Universidad de las Ciencias Informáticas

Facultad 6

Título: "Plugin para el desarrollo de aplicaciones T-arenal".

Trabajo de Diploma para Optar por el Título de Ingeniero en Ciencias Informáticas

Autores:

Jesús Padrón Puyol

Daniel Alejandro Bolaños Solana

Tutores:

MSc. Longendri Aguilera Mendoza

Ing. César Raúl García Jacas

Ing. Roberto Tellez Ibarra

Junio 2010

"***La ciencia es conocimiento organizado. La sabiduría es la propia vida organizada.***"

Immanuel Kant

Declaración de Autoría

Declaramos ser 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 firmo la presente a los _____ días del mes _____ del año _____.

Jesús Padrón Puyol

Firma del Autor

Daniel Alejandro Bolaños Solana

Firma del Autor

Ing. César Raúl García Jacas

Firma del Tutor

Ing. Roberto Téllez Ibarra

Firma del Tutor

MSc. Longendri Aguilera Mendoza

Firma del Tutor

Datos del Contacto

Autores:

Jesús Padrón Puyol.

Universidad de las Ciencias Informáticas.

e-mail: jpuyol@estudiantes.uci.cu

Daniel Alejandro Bolaños Solana.

Universidad de las Ciencias Informáticas.

e-mail: dabolanos@estudiantes.uci.cu

Tutores:

Longendri Aguilera Mendoza.

Licenciado en Ciencias de la Computación.

Máster en Bioinformática.

Profesor Asistente.

e-mail: loge@uci.cu

César Raúl García Jacas.

Ingeniero en Ciencias Informáticas.

Instructor recién graduado.

e-mail: crjacas@uci.cu

Roberto Tellez Ibarra.

Ingeniero en Ciencias Informáticas.

Instructor recién graduado.

e-mail: rtibarra@uci.cu

Agradecimientos

Agradecer a nuestros padres por los sacrificios que han hecho para que nosotros estemos hoy aquí, por ser las luces que nos han guiado siempre. Agradecer a nuestros hermanos por hacer que nos esforcemos por ser mejores cada día. Agradecer además a nuestras familias que han estado presente siempre y nos han dicho que "si se puede".

Agradecer a todos nuestros amigos que nos han apoyado y han estado con nosotros en los buenos y malos momentos, por haber hecho más amena nuestra estancia en la Universidad. A los Fantásticos que siempre nos han apoyado.

Agradecer a nuestros tutores por haber sido, más que tutores, amigos, companeros que nos han dado el aliento y depositaron enteramente su confianza en nosotros.

Agradecer al tribunal y al oponente por habernos hecho estudiar y prepararnos cada vez mejor, por ensenarnos como se hacen las cosas.

Dedicatoria

A *nuestros padres por ser la razón de nuestra existencia.*

A *nuestros hermanos.*

A *nuestros Abuelos.*

A *nuestros Amigos.*

Resumen

En la Universidad de las Ciencias Informáticas(UCI) se desarrolló la Plataforma de Tareas Distribuidas alias T-arenal que permite la programación de tareas distribuidas y la ejecución de las mismas sobre un conglomerado de estaciones de trabajo.

En la actualidad, con vista al desarrollo de tareas distribuidas, esta posee algunas limitaciones, debido que el desarrollador debe implementar prácticamente a ciegas. No tiene la posibilidad de conocer si existe algún error, hasta haber obtenido los resultados desde el servidor; estos pueden ser satisfactorios o no, en caso de serlo, envía los resultados, en caso contrario envía un archivo de errores. Además no posee conocimientos sobre las clases que debe reimplementar, las características de los métodos y el esquema que debe seguir el XML de configuración para la subida de los archivos a procesar.

En este trabajo se presenta un plugin que dota a Eclipse de nuevas herramientas que facilitan el desarrollo de tareas distribuidas, dando solución a las limitaciones que poseía la Plataforma desde el punto de vista del desarrollo de tareas. Se añaden nuevos asistentes, vistas y acciones, las cuales dan paso a la emulación de la Plataforma en el propio IDE. Brinda el esquema del XML de configuración y la generación de las clases de forma transparente, además de un sistema de ayuda haciendo que el desarrollador pueda abstraerse completamente de estos procesos y darse cuenta de sus errores. Todas estas características permiten que el sistema de desarrollo de tareas distribuidas sea sencillo y confiable.

Palabras claves:

T-arenal, cómputo, plugins.

Abstract

At the University of Informatics Sciences (UCI) developed the Platform for Distributed Tasks, that enables distributed programming tasks and carrying them on a cluster of workstations.

Nowadays, with a view to the development of distributed tasks, this has some limitations because the developer must implement almost blind. They do not have the possibility to know if there are any error or not, until they have obtained the results from the server, these may be satisfactory or not, should send the results to be satisfactory, if not send an error file. Also knows nothing about the scheme should follow the configuration XML for the upload of the files to process, the classes must be reimplemented and the characteristics of the methods.

This work presents an Eclipse plugin that gives new tools that facilitate the development of distributed tasks, providing solutions to the constraints that had the platform from the point of view of development tasks. It added new wizards, views and actions which give way to the emulation of the platform in the IDE itself. XML schema provides the configuration and generation of classes in a transparent manner, in addition to providing a support system, making the developer can completely abstract from these processes, fix their mistakes, making the development system of distributed tasks easy and reliable.

Key Words

T-arenal, compute, plugins.

Índice general

Índice de figuras

Introducción

Desde tiempos remotos hasta nuestros días el hombre ha tenido que desarrollar trabajos que le superan. Sin mucho conocimiento, fue capaz de darse cuenta que dividiendo el trabajo entre varios podían superar estas barreras.

La humanidad ha desarrollado y usado por su conveniencia accesorios que abrevian y facilitan la práctica del cálculo. Transitando así, desde el ábaco hasta las máquinas que realizaban cálculos aritméticos. Estas en particular han tenido una vertiginosa evolución, llegando así a la computadora actual. Ello vino acompañado con el desarrollo de herramientas que facilitaran el trabajo que luego se denominó software; para darle solución a esta necesidad surgieron los lenguages de programación y con ellos los Entornos de Desarrollo Integrado(IDE). Su misión consistía en evitar tareas repetitivas, facilitar la escritura del código correcto y disminuir el tiempo de depuración maximizando así la productividad de los desarrolladores.

Todavía no existía un IDE universal o perfecto capaz de reunir todas las características que un desarrollador pueda necesitar, por lo que diferentes empresas se dieron a la tarea de la contrucción de un IDE capaz de aceptar en el futuro nuevas herramientas que los desarrolladores necesitasen para llevar a cabo su trabajo de una manera eficiente. Es por eso que surge para el desarrollo de estos IDEs una arquitectura que cumple con estas necesidades denominada: “arquitectura abierta, basada en plugins” que planteaba al IDE como un núcleo al cual se le puedan adicionar elementos, que pueden realizar tareas específicas. En la actualidad existen algunos que ya implementan esta arquitectura como Eclipse y la nueva versión de Netbeans los cuales facilitan el desarrollo de las nuevas herramientas en la plataforma que posteriormente las aceptará.

Antecedentes que fundamentan este estudio

Las investigaciones científicas que se realizan en nuestro país, principalmente en la rama de la Biotecnología, necesitan potencia de cómputo para el procesamiento y análisis de datos que normalmente en computadoras personales demorarían días, semanas o meses para su culminación.

El desarrollo precipitado de las redes de computadoras en los últimos años ha hecho reconsiderar la utilización de las costosas supercomputadoras para la ejecución de aplicaciones que demanden potencia de cómputo. Una simple computadora con memoria local y procesador de capacidades moderadas no es de mucha utilidad por sí misma, pero al ser conectada a otras máquinas a través de una red de interconexión suficientemente rápida, se eleva enormemente su utilidad ya que cientos o miles de máquinas podrían trabajar como un equipo, realizando intensos cálculos para dar solución a un determinado problema.[1]

Por este motivo, se desarrolló un software en la Universidad de las Ciencias Informáticas capaz de aglutinar un conglomerado de computadoras; este software tiene como alias T-arenal[2], el cual comienza a dar solución a los grandes problemas de cómputo, siendo necesario el despliegue de aplicaciones para este software. Con el fin de dar solución a los problemas de cómputo con dicha plataforma, se deben crear dos clases con algunos requisitos específicos, la primera que define el ciclo de vida, la segunda el algoritmo del sistema, además de importar la librería de acuerdo con la versión del servidor.

Situación Problémica

En la Universidad de las Ciencias Informáticas (UCI) creada en Cuba en el año 2002, está presente la Bioinformática como una rama de investigación y producción de software. Varios son los proyectos que se desarrollan con centros del Polo Científico de Cuba, entre los que se destacan, el Centro de Inmunología Molecular, Centro de Ingeniería Genética y Biotecnología, Centro de Química Farmacéutica y la Facultad de Química de la Universidad de La Habana.

[1] Se le llama supercomputadora a la computadora con capacidades de cálculo muy superiores a las comúnes
[2] Tarenal alias de la Plataforma de Tareas Distribuidas

Estos proyectos necesitan realizar una gran cantidad de cálculos que demoran un tiempo excesivamente largo en una sola computadora. Paradójicamente, la UCI es sin lugar a dudas el centro a nivel nacional con el mayor número de computadoras personales. En estos momentos cuenta con unas miles de PCs distribuidas en toda la red universitaria.

Para satisfacer la necesidad de los cálculos, la Universidad dispone al igual que otros centros del país de un clúster compuesto por ocho nodos. No obstante, se está desaprovechando tiempo de procesamiento sobre todo en los horarios nocturnos o no laborables, ya que existe un número elevado de estaciones de trabajo que no forman parte del clúster para realizar los cálculos. Es por ello que se creó la Plataforma de Tareas Distribuidas, con el objetivo de unir en un solo conglomerado un conjunto de PCs distribuidas en una red LAN[3].

Por todo lo anterior planteado, el proceso de desarrollo de tareas distribuidas se realiza de forma compleja y engorrosa, ya que debe tene pleno conocimiento de como es el proceso de desarrollo, teniendo en cuenta el particionado de los datos y como se ejecurá el algoritmo.

En la actualidad, este sistema cuenta con algunas limitaciones, entre ellas podemos mencionar: que los desarrolladores deben buscar la librería de acuerdo con la versión del servidor de la Plataforma de Tareas Distribuidas y con ella crear dos clases que hereden de DataManager y Task residentes en la librería, teniendo en cuenta como debe llamarse el paquete donde estarán dichas clases, esto está dado ya que en las versiones diferentes de la plataforma este toma nombres diversos. Después de reimplementar los métodos de las clases se deben subir al servidor y esperar la respuesta del mismo. Este analizará si está correcto o no, pero no pueden observar como sucede el proceso, ni como transcurre la división de las unidades de trabajo al no poseer dentro del IDE[4] una herramienta que les permita visualizar la ejecución de los procesos que realiza la Plataforma de Tareas Distribuidas, no pueden a través de los IDEs enviar los elementos hacia los servidores de la plataforma. No posee una herramienta que les permita de una manera sencilla definir los elementos que deben componer el XML de configuración, ya que este debe ser definido con una estructura específica de acuerdo con el esquema que define el servidor y es el encargado de decirle a la plataforma como son los ficheros que se van ha subir

[3]LAN: Local Area Network
[4]IDE: Entorno de Desarrollo Integrado

hacia el servidor.

Problema Científico

¿Cómo lograr que se desarrollen aplicaciones para la Plataforma de Tareas Distribuidas de una forma más amigable, sencilla y confiable?

Objeto de Estudio y Campo de Acción

Para dar respuesta al problema planteado se definió:

Como **objeto de estudio**: Plugins para Entornos de Desarrollo Integrado basados en Java.

Como **campo de acción**: Plugins para T-arenal basados en los IDEs de Java.

Objetivo General

Desarrollar un plugin para su integración con un IDE[5] basado en Java que permita emular la plataforma T-arenal para facilitar la programación de tareas y la detección temprana de posibles errores.

Objetivos Específicos

- Definir los requisitos funcionales y no funcionales del plugin.
- Realizar el análisis del plugin.
- Diseñar el plugin.
- Implementar el plugin diseñado.
- Validar las pruebas exploratorias al plugin implementado.

[5]IDE: Entorno de Desarrollo Integrado

Tareas

1. Estudio de las principales características de la Plataforma de Tareas Distribuidas.
2. Estudio de las arquitecturas de los Entornos de Desarrollo Integrado.
3. Definición del Modelo de Dominio del plugin.
4. Descripción de los requisitos funcionales y no funcionales del plugin.
5. Desarrollo del diagrama de casos de uso del plugin.
6. Descripción de los casos de uso del plugin.
7. Descripción de la arquitectura del plugin.
8. Desarrollo del diagrama de clases del diseño del plugin.
9. Implementación del plugin.
10. Realización de las pruebas exploratorias del plugin implementado.

Importancia

Con el presente trabajo se pretende desarrollar un plugin que de solución al problema científico propuesto, el cual ofrecerá diferentes bondades para el desarrollo con los esquemas de la Plataforma de Tareas Distribuidas. Brindará valiosas ayudas, entre las que se destacan la abstracción total del desarrollador de lo que es T-arenal y todos los procesos que en ella ocurren ya que posibilita el mejor entendimiento del desarrollador de los recursos que va a utilizar, haciendo que el proceso sea completamente transparente.

Estructura del documento

- **Capítulo 1 "Fundamentación Teórica":** se presenta una reseña bibliográfica donde se hace un análisis crítico de la literatura y una presentación de la información recopilada que está estrechamente relacionada con el tema tratado, así como las herramientas y las tecnologías a emplear durante el desarrollo del trabajo.

- **Capítulo 2 "Características del Sistema":** se presenta la descripción del sistema a desarrollar, los requisitos funcionales y no funcionales, los actores y el diagrama de casos de uso del sistema de cada módulo.
- **Capítulo 3 "Diseño del sistema":** se explican los patrones de desarrollo de software usados, se presentan los diagramas de clases del diseño y los diagramas de interacción. Finalmente se muestra el diagrama de despliegue con el objetivo de tener una idea más clara de como será desplegado el sistema.
- **Capítulo 4 "Implementación del Sistema":** se presentan los diagramas de componentes que representan cada uno de los módulos que componen el plugin final, así como los resultados de las pruebas exploratorias realizadas, además de las vistas del producto final.

Capítulo 1

Fundamentación Teórica

En este capítulo se presenta una reseña bibliográfica donde se hace un análisis crítico de la literatura y una presentación de la información recopilada que está estrechamente relacionada con el tema tratado, además de las herramientas y las tecnologías a emplear durante el desarrollo del presente trabajo.

1.1. Plataforma de Tareas Distribuidas

La Plataforma de Tareas Distribuidas es un sistema de cómputo distribuido programado en java y basado en el software Java Based Heterogeneous Distributed Computing System. Ha sido utilizado para dar solución a varios problemas de la Bioinformática y brinda un modelo de programación de alto nivel basado en el paradigma de la POO[1] utilizando a RMI[2] para el paso de mensajes.
El sistema T-arenal consiste en un servidor central (T-arenal server) y uno o más clientes (T-arenal client). El servidor central maneja toda la información concerniente al sistema, gestiona la transferencia de ficheros vía sockets TCP y planifica el orden en que los trabajos serán atendidos. Estos trabajos son ejecutados por los clientes los cuales pueden estar dedicados o no. Las personas que envían trabajos para el sistema son denominados "usuarios" [1].

[1]POO:Programación Orientada a Objetos
[2]RMI: API para la Invocación de Métodos Remotos

1.1.1. Características que propone Tarenal para el desarrollo de aplicaciones.

La Plataforma de Tareas Distribuidas plantea las siguientes características para el desarrollo de los problemas de tipo T-arenal:

1. Requerimientos de estructura de código
2. Convenciones o estándares de código y recursos

A continuación se describen las características de cada paso.

