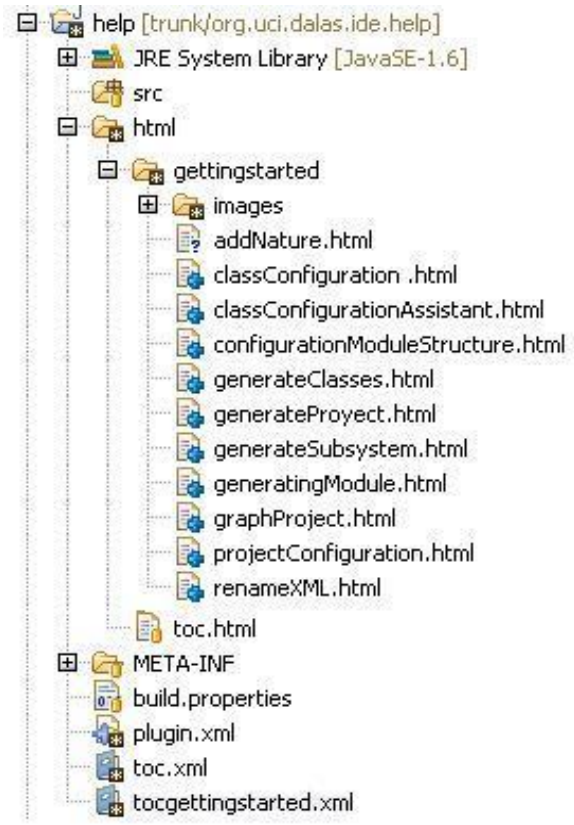

Fig. 9: Estructura del plugin Help

Diagramas de clases de las principales funcionalidades del plugin DalasGraphicalEditor

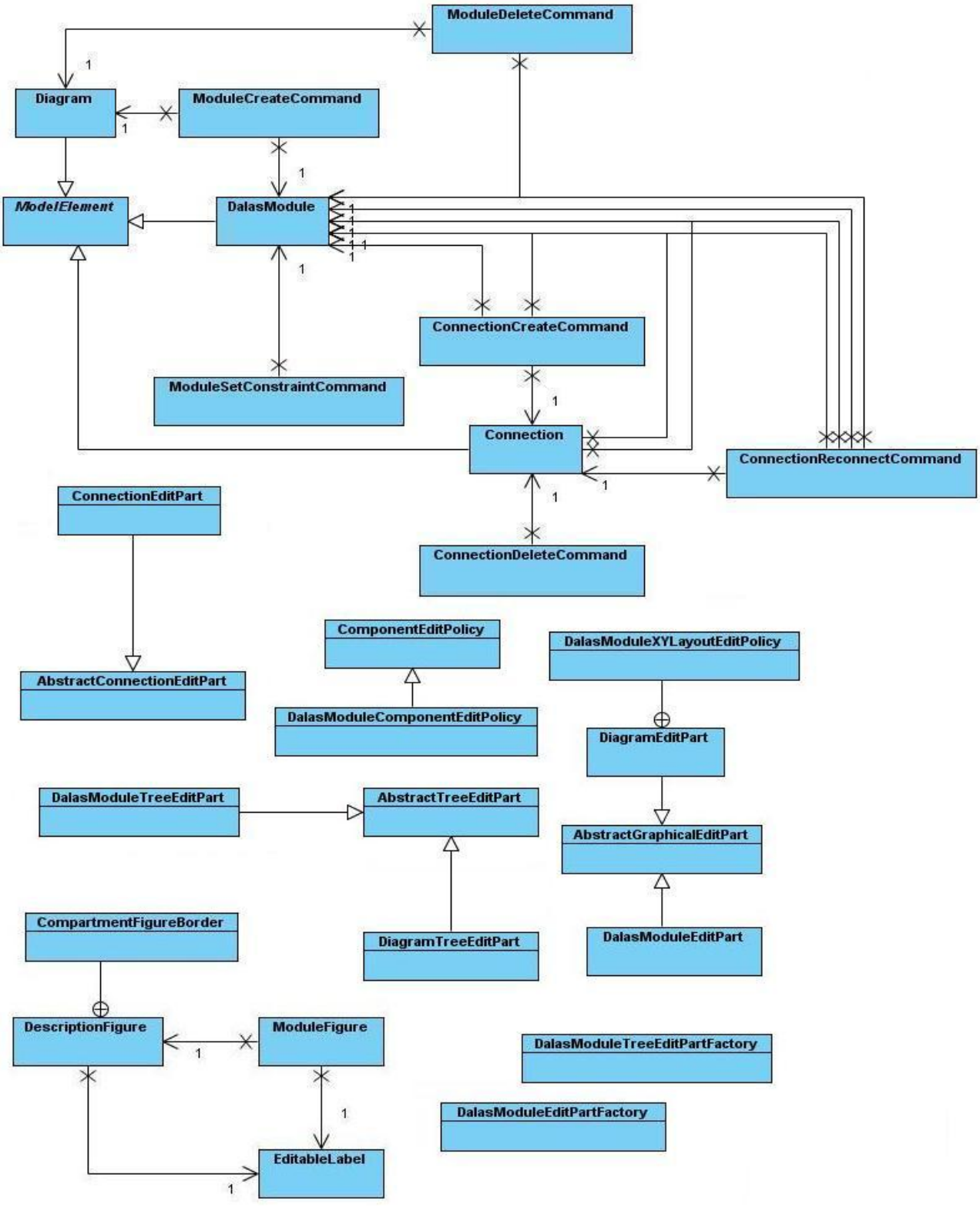

Fig. 10: Diagrama de clases del DalasGraphicalEditor

Descripción de las principales clases del DalasGraphicalEditor

ConnectionEditPart: Controlador que manipula todas las operaciones que se hacen con las conexiones entre los módulos. Se encarga de dibujar la conexión con los elementos que posee y asocia cada uno de los módulos vinculados por esta conexión.

