VISTA GRÁFICA PARA LA ADMINISTRACIÓN DE PROCESOS DE DESARROLLO PERSONALIZADOS BASADOS EN AGENTES EN LA HERRAMIENTA *agentTool Process Editor (APE)*: SOPORTANDO LA TÉCNICA *EARNED VALUE ANALYSIS.*

ING. ANDREA PATRICIA GARCIA PRADA

MAESTRÍA EN SOFTWARE LIBRE
FACULTAD DE INGENIERÍA DE SISTEMAS
UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
EN CONVENIO CON LA UNIVERSITAT OBERTA DE CATALUNYA
BUCARAMANGA, 2011

TÍTULO

VISTA GRÁFICA PARA LA ADMINISTRACIÓN DE PROCESOS DE DESARROLLO PERSONALIZADOS BASADOS EN AGENTES EN LA HERRAMIENTA *agentTool Process Editor (APE)*: SOPORTANDO LA TÉCNICA *EARNED VALUE ANALYSIS.*

PROYECTO DE INVESTIGACIÓN PARA OPTAR AL TÍTULO DE:
MAGÍSTER EN SOFTWARE LIBRE

AUTOR
ING. ANDREA PATRICIA GARCIA PRADA

DIRECTOR
MSc. JUAN CARLOS GARCÍA OJEDA

MAESTRÍA EN SOFTWARE LIBRE
FACULTAD DE INGENIERÍA DE SISTEMAS
UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
EN CONVENIO CON LA UNIVERSITAT OBERTA DE CATALUNYA
BUCARAMANGA, 2011

AGRADECIMIENTOS

De manera muy sincera expreso mi agradecimiento a la Universidad Autónoma de Bucaramanga (UNAB) y a la Universitat Oberta de Catalunya (UCO), por su modelo educativo y tecnológico en el campo de la educación virtual, el cual me ha permitido ampliar y fortalecer mi formación como profesional, en el área de la informática y las tecnologías de la información y comunicación.

A Juan Carlos García Ojeda MSc. Ing., director de proyecto, por brindarme la oportunidad de trabajar en el desarrollo de esta propuesta, la cual me ha permitido poner en práctica los conocimientos adquiridos durante este proceso de formación y sobre todo por sus valiosos aportes y sugerencias durante su desarrollo.

TABLA DE CONTENIDO

LISTADO DE TABLAS

LISTADO DE FIGURAS

RESUMEN

TITULO: VISTA GRÁFICA PARA LA ADMINISTRACIÓN DE PROCESOS DE DESARROLLO PERSONALIZADOS BASADOS EN AGENTES EN LA HERRAMIENTA *agentTool Process Editor (APE)*: SOPORTANDO LA TÉCNICA *EARNED VALUE ANALYSIS.*

AUTOR: ANDREA PATRICIA GARCÍA PRADA.

PALABRAS CLAVE: agentTool Process Editor, Eclipse, Java, JFreeChart, plug-ins, Técnica del Earned Value Analysis.

DESCRIPCIÓN:

El siguiente documento describe el proceso de desarrollo de una nueva funcionalidad en la herramienta agentTool Process Editor (APE). Esta nueva funcionalidad consiste en la creación de una vista gráfica que facilitará a los Directores de Proyecto sus labores de control y seguimiento durante el desarrollo de Sistemas Multiagente (SMA), mediante la Técnica del Earned Value Analysis (EVA). Esta funcionalidad representa gráficamente la información relacionada con el alcance, tiempo y costo del proyecto con el objetivo de evaluar su desempeño en cualquier fase o período de evaluación. La vista gráfica desarrollada permitirá a los Directores de Proyecto:

- Determinar el costo, el cronograma y el trabajo realizado de las actividades planificadas.
- Controlar las actividades y tareas a realizar, así como los productos a generar en función de las características propias del proyecto.
- Identificar de manera oportuna, los posibles cambios o variaciones que pueden afectar el normal desarrollo de un Sistema Multiagente (SMA).
- Implementar medidas correctivas o preventivas de manera oportuna.
- Identificar el avance del proyecto en cualquier fase de su desarrollo.
- Medir y analizar los resultados.
- Ajustar el proyecto con el presupuesto y tiempo establecido inicialmente.
- Cumplir las especificaciones y necesidades del cliente.

SUMMARY

TITLE: A GRAPHICAL VIEW FOR ASSESSING CUSTOMIZED AGENT-ORIENTED PROCESS ON THE agentTool Process Editor (APE): SUPPORTING THE EARNED VALUE ANALYSIS TECHNIQUE.

AUTHOR: ANDREA PATRICIA GARCÍA PRADA

KEY WORDS: agentTool Process Editor, Eclipse, Java, JFreeChart, plug-ins, Earned Value Analysis Technique.