Requerimientos de estructura de código

Todo problema a desarrollar orientado a T-arenal debe cumplir con la estructura que ella define, como se muestra en la figura 1.1:

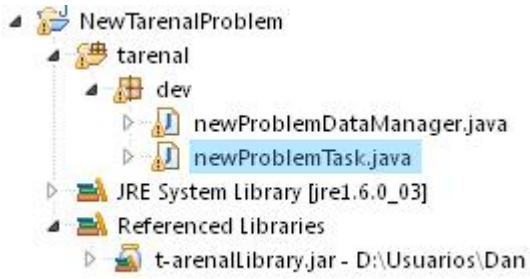

Figura 1.1: Arquitectura definida para un problema de tipo T-arenal

La estructura que define T-arenal para la implementación de sus problemas, es únicamente extender dos clases definidas por el sistema, que son DataManager y Task(estas clases serán analizadas posteriormente). Estas clases deben encontrarse definidas dentro de un paquete con un nombre específico que va ha estar dado según la versión del sistema, en este caso como se trabaja con la versión 2.0 el nombre del paquete es *tarenal.dev*. En estas aplicaciones se pueden utilizar librerías que deben ser incluidas en el sistema posteriormente. A continuación se muestra una breve descripción de la responsabilidad de las clases definidas que deben ser extendidas y se argumentan varios métodos que son de vital importancia para dar solución a los problemas a resolver.

Clase DataManager

La clase DataManager define el ciclo de vida del problema, o sea, es la encargada de generar las unidades de trabajo, procesar todos los resultados devueltos, ajustar el particionado de las unidades de trabajo generando la información de estado entre otras actividades. La definición de la clase se muestra en la figura 1.2 y el constructor no debe recibir parámetros. Las inicializaciones deben hacerse a través de ficheros. [2]

```
public class CustomDatamanager extends DataManager {
//declaración de los atributos...
        public newProblemDataManager() throws Throwable {
        // inicialización de los datos del objeto...
        }
//resto de la clase...
}
```

Figura 1.2: Declaración de la clase y el constructor de un DataManager extendido

Luego de haber implementado e inicializado las variables se debe pasar a definir o redefinir los métodos siguientes:

- generateWorkUnit: este método es llamado por el sistema cada vez que un cliente solicita una unidad de trabajo.
- processResults: este método es llamado cada vez que el cliente devuelve un conjunto de resultados.
- adjustGranularity: es el que se encarga del particionado dinámico del problema.
- getStatus: permite ver el estado en que se encuentra el problema.
- closeResources: este método es el encargado de cerrar todo recurso (ficheros, flujo de entrada y salida, conexiones) que pueda estar abierto.

Clase Task

La clase Task posee parte del código que corre en los clientes y procesa las unidades de trabajo generadas por el método generateWorkUnit de la clase DataManager. Solo un método se debe reimplementar en esta clase, este es processUnit el cual se muestra en la figura 1.3 [2].

```
public Vector processUnit( Vector workUnit ) throws Throwable {
        // procesar la unidad de trabajo enviada por el servidor y retornar un vector
        //que represente el conjunto de resultados
        Vector results = new Vector();
        //...
        return results;
}
```

Figura 1.3: Método processUnit de la clase Task

Convenciones o Estándares de código y recursos

T-arenal establece convenciones de nombres y estándares de código para los distintos recursos con el objetivo de lograr un lenguaje común y uniforme. Para las clases Java se hacen uso de las convenciones y estándares presentados en la especificación del lenguaje Java por la Sun Microsystem.

Clases Extendidas y paquetes

Las clases extendidas son las descendientes de DataManager y Task, las cuales deben poseer nombres compuestos:

[Custom] + [Nombre de la Clase Padre].java

- [Custom]: es el nombre del problema predefinido para un Proyecto T-arenal.
- [Nombre de la Clase Padre]: es el nombre de la cual heredan.

Estas clases se encontrarán en un paquete cuyo nombre estará definido por la versión del servidor. Para la versión 2.0 del servidor que es el utilizado en el presente trabajo será denominado *tarenal.dev*.

XML de configuración

El XML de configuración de un proyecto T-arenal deberá tener por nombre config.xml.

1.2. Herramientas que facilitan el desarrollo en entornos distribuidos.

En esta sección se mencionan algunas herramientas que se utilizan para la Simulación de Entornos Distribuidos(Distributed Simulation Enviroment).

1.2.1. SimGrid

SimGrid es una herramienta que provee diferentes funcionalidades para la simulación de aplicaciones distribuidas en entornos heterogéneos. El logro específico de este proyecto es el de facilitar la investigación en el área de aplicaciones distribuidas y paralelas en Plataformas de Cómputo [3]. Este proyecto está constituido por varios niveles como se muestra en la figura 1.4.

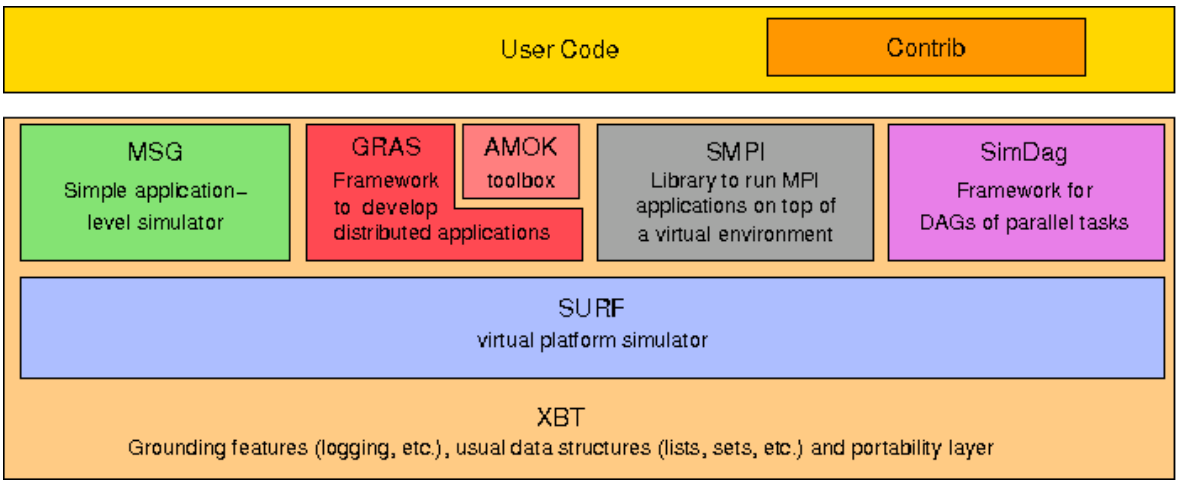

Figura 1.4: Niveles y módulos del proyecto SimGrid

SimGrid provee varios entornos de programación, cada entorno especifica y constituye diferentes paradigmas para que el desarrollador pueda elegir el que necesite, teniendo en cuenta lo que se quiere realizar y cual sería el resultado de su trabajo. Brevemente se describe en qué consiste cada uno de sus componentes:

- UserCode: es el código que brinda el desarrollador.
- MSG: para realizar un estudio de un problema teórico, y tener comparaciones heurísticas.
- SMPI: para realizar estudios con aplicaciones paralelas, usando técnicas de emulación y ejecución en entornos controlados de desarrollo.

- GRAS: para realizar estudios con aplicaciones distribuidas reales.
- SURF: este elemento es el que provee de las herramientas para la simulación.

GRAS

Este módulo de Simgrid nos proporciona un API completo para implementar aplicaciones distribuidas en la parte superior de plataformas heterogéneas. Además esta interfaz permite trabajar con la aplicación SimGrid en su interior, dándole al usuario comodidad para el desarrollo con el simulador, también le proporciona una eficaz ejecución adaptada a las plataformas reales. Por lo tanto, constituye una completa aplicación en el marco de la red de desarrollo, ya que abarcan tanto las herramientas del desarrollador y las de ayuda (el simulador y las herramientas asociadas), como una ejecución eficaz en tiempo de ejecución portátil.

1.2.2. g-Eclipse

El proyecto g-Eclipse tiene como objetivo construir e integrar frameworks para acceder al poder de las existentes infraestructuras Grid. Este será construido en la parte superior del ecosistema fiable de la comunidad Eclipse para permitir un desarrollo sostenible. Este proporciona herramientas para personalizar las aplicaciones de usuarios de la red, administrar los recursos Grid y apoyar el ciclo de desarrollo de nuevas aplicaciones. Por lo tanto, las herramientas ya existentes (como el escritorio de migración, la suite de GridBench y el núcleo de la visualización de cuadrícula (GVK)) será integrado. Este proyecto tiene como objetivo generar y proporcionar herramientas de trabajo Grid que puedan ser extendidos por diferentes middlewares (como gLite, Unicore, Globus). Utilizará la infraestructura de red ya existente dando acceso al poder de la infraestructura grid de una forma más intuitiva y sencilla. Los usuarios son capaces de acceder a la red con estandarizados, los proveedores de recursos reducen el costo de sus operaciones y los desarrolladores se benefician porque se acelera el proceso de desarrollo [4].

1.3. IDEs basados en Java con arquitectura de plugins.

Los IDEs son un conjunto de herramientas para el programador que suelen incluir en una misma suite, un buen editor de código, administrador de proyectos y archivos, enlace transparente a compiladores y debuggers

integrado con sistemas controladores de versiones o repositorios [5].

Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes, estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación, tales como C++, C#, Java y Pyton por sólo citar algunos.

En muchos casos puede dedicarse a un solo lenguaje de programación o a varios como por ejemplo la plataforma Eclipse y la Plataforma Netbeans que soportan C++, Java y otros.

1.3.1. Plataforma Netbeans

Netbeans es un IDE de código abierto cuya principal misión es hacer fácil el desarrollo de todo tipo de aplicaciones en la plataforma JAVA (JVM[3]). Con Netbeans es posible desarrollar tanto aplicaciones de escritorio como aplicaciones empresariales en varias capas, o programas para todo tipo de dispositivos móviles. Netbeans tiene una arquitectura modular que permite que con posterioridad se le añadan complementos (plugins). [6]

Además de ser un IDE especialmente indicado para desarrollar aplicaciones en java, Netbeans también es una plataforma de desarrollo. El IDE Netbeans está escrito sobre la plataforma Netbeans. Se puede escoger esta plataforma como base para desarrollar la aplicación que se necesite. La naturaleza modular de Netbeans haría posible desactivar aquellos módulos que no fueran útiles, y a su vez acoplar aquellos otros que si se necesiten, pues construir estos módulos para que se ejecuten de manera armoniosa y transparente dentro de Netbeans es muy fácil al proporcionar una API[4] simple y muy bien documentada para la creación de estos módulos. Esto hace que el uso de Netbeans como plataforma y no como IDE sea otro de sus puntos a destacar, lo cual proporcionaría gran cantidad de ahorro en tiempo de desarrollo al permitir construir aplicaciones con componentes ya probados y estables.

Una de las características que más se agradecen es la manera de obtener los plugins que cada día se parece más a los repositorios de software de las distribuciones Linux, lo cual hace comodísimo tener el IDE perfecta-

[3] Java Virtual Machine
[4] Application Programming Interface

mente adaptado a la tarea que se esté realizando. Con sólo un click se activan o desactivan módulos de modo que aquellos que no se utilicen no se cargan, aligerando así los recursos consumidos.

1.3.1.1. Diseño

La plataforma de Netbeans hace fuerte hincapié en la construcción del software de forma modular, módulo sobre módulo y así es como se le puede sacar mejor provecho ya que brinda implementados los mecanismos de descubrimiento de nuevos módulos (y de actualizaciones de los existentes) desde sitios remotos, resolución de dependencias, activación/desactivación de módulos en caliente, comunicación entre ellos, permitiendo así preocuparse por la lógica y rápidamente desplegar aplicaciones, pudiendo ir extendiendo su funcionalidad a medida que pasa el tiempo.

Se pueden crear aplicaciones conformadas por *varios* módulos diferentes, cada uno responsable de llevar a cabo determinadas funcionalidades, según el rol de la persona que vaya a utilizarlo. Solo se cargan en la aplicación los módulos para cumplir su tarea, permitiendo tener un abanico de aplicaciones sin tener que tirar una línea de código innecesaria.

1.3.1.2. Módulos

Un módulo es simplemente un típico JAR (Java Archive) con cierta metainformación almacenada en el manifiesto. Los módulos poseen la extensión NBM (Netbeans Module).

Dentro de la metainformación se encuentra la versión del módulo, las dependencias, descripción de funcionalidad y datos del autor. Un módulo puede ser usado en cualquier proyecto desarrollado sobre la Plataforma Netbeans (siempre que se cumplan las restricciones de dependencias del mismo), incluso sobre el IDE Netbeans. Se puede observar la estructura de un módulo en la figura 1.5.

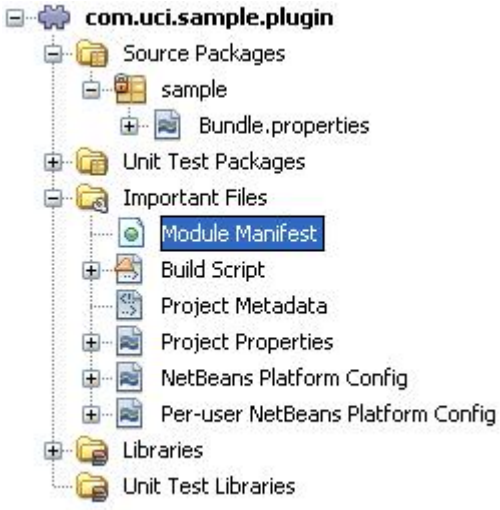

Figura 1.5: Composición de un módulo de Netbeans

1.3.2. Plataforma Eclipse

La plataforma Eclipse está diseñada para la construcción de Entornos de Desarrollo Integrados (IDEs) que puedan ser utilizados para la construcción de aplicaciones web, aplicaciones java de todo tipo, aplicaciones en C++, Enterprise JavaBeans (EJBs) [7].

A pesar de que la funcionalidad de Eclipse es mucha, gran parte es genérica. Permite que los componentes puedan utilizar nuevos tipos de contenido para realizar tareas con contenidos existentes. La Plataforma Eclipse permite descubrir e invocar funcionalidades implementadas en componentes llamados plugins. Un fabricante puede proporcionar una herramienta independiente como un plugin que permita llevar a cabo una determinada actividad. Cuando la Plataforma se inicializa, se muestran al usuario además del entorno de Eclipse todos los plugins instalados en el entorno. La calidad de la experiencia del usuario depende de cómo se integren los diferentes plugins con la Plataforma y cómo aquellos puedan comunicarse entre sí [8].

1.3.2.1. Diseño

La Plataforma Eclipse [9] (o simplemente "La Plataforma" cuando no haya riesgo de confusión) está diseñada para afrontar las siguientes necesidades:

- Soportar la construcción de gran variedad de herramientas de desarrollo.
- Soportar las herramientas proporcionadas por diferentes fabricantes de software independientes (ISV's).
- Soportar herramientas que permitan manipular diferentes contenidos (HTML, Java, C, JSP, EJB, XML, y GIF).
- Facilitar una integración transparente entre todas las herramientas y tipos de contenidos sin tener en cuenta el proveedor.
- Proporcionar entornos de desarrollo gráfico (GUI) o no gráficos.
- Ejecutarse en una gran variedad de sistemas operativos, incluyendo Windows® y Linux™.
- Hacer hincapié en que el lenguaje de programación sea Java para la construcción de nuevos plugins.

El principal objetivo de la Plataforma Eclipse es proporcionar mecanismos y reglas que puedan ser seguidas por los fabricantes para integrar de manera transparente sus herramientas. Mediante APIs[5], interfaces, clases y métodos se exponen estos mecanismos. La Plataforma también posibilita la construcción de herramientas que extenderán la funcionalidad de la Plataforma. La figura 1.6 muestra los componentes (APIs), de la Plataforma Eclipse. [10]

La plataforma Eclipse posee una arquitectura madura [11] [12], bien diseñada, extensible y adaptable para todo aquel que lo necesite. Con la excepción del núcleo, todo lo demás que compone a Eclipse son plugins como por ejemplo Worbench [13], Workspace por solo mencionar algunos, estos se pueden evidenciar en la figura 1.6.