DalasModuleTreeEditPart: Controlador que manipula las operaciones que se realizan sobre los elementos DalasModule, vincula el modelo con la vista, o sea, el elemento DalasModule con la figura que le corresponde, ModuleFigure.

DiagramTreeEditPart: Controlador que contiene todos los elementos del gráfico.

DalasModuleEditPartFactory: Clase que se encarga de crear una instancia de un controlador según sea la figura que el mismo manipula.

DalasModule: Objeto del modelo que contiene la información necesaria de un módulo representado en el diagrama.

ModuleFigure: Figura que contiene los componentes gráficos que corresponden con el elemento DalasModule.

2.3 Conclusiones del capítulo

Siguiendo la metodología OpenUP a través de una óptima planificación y diseño guiados por los artefactos generados, se ha logrado obtener el plugin Dalas IDE. A lo largo del capítulo se resume el trabajo realizado donde se exponen las principales características y funcionalidades del plugin, además de los componentes que lo conforman, estructura y funcionalidades que poseen. También se ha realizado una breve descripción de las principales clases de cada uno de los flujos de eventos del plugin. Como resultado, el plugin Dalas IDE cumple con todos los requisitos y funcionalidades definidas.

CAPÍTULO 3: PRUEBA Y VALIDACIÓN DE LA SOLUCIÓN

Introducción

En el presente capítulo se exponen las pruebas realizadas al software desarrollado, con el objetivo de comprobar las funcionalidades del plugin en los diferentes escenarios, verificando en todos los casos que los resultados de las pruebas sean los esperados.

Las pruebas tienen gran importancia en el desarrollo de un software, ya que mediante éstas se pueden detectar y corregir errores tempranamente. Antes de entregar el producto al cliente final se debe garantizar que el software cumpla con todos los requerimientos y se han corregido todos los errores, pues cada vez que el programa se ejecuta, el cliente lo está probando, por tanto, debemos hacer un intento especial para que se sienta satisfecho. Con el objetivo de encontrar el mayor número posible de errores, las pruebas deben conducirse sistemáticamente y los casos de prueba deben diseñarse utilizando técnicas definidas.

Un caso de prueba especifica cómo probar un Caso de Uso o un escenario específico del mismo. Un caso de prueba incluye la verificación del resultado de la interacción entre actores y el sistema, la satisfacción de las precondiciones y poscondiciones especificadas por el Caso de Uso y el seguimiento de la secuencia de acciones especificadas por el mismo.

3.1 Casos de prueba

Nombre de la Prueba:	Generar Proyecto.
Caso de Uso Probado:	Generar Proyecto.
Descripción de la Prueba:	Se crea un proyecto Web integrado con Dalas.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se crea un proyecto Web que contiene todas las librerías definidas por el usuario, además de las librerías que incluye Dalas.
Notas:	

Resultado: **(Pasa/Falla/aviso/incompleto)**				
	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	En el asistente para crear un proyecto Web seleccionar en Configuración la opción Dalas Web Project.	No se muestra nada.	**X**	
2.	Presionar el botón Finish para generar el proyecto.	Se genera un proyecto Web con todas las librerías definidas por el usuario además de las que requiere dalas.	**X**	

Tabla 4: Prueba Generar Proyecto

Nombre de la Prueba:	Editar Preferencias.			
Caso de Uso Probado:	Editar preferencias->Clases de Patrones.			
Descripción de la Prueba:	Se define qué clases pueden ser generadas dentro de la estructura del módulo así como los datos de estas clases.			
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.			
Post-condiciones:	Se guarden los datos de las clases definidas en la estructura del módulo.			
Notas:	Mediante un pop up menú se puede crear una nueva clase o una nueva interfaz así como eliminar o editar alguna ya existente.			
Resultado: **(Pasa/Falla/aviso/incompleto)**				
	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Clases de Patrones.
Descripción de la Prueba:	Se define qué clases pueden ser generadas dentro de la estructura del módulo así como los datos de estas clases.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guarden los datos de las clases definidas en la estructura del módulo.
Notas:	Mediante un pop up menú se puede crear una nueva clase o una nueva interfaz así como eliminar o editar alguna ya existente.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	Resultados Esperados de Prueba	P	F
1.	Seleccionar mediante el clic derecho nueva clase.	El sistema muestra una ventana para entrar los datos de la nueva clase.	**X**	
2.	Seleccionar mediante el clic derecho nueva interfaz.	El sistema muestra una ventana para entrar los datos de la nueva interfaz.	**X**	
3.	Llenar los datos introduciendo caracteres extraños en los campos obligatorios.	El sistema muestra un mensaje que los campos contienen caracteres extraños.	**X**	
4.	Llenar los datos dejando algún campo obligatorio en blanco.	El sistema muestra un mensaje que existen campos vacíos.	**X**	
5.	Llenar los datos correctamente.	El sistema habilita el botón Finish.	**X**	
6.	Presionar el botón Finish en la ventana de crear una nueva clase.	Se cierra la ventana y se muestra en la estructura la clase creada.	**X**	