DESCRIPTION:

The following document describes the development of a new feature for agentTool Process Editor (APE). This new feature is a graphical view where Project Managers can control and check activities during the development of Multiagent Systems (MAS) by resorting to the Earned Value Analysis Technique. This feature graphically represents the scope, time and cost of the project with the aim of evaluating the performance of theproject at any time. This feature will allow Project Managers to:

- Determinate the cost, the schedule and the work performed of the planning activities.
- Control the activities and things to do, as well as the products regarding the requirements of the project.
- Identify changes (in timely fashion) or variations than can affect the normal development of a Multiagent System (SMA)
- Implement corrective or preventive actions in an acceptable amount of time.
- Identify the project's progress at any phase of development.
- Measure and analyze the results.
- Adjust the project in time and budget.
- Satisfy the specifications and needs of the clients.

INTRODUCCIÓN

La herramienta **agentTool Process Editor (APE)** [1] se ha desarrollado como un complemento para el entorno de desarrollo Eclipse, con el objetivo de facilitar a los Directores de Proyecto sus labores de control y seguimiento en la construcción de un **Sistema Multiagente (SMA),** mediante la implementación de la técnica ***Earned Value Analysis (EVA),*** lo cual ha permitido:

- Conocer el tamaño aproximado del sistema a desarrollar.
- Estimar el esfuerzo requerido para su desarrollo.
- Planificar las actividades que conforman el desarrollo de un proyecto.
- Establecer el costo, la duración y los recursos necesarios para la ejecución del proyecto.
- Controlar las actividades y tareas a realizar, así como los productos a generar, en función de las características propias del proyecto.
- Documentar las actividades a realizar, con los datos necesarios para su control posterior.

La herramienta agentTool Process Editor (APE) va a ser complementada con el desarrollo de este proyecto, mediante la implementación de una nueva funcionalidad que permitirá a los Directores de Proyecto, analizar y comprender más fácilmente la información y los datos generados durante el proceso de control y seguimiento de este tipo de proyectos, al hacer uso de la técnica *Earned Value Analysis (EVA).*

Esta nueva funcionalidad consiste en la creación de una vista, que permitirá representar gráficamente toda la información relacionada con el alcance, costo y tiempo del proyecto, de un modo más eficiente y por consiguiente, permite detectar oportunamente situaciones de mejora, facilita la toma de decisiones y se asimila de manera rápida y eficaz una gran cantidad de información que gira alrededor de la ejecución de este tipo de proyectos.

Asimismo, los Directores de Proyecto podrán detectar más fácilmente cualquier desviación que pueda afectar el normal desarrollo del proyecto, y prestar atención de manera rápida y oportuna a las posibles incidencias positivas o negativas que puedan ser detectadas, pues en cualquier momento puede existir una fuente de situaciones indeseadas que precise de atención inmediata.

[1] Herramienta desarrollada por el Dr. Juan Carlos García Ojeda, Docente de la Universidad Autónoma de Bucaramanga (UNAB).

1. DESCRIPCIÓN DE LA INVESTIGACIÓN

1.1. ANTECEDENTES

La Gestión de Proyectos es una de las actividades básicas dentro de la Ingeniería del Software que cubre todo el proceso de desarrollo, el éxito de un proyecto depende en gran medida si se logra identificar y comprender claramente el entorno del trabajo, los objetivos a alcanzar, los riesgos en que se puede incurrir, los recursos requeridos, las actividades o tareas a realizar, el tiempo de ejecución, etc. Cada una de estas variables permite definir un plan de trabajo sin importar el tamaño del proyecto, una vez definido el plan de trabajo, el líder de proyecto requiere de habilidades técnicas y administrativas, que le permitan afrontar los diferentes cambios o situaciones indeseadas que se le pueden presentar durante la ejecución de un proyecto, con el fin de garantizar el cumplimiento de los objetivos propuestos en el tiempo estipulado.

Entre los principales inconvenientes que surgen en la dirección de proyectos tenemos:

- Lograr una comunicación efectiva entre los miembros del equipo y las personas interesadas en su desarrollo.
- Contar con las técnicas y herramientas apropiadas para facilitar las labores de control y seguimiento del proyecto.
- Identificar los posibles cambios y variaciones que pueden afectar el normal desarrollo del proyecto.
- Definir y ejecutar acciones correctivas o preventivas de manera oportuna.
- Lograr identificar el avance del proyecto en cualquier fase de su desarrollo.
- Lograr medir y analizar resultados.
- Lograr finalizar los proyectos en el tiempo establecido.
- Ajustar el proyecto con el presupuesto establecido inicialmente.
- Cumplir con las especificaciones y las necesidades del cliente.
- Baja calidad en el software generado.