[5] Application Programming Interface

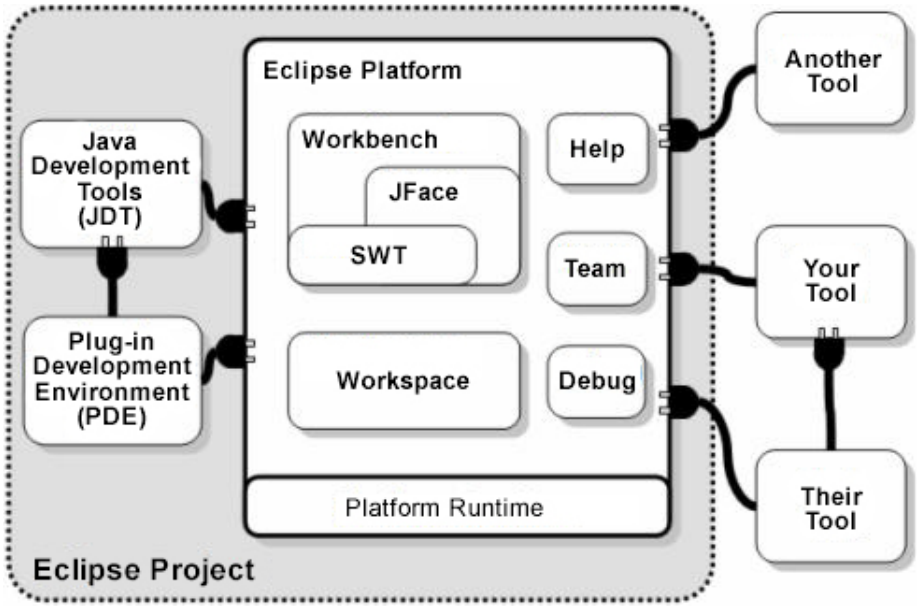

Figura 1.6: Arquitectura de la Plataforma Eclipse

El 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 (en inglés workspace), además contiene los puntos de extensión (ver epígrafe 1.4.2) para que otros plugins interactúen con los recursos de los usuario.

El Worbench[6] es el componente de la plataforma que proporciona toda la estructura y presenta una interfaz de usuario (UI) extensible al usuario, este posee dos herramientas de interfaz que son:

SWT: brinda un conjunto de utilidades y librerías gráficas integradas con el sistema nativo de ventanas pero con un API independiente al sistema operativo.

JFace: una interfaz gráfica implementado con SWT que nos simplifica las tareas de desarrollo.

Los plugins anteriormente mencionados son componentes escenciales de la plataforma Eclipse, además, la plataforma tiene integrados otros plugins como el Team que facilita el uso de control de versiones, para el

[6]Worbench: Banco de Trabajo

desarrollo en equipo y la Ayuda que proporciona la documentación online y además permite contribuir a ella, teniendo en cuenta el formato HTML que presenta y su lectura a través de XMLs externos.

Además de los plugins ya referidos se encuentran otros que aunque no son parte integral de la plataforma, se encuentran integrados con Eclipse SDK[7] que son Java Develop Tooling (JDT) y el Ambiente de Desarrollo de Plugin o en inglés Plugin Development Enviroment (PDE), que como sus nombres lo indican son módulos destinados para la implementación de nuevas herramientas o sea para crearle al Eclipse nuevas herramientas.

Hasta el momento se hace referencia a plugins y puntos de extensión pero no se han definido concretamente los conceptos anteriores por lo que en el próximo epígrafe los abordaremos.

1.3.2.2. Plugins y Puntos de Extensión

Imaginen que la Plataforma Eclipse no es más que un barco y que los plugin son las tareas que puede adquirir el barco, primeramente él está provisto de algunos componentes como la máquina, la cual le da la función de moverse. Si usted se percata bien el plugin que brinda cada herramienta lo podemos utilizar según las necesidades del mismo. Estos lugares serían los puntos de extensión.

Un plugin es la unidad más atómica y extensible de Eclipse. Está escrito puramente en el lenguaje Java y típicamente consiste en código empaquetado en un archivo de Java (JAR), con algunos archivos de solo lectura, y otros recursos como imágenes, catálogo de mensajes, entre otros.

Anatomía de los Plugins

Cada plugin está compuesto por una serie de ficheros y directorios que les dan la estructura, función y comunicación con otros plugins, además de los que definen su ciclo de vida. En el presente epígrafe se dará una breve descripción de los componentes medulares como se muestra en la figura 1.7.

[7] se denomina también proyecto Eclipse. Es la forma entregable de la plataforma Eclipse, y que al descargarse contiene también además de la Plataforma Eclipse los plugins JDT y PDE.

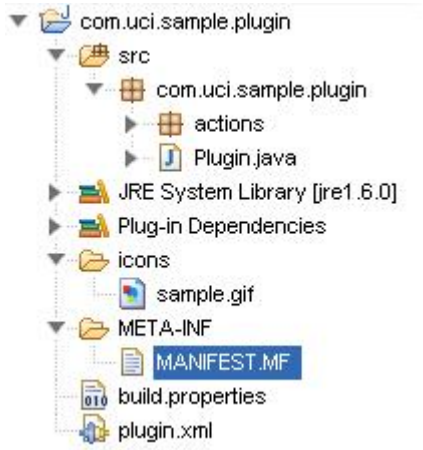

Figura 1.7: Estructura de un plugin de Eclipse

Cada plugin está compuesto por un manifiesto, en el cual se definen las relaciones con otros plugins, se declaran un número determinado de extensiones que se han realizado de otros plugins.

El manifiesto del plugin está representado principalmente por un par de ficheros, el *MANIFEST.MF* y el *plugin.xml*. A continuación se hará una breve descripción de los mismos:

- *MANIFEST.MF:* es generado dentro del directorio META-INF cuando se crea un proyecto de tipo plugin (en inglés Plugin Project). Es un manifiesto del paquete OSGi[8] [14] en donde se describen las dependencias que posee el plugin con otros en tiempo de ejecución y sus atributos como se muestra a continuación en la figura 1.8.

[8]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.

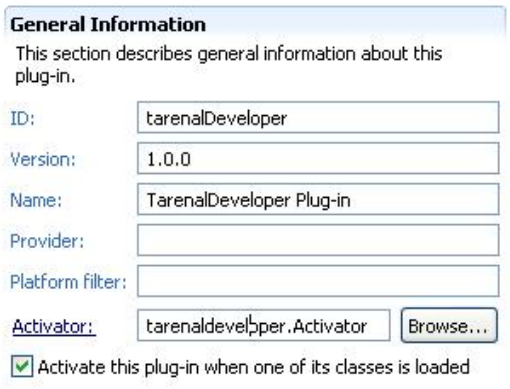

Figura 1.8: Sección del Manifest.MF

- *plugin.xml:* este archivo se encuentra en el directorio raíz del proyecto con formato XML que describe todas las extensiones realizadas por el plugin y se declaran además los propios puntos de extensión del mismo.

Además de estos dos archivos, existen una serie de directorios y ficheros con los cuales el plugin interactúa, como los iconos que se encuentran en el directorio definido para los mismos, usualmente en formato GIF, JPG o PNG; *about.xml* que es el fichero con formato HTML que brinda información acerca de las licencias y el *plugin.properties* el cual posee cadenas externalizadas que son utilizadas en el archivo *plugin.xml*.

1.4. Plugins existentes que falicitan la ejecución de tareas.

A través de los años, los desarrolladores han buscado la manera de implementar herramientas que les permitan generar tareas que deben realizar a diario, las cuales pueden ser desde un pequeño archivo hasta grandes proyectos. Mucho se ha avanzado al respecto y actualmente plataformas de desarrollo como Eclipse y Netbeans poseen cientos de herramientas que le brindan facilidades a los desarrolladores, estas herramientas son llamadas plugins.

El objetivo de este epígrafe es analizar los plugins desarrollados en la universidad como en el mundo con relación a la generación de elementos para una tarea determinada, al igual que aquellos que las ejecuten; de ellos se abordarán las diferentes características que ofrecen.

1.4.1. Java Code Generator

Java Code Generetor es un plugin de la Plataforma Eclipse para la generación de códigos Java a través de wizards[9]. Es muy flexible en cuanto a la entrada y salida de datos, esto se logra por una arquitectura altamente configurable mediante plantillas. Un escenario sencillo es generar una clase Bean para algunas tablas de base de datos. Este proceso se divide en dos iteraciones.

La primera, es el análisis de la entrada, la cual consiste en que cualquier tipo de archivo puede ser desde una tabla dentro de una base de datos física, hasta un script para generar las tablas. La entrada es analizada e insertada en un archivo con formato XML. Algunos métodos de insumo son suministrados por el plugin, otros son para las extensiones personalizadas basados en el posterior tratamiento del archivo XML generado o el DataModel[10] subrayado y su representación como un objeto que se utilizará.

La segunda, es la generación de código, por la cual hay varias maneras de hacerlo: la primera es la generación de las clases java-JET con una transformación basada en la entrada XML ya creados anteriormente y la otra generando un árbol de sintaxis abstracta basada en una implementación de referencias con el modelo de entradas combinadas [15].

1.4.2. TPTP Platform Plugin

TPTP es un plugin que brinda una plataforma de Ecuaciones en un Entorno Controlado, el cual utiliza componentes de recopilación de datos y no brinda la posibilidad de crear y consumir servicios de recolección de datos. Este posee una arquitectura orientada Cliente-Servidor lo cual permite hacer transferencias de datos en ambos sentidos.

El agente, se encuentra de manera independiente al cliente es un demonio que se encuentra de manera remota o local, el controlador del agente proporciona la arquitectura para que posea la capacidad de controlar y recoger la información en un momento dado. El marco de recopilación de datos se compone de un conjunto

[9] wizards: asistentes
[10] DataModel: modelo de datos

de clases abstractas y concretas, el cual guiará el procesamiento, recolección y transferencia de datos [16].

1.5. Herramientas y tecnologías

Existen creencias de que un grupo de desarrollo debería organizarse en torno a las habilidades de los individuos altamente calificados, que saben cómo hacer el trabajo, lo hacen bien y que raramente necesitan dirección. Esto constituye una equivocación en la mayoría de los casos, y un grave error en el desarrollo de software. Por lo tanto, es necesario un proceso que esté ampliamente disponible de forma que todos los interesados puedan comprender su papel en el desarrollo donde se encuentran implicados. Todo esto unido a una correcta selección de las herramientas, métodos, técnicas y procedimientos que ayuden a obtener un producto de elevada calidad.

1.5.1. Metodología de desarrollo de software

Uno de los principales retos que enfrentan los desarrolladores de software es obtener un producto con calidad y de manera eficiente, lo cual supone un paso importante hacer una selección correcta de las herramientas y procesos necesarios. Con este propósito se crearon diferentes metodologías de desarrollo, las cuales son un conjunto de pasos y procedimientos que deben seguirse. Con los años y las experiencias que han ido teniendo las diferentes instituciones productoras de software, se han ido mejorando algunas metodologías y se han creado otras con el objetivo de aumentar la productividad.

Entre la familia de metodologías existen las denominadas ágiles, las cuales intentan evitar los intrincados y burocráticos caminos de las metodologías tradicionales, enfocándose en las personas y los resultados. Estas metodologías recogen ventajas de las metodologías tradicionales e incorporan características nuevas que hacen que el proceso de desarrollo sea más simple, basándose en que lo más importante en un proyecto es valorar más a los individuos que a los procesos y herramientas, al software que funciona más que a la documentación exhaustiva, a la colaboración del cliente más que a la negociación contractual, a la respuesta al cambio más que al seguimiento de un plan.

Elegir la metodología adecuada es vital para lograr un sistema de alta calidad en un tiempo razonablemen-

te corto. Existen varias metodologías de desarrollo de software, dentro de las cuales se encuentran RUP [17], XP [18] y OpenUP [19].

Esta última es una de las metodologías ágiles de desarrollo de software. Ofrece las mejores prácticas de una variedad de líderes en ideas sobre producción de software y comunidades de desarrollo que cubren un diverso conjunto de perspectivas y necesidades de desarrollo. Preserva las características esenciales de RUP que incluye el desarrollo interactivo, casos de uso y escenarios de conducción de desarrollo, la gestión de riesgo y el enfoque centrado en la arquitectura.

OpenUP/Basic es la forma más ágil y ligera de OpenUP, y se basa en una donación de código abierto al proceso de contenido, conocido como el Proceso Unificado Básico (BUP). La mayoría de las partes de RUP han sido excluidas de esta metodología y muchos elementos se han fusionado, siendo el resultado un proceso mucho más sencillo que coincide con los principios de RUP. OpenUP/Basic es aplicable a proyectos pequeños con grupos de 3 a 6 personas interesadas en el desarrollo rápido e interactivo.

Además de lo anterior, OpenUP/Basic define un proceso de desarrollo de software mínimo y completo. Mínimo porque solamente lo fundamental es incluido dentro del proceso, y completo porque define un conjunto de componentes que guían y definen dicho proceso de desarrollo hasta la obtención del producto. Es extensible, de manera que se pueden añadir artefactos, actividades u otro componente a la metodología, según lo requiera el sistema que se desarrolla. Por las razones antes expuestas y por el desarrollo del proyecto T-arenal con esta metodología, OpenUP/Basic constituye la metodología seleccionada para el desarrollo de la aplicación que se propone en el presente trabajo.

1.5.2. Lenguaje de modelado

Para dar solución al problema planteado se usa como lenguaje de modelado UML [20] [21]. El Lenguaje Unificado de Modelado (UML) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir. Se usa para entender, diseñar, hojear, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del

ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. UML incluye conceptos semánticos, notación, y principios generales. Tiene partes estáticas, dinámicas, de entorno y organizativas. Está pensado para ser utilizado en herramientas interactivas de modelado visual que tengan generadores de código así como generadores de informes. La especificación de UML no define un proceso estándar pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos.

1.5.3. Herramienta CASE

Las herramientas CASE (Computer Aided Software Engineering) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo del sistema de información, completamente o en algunas fases. Como ejemplo de herramientas CASE tenemos MagicDraw, Rational Rose, Umbrello, Visual Paradigm for UML, ArgoUML y otras. Por decisión del proyecto T-arenal, la Herramienta CASE a utilizar es Visual Paradigm.

Visual Paradigm es una herramienta CASE que soporta la última versión del Lenguaje de Modelado Unificado (UML) y la Notación del Proceso de Modelado de Negocio (BPMN), y genera código para un gran número de lenguajes de programación. Además brinda una versión libre para uso no comercial. La herramienta fue desarrollada para una amplia gama de usuarios incluyendo ingenieros de software, analistas de sistemas, analistas del negocio y arquitectos de sistemas. Permite la integración con varias herramientas de Java, y brinda además una gran interoperabilidad con otras herramientas CASE como Rational Rose.

1.5.4. IDE y lenguaje de programación

El IDE Eclipse es un entorno de desarrollo de Java que emplea módulos (en inglés plugin) para proporcionar toda su funcionalidad, a diferencia de otros entornos monolíticos donde las funcionalidades están generalmente prefijadas, las necesite el usuario o no. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes de programación además de Java, así como introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo.

Este IDE fue seleccionado principalmente por su potente editor de código, la interfaz amigable que presenta y la existencia en el ciberespacio de una gran cantidad de plugins que hacen del IDE una herramienta potente, además de su arquitectura basada en plugins. Entre otras de sus características tenemos que es un software libre y multiplataforma.

El lenguaje de programación seleccionado es Java. La razón principal de esta selección es que es independiente de plataforma y arquitectura de red. Por eso una aplicación escrita en Java, puede ejecutarse en cualquier sistema. Esta portabilidad es extremadamente importante para cualquier sistema distribuido, ya que se espera que los clientes puedan correr en múltiples sitios en diferentes plataformas.