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Clases de Patrones.
Descripción de la Prueba:	Se define qué clases pueden ser generadas dentro de la estructura del módulo así como los datos de estas clases.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guarden los datos de las clases definidas en la estructura del módulo.
Notas:	Mediante un pop up menú se puede crear una nueva clase o una nueva interfaz así como eliminar o editar alguna ya existente.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	Resultados Esperados de Prueba	P	F
7.	Presionar el botón Finish en la ventana de crear una nueva interfaz.	Se cierra la ventana y se muestra en la estructura la interfaz creada.	**X**	
8.	Seleccionar mediante el clic derecho la opción Editar.	El sistema muestra una ventana con los datos de la clase o interfaz seleccionada para ser editados.	**X**	
9.	Seleccionar mediante el clic derecho la opción Eliminar.	El sistema confirma si desea eliminar.	**X**	
10.	Confirmar que desea eliminar.	Se elimina la clase o interfaz de la estructura.	**X**	
11.	Cancelar la confirmación de eliminar.	El sistema no hace nada.	**X**	

Tabla 5: Prueba Editar Clases de Patrones

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Generador de Asistentes.
Descripción de la Prueba:	Se crean asistentes con las clases definidas en la estructura que el usuario desee que se muestren a la hora de generar clases.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se crea un asistente que contiene un grupo de clases seleccionadas.
Notas:	La vista contiene una tabla con todos los asistentes creados y permite crear un nuevo asistente, editar una existente y eliminarlo mediante botones en la parte derecha de la tabla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	Presionar el botón Nuevo.	El sistema muestra una ventana para entrar los datos del nuevo asistente.	**X**	
2.	Llenar los datos introduciendo caracteres extraños en los campos obligatorios.	El sistema muestra un mensaje que los campos contienen caracteres extraños.	**X**	
3.	Llenar los datos dejando algún campo obligatorio en blanco.	El sistema muestra un mensaje que existen campos vacíos.	**X**	
4.	Llenar los datos correctamente.	El sistema habilita el botón Finish.	**X**	
5.	Presionar el botón Finish en la ventana de nuevo asistente.	Se cierra la ventana y se muestra en la tabla el asistente creado.	**X**	

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Generador de Asistentes.
Descripción de la Prueba:	Se crean asistentes con las clases definidas en la estructura que el usuario desee que se muestren a la hora de generar clases.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se crea un asistente que contiene un grupo de clases seleccionadas.
Notas:	La vista contiene una tabla con todos los asistentes creados y permite crear un nuevo asistente, editar una existente y eliminarlo mediante botones en la parte derecha de la tabla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
6.	Presionar el botón Editar.	El sistema muestra una ventana con los datos del asistente seleccionado para ser editados.	**X**	
7.	Presionar el botón Eliminar.	El sistema confirma si desea eliminar.	**X**	
8.	Confirmar que desea eliminar.	Se elimina el asistente de la tabla.	**X**	
9.	Cancelar la confirmación de eliminar.	El sistema no hace nada.	**X**	

Tabla 6: Prueba Editar Generador de Asistentes

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Module Template.

Descripción de la Prueba:	Se crean plantillas de estructuras de módulos, se editan, eliminan, se importan y exportan, además de predeterminar una plantilla.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guardan todos los cambios realizados.
Notas:	La vista contiene una tabla con todas las plantillas existentes además, en la parte derecha de la tabla mediante botones, brinda la posibilidad de crear una nueva plantilla, editarla, eliminarla, importarla, exportarla y predeterminarla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	Presionar el botón Nuevo.	El sistema muestra una ventana para entrar los datos de la nueva plantilla.	**X**	
2.	Llenar los datos introduciendo caracteres extraños en los campos obligatorios.	El sistema muestra un mensaje que los campos contienen caracteres extraños.	**X**	
3.	Llenar los datos dejando algún campo obligatorio en blanco.	El sistema muestra un mensaje que existen campos vacíos.	**X**	
4.	Llenar los datos correctamente.	El sistema habilita el botón Finish.	**X**	
5.	Presionar el botón Finish en la ventana de nuevo asistente.	Se cierra la ventana y se muestra en la tabla la plantilla creada.	**X**	
6.	Presionar el botón Editar.	El sistema muestra una ventana con los datos de la plantilla seleccionada para ser editados.	**X**	

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Module Template.
Descripción de la Prueba:	Se crean plantillas de estructuras de módulos, se editan, eliminan, se importan y exportan, además de predeterminar una plantilla.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guardan todos los cambios realizados.
Notas:	La vista contiene una tabla con todas las plantillas existentes además, en la parte derecha de la tabla mediante botones, brinda la posibilidad de crear una nueva plantilla, editarla, eliminarla, importarla, exportarla y predeterminarla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
7.	Presionar el botón Eliminar.	El sistema confirma si desea eliminar.	**X**	
8.	Confirmar que desea eliminar.	Se elimina la plantilla de la tabla.	**X**	
9.	Cancelar la confirmación de eliminar.	El sistema no hace nada.	**X**	
10.	Presionar el botón Importar.	El sistema permite seleccionar de un directorio la plantilla que desea importar.	**X**	
11.	Seleccionar para importar un archivo no válido.	El sistema muestra un mensaje que el archivo no corresponde con la estructura de la plantilla.	**X**	
12.	Seleccionar para importar un archivo válido.	El sistema muestra en la tabla la nueva plantilla importada.	**X**	