En respuesta a estos inconvenientes se han venido desarrollando aplicaciones dentro de una nueva área de trabajo y del conocimiento, que ha sido capaz de revolucionar el tratamiento de los sistemas complejos, mediante la construcción de Sistemas Multiagentes (SMA). Entre las aplicaciones desarrolladas para la construcción de un Sistema Multiagente (SMA) se encuentra [2], [3]: el entorno de desarrollo agentTool III, que apoya el proceso de desarrollo de software, y la herramienta agentTool Process Editor (APE), que apoya los procesos de control y seguimiento de este tipo de proyectos.

Estos desarrollos se deben en gran medida, a la dificultad que se presenta a la hora de construir un Sistema Multiagente (SMA), al ser considerado como uno de los paradigmas de software más recientes en comparación con sus competidores, como por ejemplo lo es la Programación Orientada a Objetos (POO) o la Programación Orientada a Servicios (SOA), por lo tanto, se requiere de aplicaciones, estándares y metodologías de desarrollo que garanticen el éxito en la construcción de este tipo de proyectos.

1.2. PLANTEAMIENTO DEL PROBLEMA

El desarrollo de software es uno de los pilares fundamentales en el área de la informática y a lo largo de los últimos años, conforme la tecnología va avanzando van apareciendo nuevas soluciones, nuevas formas de programación, nuevos lenguajes y un sin fin de herramientas que intentan hacer de la labor del programador algo más fácil. Generalmente el software que se construye no satisface los requerimientos ni las necesidades del cliente, sino que además excede los presupuestos y los tiempos de desarrollo que han sido establecidos, estos inconvenientes han dado origen a lo que se denomina la crisis del software, pues no se obtienen los resultados deseados.

Todo proyecto por definición necesita ser ejecutado y entregado bajo ciertas restricciones, tradicionalmente estas restricciones han sido alcance, tiempo y costo, ninguna restricción puede ser modificada sin que esta impacte a las demás, por eso es de gran importancia, el tiempo y la dedicación que el líder del proyecto pueda ofrecer a las actividades de control y seguimiento, con el fin de obtener los objetivos propuestos.

Cuando el líder del proyecto no da importancia y no dispone del tiempo necesario para ejecutar esta labor, se presentan riesgos y situaciones indeseadas que impiden el normal desarrollo del proyecto, aunque no hay nada de malo en que se presenten estas situaciones pues no es posible pretender que el proyecto no los tenga, lo importante es la respuesta y la actitud que asume la administración del proyecto en su desarrollo y frente al riesgo, para asegurar que el proyecto siga de acuerdo a lo planificado.

Para apoyar la construcción de un Sistema Multiagente (SMA) se ha desarrollado la herramienta agentTool Process Editor (APE), para facilitar a los Directores de Proyecto sus labores de control y seguimiento, esta herramienta actualmente requiere ser complementada con una nueva vista gráfica, que permitirá representar gráficamente toda la información relacionada con el alcance, tiempo y costo del proyecto, con el objetivo de facilitar el análisis de los resultados a la hora de implementar la técnica *Eanerd Value Analysis (EVA)*, y a la vez ofrecer al líder de proyecto la posibilidad de identificar oportunamente los posibles riesgos y dificultades que pueden afectar el normal desarrollo del proyecto.

1.3. OBJETIVOS

1.3.1. OBJETIVO GENERAL

Analizar y diseñar una vista gráfica para la administración de procesos de desarrollo personalizados basados en agentes siguiendo la técnica del *Earned Value Analysis (EVA).*

1.3.2. OBJETIVOS ESPECÍFICOS

- Estudiar y comprender la técnica Earned Value Analysis (EVA).
- Estudiar y comprender la herramienta Eclipse Process Framework.
- Estudiar y comprender la herramienta Eclipse.
- Estudiar y comprender la herramienta agentTool Process Editor (APE).
- Proponer e implementar una solución viable para la construcción de una nueva vista gráfica en APE.
- Documentar la solución propuesta.

1.4. JUSTIFICACIÓN

Los Sistemas Multiagente (SMA) así como la Programación Orientada a Objetos (POO) o la Programación Orientada a Servicios (SOA), son nuevos paradigmas de desarrollo de Software que se orientan a ciertos sectores y necesidades específicas, por lo tanto, deben contar con mecanismos y herramientas que apoyen y guíen la construcción de estas soluciones para reducir su complejidad y aumentar su eficiencia y calidad. Sin embargo, los Sistemas Multiagentes (SMA) cuentan con una desventaja inicial al ser más reciente que sus competidores, lo cual minimiza la generación de prácticas y ciclos de vida estandarizados.