En los primeros días de Java, gran parte de sus críticas se originaban de su pobre rendimiento respecto a lenguajes nativos como C y Fortran. Mucho ha cambiado desde entonces. Actualmente se han producido enormes mejoras en el rendimiento de la Java Virtual Machine (JVM), principalmente atribuida a la introducción del compilador just-in-time y la tecnología hotspot [22]. Estas mejoras han dado lugar a que la ejecución de la JVM, sea comparable a la de otros lenguajes nativos [23]. Otro de los aspectos más importante de Java es el modelo de seguridad [24] [25]. Este simplifica enormemente la implementación de una estricta política de seguridad para una aplicación Java. Esto posibilita que las aplicaciones puedan cargar dinámicamente código, sin tener que preocuparse por las posibles implicaciones. Además de las características mencionadas con anterioridad, Java constituye un lenguaje *"simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico"* [26].

1.5.5. Herramientas para el desarrollo

JDT (Java Development Tooling)

Es uno de los principales subproyectos de Eclipse, ya que el mismo hace al IDE una potente herramienta. Es uno de los conjuntos de plugins más avanzados y elaborados que posee. Asiste a los usuarios en los procesos de escribir, depurar, compilar, probar y editar programas en el lenguaje de programación JAVA incluyendo el desarrollo de plugin de utilidad para el usuario. Esta posee varias propiedades y funcionalidades de las cuales solo vamos a mencionar algunas de ellas. Ofrece editores para todos los archivos de un proyecto Java brindándole funcionalidades como completamiento de código.Provee de un conjunto de vistas para navegar

y administrar proyectos Java como por ejemplo el Explorador de Paquetes. Incluye un amplio soporte para configurar el classpath[11], dependencias, librerías entre otros. Además permite realizar búsquedas complejas de métodos, atributos, ficheros e incluso plugins. Posee una enorme cantidad de wizards[12] [27] que permite la posibilidad de crear diversos elementos y posee además un ambiente enriquecido para depurar errores.

Además de que está dividido en tres secciones; la primera JDT Core que define el núcleo de los elementos Java y provee clases útiles para manipular archivos de extensión .class[13] y el modelo de elementos Java. La segunda JDT UI que contiene las interfaces de usuario que provee el IDE y la tercera JDT Debug que brinda un conjunto de clases que nos permiten correr y depurar código Java [28].

PDE (Plugin Development Environment)

El PDE [29] no es más que el ambiente de desarrollo para plugins que provee Eclipse. Esta compuesto por un conjunto de herramientas que cubren todo el ciclo de desarrollo de los plugins. Facilita todo el proceso de crear, desarrollar, probar, depurar, construir y desplegar plugins. Algunas de estas herramientas incluyen Editores de Manifiesto, asistentes para la creación de nuevos proyectos, asistentes para importar y exportar, launchers[14] y vistas por solo citar algunos. Todas estas Herramientas hacen que el proceso de desarrollo de un plugin, sea más fácil, mas legible y las interfaces más amigables.

SubEclipse

Es un plugin que agrega soporte subversion al proyecto Eclipse y que permite realizar entre otras tareas la de sincronizar y actualizar los elementos de un proyecto, posee una perspectiva llamada “SVN Repository Exploring” para trabajar con varios repositorios SVN al mismo tiempo, posee vistas para el manejo de los mismos y da la opción de interactuar con ellos desde el mismo eclipse.

[11] classpath: es un argumento que le muestra a la Maquina Virtual de Java donde buscar las clases y paquetes definidos por el usuario en un programa Java.
[12] wizards: asistentes.
[13] .class: extensión de archivos generados por el compilador por cada archivo .java.
[14] launchers: Probadores

JUnit

JUnit permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de una clase se comporta como se espera. Es decir, en función de algún valor de entrada se evalúa el valor de retorno esperado, si la clase cumple con la especificación, entonces JUnit devolverá que el método de la clase pasó exitosamente la prueba; en caso de que el valor esperado sea diferente al que regresó el método durante la ejecución, JUnit devolverá un fallo en el método correspondiente. En la actualidad las herramientas de desarrollo como Netbeans y Eclipse cuentan con plugins que permiten que la generación de las plantillas necesarias para la creación de las pruebas de una clase Java se realice de manera automática, facilitando al programador enfocarse en la prueba y el resultado esperado, y dejando a la herramienta la creación de las clases que permiten coordinar las pruebas.

1.6. Conclusiones

Los software analizados anteriormente en la sección 1.2 proveen de una serie de características para la tecnología *Grid*, en caso del *SimGrid*, y en el caso del *gEclipse* es un IDE con plugins de conexión pero no posee las caracteristicas necesarias para darle solución a nuestro problema, ahunque aportaron conocimientos para el desarrollo del presente trabajo. Para el desarrollo de plugins en java, existen incontables IDEs que lo posibilitan, entre los que se encuentran *Netbeans* y Eclipse, los cuales fueron abordados anteriormente en la sección 1.3, teniendo en cuenta las características de cada uno de ellos, la documentación de las tecnologías utilizadas por ellos y las facilidades de desarrollo, en el presente trabajo se seleccionó como IDE a *Eclipse* ya que este cuenta con una amplia documentación, ejemplos de código, una buena descripción de las tecnologías utilizadas, entre otras, entre las limitaciones de Netbeans, es que el mismo se encuentra restringido para nuestro país y entorpecería el proceso de búsqueda de unformación para la construcción del plugin. Para el proceso de desarrollo de la aplicación se utilizará la metodología OpenUP/Basic para lograr una mayor organización. Como lenguaje de modelado UML[15] y como herramienta CASE se seleccionó Visual Paradigm. Dando paso así al Diseño del Sistema.

[15]UML: Lenguaje Unificado de Modelado

Capítulo 2

Características del Sistema

En general, cuando las personas abordan el desarrollo de cualquier sistema, este evoluciona desde ideas abstractas hasta concreciones realizables, por ello en este capítulo se describen las principales características del sistema a desarrollar, sus requisitos funcionales y no funcionales y los actores que intervienen en el mismo. Además, se presenta el diagrama de casos de usos del sistema, así como una breve descripción de los casos de usos identificados.

2.1. Breve Descripción del Sistema

El plugin a desarrollar permitirá a los desarrolladores de una forma confiable, amigable y sencilla la realización de aplicaciones distribuidas, teniendo como base la creación, ejecución y monitoreo de problemas T-arenal, brindando nuevas funcionalidades y una arquitectura definida (por las librerías T-arenal) para el desarrollo de aplicaciones en el IDE Eclipse, definiendo así una estructura la cual se aborda en los próximos epígrafes.

2.1.1. Estructura del plugin

El diseño que se propone está basado en las características expuestas anteriormente. El plugin se denomina T-arenal Developer Tools (TDT) o Herramienta de desarrollo para T-arenal. En la figura 2.1 se muestra la estructura básica de TarenalDeveloperTools. El mismo contiene tres subproyectos: TarenalDeveloperTools, TarenalDeveloperToolsLibrary, TarenalDeveloperToolsHelp. El segundo provee librerías y el sistema de clases

que posteriormente se generará en el primero. El tercero es el sistema que guiará al usuario en la utilización del plugin.

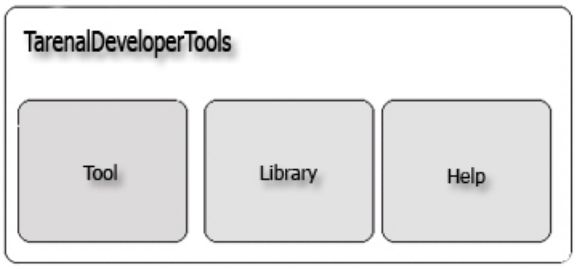

Figura 2.1: Estructura del Plugin

El componente Tools, es un plugin que permite en el proceso de creación de un proyecto se le agreguen asistentes para un nuevo tipo de proyecto que se denomina T-arenal. Otra de las responsabilidades que toma es la de encargarse de adicionar nuevas funcionalidades en el sistema de menú que contienen las acciones a realizar, estas accederán a recursos como son los asistentes(wizards) y vistas(views). Además tendrá la responsabilidad de cuando exista un proyecto creado poder ejecutarlo en un entorno controlado de desarrollo, mostrando lo que sucede. El subproyecto Library, tiene adscrita la librería proporcionada para el desarrollo de sistemas T-arenal, la cual nos provee de la arquitectura a seguir para el desarrollo de aplicaciones de este tipo.
El subproyecto Library, tiene adscrita la librería proporcionada para el desarrollo de sistemas T-arenal. La cual provee de la arquitectura a seguir para el desarrollo de aplicaciones T-arenal.

El subproyecto Help es el componente que se responsabiliza en ofrecer la documentación necesaria para desarrollar en TDT. Muestra a través de un manual de usuario integrado a la Ayuda de Eclipse con todos los pasos a seguir, conceptos y ejemplos.

2.2. Modelo de Dominio

En la figura 2.2 se muestran los conceptos de mayor importancia en el área de la aplicación así como las relaciones entre ellos.

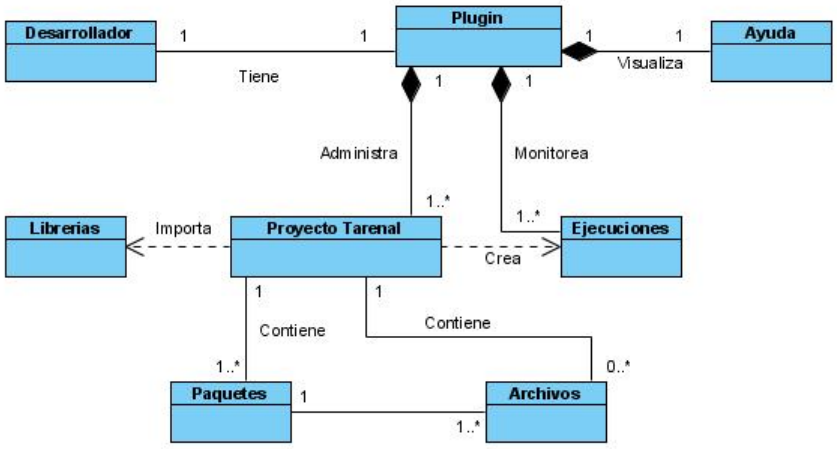

Figura 2.2: Modelo de Dominio

2.2.1. Descripción de los Objetos

DESARROLLADOR: El desarrollador representa al ente que tiene un determinado problema T-arenal que desarrollar.

PLUGIN: Representa la herramienta con la cual el desarrollador interactúa para darle solución a un determinado problema.

AYUDA: Representa uno de los componente de la herramienta, que es la encargada de brindar la documentación sobre mismo.

LIBRERÍAS: Representa aquellos JAR que serán incluidos en un Proyecto.

PROYECTO TARENAL: Representa un problema T-arenal a modo de proyecto.

PAQUETES: Representa la Entidad que encapsula archivos.

ARCHIVOS: Representa Clases, XMLs, cualquier tipo de elemento necesario para un proyecto.

EJECUCIONES: Representa las ejecuciones de los proyectos T-arenal.

2.3. Especificación de los Requisitos del Sistema

Los requisitos o requerimientos no funcionales son aquellas características o propiedades que el producto debe poseer y que harán del mismo confiable y seguro.

2.3.1. Requisitos Funcionales

Los requisitos funcionales no son más que los encargados de describir las funcionalidades o servicios que proveerá el sistema, o sea las propiedades o cualidades que se relacionen. En vista al desarrollo del presente trabajo se obtuvieron los siguientes:

1. Administrar Proyecto T-arenal
   1.1 Crear Proyecto T-arenal
   1.2 Generar Paquete (tarenal.dev)
   1.3 Adicionar Librerías (Nativas + tarenalLibrary)
   1.4 Generar Clases (DataManager + Task)
   1.5 Generar XML de Configuración

2. Ejecutar Problema

3. Monitorear Ejecución
   3.1 Devolver el Estado de la Ejecución
   3.2 Devolver los Errores de la Ejecución

4. Mostrar Ayuda

2.3.2. Requisitos no Funcionales

Los requerimientos no funcionales son las propiedades emergentes que va a poseer el sistema y que harán del mismo un producto seguro, confiable y de respuesta rápida.

Transparencia

- El plugin ocultará la naturaleza distribuida, permitiendo que los usuarios interactúen con un conjunto de interfaces que posibilitan la abstracción de cómo está conformado T-arenal.
- Los desarrolladores o usuarios no tienen que encargarse de repartir el cálculo entre los nodos y no tiene

que interactuar con los servidores T-arenal durante el proceso de desarrollo ya que el plugin ejecuta la tarea en un entorno controlado.

Eficiencia

- La solución a los problemas se obtienen más rápido haciendo uso del plugin ya que la interacción desarrollador-servidores (T-arenal) es mínima.

Flexibilidad

- Un proyecto en desarrollo como es la implementación de un plugin debe estar abierto a modificaciones que mejoren su funcionamiento o que pueda añadírsele nuevas funcionalidades. El presente trabajo es lo suficientemente flexible para que cualquier cambio no requiera de la parada total del sistema y la recopilación de todo el código. Si se realizaran cambios en la librería, ese solo fragmento es el que se debe recompilar, al igual que los otros ya que ellos están relacionados pero existe su independencia particular.

Fiabilidad

- Asegurará al 100 % que el problema desarrollado no presentará problemas a la hora de desplegarlo en los servidores T-arenal, exceptuando los problemas externos como fallas en la red, de fluido eléctrico, entre otros.

Portabilidad

- El sistema será multiplataforma ya que el plugin se desarrollará sobre Eclipse y para él. El plugin puede ser incorporado a cualquier versión del Eclipse, por lo cual podrá ser utilizado en cualquier Sistema Operativo.

Software

- Para el uso del plugin se debe disponer de la versión 3.2 del Eclipse o superior.

2.4. Definición de los Casos de Usos del Sistema

Los casos de usos representan las acciones que hace el sistema desde el punto de vista del usuario. Es decir, describen el uso del sistema y cómo este interactúa con el usuario. Para ello se identifican actores y casos de usos que participarán en las acciones del sistema, los cuales se argumentan a lo largo de este capítulo.

2.4.1. Actores del Sistema

Los actores del sistema representan a terceros fuera del sistema que interactúan con él. En el sistema que se describe se identificaron los siguientes (ver Tabla 2.1):

Actor	Descripción
Desarrollador	El desarrollador es el encargado de iniciar los casos de usos del sistema.
Eclipse	Es el IDE con el cual se integra el plugin y es iniciado por los casos de uso.

Cuadro 2.1: Actores del Sistema

2.4.2. Listado de Casos de Usos del Sistema

Los casos de usos identificados en este sistema se enuncian en la Tabla 2.2 :

Orden	Nombre	Prioridad	Breve Descripción
1	Administrar Proyecto T-arenal (Ver Anexo A)	Crítico	El caso de uso inicia con la necesidad del desarrollador, de realizar una aplicación para desplegarla en un servidor T-arenal.
2	Ejecutar Problema (Ver Anexo A)	Crítico	El caso de uso inicia con la necesidad del desarrollador de ejecutar el proyecto T-arenal.
3	Monitorear Ejecución (Ver Anexo A)	Crítico	El caso de uso inicia con la necesidad de ver qué es lo que esta sucediendo en la ejecución, o sea, como se va comportando.
4	Mostrar Ayuda	Secundario	El caso de uso inicia con la necesidad del desarrollador de comprender como interactuar con el plugin.

Cuadro 2.2: Descripción de Casos de Uso

2.4.3. Diagrama de Casos de Usos del Sistema

A continuación se muestra en forma de diagrama lo que hará el sistema desde el punto de vista del usuario, o sea describe lo que hará y la interacción con los usuarios.