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Module Template.
Descripción de la Prueba:	Se crean plantillas de estructuras de módulos, se editan, eliminan, se importan y exportan, además de predeterminar una plantilla.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guardan todos los cambios realizados.
Notas:	La vista contiene una tabla con todas las plantillas existentes además, en la parte derecha de la tabla mediante botones, brinda la posibilidad de crear una nueva plantilla, editarla, eliminarla, importarla, exportarla y predeterminarla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
13.	Presionar el botón Exportar.	El sistema permite buscar un directorio para salvar la plantilla.	**X**	
14.	Presionar el botón Predeterminar.	El sistema marca en negrita la plantilla seleccionada y debe mostrar esta plantilla a la hora de crear una nueva y cuando se vaya a generar un nuevo módulo.	**X**	

Tabla 7: Prueba Editar Template

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Project.
Descripción de la Prueba:	Se definen las librerías que se desean adicionar cuando se cree un nuevo proyecto, se especifica si se desea registrar clases en el web.xml y de ser así el nombre con que se registrará y si desea generar el paquete de configuración y su respectivo nombre.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guardan todos los datos especificados por el usuario.
Notas:	Se muestra una tabla para las librerías y se puede crear una nueva, eliminar o editar una existente en los botones situados en la parte derecha de la tabla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	Se selecciona o no si se desea registrar las clases en el web.xml y su respectivo nombre en caso de desearlo.	No se muestra nada.	**X**	
2.	Se selecciona o no si se desea generar un paquete de configuración y su respectivo nombre en caso de desearlo.	No se muestra nada.	**X**	
3.	Presionar el botón Nuevo en la sección librerías.	El sistema muestra una ventana para entrar los datos de la nueva librería.	**X**	
4.	Entrar los datos de la librería introduciendo caracteres extraños en los campos obligatorios.	El sistema muestra un mensaje que los campos contienen caracteres extraños.	**X**	

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Project.
Descripción de la Prueba:	Se definen las librerías que se desean adicionar cuando se cree un nuevo proyecto, se especifica si se desea registrar clases en el web.xml y de ser así el nombre con que se registrará y si desea generar el paquete de configuración y su respectivo nombre.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guardan todos los datos especificados por el usuario.
Notas:	Se muestra una tabla para las librerías y se puede crear una nueva, eliminar o editar una existente en los botones situados en la parte derecha de la tabla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
5.	Entrar los datos de la librería dejando campos obligatorios en blanco.	El sistema muestra un mensaje alertando que existen campos obligatorios vacíos.	**X**	
6.	Entrar los datos de la librería correctamente.	Se habilita el botón Finish.	**X**	
7.	Presionar el botón Finish en la ventana de nueva librería.	Se cierra la ventana y se muestra la nueva librería en la tabla.	**X**	
8.	Presionar el botón Editar.	El sistema muestra una ventana con los datos de la librería seleccionados para ser editados.	**X**	

Nombre de la Prueba:	Editar Preferencias.
Caso de Uso Probado:	Editar preferencias->Project.
Descripción de la Prueba:	Se definen las librerías que se desean adicionar cuando se cree un nuevo proyecto, se especifica si se desea registrar clases en el web.xml y de ser así el nombre con que se registrará y si desea generar el paquete de configuración y su respectivo nombre.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	Se guardan todos los datos especificados por el usuario.
Notas:	Se muestra una tabla para las librerías y se puede crear una nueva, eliminar o editar una existente en los botones situados en la parte derecha de la tabla.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
9.	Presionar el botón Eliminar.	El sistema confirma si desea eliminar.	**X**	
10.	Confirmar que desea eliminar.	Se elimina la librería de la tabla.	**X**	
11.	Cancelar la confirmación de eliminar.	El sistema no hace nada.	**X**	

Tabla 8: Prueba Editar Project

Nombre de la Prueba:	Generar Módulos.
Caso de Uso Probado:	Generar Módulos.
Descripción de la Prueba:	Se crea la estructura del módulo, adicionando paquetes y tipos de elementos a una estructura que se propone inicialmente.

Pre-condiciones:	El Eclipse IDE 3.4 se encuentra abierto y el plugin Dalas IDE instalado.
Post-condiciones:	Se crea el Módulo.
Notas:	En la primera vista se llenan datos que son propios del módulo, luego en la otra vista se podrá conformar la estructura del módulo. En la segunda vista tanto los tipos de elementos como los paquetes se crean utilizando un pop up menú o un toolbar. También se puede, eliminar paquetes y tipos de elementos mediante el pop up menú o el toolbar.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	No introduce algún dato requerido.	Se muestra un aviso de error: Existen campos vacíos que son requeridos.	**X**	
2.	Introduce caracteres extraños.	Se muestra un aviso de error: El campo tiene caracteres extraños.	**X**	
3.	Llenar los datos solicitados.	Se debe habilitar el botón Next.	**X**	
4.	Crear un paquete.	Se deben crear el paquete dentro del paquete seleccionado como padre en la estructura.	**X**	
5.	Crear tipos de elementos dentro de los paquetes.	Se deben crear los tipos de elementos dentro del paquete seleccionado como padre.	**X**	
6.	Crear un tipo de elemento dentro de otro Tipo de elemento.	No permite hacerlo.	**X**	