Una de las labores más importantes en el desarrollo de Sistemas Multiagente (SMA) es el seguimiento y control de las actividades propuestas, para lo cual se requiere que el jefe de proyecto vigile constantemente, el cumplimiento de una planificación previamente establecida, mediante el desarrollo y entrega de las actividades o tareas que han sido asignadas dentro del calendario previsto, de lo contrario se producirán desviaciones en el proyecto, en cuanto a costos y tiempos de entrega.
La utilización de herramientas automatizadas facilitan el seguimiento y control de las actividades o tareas a realizar, contrastando la situación en la que se encuentran actualmente con la prevista en la planificación del proyecto, por lo tanto, es de gran importancia dar continuidad a los proyectos realizados que suplen estas necesidades, como lo es la herramienta agentTool Process Editor (APE).

La nueva funcionalidad a implementar en la herramienta agentTool Process Editor (APE) permitirá:

- Conocer de manera rápida y oportuna el estado actual de un proyecto, mediante la visualización de las diferentes variables de estudio.
- Identificar los riesgos y situaciones indeseadas de manera oportuna durante la ejecución de un proyecto.
- Facilitar la toma de decisiones y definición de acciones correctivas o preventivas.
- Mejorar el proceso de desarrollo con el fin de garantizar el cumplimiento de los objetivos propuestos.
- Mejorar y rediseñar los procesos de negocio proporcionando ahorros en tiempo y costo.
- Comprender y analizar más fácilmente el avance de un proyecto.
- Contar con un medio de comunicación más eficiente entre las personas interesadas en el desarrollo del proyecto.
- Aprender de lecciones pasadas, al usar la experiencia adquirida en la planificación y realización de proyectos futuros.

Por lo tanto, es de gran importancia la correcta utilización y aplicación de las nuevas tecnologías en el desarrollo de esta propuesta de investigación, para garantizar el correcto funcionamiento de la vista gráfica en la herramienta agentTool Process Editor (APE) y así ofrecer a los líderes de proyecto una nueva funcionalidad que facilitará sus labores de control y seguimiento durante la construcción de un Sistema Multiagente (SMA).

La ejecución de esta propuesta es en gran medida posible, gracias a la filosofía de desarrollo que sigue este tipo de aplicaciones, las cuales se encuentran catalogadas como herramientas de Software Libre, esto nos permite acceder al código fuente de la aplicación para realizar mejoras y ajustes, de acuerdo a las necesidades específicas del cliente, y a su vez, pone a disposición este tipo de desarrollos para que pueda ser aprovechado por un amplio número de usuarios.

1.4.1. IMPACTO

Es importante resaltar que el desarrollo de esta nueva funcionalidad permitirá conocer el avance y el estado actual de un proyecto en cualquier fase de su desarrollo, detectar oportunamente cualquier desviación que se pueda presentar durante la construcción de un Sistema Multiagente (SMA), identificar y analizar las posibles causas que la originaron e implementar las acciones correctivas o preventivas que se requieran y evitar que se vea afectado el normal desarrollo del proyecto. También es importante resaltar que el Director de Proyecto dedique buen parte de su tiempo al control y seguimiento de cada una de las tareas o actividades que están siendo ejecutadas,

prestando especial interés en aquellas que presenten algún retraso para que de alguna manera pueda recuperar el costo o tiempo perdido.

1.4.2. VIABILIDAD

- **TÉCNICA:** Para el desarrollo de esta propuesta de investigación se cuenta con el personal calificado y con la experiencia necesaria, en la dirección, diseño e implementación de proyectos software en el lenguaje de programación JAVA, también se cuenta con la infraestructura tecnológica que soporta el proceso de desarrollo de la propuesta y que está conformada por el hardware y software necesario que garantiza su optimo desarrollo.

- **ECONÓMICA:** En relación al valor de adquisición del software, este concepto no genera ningún tipo costo en el desarrollo de la propuesta, al utilizarse herramientas de Software Libre.

- **SOCIAL:** Con el desarrollo de este proyecto se logra complementar y mejorar una herramienta de Software Libre, para facilita el control y seguimiento de un Sistema Multiagente (SMA). Este desarrollo se le considera de alto impacto, porque beneficia en gran parte a la comunidad del Software Libre y las personas u organizaciones que utilizan estas herramientas en el desarrollo de sistemas complejos.

2. MARCO TEÓRICO

En este capítulo se presenta una descripción general de las bases teóricas que han sido consideradas para el buen desarrollo de la propuesta de investigación, las cuales se centran principalmente en el área de la Ingeniería del Software y su aplicación en entornos de desarrollo de Software Libre.