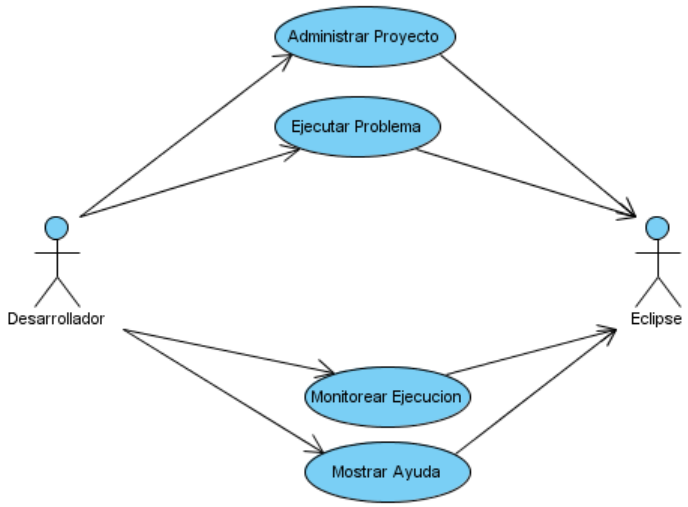

Figura 2.3: Diagrama de Casos de Uso del Sistema

2.5. Conclusiones

En este capítulo se identificaron los requerimientos funcionales y no funcionales, los actores y casos de usos que sirvieron de base para realizar el diseño del producto que se muestra a continuación. Se obtuvo la descripción de los casos de usos identificados y el diagrama que los representa. Se define además la estructura que poseerá el plugin y qué representa cada uno de los subproyectos.

Capítulo 3

Diseño del Sistema

Durante el proceso de diseño se sintetizan representaciones de la estructura de los datos, la estructura del sistema y los detalles de procedimiento ante los requisitos, tanto funcionales como no funcionales, todo ello refinado con los patrones arquitectónicos y de diseño, brindando así un producto con alta calidad. En este capítulo se describe la representación arquitectónica del sistema utilizada, así como en los patrones de diseño más importantes. Se muestran además los diagramas de clases y los de interacción, necesarios para llegar a una mejor comprensión del presente trabajo.

3.1. Estilo de Arquitectura

Un estilo de arquitectura de software describe un problema en particular y recurre del diseño, que surge en un contexto específico y presenta un esquema genérico y probado de su solución, en los cuales se define un vocabulario de tipos de componentes y conectores como también un conjunto de restricciones sobre cómo combinar estos. El estilo de arquitectura seleccionado para el desarrollo de este sistema se encuentra inmerso en los de llamada y retorno, es el Modelo Vista Controlador (MVC) [30][1]. Este estilo o patrón estructura el desarrollo de la aplicación en tres componentes distintos, separando así la lógica de control con la presentación. Para llegar a una mejor comprensión de la selección, ubicaremos antes el problema.

- Contexto: Plugins de Eclipse.

[1]MVC: Modelo-Vista-Controlador

- Problema: Es frecuente que se soliciten cambios en el plugin cuando se genere alguna nueva versión de la arquitectura, o de T-arenal. Los cambios al plugin deberían realizarse de una forma fácil y en el menor tiempo posible.

- Solución: Se separa el Sistema en tres componentes: Modelo, Vista, Controlador.

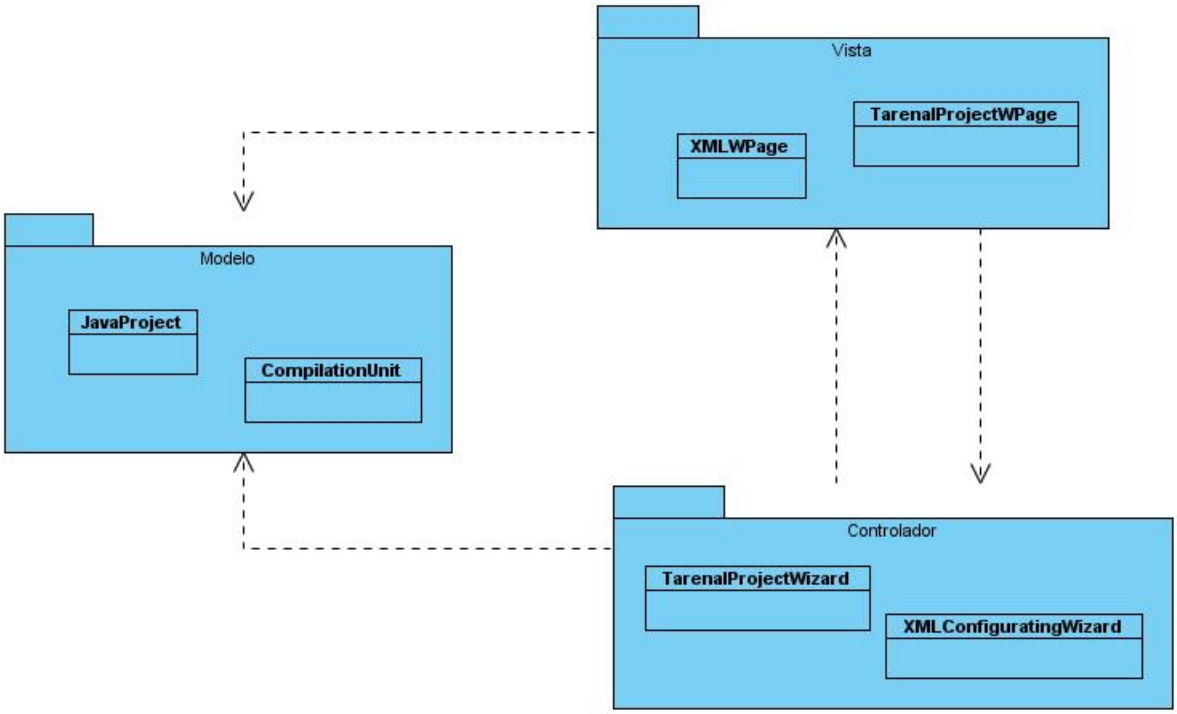

Figura 3.1: Modelo Vista Controlador

Modelo: Es la presentación específica de la información con la cual el plugin opera. La lógica de datos asegura la integridad de estos datos y permite derivar nuevos.
Vista: Despliega la información contenida en el modelo. Presenta el modelo en un formato adecuado para interactuar con el sistema y usualmente es un elemento de interfaz de usuario (entre ellas vistas y asistentes).
Controlador: Este responde a eventos o usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista. El flujo de control que sigue el patrón seleccionado se muestra en la figura 3.2:

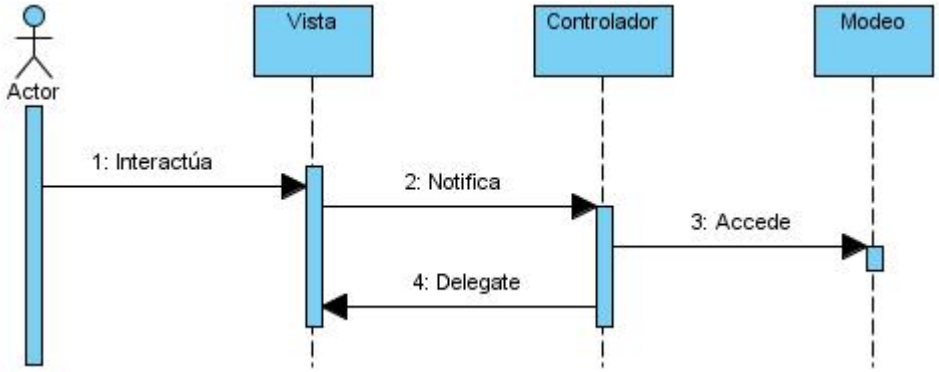

Figura 3.2: Flujo de Control del MVC

3.2. Patrones de Diseño

3.2.1. Patrón Singleton (Instancia Única)

PROBLEMA: en el subsistema tarenalDeveloperTools se necesita una única instancia de la Clase Server y que este tuviera un único punto de acceso, de manera que varias clases pudiesen utilizar el mismo objeto.
SOLUCIÓN: se diseña la clase *Server* con una referencia de ella misma y un método estático devolviendo el mismo objeto necesitado.

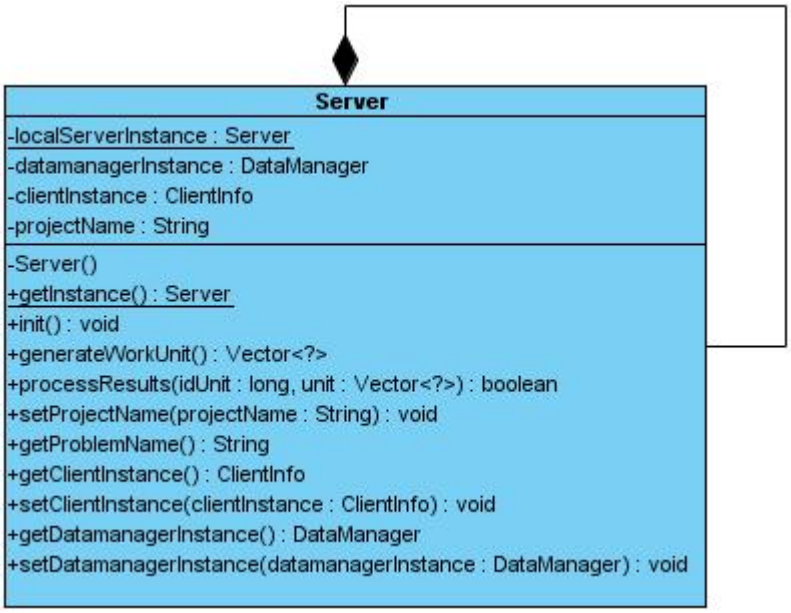

Figura 3.3: Ejemplo del patrón Singleton en tarenalDeveloperTools

3.3. Vista Lógica

En esta vista se describen las partes arquitectónicamente significativas del modelo de diseño, como son la descomposición en capas, paquetes y subsistemas. La descripción de la Vista Lógica del sistema se realiza por módulos para llegar a una mejor comprensión de las características del mismo. Se denomina módulo a cada uno de los subsistemas que lo componen. Se muestran por cada módulo los diagramas de paquetes que son arquitectónicamente significativos teniendo en cuenta su relación y composición. Se brindaran además los diagramas de clases del diseño de los paquetes que se consideran necesarios, acompañados con una breve descripción de las clases e interfaces que lo componen.

3.3.1. Subsistemas

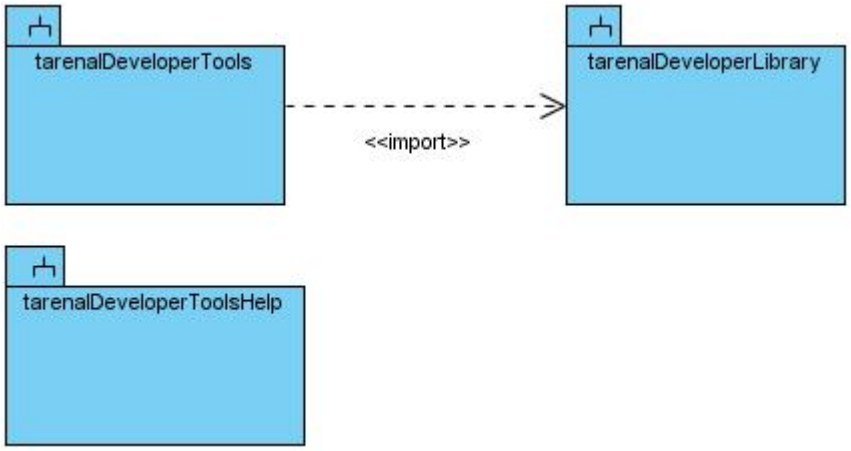

Figura 3.4: Vista General de los Subsistemas y sus relaciones.

Cada uno de los subsistemas representa una parte arquitectónicamente significativa dentro del sistema.

TARENALDEVELOPERTOOLS: encierra todo el negocio de nuestro sistema, así como nuevas interfaces, nuevos asistentes, nuevas vistas, y el proceso de ejecución en el Entorno Controlado de Desarrollo.
TARENALDEVELOPERTOOLSLIBRARY: está constituido medularmente por la librería de la Plataforma de Tareas Distribuidas.
TARENALDEVELOPERTOOLSHELP: posee toda la documentación sobre los dos primeros subsistemas y el modo de trabajo con ellos.

3.3.1.1. TarenalDeveloperTools

Sus principales funciones son las de brindar nuevos elementos para el desarrollo orientado a T-arenal como por ejemplo; los asistentes para la creación del nuevo tipo de proyecto y asociado a él se añade el que brinda la estructura del xml. Además de visualizar la nueva vista que contendrá los resultados de la ejecución controlada.

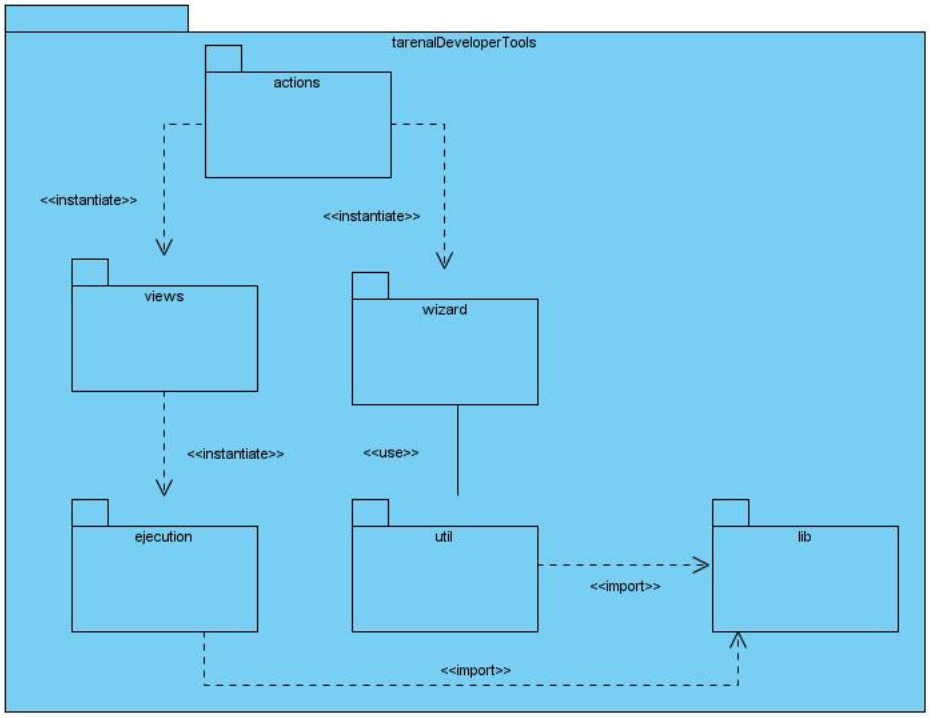

Figura 3.5: Paquetes relevantes para la arquitectura de tarenalDeveloperTools. Vista de Composición

Descripción de los paquetes de tarenalDeveloperTools

- Action: define los nuevos menús(acciones) y elementos que se añadirán posteriormente al Eclipse. Cada clase representa una acción diferente y van ha tener responsabilidades diferentes.
- Wizards: define los nuevos asistentes y con ellos los elementos de cada asistente.
- Views: define las nuevas vistas que se añadirán al Eclipse.
- Excecution: implementa las interfaces del Cliente y el Servidor estableciendo así cada elemento por separado, brindando todas las funcionalidades necesarias para ellos, teniendo en cuenta que el servidor posee una instancia única y del cliente habrán varias referencias abiertas.
- Lib: contiene la librería T-arenal.
- Utils: posee las interfaces y clases que facilitan el desarrollo de todos los elementos.

3.3.2. Diagramas de Clases del Diseño

Un diagrama de clases de diseño es un diagrama de estructura estática que describe gráficamente las especificaciones para las clases e interfaces de una aplicación y además contiene información como clases, asociaciones y atributos, interfaces con sus operaciones y constantes, métodos o funciones, navegabilidad, dependencias y multiplicidad en algunas de sus relaciones.