Nombre de la Prueba:	Generar Módulos.
Caso de Uso Probado:	Generar Módulos.
Descripción de la Prueba:	Se crea la estructura del módulo, adicionando paquetes y tipos de elementos a una estructura que se propone inicialmente.
Pre-condiciones:	El Eclipse IDE 3.4 se encuentra abierto y el plugin Dalas IDE instalado.
Post-condiciones:	Se crea el Módulo.
Notas:	En la primera vista se llenan datos que son propios del módulo, luego en la otra vista se podrá conformar la estructura del módulo. En la segunda vista tanto los tipos de elementos como los paquetes se crean utilizando un pop up menú o un toolbar. También se puede, eliminar paquetes y tipos de elementos mediante el pop up menú o el toolbar.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
7.	Crear un paquete dentro de un tipo de elemento.	No permite hacerlo.	**X**	
8.	Eliminar un paquete.	Si confirma que desea eliminarlo, se debe eliminar el paquete seleccionado.	**X**	
9.	Eliminar un elemento.	Si confirma que desea eliminarlo, se debe eliminar el elemento seleccionado.	**X**	
10.	Renombrar un elemento o paquete.	Se debe de renombrar el elemento o paquete con el nombre introducido.	**X**	

Nombre de la Prueba:	Generar Módulos.
Caso de Uso Probado:	Generar Módulos.
Descripción de la Prueba:	Se crea la estructura del módulo, adicionando paquetes y tipos de elementos a una estructura que se propone inicialmente.
Pre-condiciones:	El Eclipse IDE 3.4 se encuentra abierto y el plugin Dalas IDE instalado.
Post-condiciones:	Se crea el Módulo.
Notas:	En la primera vista se llenan datos que son propios del módulo, luego en la otra vista se podrá conformar la estructura del módulo. En la segunda vista tanto los tipos de elementos como los paquetes se crean utilizando un pop up menú o un toolbar. También se puede, eliminar paquetes y tipos de elementos mediante el pop up menú o el toolbar.
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
11.	Presionar el botón Salvar template para guardar la estructura del Módulo.	Se muestra una vista para escoger el directorio donde se guardará.	**X**	
12.	Presiona el botón Guardar.	Se guarda la estructura en el directorio escogido.	**X**	
13.	Presionar el botón Cargar template para cargar la estructura del Módulo que desee.	Se muestra una vista para que escoja el directorio donde se encuentra.	**X**	
14.	Selecciona una estructura que no corresponde.	Se muestra un aviso de error: La estructura seleccionada no es la que se esperaba.	**X**	

Nombre de la Prueba:	Generar Módulos.			
Caso de Uso Probado:	Generar Módulos.			
Descripción de la Prueba:	Se crea la estructura del módulo, adicionando paquetes y tipos de elementos a una estructura que se propone inicialmente.			
Pre-condiciones:	El Eclipse IDE 3.4 se encuentra abierto y el plugin Dalas IDE instalado.			
Post-condiciones:	Se crea el Módulo.			
Notas:	En la primera vista se llenan datos que son propios del módulo, luego en la otra vista se podrá conformar la estructura del módulo. En la segunda vista tanto los tipos de elementos como los paquetes se crean utilizando un pop up menú o un toolbar. También se puede, eliminar paquetes y tipos de elementos mediante el pop up menú o el toolbar.			
Resultado: **(Pasa/Falla/aviso/incompleto)**				
	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
15.	Selecciona una estructura correcta.	Se reemplaza la estructura existente por la seleccionada.	**X**	

Tabla 9: Prueba Generar Módulos

Nombre de la Prueba:	Generar Subsistema.
Caso de Uso Probado:	Generar Subsistema.
Descripción de la Prueba:	Se crea un subsistema en un proyecto seleccionado.
Pre-condiciones:	El Eclipse IDE 3.4 se encuentra abierto y el plugin Dalas IDE instalado.

Post-condiciones:	Se crea el Subsistema.
Notas:	
Resultado: **(Pasa/Falla/aviso/incompleto)**	

	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	No introduce algún dato requerido.	Se muestra un aviso de error: Existen campos vacíos que son requeridos.	**X**	
2.	Introduce caracteres extraños.	Se muestra un aviso de error: El campo tiene caracteres extraños.		
3.	Llena el campo del proyecto incorrectamente.	Se muestra en aviso de error: Proyecto no válido.	**X**	
4.	Se llenan los campos correctamente.	Se debe crear el subsistema en el proyecto.	**X**	

Tabla 10: Prueba Generar Subsistema

Nombre de la Prueba:	Generar Clases de Patrones.
Caso de Uso Probado:	Generar Clases de Patrones.
Descripción de la Prueba:	Se crean las clases de patrones.
Pre-condiciones:	El Eclipse IDE 3.4 se encuentra abierto y el plugin Dalas IDE instalado.
Post-condiciones:	Se crean las clases.
Notas:	Las clases se generan a partir de los asistentes creados en las preferencias, aquí solo se valida en caso de que se cambie uno de los valores antes definidos.