El contenido del marco teórico que describe el proceso de desarrollo de la vista gráfica en la herramienta agentTool Process Editor (APE) se puede apreciar en la siguiente figura:

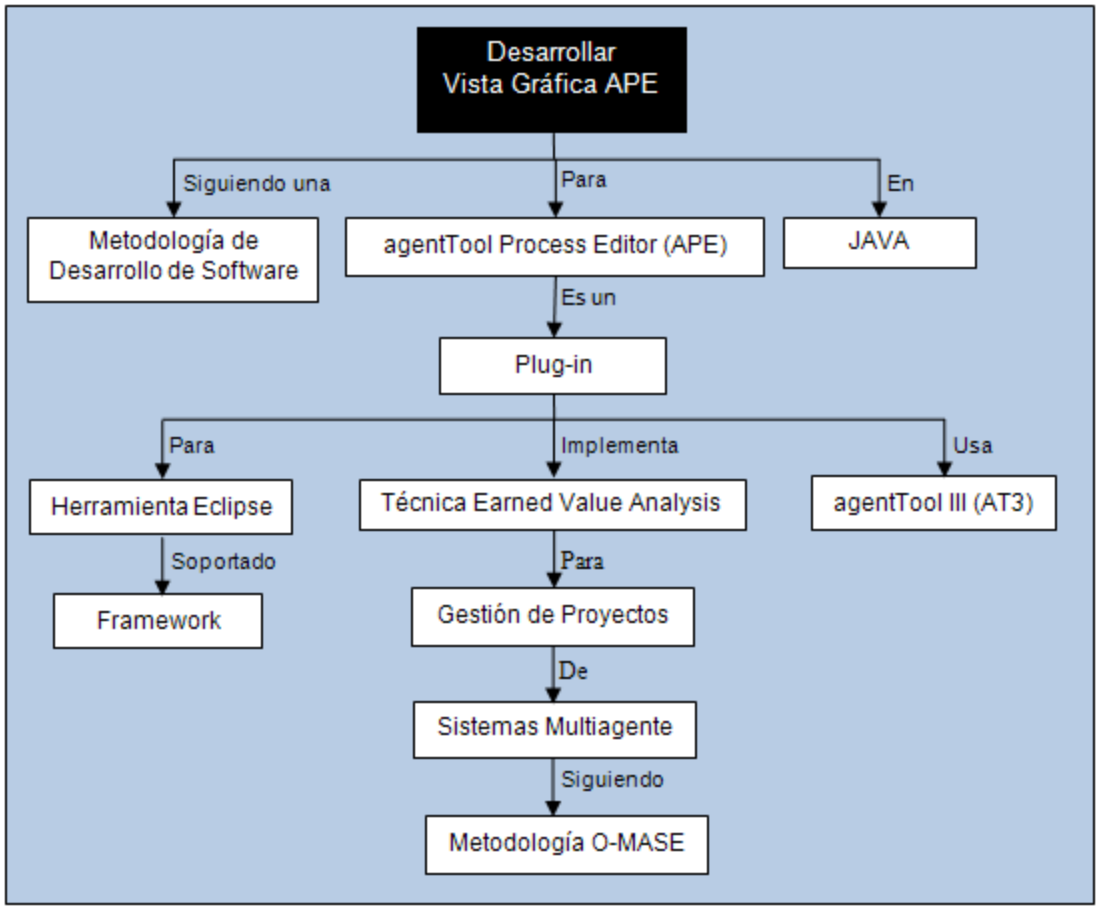

Figura 1. Marco Teórico de la Propuesta de Investigación.

2.1. SOFTWARE LIBRE

2.1.1. HISTORIA

En los años 70 el software era considerado como un valor añadido al hardware, que ofrecían las grandes empresas para que sus clientes pudieran manejarlos adecuadamente, era común que los programadores y desarrolladores de software compartieran libremente sus aplicaciones y código fuente, pero a finales de los años 70 las compañías empezaron a definir restricciones a sus usuarios, mediante el manejo de acuerdos de licencia, lo cual impedía adaptar el software a sus necesidades y corregir errores sin la respectiva autorización del productor, a pesar que el software es considerado como el elemento tecnológico más flexible y adaptable que ha existido hasta el momento.

Debido a esto, en 1984 Richard Stallman comenzó a trabajar en el proyecto GNU y un año más tarde fundó la Free Software Foundation (FSF) introduciendo la definición de free software y el concepto de copyleft[2], para otorgar libertad a los usuarios en el momento de utilizar y adaptar el software a sus necesidades logrando restringir las posibilidades de apropiación del software.

2.1.2. DEFINICIÓN

El software libre [8] tal como fue concebido por Richard Stallman, es toda aplicación o programa informático que garantiza a los usuarios las siguientes libertades:

Libertad 0: La libertad de usar el programa, con cualquier propósito.
Libertad 1: La libertad de estudiar cómo funciona el programa y modificarlo, adaptándolo a tus necesidades.
Libertad 2: La libertad de distribuir copias del programa, con lo cual puedes ayudar a tu prójimo.
Libertad 3: La libertad de mejorar el programa y hacer públicas esas mejoras a los demás, de modo que toda la comunidad se beneficie.