A continuación se muestran los diagramas de clases del diseño que dan solución al problema con una breve descripción de las clases que lo componen:

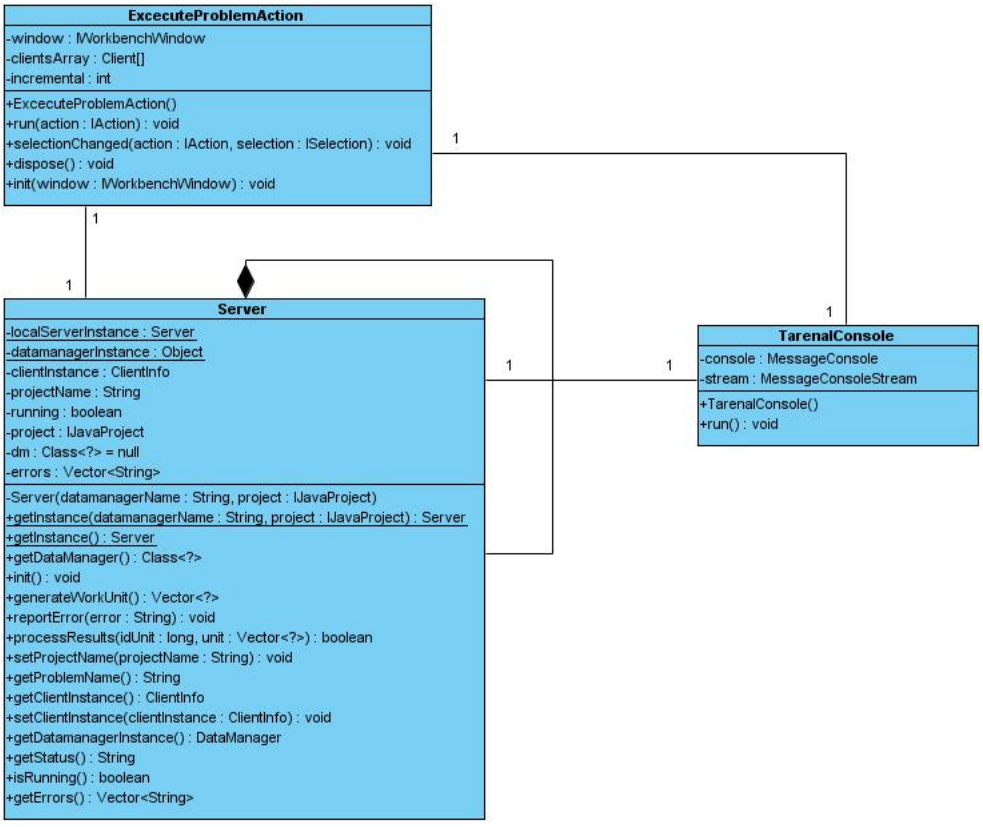

Figura 3.6: DCD del CUS Monitorear Ejecución

- ExecuteProblemAction: tiene la responsabilidad de iniciar la Acción ejecutar Problema e inicializar previamente los parámetros necesarios.

- ExcecutionView: tiene la responsabilidad de brindarle al desarrollador visualmente las devoluciones.
- Client: tiene la responsabilidad del procesamiento de los datos.

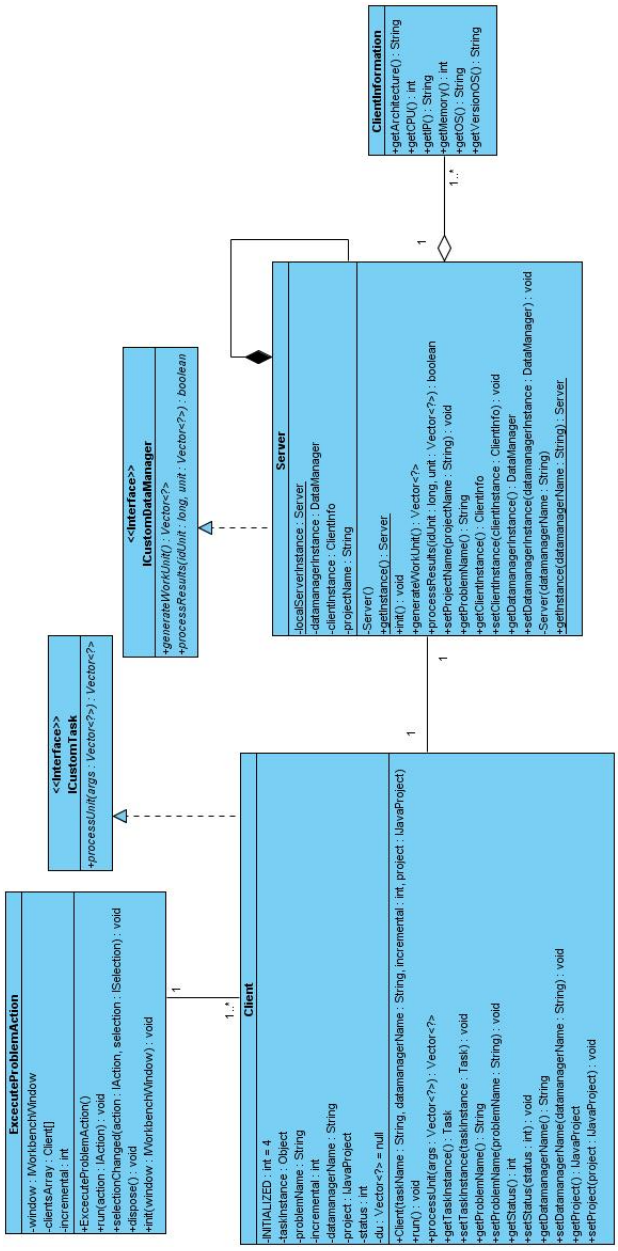

Figura 3.7: DCD del CUS Ejecutar Problema

- ICustomTask: interfaz que determina el procesamiento de datos, por lo cual la clase que la implementa

es Client.

- ICustomDatamanager: interfaz que determina el desarrollo de los elementos, por lo cual la clase que la implementa es Server.

- ClientInformation: define la Información de los Clientes, recogiendo las propiedades de CPU para su conformación.

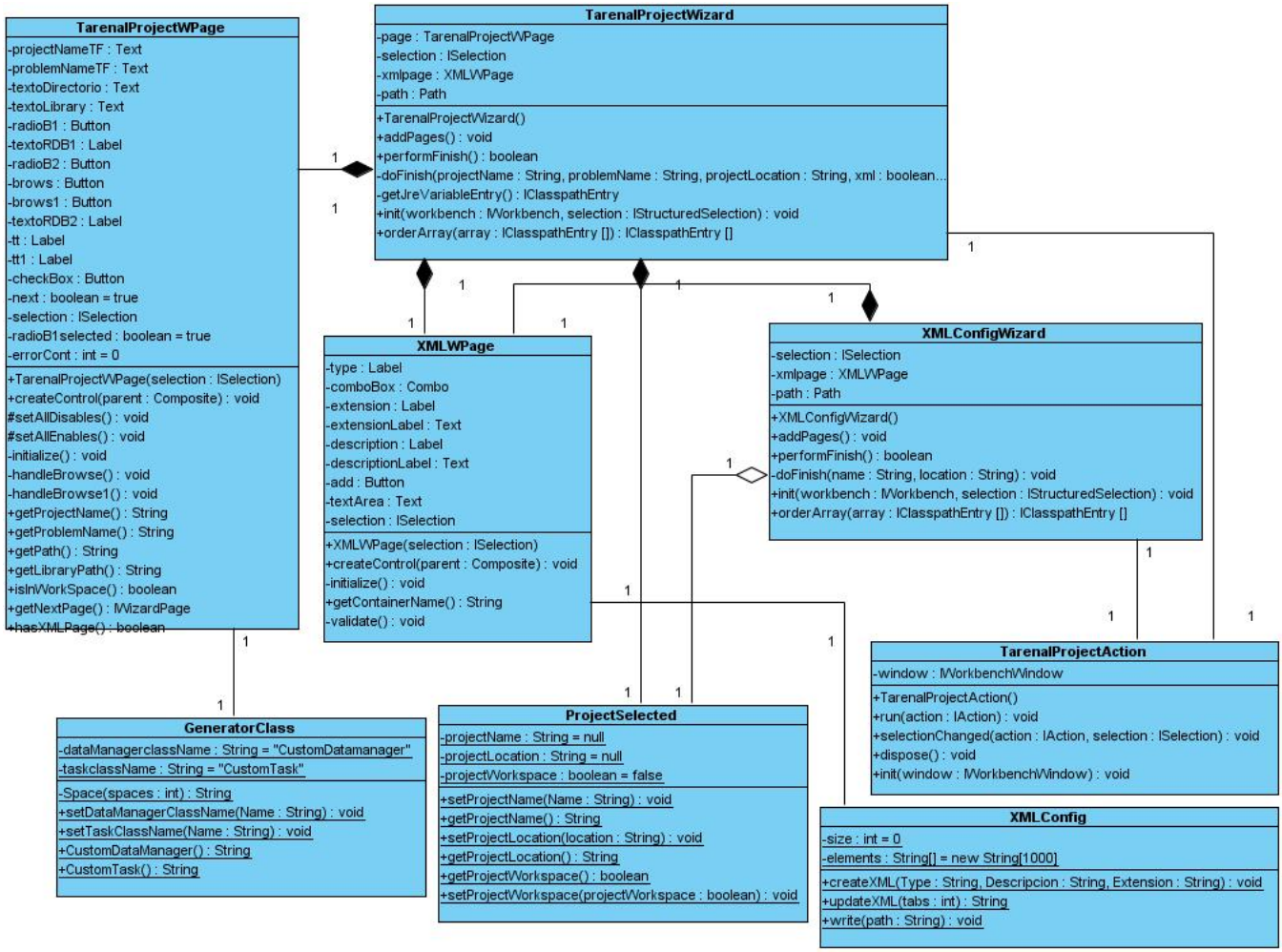

Figura 3.8: DCD del CUS Administrar Proyecto

- TarenalProjectWizard: se encarga de manejar el nuevo asistente para crear nuevos Proyectos T-arenal.

- XMLConfigWizard: se encarga de manejar el nuevo asistente para crear nuevos XML de configuración para los proyectos T-arenal.
- TarenalProjectWPage: posee los componentes necesarios para la creación de los proyectos.
- XMLWPage: posee los componentes necesarios para la creación del XML y la adición de elementos.
- GeneratorClases: tiene la responsabilidad de generar las nuevas clases dentro de los proyectos, aquellas que heredaran de DataManager y Task respectivamente.
- ProjectSelected: se encarga de obtener el proyecto T-arenal seleccionado en un momento dado.
- TarenalProjectAction: define la acción responsable de inicializar todos los elementos para cargar el nuevo asistente.
- XMLConfig: tiene la responsabilidad de generar los xml y adicionarle elementos.

3.3.3. Diagramas de Interacción

Los diagramas de interacción describen secuencias de intercambios de mensajes entre los roles que implementan el comportamiento de un sistema. Son utilizados para el modelado de los aspectos dinámico de un sistema, proporcionando una vista integral de su comportamiento. Esto conlleva modelar instancias concretas, componentes y nodos junto con los mensajes enviados entre ellos que representan su interacción.

Existen dos tipos de estos diagramas ambos basados en la misma información pero cada uno enfatizando un aspecto particular, ellos son:

- Diagramas de Secuencia
- Diagramas de Colaboración

Un diagrama de secuencia muestra la forma en que los objetos se comunican entre sí al transcurrir el tiempo y destaca la muestra de una secuencia ordenada en los mensajes intercambiados entre los objetos. Los diagramas de colaboración destaca la organización estructural de los objetos que envían y reciben mensajes.

Se muestran los principales diagramas de Interacción:

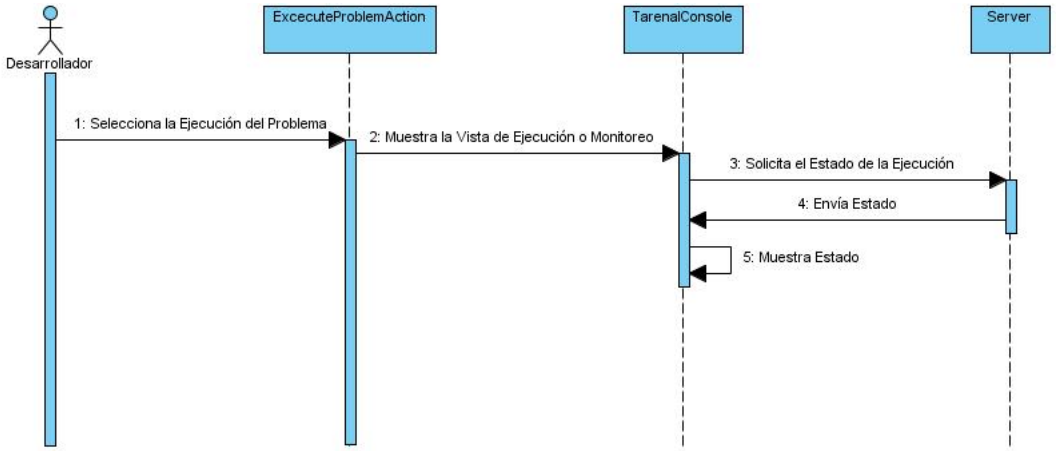

Figura 3.9: Diagrama de Secuencia del CUS Monitorear Ejecución

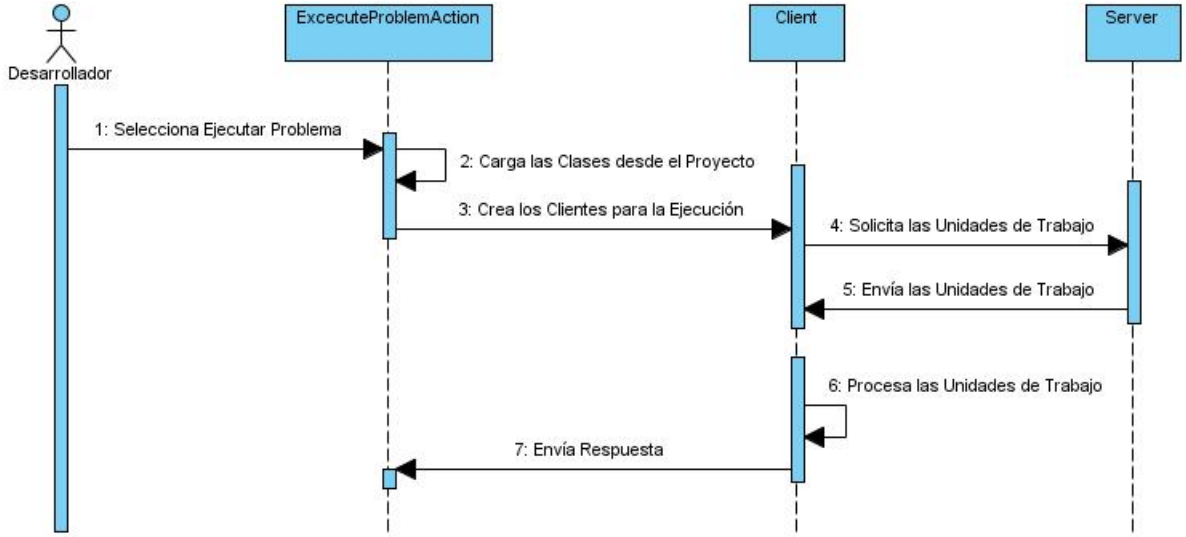

Figura 3.10: Diagrama de Secuencia del CUS Ejecutar Problema

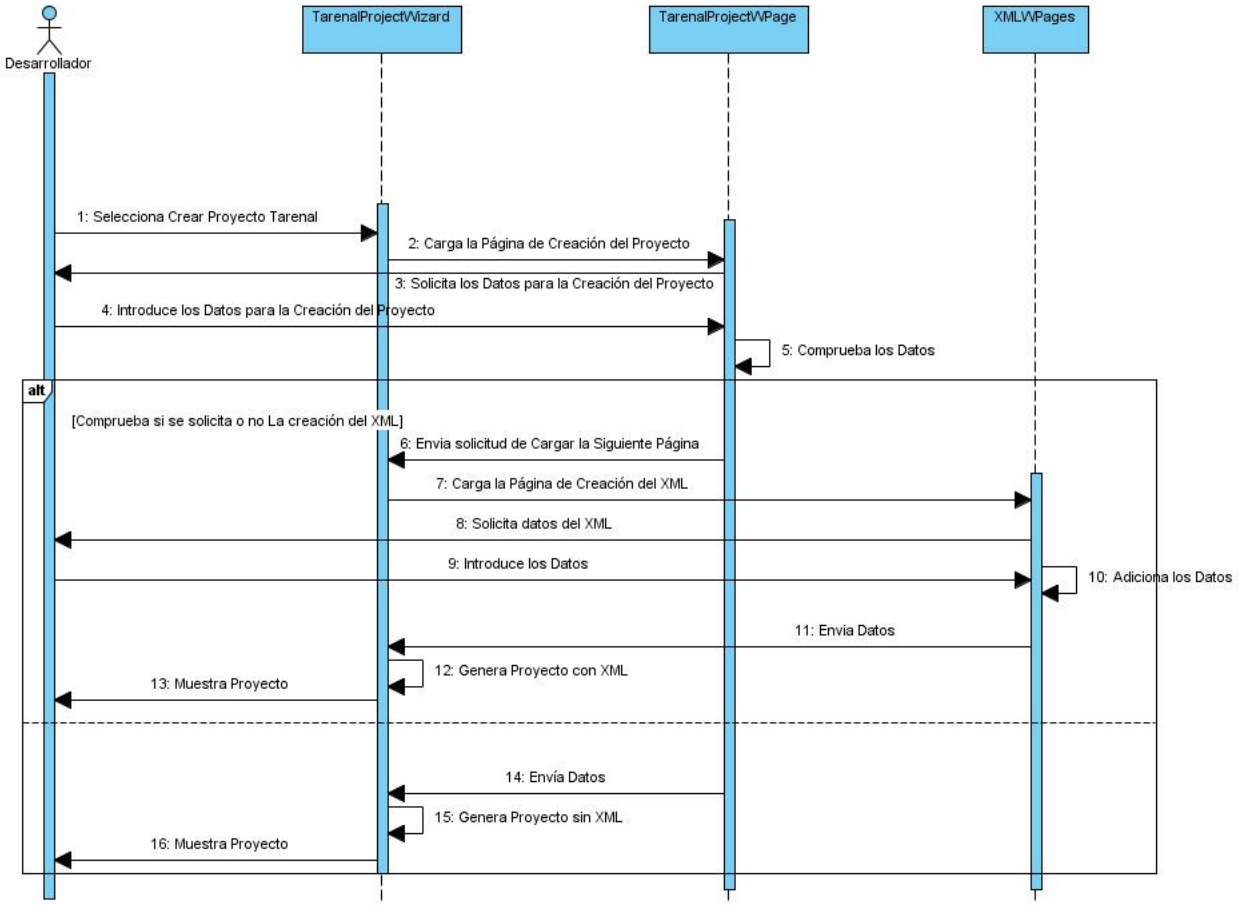

Figura 3.11: Diagrama de Secuencia del CUS Administrar Proyecto T-arenal

3.4. Vista de Despliegue

Esta vista constituye la visión física del sistema, donde se describe este como un conjunto de nodos y sus conexiones, donde se establece la comunicación entren ellos a través de protocolos o enlaces.

3.4.1. Diagrama de Despliegue del Sistema

En el diagrama de despliegue se describe la arquitectura física del sistema durante la ejecución, en términos de procesadores, dispositivos y componentes de software. Describen la topología del sistema, la estructura de los elementos de hardware y software que ejecuta cada uno de ellos.

Figura 3.12: Diagrama de Despliegue

Descripción de los nodos

El nodo existente es la Computadora Personal en la cual deberá poseer la Plataforma Eclipse para la ejecución del plugin.

3.5. Conclusiones

Como resultado de este capítulo se obtuvo la arquitectura del sistema sustentada en el estilo Modelo Vista Controlador. Se elaboró además el diseño del sistema. Desde la estructura de paquetes y subsistemas hasta los diagramas de clases que representan el paquete, mostrando en este lo más significativo. También se muestra la vista de datos con una breve descripción de cada una de las entidades que la integran, así como la distribución física del sistema.

Capítulo 4

Implementación y Pruebas

En este capítulo se describe la implementación del sistema en términos de componentes y la forma en que estos componentes serán desplegados. Se ilustrarán los principales resultados del desarrollo y los casos de pruebas para los principales casos de uso.

4.1. Diagrama de Componente

Modelan la vista estática de un sistema. Muestra la organización y las dependencias entre un conjunto de componentes. La figura (4.1) se muestra los componentes relevantes del sistema y las interfaces que estos implementan y acceden para mostrar su interacción. Existen tres componentes principales; tarenalDeveloperTools, tarenalDeveloperToolsHelp y tarenalDeveloperToolsLibrary. Cada uno de estos componentes estará desplegado en el lugar potencial que se designe como se muestra en la sección (4.2).

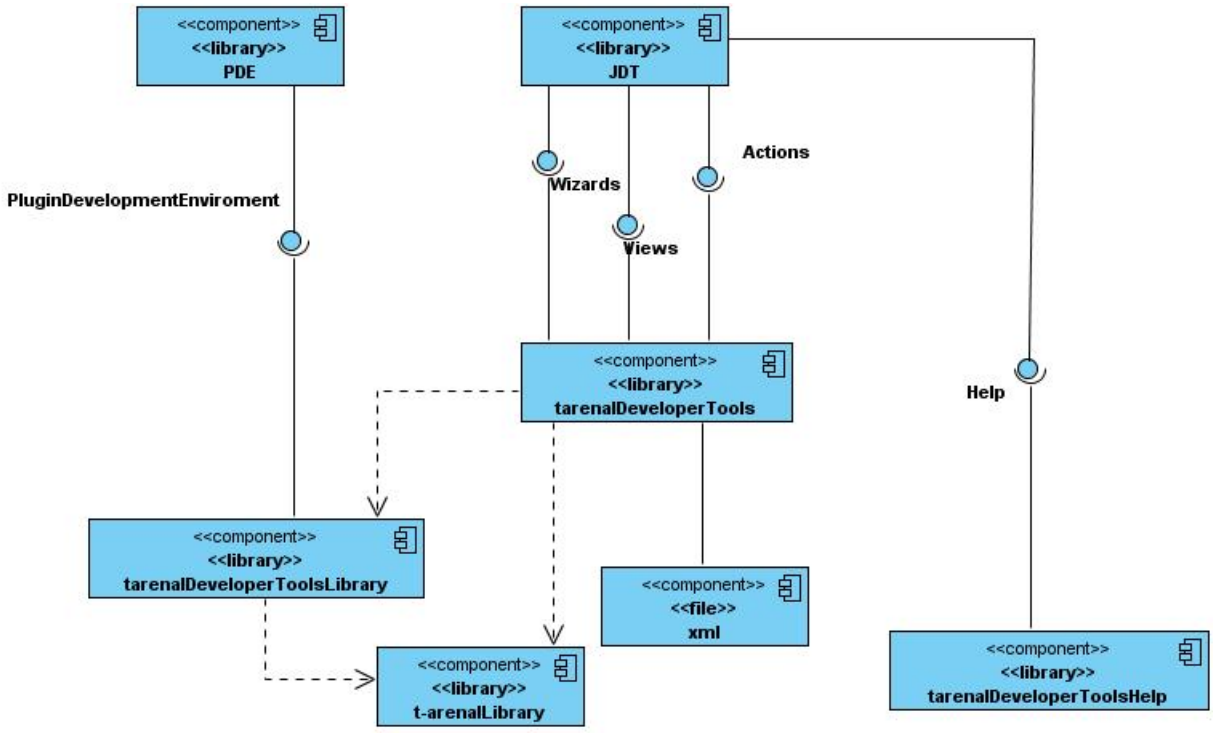

Figura 4.1: Diagrama de Componentes del Sistema

La mayoría de los componentes ilustrados son bibliotecas, ya que estas son cargadas por Eclipse para obtener nuevas funciones. La biblioteca tarenalDeveloperTools implementa las interfaces que provee JDT que son: Actions, Views, Wizards desarrollando nuevos asistentes, vistas y acciones. La biblioteca tarenalDeveloperLibrary encapsula la biblioteca t-arenalLibrary para su utilización dentro del sistema. La biblioteca tarenalDeveloperHelp implementa la interfaz Help brindada por JDT[1] para la construcción de la ayuda.

4.2. Diagrama de Despliegue de los Componentes

El diagrama de despliegue de componentes se indica la situación física de los componentes lógicos desarrollados. Es decir, situar el software en el hardware que lo contiene.

[1]JDT: Java Development Tooling

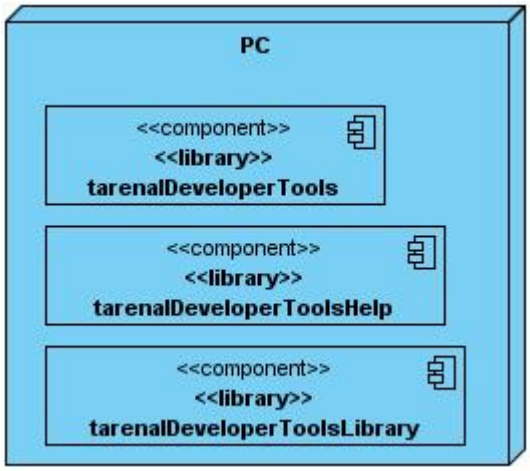

Figura 4.2: Diagrama de Despliegue de los Componentes

4.3. Pantallas de la Aplicación y Resultados

La aplicación tiene como resultado principal la integración con Eclipse y la solución a los requerimientos anteriormente planteados. Por ello, se muestran las pantallas de la aplicación encargadas de dar solución a cada uno de los problemas.

La figura 4.3 ilustra las nuevas acciones que se añaden a la barra de herramienta y el nuevo sistema de menú que incorpora a Eclipse.

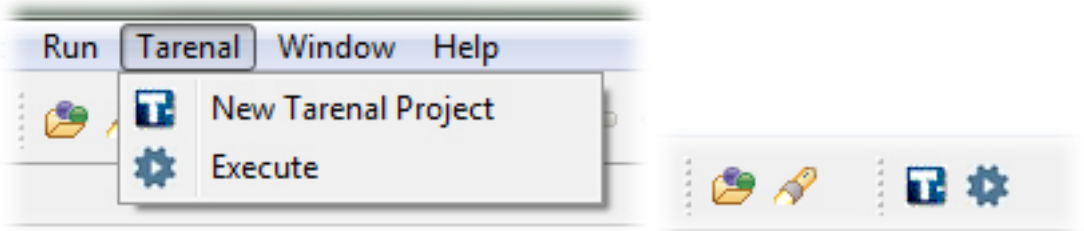

Figura 4.3: Nuevas Acciones y elementos en la barra de herramientas

La figura 4.4 muestra la IDE Eclipse con el nuevo menú incorporado, uno de los nuevos asistentes, las nuevas acciones en la barra de herramienta y una de las nuevas vistas donde se monitorea el proceso de ejeción de las tareas.

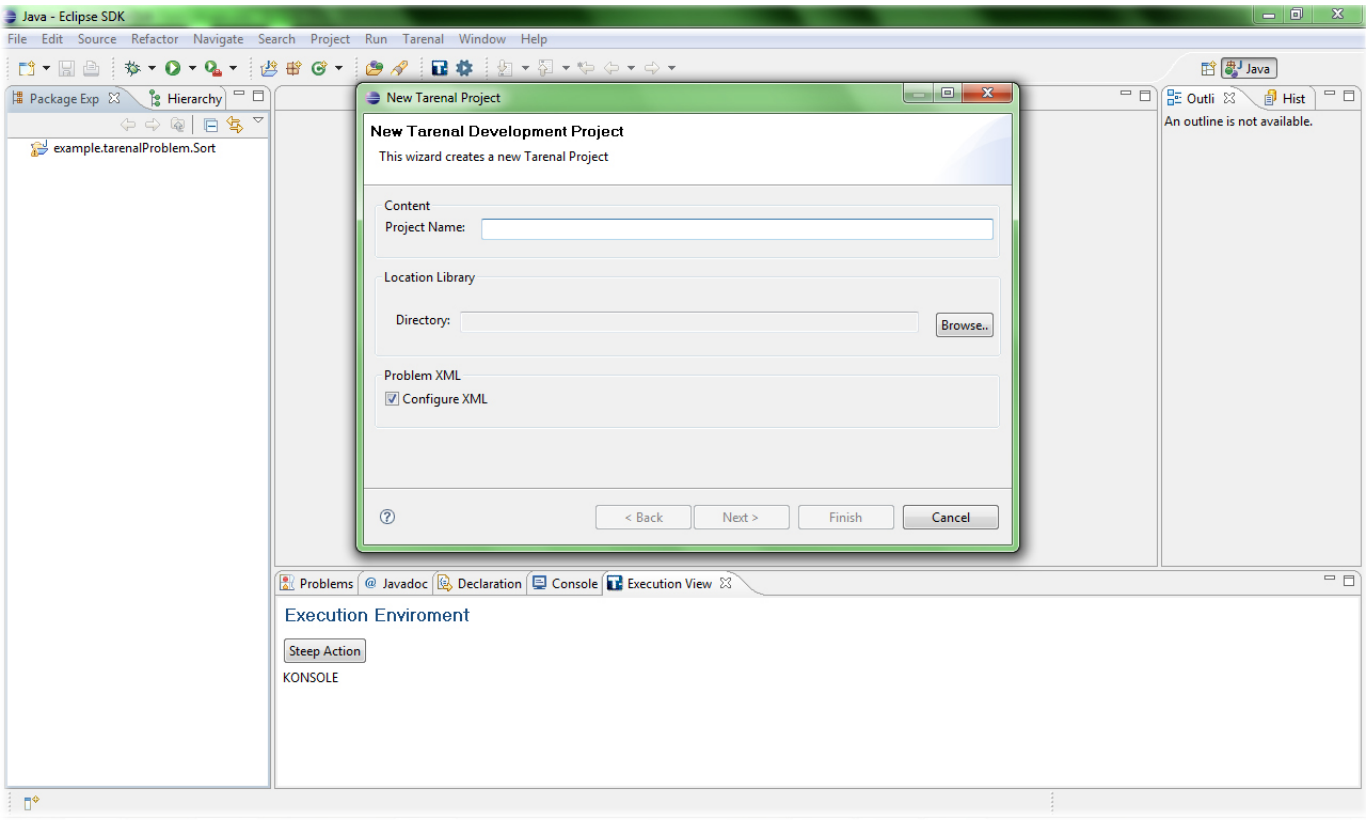

Figura 4.4: Plugin TarenalDeveloperTools en Eclipse 3.3

4.4. Modelos de Prueba

El flujo de trabajo de pruebas le presta servicios a los demás flujos de trabajo. su principal objetivo es la búsqueda y documentación de errores, validando así el cumplimiento de los requerimientos, el desempeño y dando una indicación de calidad.

Las pruebas son aquellas acciones que se llevan acabo sobre un software para verificar o revelar la calidad del producto de software, identificando en estas posibles fallos de implementación calidad o usabilidad. Entre ellas existen las pruebas de caja blanca y las pruebas de caja negra.

Las pruebas de caja negra son aquellas que se realizan sobre la interfaz de software, teniendo en cuenta parámetros como las entradas y estudiando las salidas a ver si son las esperadas o no.