Resultado: **(Pasa/Falla/aviso/incompleto)**				
	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	Cambia el nombre por uno con caracteres extraños.	Se muestra un aviso de error: El campo no puede contener caracteres extraños.	**X**	
2.	Si no llena el campo nombre.	Se muestra un aviso de error: El campo no puede estar vacío.	**X**	
3.	Si cambia el sufijo por uno con caracteres extraños.	Se muestra en aviso de error: El campo no puede contener caracteres extraños.	**X**	
4.	Si no llena el campo sufijo.	Se muestra un aviso de error: El campo no puede estar vacío.	**X**	
5.	Si no se selecciona el paquete contenedor.	Se muestra un aviso de error: El campo no puede estar vacío.	**X**	
6.	Todos los campos son válidos.	Se crea la clase de patrones en el paquete que corresponde.	**X**	

Tabla 11: Prueba Generar Clases de Patrones

Nombre de la Prueba:	Consultar Ayuda.
Caso de Uso Probado:	Consultar Ayuda.
Descripción de la Prueba:	Se consulta la ayuda del plugin integrada a Eclipse.
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.
Post-condiciones:	
Notas:	

Resultado: **(Pasa/Falla/aviso/incompleto)**				
	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	Seleccionar en el menú Help Contents la temática Dalas IDE.	El sistema muestra los temas referentes a la ayuda del plugin Dalas.	**X**	

Tabla 12: Prueba Consultar Ayuda

Nombre de la Prueba:	Graficar Estructura.			
Caso de Uso Probado:	Graficar Estructura.			
Descripción de la Prueba:	Se grafica la estructura de un proyecto.			
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.			
Post-condiciones:				
Notas:				
Resultado: **(Pasa/Falla/aviso/incompleto)**				
	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
1.	Seleccionar mediante el clic derecho sobre un proyecto la opción DalasTools -> Open Grahp.	El sistema muestra una vista gráfica de todos los módulos que conforman el proyecto, así como las relaciones entre los mismos.	**X**	
2.	Seleccionar mediante el clic derecho sobre el nodo que representa el módulo la opción Open module.properties.	El sistema abre el fichero module.properties correspondiente al módulo.	**X**	

Nombre de la Prueba:	Graficar Estructura.			
Caso de Uso Probado:	Graficar Estructura.			
Descripción de la Prueba:	Se grafica la estructura de un proyecto.			
Pre-condiciones:	El Eclipse IDE 3.5 se encuentra abierto y el plugin Dalas instalado.			
Post-condiciones:				
Notas:				
Resultado: **(Pasa/Falla/aviso/incompleto)**				
	Pasos de Prueba	**Resultados Esperados de Prueba**	**P**	**F**
3.	Seleccionar mediante el clic derecho sobre la vista la opción Save as image.	El sistema salva la imagen de la vista en el lugar seleccionado por el usuario dentro del workspace.	**X**	

Tabla 13: Prueba Graficar Estructura

3.2 Validación de la solución

Con el objetivo de validar el plugin Dalas IDE se utilizaron dos grupos de desarrolladores compuestos por 3 integrantes. El primer grupo de desarrolladores (Grupo1) se le brindó la posibilidad de hacer uso del plugin Dalas IDE para el desarrollo, pudiendo hacer uso anteriormente de la ayuda del mismo para conocer su funcionamiento, mientras que el segundo grupo (Grupo2) debe realizar este trabajo de forma manual.

A ambos grupos de les asignó la tarea de crear un proyecto Web utilizando la arquitectura Dalas, el proyecto debe estar conformado por un subsistema, que contenga a su vez 5 módulos y cada módulo 4 clases, la estructura de cada módulo debe regirse por una estructura que fue definida para ambos grupos y para las clases se definió en qué paquetes se crearán y qué estructura deberán tener.