Las libertades 1 y 3 requieren el acceso al código fuente, pues estudiar y modificar el software sin su código fuente, es muy poco viable.

El Software libre en la actualidad permite disponer con total confianza y seguridad, de sistemas operativos totalmente estables y aplicaciones de escritorio de alta calidad,

[2] Copyleft es un método que garantiza que un programa y sus versiones modificadas o extendidas sigan siendo libres.

facilitando el uso del software, en nuestras labores cotidianas. El Software libre, está siempre en proceso de mejora continua al contar con una comunidad de desarrolladores totalmente activa, que respaldan la continuidad del proyecto y que aportan mejoras de manera permanente, colocando a disposición de los usuarios un producto actualizado e innovador, por lo cual ha sido considerado, como una de las estrategias más viables para apropiarnos realmente de las nuevas tecnologías.

2.1.3. TIPOS DE LICENCIA.

Una licencia es una autorización formal de carácter contractual donde el autor establece los términos bajo los cuales se libera dicho software, las licencias de software libre se caracterizan por permitir a sus usuarios utilizar, estudiar, modificar o redistribuir el software sea modificado o no.

Una de las licencias más utilizadas en Software Libre es la Licencia Pública General de GNU (GNU GPL) y debido a la gran variedad de licencias, el proyecto GNU ha establecido en su página oficial: una lista de licencias que pueden considerarse de software libre y que son compatibles con las condiciones establecidas por la GNU GPL y una lista de licencias que no cumplen con estos requerimientos. Estas licencias se encuentran disponibles para su consulta en el siguiente enlace:
http://www.gnu.org/licenses/license-list.es.html#GPLCompatibleLicenses

2.1.4. SOFTWARE LIBRE Y SOFTWARE PROPIETARIO

A diferencia del Software Libre, el Software Propietario es cualquier aplicación o programa informático donde el usuario final tiene ciertas limitaciones en cuanto a su uso, modificación y distribución, o cuando su código fuente no está disponible porque se encuentra restringido por un acuerdo de licencia.

A continuación se presentan algunas diferencias y similitudes entre el software libre y software propietario.

- **DIFERENCIAS**

Característica	Software Libre	Software Propietario
Origen	*El software libre tuvo su origen en el mundo académico.*	*El software propietario tuvo su origen en el mundo empresarial.*
Modelo de negocio	*El software libre plantea un modelo de negocio basado en servicios.*	*El software propietario plantea un modelo de negocio muy lucrativo para las empresas que lo desarrollan y poco ventajoso para*

		los usuarios del software.
Precio	*El cliente no paga por el uso del software sino por los servicios que pueden ir asociados con él.*	*El cliente paga por el uso exclusivo del software.*
Licencias	*Las licencias de software libre ofrecen libertad de uso y redistribución.*	*Las licencias expresan claramente que el cliente sólo adquiere la facultad para utilizar la aplicación en determinada cantidad de equipos.*
	Libre distribución por parte del usuario.	*No se le permite al usuario copiar o transferir la titularidad de la licencia a un tercero.*
	Permite a los usuarios contar con software actualizado de manera permanente.	*Si deseas actualizar el software propietario debes hacerlo mediante la adquisición de otra licencia.*
	Actualmente la curva de aprendizaje es mayor, debido a la gran cantidad de software propietario en el mercado.	*Actualmente la curva de aprendizaje es menor debido a la gran cantidad de software propietario en el mercado.*
Proceso de desarrollo	*Sus procesos de desarrollo suelen ser más informales, debido a que la mayoría de los desarrolladores realizan sus actividades de manera voluntaria.*	*Generalmente se sigue un proceso formal de desarrollo.*
	Publicación temprana de versiones y actualizaciones.	*El desarrollo de nuevas versiones no se realiza de manera frecuente.*
	El software libre garantiza el respeto a los estándares en los formatos, protocolos e interfaces.	*El software propietario generalmente los modifica, para obligar al usuario a adquirir una nueva versión.*
	Las interfaces amigables para el usuario y procesos multimedia apenas se están estabilizando.	*Cuenta con mejores diseños para las interfaces de usuario y facilidades multimedia.*
	El desarrollo de herramientas se realiza de forma cooperativa y abierta, mediante el libre intercambio de su código fuente. Es decir, permite la reutilización de código.	*No permite el intercambio de código fuente.*
Código fuente	*Disponibilidad del código fuente.*	*El usuario no tiene acceso al código fuente.*
	El software se encuentra a disponibilidad de cualquier persona u organización.	*El software es y seguirá siendo propiedad de la empresa.*