4.4.1. Casos de Pruebas

Los escenarios principales de los casos de uso críticos fueron probados para detectar no conformidades. A continuación se muestran algunas de las pruebas realizadas:

id del Escenario	Escenario	Variable	Respuesta del Sistema	Resultados de la Prueba
EC 1.1	Crear Proyecto	- tarenalTest - archivo diferente a t-arenalLibrary	El sistema verifica los datos, que estén semánticamente correctos y que la dirección de la bibleoteca sea válido.	El sistema pide que entre correctamente la dirección de la bibleoteca, y no permite finalización del asistente.
EC 1.3	Generar XML de Configu-ración	- GroupFile - “ ” - grupo de archivos	El sistema verifica los datos a añadir en el XML si son correctos.	El sistema solicita que se entre la extensión de los ficheros a cargar y no permite la finalización del asistente

Cuadro 4.1: Casos de Pruebas

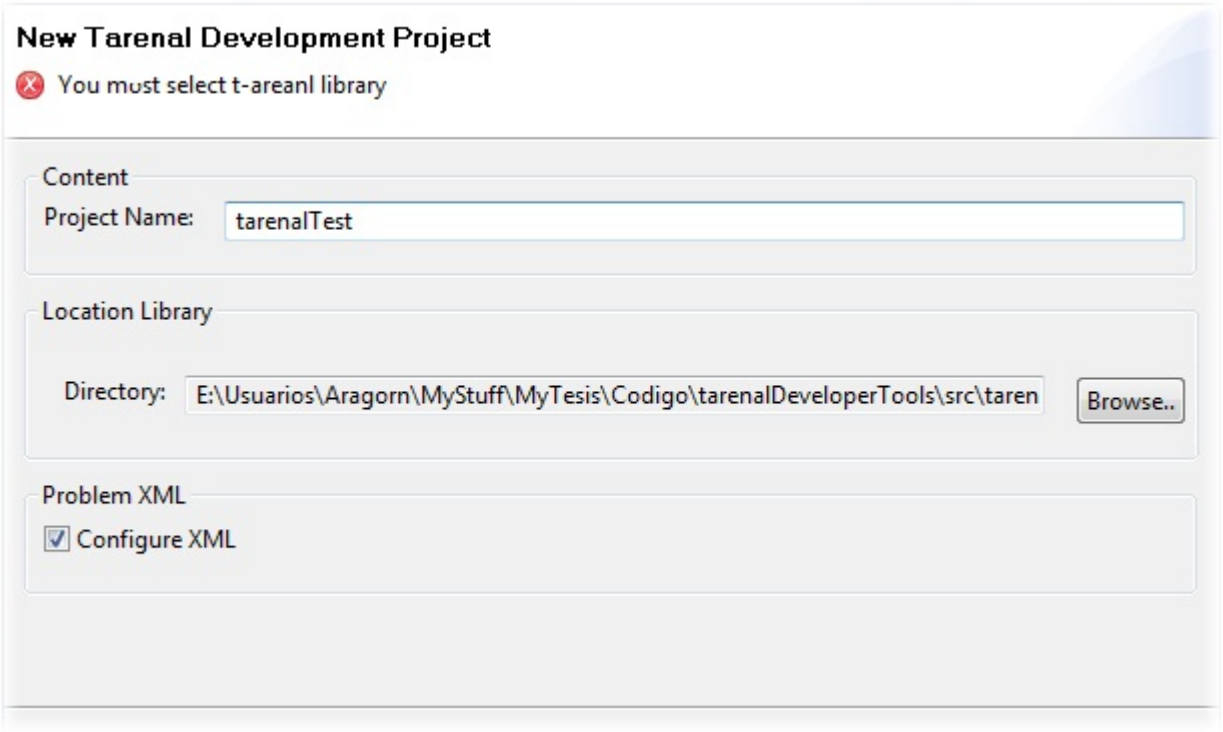

Figura 4.5: Resultados de la Prueba del Escenario EC 1.1

Nuevo asistente que se incorpora a Eclipse donde se muestra un mensaje de error ya que la dirección donde se ubica t-arenalLibrary.jar es incorrecta.

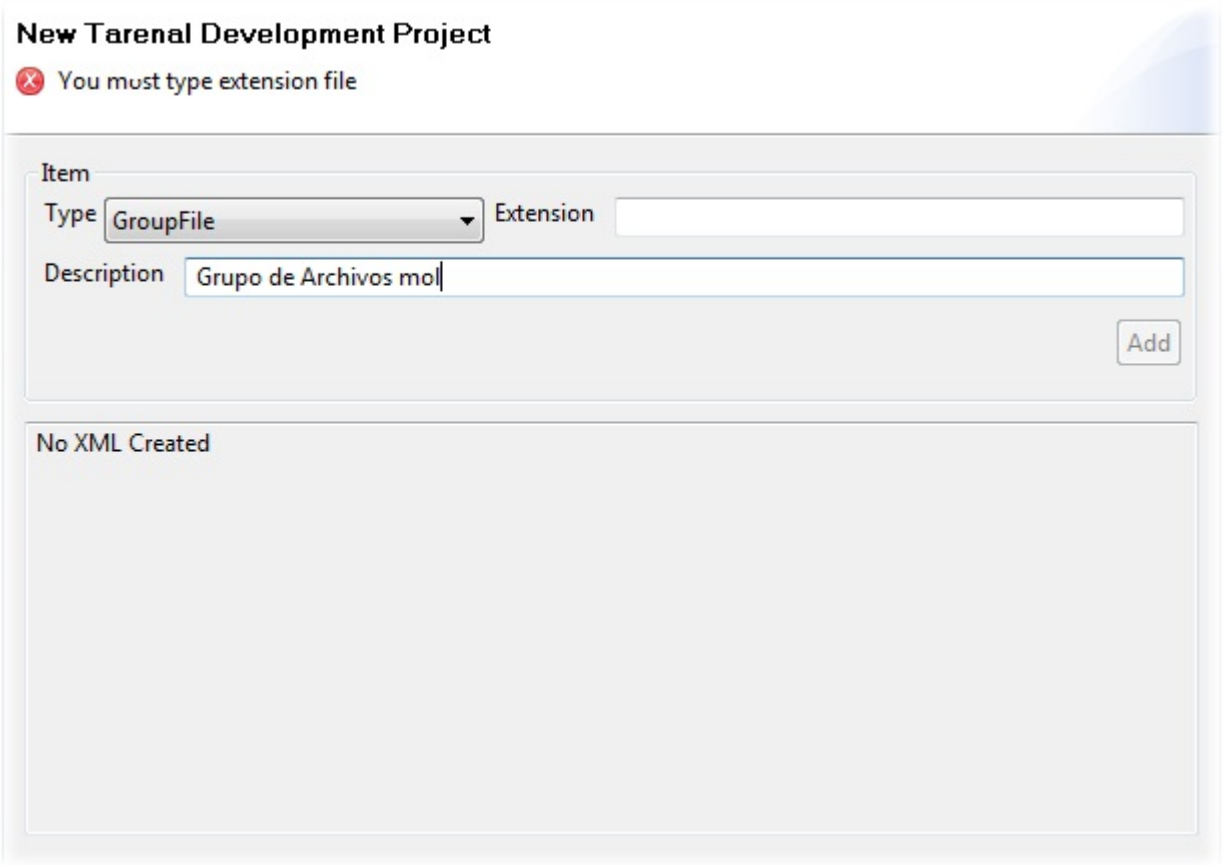

Figura 4.6: Resultados de la prueba del Escenario EC 1.2

Nuevo asistente que se incorpora a Eclipse donde muestra un mensaje de error debido a una entrada de datos incorrecta, ya que ningun campo de los elementos a adicionar en el XML de configuración puede estar en blanco.

4.5. Conclusiones

Como resultado de este capítulo se obtuvo el Diagrama de Componentes del sistema mostrando la dependencia entre los componentes, se obtuvo además el Diagrama de Despliegue de los Componentes que nos ilustra donde se ubicarán físicamente nuestros componentes. Se ilustran los resultados del sistema con pantallas a los escenarios más significativos y se muestran una porción de las pruebas de caja negra realizadas a los escenarios críticos.

Conclusiones

1. Se definió como IDE para la integración y desarrollo del plugin a Eclipse por las facilidades que brinda para el desarrollo de los mismos.
2. Se identificaron los requisitos funcionales y no funcionales.
3. Se realizó el análisis del plugin.
4. Se realizó el diseño del plugin, separándolos en 3 módulos, Tools, Help y Library.
5. Se implementaron los módulos diseñados.
6. Se validaron las pruebas exploratorias al sistema, utilizando el método de caja negra.

Recomendaciones

1. Añadir la funcionalidad de envío de la tarea al servidor.
2. Añadir al plugin un sistema de ejecución paso a paso.
3. Añadir un nuevo asistente que solicite los ficheros necesarios para la ejecución de la tarea.
4. Mantener actualizado el plugin de acuerdo a las versiones del servidor.
5. Utilizar el plugin TarenalDeveloperTools para el desarrollo de aplicaciones T-arenal.

Referencias Bibliográficas

[1] Cesar Raul Garcias Jacas DMMT. T-arenal v2.0: Desarrollo del back - end; 2009.

[2] Mendoza LLA. Sistema de cómputo distribuido aplicado a la Bioinformática; 2008.

[3] Henri Casanova AL, Quinson M. SimGrid;. Available from: `http://simgrid.gforge.inria.fr` [cited 2010 Feb 15].

[4] Stumpert DM. gEclipse: Access the power of the Grid;. Available from: `http://www.geclipse.eu` [cited 2010 Feb 15].

[5] Entornos de Desarrollo Integrado para Java;. Available from: `http://luauf.com` [updated 2008 May 13; cited 2009 Nov 21].

[6] Microsystem S. Netbeans;. Available from: `http://www.netbeans.org` [cited 2010 Feb 15].

[7] Erich Ganma B Kent. Contributing to Eclipse. Eclipse Wiki; 2003.

[8] Beaton DW. Eclipse Platform Technical Overview;. Available from: `http://www.eclipse.org/articles/Whitepaper-Platform-3.1/eclipse-platform-whitepaper.html` [cited 2010 May 19].

[9] S Shavor JD. The Java Developer's Guide to Eclipse. Addison-Wesley; 1995.

[10] Eric Clayberg DR. Eclipse Plugins (3rd Edition). Addison Wesley; 2009.

[11] Fundation E. Eclipse Architecture;. Available from: `http://www.eclipse.org/articles/Article-Plug-in-architecture/plugin_architecture.html` [cited 2009 Feb 15].

[12] Estberg D. How the Minimum Set of Platform Plugins Are Related. Eclipse Wiki; 2004.

[13] Creasey T. Model-View-Controller (MVC) Architecture; 2002. Available from: http://www.eclipse.org/articles/Article-Preferences/preferences.htm [cited 2002 Aug].

[14] Barttlet N. Comparison of Eclipse Extensions and OSGi Services; 2007. Available from: http://www.eclipsezone.com/articles/extensions-vs-services [cited 2009 Feb].

[15] Company SG. Java Code Generator;. Available from: http://www.sahits.ch/project/javacodegen [cited 2010 Ene 21].

[16] Whebber DS. Java Code Generator;. Available from: http://www.eclipse.org/tptp/platform [cited 2010 Ene 19].

[17] James Rumbaugh IJ, Booch G. El Proceso Unificado de Desarrollo de Software. Addison Wesley; 1999.

[18] Extreme Programming: A gentle introduction;. Available from: http://www.extremeprogramming.org/ [updated 2006 Feb 17; cited 2009 Dic 10].

[19] Eclipse Process Framework Project;. Available from: http://www.eclipse.org/epf/ [cited 2009 Nov 20].

[20] Unified Modeling Language;. Available from: http://www.uml.org/ [cited 2010 Ene 20].

[21] James Rumbaugh IJ, Booch G. El Lenguaje Unificado de Modelado. Manual de Referencia. Segunda Edición. Addison Wesley; 2007.

[22] Armstrong E. HotSpot: A new breed of virtual machine, Javaworld;. Available from: http://www.javaworld.com/jw-03-1998/jw-03-hotspot.html [cited 2010 Ene 15].

[23] J M Bull LP L A Smith, Freeman R. Benchmarking Java against C and Fortran for scientific applications. Journal of the ACM. 2001;.

[24] Oaks S. Java Security (2nd Edition). O'Reilly Media, Inc.; 2001.

[25] Java SE Security; 2010 Feb 10. Available from: http://java.sun.com/security/.

[26] Zukowski J. Programación Java 2 J2SE 1.4. vol. 1. SYBEX, Inc.; 2003.

[27] Klinger D. Creating JFaces Wizards; 2002. Available from: http://www.eclipse.org/articles/article.php?file=Article-JFaceWizards/index.html [cited 2009 Feb].

[28] Fundation E. Eclipse Java development tools (JDT) Overview; 2008. Available from: http://www.eclipse.org/jdt/overview.php [cited 2010 May 19].

[29] Fundation E. Plug-in Development Environment Overview; 2008. Available from: http://help.eclipse.org/help32/index.jsp?topic=/org.eclipse.pde.doc.user/guide/intro/pde_overview.htm [cited 2010 May 19].

[30] Deacon J. Model-View-Controller (MVC) Architecture; 2002. Available from: http://www.jdl.co.uk/briefings/MVC.pdf [cited 2009 May].

Apéndice A

Descripción de Casos de Usos Críticos

Caso de uso Ejecutar Problema

Nombre del CU	**Administrar Proyecto T-arenal**
Actor	Desarrollador (inicia)
Propósito	Crear proyecto T-arenal.
Resumen	El caso de uso inicia con la necesidad de crear un proyecto T-arenal, en ello se crea un proyecto T-arenal, se generan los paquetes y clases además de importarse las librerías nativas y la del sistema T-arenal. Con ello se generará o no el XML de configuración.
Referencias	RF 2
Curso Normal de Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El desarrollador selecciona la opción de crear un nuevo proyecto Tarenal.	2. El sistema verifica que el nombre del proyecto y el nombre del problema sea válido.
	3. El sistema crea el proyecto, genera el paquete tarenal.dev donde van las clases DataManager y Task que se generarán posteriormente. Luego se importan las librerías nativas y la del sistema T-arenal.

4. El desarrollador selecciona la opción de generar XML de configuración.	5. El sistema solicita el tipo de elemento que se insertará en el XML además de la descripción y la extensión.
6. El desarrollador selecciona el tipo, inserta la descripción y la extensión	7. El sistema verifica si la descripción y la extensión no posee caracteres no reconocidos.
	8. El sistema genera el XML de configuración.
	9. Finaliza el caso de Uso.
Cursos Alternativos **CA1**	
	2.1. Si el nombre del proyecto o el nombre del problema no son válidos se lanza un mensaje de error.
Cursos Alternativos **CA2**	
4.1 Si el desarrollador no selecciona la opción de generar XML de configuración.	4.2 Pasa al paso 9 del Flujo Normal de Eventos.
Cursos Alternativos CA3	
	7.1 Si la descripción y la extensión poseen caracteres inválidos se lanza un mensaje de error.
Prioridad	**Crítico**

Caso de uso Ejecutar Problema

Nombre del CU	**Ejecutar Problema**
Actor	Desarrollador (inicia)
Propósito	Permitir ejecutar un problema en el sistema.
Resumen	El caso de uso inicia cuando el desarrollador selecciona la opción de ejecutar un problema en el sistema. Luego extrae los datos necesarios para el funcionamiento de la ejecución creada y finalmente le indica al sistema que ya puede visualizarla.

Referencias	RF 2
Curso Normal de Eventos	
Acciones del Actor	**Respuesta del Sistema**
1. El desarrollador hace una petición para ejecutar un problema.	2. El sistema verifica que el proyecto seleccionado por el usuario sea de tipo T-arenal.
	3. El sistema extrae los datos necesarios para realizar la ejecución en el ECD[1].
	4 El sistema adiciona la Ejecución a la lista de ejecuciones, manda a Ejecutar el Algoritmo con los datos recogidos en el ECD y Finaliza el Caso de Uso.
Cursos Alternativos **CA1**	
	2.1. Si el proyecto no es de tipo T-arenal, se lanza un mensaje de error y finaliza el caso de uso.
Prioridad	**Crítico**

Caso de uso Monitorear Ejecución

Nombre del CU	**Monitorear Ejecución**
Actor	Desarrollador (inicia)
Propósito	Mostrar el estado de la Ejecución, y como va sucediendo.
Resumen	El caso de uso inicia cuando el desarrollador ejecuta un problema, ya que esta muestra lo que esta sucediendo con el problema.
Referencias	RF 3.1, RF 3.2
Curso Normal de Eventos	
Acciones del Actor	**Respuesta del Sistema**

[1]ECD: Entorno Controlado de Desarrollo

1. El desarrollador hace una petición para ejecutar un problema (*ver Caso de Uso: Ejecutar Problema*).	2. El sistema toma por cada nodo(thread) los datos que se enviaron a ejecutar.
	3. Muestra los datos de los nodos.
	4. El sistema brinda los estados de cada nodo, de como se comportan y Finaliza el caso de Uso.
Cursos Alternativos **CA1**	
	2.1. Si no se obtiene ninguna información de los nodos, lanzar mensaje de error y pasa al paso 4 del Flujo Normal de Eventos.
Prioridad	**Crítico**