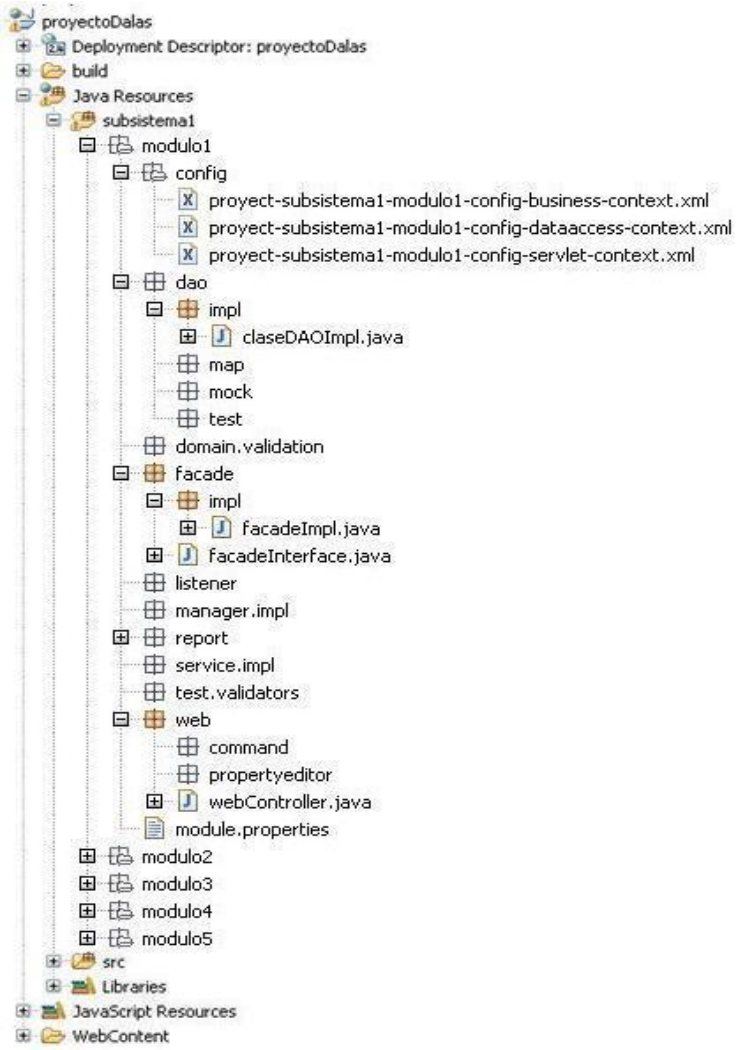

Fig. 11: Estructura del proyecto

En la figura anterior se muestra la estructura del proyecto creado, donde se encuentra expandido el módulo1 y dentro de él, los paquetes que contienen las clases y los ficheros. Los módulos que no se encuentran expandidos presentan la misma estructura que el módulo1.

Después de concluida las tareas por parte de ambos grupos, se obtuvo como resultado que el que hizo uso del plugin Dalas IDE concluyó sus tareas en un tiempo de 6 minutos con 20 segundos. Del tiempo total consumió 4 minutos y 40 segundos en la creación del proyecto con un subsistema y configurando el primer módulo, a partir de haber creado la estructura del primer módulo haciendo uso de las funcionalidades del plugin, el tiempo que consume generar un módulo es de sólo 20 segundos y teniendo en cuenta que se generaron 5 módulos se consumió un tiempo de 1 minuto y 40 segundos en esta tarea. Por otra parte, el grupo que realizó las tareas de forma manual le tomó concluir sus tareas 32 minutos, de los cuales invirtió 2 en la creación del proyecto con el subsistema y 6 minutos por cada módulo, o sea, 30 minutos para crear los 5 módulos.

Con respecto a la creación de las clases definidas para cada módulo, el grupo que hizo uso del plugin las terminó en un tiempo de 3 minutos y 55 segundos, de los cuales invirtió 3 minutos y 15 segundos en configurar las clases en las preferencias y el asistente en el cual iban a estar dichas clases, mientras que generarlas en cada módulo tomó un tiempo sólo de 8 segundos por módulo, lo cual completa los 40 segundos restantes. Por su parte, el grupo que no pudo realizar el proyecto con la ayuda del plugin, tardó un tiempo de 3 minutos y 50 segundos en crear las 4 clases de un módulo, por lo que en total para los 5 módulos tardó un tiempo de 17 minutos y 50 segundos.

Teniendo en cuenta los resultados obtenidos, se puede apreciar que el uso del plugin Dalas IDE agiliza en un tiempo considerable el desarrollo de aplicaciones Web que hagan uso de la arquitectura Dalas.

Luego de haber concluido los dos grupos de desarrolladores, pudieron hacer uso del plugin DalasGraphicalEditor para obtener una vista global de las relaciones entre los módulos y en ambos casos se mostraron los módulos y las relaciones correctamente.

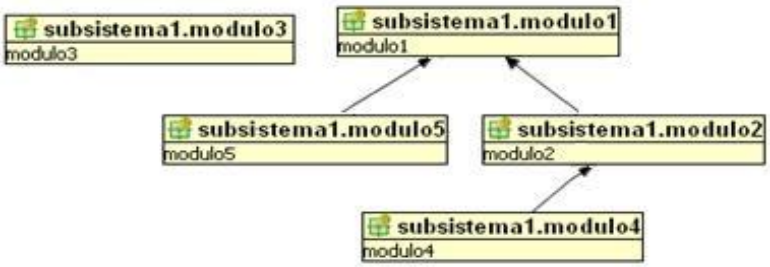

Fig. 12: Vista global proporcionada por el plugin DalasGraphicalEditor

3.3 Conclusiones del capítulo

A lo largo de este capítulo los resultados obtenidos por las pruebas realizadas al plugin Dalas IDE fueron excelentes, ya que éste agiliza el desarrollo del software al permitir configurar los diferentes elementos de un proyecto brindando la posibilidad de reutilizar esta configuración, además el desarrollador se abstrae de pensar en la creación de los tipos de elementos arquitectónicos y se enfoca en la implementación. Por tanto, se puede afirmar que los objetivos definidos en el presente trabajo se han cumplido satisfactoriamente.

CONCLUSIONES

El proceso de configurar la arquitectura Dalas para un proyecto de forma manual provoca un aumento del tiempo de desarrollo del software y mientras más grande sea el proyecto más tiempo consumirá esta labor.

Con el desarrollo de este plugin se ha logrado obtener un producto que cumple con los objetivos previstos, es decir, facilitar la configuración de la arquitectura Dalas en proyectos Web que utilicen la plataforma de desarrollo Eclipse, además de brindar un grupo de funcionalidades que facilitan la labor de los desarrolladores.