	Permite ajustar el código fuente a las necesidades del usuario.	*No se le permite al usuario modificar libremente el código fuente.*
	Independencia del proveedor y en el uso de tecnología.	*La empresa proveedora es la única capacitada en atender los requerimientos de un cliente insatisfecho con el producto adquirido.*
Corrección más rápida y eficiente de fallos	*El proceso de corrección de errores es un proceso dinámico, rápido y eficiente, al contar con una amplia comunidad de usuarios que respaldan el proceso.*	*El proceso de corrección de errores depende exclusivamente del proveedor de software.*
	Considera a los usuarios como colaboradores y es la forma más efectiva y apropiada de mejorar el código y depurarlo.	*Considera a los usuarios como simples consumidores.*
Expansión	*El usuario promedio no reconoce los entornos de software libre.*	*El usuario promedio reconoce fácilmente los entornos de software propietario más comunes.*

Tabla 1. Diferencias entre Software Libre y Software Propietario

- **SIMILITUDES**

- ✓ El desarrollo de sus proyectos tienen como objetivo lograr satisfacer las necesidades de sus clientes.
- ✓ El usuario adquiere mediante un contrato denominado licencia, la facultad para hacer uso del programa.
- ✓ Presentan dificultad con el intercambio de archivos.
- ✓ Cuentan con una gran variedad de software que se ajusta a las necesidades de cualquier usuario doméstico u organización.
- ✓ Ofrecen soporte y documentación necesaria para el manejo de sus productos.
- ✓ Sus procesos de desarrollo se encuentran altamente definidos.
- ✓ La mayoría de sus proyectos importantes y de trayectoria cuentan con un buen soporte.
- ✓ Mediante el uso de sus desarrollos se logran beneficios sociales y tecnológicos para la región.
- ✓ Ambos modelos presentan una clara distribución de actividades y responsabilidades.

2.2. SISTEMAS MULTIAGENTE (SMA)

Durante los últimos años se ha venido consolidando una nueva área de trabajo y de conocimiento capaz de revolucionar el tratamiento de los sistemas complejos, gracias a la interacción realizada entre la Inteligencia Artificial y la Informática Distribuida, y la aplicación del nuevo concepto "agente inteligente" en la investigación básica y aplicada.

2.2.1. DEFINCIÓN

Un Sistema Multiagente es un conjunto de agentes autónomos, generalmente heterogéneos e independientes que trabajan en común para la solución de un problema en particular.

Según [7] un Sistema Multiagente reúne los siguientes elementos:

- Un entorno.
- Un conjunto de objetos que se encuentran integrados en el entorno. Estos objetos pueden ser pasivos, percibidos, creados, destruidos y modificados por agentes.
- Un conjunto de agentes que se consideran como objetos especiales que representan las partes activas del sistema.
- Un conjunto de relaciones que unen objetos y por lo tanto agentes.
- Un conjunto de operaciones que hacen posible que los agentes perciban, produzcan, consuman, transformen y manipulen objetos.
- Operadores que representan la aplicación de operaciones sobre el mundo y la reacción de éste al ser alterado. Estos operadores se pueden entender como las leyes del universo.

2.2.2. CONCEPTO DE AGENTE

Woodridge y Jennings [23] definen un agente como un sistema informático hardware o más frecuentemente software, que posee las siguientes propiedades:

- **Autonomía:** los agentes actúan sin la intervención directa de humanos u otros agentes y tienen algún tipo de control sobre sus acciones y estado interno.
- **Habilidad social:** los agentes interactúan con otros agentes (e incluso con humanos) por medio de algún tipo de lenguaje de comunicación de agentes.
- **Reactividad:** un agente percibe su entorno y responde de forma apropiada en un tiempo razonable a los cambios que ocurren en él.
- **Pro-actividad:** los agentes no actúan simplemente en respuesta a su entorno, sino que también deben exhibir un comportamiento dirigido por objetivos tomando la iniciativa.

2.2.3. METODOLOGÍAS DE DESARROLLO

Las metodologías de desarrollo son un conjunto de guías que cubren todo el ciclo de vida del desarrollo de un sistema complejo. Para el desarrollo de Sistemas Multiagente (SMA) existe una gran variedad de metodologías que se diferencian en el enfoque utilizado para su desarrollo, estos enfoques son:

- El enfoque orientado a objetos, como por ejemplo: INGENIAS, MaSe, MESSAGE/UML.
- El enfoque de ingeniería del conocimiento, como por ejemplo: CoMoMAS, MAS-CommonKADS, TROPOS.
- El enfoque de teoría de agentes, como por ejemplo: Gaia, KAoS, Vowel Engineering.