Beneficios que aporta el plugin:

- Integrar el Plugin con el WTP para crear proyectos Web dinámicos
- Gestionar las librerías a utilizar por el proyecto Web integrado al framework arquitectónico Dalas.
- Brindar la posibilidad de personalizar la generación de estructuras lógicas de un proyecto Web a la hora de crear los elementos que componen el mismo.
- Generar las estructuras arquitectónicas definidas por el framework Dalas.
- Registrar los elementos del proyecto con la semántica especificada por Dalas Framework en los properties.
- Brindar la posibilidad de personalizar los sufijos, paquetes, nombre de clases, así como nuevas clases dado el patrón de diseño especificado.
- Permitir a un arquitecto configurar desde cero, mediante wizards o asistentes, las estructuras que puede tener un proyecto Web que haga uso de Dalas.
- Graficar las relaciones entre los módulos que componen el proyecto Web integrado con Dalas.
- Posee una ayuda integrada al Eclipse.
- Posee asistentes y preferencias que ayudan al desarrollador en la creación de elementos totalmente personalizados.
- Reduce el tiempo de creación de los elementos así como la cantidad de errores.

RECOMENDACIONES

- Continuar el desarrollo del plugin brindando la posibilidad de que a la hora de generar clases se pueda configurar o crear un asistente si lo desea.
- Internacionalizar el plugin para otros idiomas.
- Adicionar funcionalidades de drag and drop para facilitar el trabajo con los Trees.

BIBLIOGRAFÍA

1. **Gueorgui, Obregón Obregón y Liyanis, Velázquez Galá.** *"ArBaWeb Integrator Tools: Plugin de Eclipse para desarrollar proyectos web integrados con ArBaWeb".* La Habana : s.n., 2008.

2. **Mabel, Navarro Bermúdez y Yissel, Rodríguez Aldana.** *"BioSyS: Implementación del Módulo de Simulación.".* La Habana : s.n., 2008.

3. **Lisbeth, León Téllez y Alain, Meneses Jiménez.** *BioSyS. Desarrollo de Portlets para la Gestión de la Información en la Simulación de Sistemas Biológicos.* La Habana : s.n., 2008.

4. **Guillermo, Isse Reyes y Héctor, Pérez Hernandez.** *"Sistema de Gestión de la Dirección de Deporte.".* La Habana : s.n., 2009.

5. **González, Héctor Luis López.** *Plug-in para Eclipse para la generación de elementos arquitectónicos personalizados.* La Habana : s.n., 2009. (1)

6. www.eclipse.org. *Eclipse Java development tools (JDT).* [En línea] [Citado el: 15 de 1 de 2010.] http://eclipse.org/jdt

7. **Altamiranda, M.J.** *Manual de usuario para el uso de Gforge.* Mérida,Venezuela : s.n., 2006.

8. www.visual-paradig.com. *UML CASE tools - Free for Learning UML, Cost-Effecive for Business Solutions.* [En línea] [Citado el: 15 de 1 de 2010.] www.visual-paradig.com/product/vpuml.

9. **Clayberg, Dan Rubel Eric.** *Eclipse:Building Commercial-Quality Plug-ins,Second Edition.* 2006.

10. www.eclipse.org. *About the Eclipse Fundation.* [En línea] [Citado el: 18 de 1 de 2010.] http://www.eclipse.org/org.

12. www.eclipse.org. *PDE.* [En línea] [Citado el: 16 de 4 de 2010.] http://www.eclipse.org/pde .

13. **Erich Gamma, Kent Beck.** *Contributing to Eclipse: Principles, Patterns, and Plug-Ins.* s.l. : Addison Wesley, 2003.

14. **Guojie, Jackwind Li.** *Professional Java Interfaces with SWT/JFace.* s.l. : John Wiley & Sons , 2005.

15. www.eclipse.org. *Extending WTP Using Project Facets.* [En línea] 2006. [Citado el: 5 de 4 de 2010.] http://www.eclipse.org/articles/Article-BuildingProjectFacets/tutorial.html.

16. epf.eclipse.org/. *OpenUp.* [En línea] [Citado el: 24 de 3 de 2010.] http://epf.eclipse.org/wikis/openup/index.htm.

17. **Gallardo, David.** Developing Eclipse plug-ins. [En línea] 1 de 12 de 2002. [Citado el: 25 de 1 de 2010.] http://www.ibm.com/developerworks/java/library/os-ecplug/.

18. **John Arthorne, Chris Laffra.** *Official Eclipse 3.0 Faqs.* s.l. : Addison-Wesley Professional, 2004.

19. **Kersten, Mik.** *Mylyn 2.0, Part 1: Integrated task management.* 2007.

20. **Pimentel González, Luis Alberto y Hernandez Suárez, Eivys.** *Framework Dalas.* La Habana : s.n.

21. **Pimentel González, Luis Alberto y Hernandez Suárez, Eivys.** *Framework Dalas.* La Habana : s.n.

22. **Cernosek, Gary.** IBM. [En línea] 15 de 11 de 2005. [Citado el: 24 de 2 de 2010.] http://www.ibm.com/developerworks/rational/library/nov05/cernosek/index.html.

23. eclipse.org. *About the Eclipse Foundation.* [En línea] [Citado el: 24 de 2 de 2010.] http://www.eclipse.org/org/.

24. **Roberth G. Figueroa, Camilo J. Solís.** *Metodologías Tradicionales vs. Metodologías Ágiles.* [En línea] [Citado el: 10 de 2 de 2010.] http://www.mygnet.net/manuales/software/metodologias_tradicionales_vs_dot_metodologias_agiles.1515.

25. **Rosales, Rolando Gutiérrez y Fleites, Dayana Cabrera.** *Adaptación de OpenUp/Basic para el Polo.* La Habana : s.n., 2009.