Pare el desarrollo de Sistemas Multiagente (SMA) el Grupo de Investigación Ingeniería de Software y Sistemas de Información de la UNAB, también sigue entre sus metodologías de desarrollo, la metodología O-MaSE, la cual se deriva de una integración realizada entre la metodología MaSE y el Open Process Framework (OPF). La metodología MaSE (Multi-agent systems Software Engineering) se concibe como una abstracción del paradigma orientado a objetos donde los agentes son especializaciones de objetos y el Open Process Framework (OPF) es un enfoque estandarizado para la creación de un método de ingeniería de requerimientos.

Por lo tanto, la metodología O-MASE [11] propone crear agentes como extensiones de objetos y se divide fundamentalmente en tres fases: Requisitos de Ingeniería, Análisis y Diseño. A su vez, la primera fase se subdivide en 3 etapas: captura de objetivos, aplicación de casos de uso y refinamiento de roles. La fase de diseño también está subdividida en 4 etapas: creación de las clases de agentes, construcción de las conversaciones, ensamblaje de las clases y diseño del sistema.

2.2.4. HERRAMIENTAS DE DESARROLLO

- **Entorno de Desarrollo agentTool III (aT3)**

aT3 [9] es un entorno de desarrollo que soporta los procesos de análisis y diseño en la construcción de un Sistema Multiagente (SMA) mediante la implementación de la metodología O-MaSE. Es una herramienta que nos permite crear de modo visual los diagramas del proceso de desarrollo de este tipo de proyectos y pone a disposición de los usuarios una base de conocimiento persistente, un verificador de conversaciones y la generación automática de código.

Es un proyecto patrocinado por la National Science Foundation and the Air Force Office of Scientific Research, que está siendo desarrollado utilizando la versión de Java 1.5 y funciona como un complemento para el entorno de desarrollo Eclipse 3.3, 3,4 y 3,5 (Europa, Ganímedes, y Galileo). Esta herramienta se encuentra disponible en la página oficial del proyecto en su página de descargas.

- **Herramienta agentTool Process Editor (APE)**

APE [10] es una herramienta que ha sido desarrollada como un complemento para el entorno de desarrollo Eclipse, con el objetivo de facilitar la gestión de los procesos que conforman el desarrollo de un Sistema Multiagente (SMA). Esta herramienta proporciona tres funcionalidades básicas: en primer lugar, proporciona interfaces gráficas que facilitan al usuario guardar la información relacionada con los requisitos del proyecto, en segundo lugar se integra a la herramienta aT3 para facilitar a los administradores y desarrolladores, ver y editar directamente los productos de trabajo que se esperan de cada tarea que ha sido relacionada y en tercer lugar, proporciona soporte para el acceso y el seguimiento de las actividades planeadas mediante la implementación de la Técnica del Valor Ganado.

- **Entorno de Desarrollo TAOM4E [22]**

Esta herramienta también ha sido desarrollada como un plug-in para el entorno de desarrollo eclipse, para apoyar el proceso de desarrollo de software orientado a agentes, soportando la metodología Tropos y se considera como una solución flexible al problema de integración de componentes. Esta metodología se conforma por dos diagramas que permiten modelar el ambiente organizacional del sistema con un enfoque basado en metas y dependencias.

- **Entorno de Desarrollo APTK**

APTK es una herramienta que ha sido desarrollada para apoyar el proceso de desarrollo de software orientado a agentes, soportando la metodología PASSI, aunque todavía tiene que ser significativamente mejorada para alcanzar la flexibilidad y amplio soporte que ofrece la herramienta de soporte convencional PASSI (PTK). La metodología PASSI se conforma por las siguientes fases principales: Requisitos del sistema, Sociedad del Agente, Aplicación del agente, Código e implementación.

2.3. TÉCNICA EARNED VALUE ANALYSIS (EVA)

2.3.1. HISTORIA

La Técnica *Earned Value Analysis (EVA)* o el Método de Valor Ganado ha sido desarrollado realmente en el siglo XIX, cuando surgió la necesidad de medir los rendimientos de las factorías[3], y no fue sino hasta el año 1962 que el departamento de defensa de los Estados Unidos la adoptó como una metodología estándar para medir el rendimiento de sus proyectos. Durante todo este tiempo muchos de sus criterios y políticas de control costo/tiempo han ido evolucionando y aunque algunos de sus acrónimos han sido modificados prevalecen sus fundamentos. El Método de Valor Ganado ha logrado alcanzar una notable popularidad en el mundo de la gestión de proyectos y es por tal razón que en el año 2005 el *Project Management Institute (PMI)*[4] decide publicar el estándar del Método del Valor Ganado como buena práctica en la dirección de proyectos[5].

2.3.2. DEFINICIÓN

La Técnica del Valor Ganado o el Método de Valor Ganado según el PMBOK[6] es *"un método objetivo para medir el desempeño del proyecto en lo referente al alcance, tiempo y costo",* es decir, es un método que permite comparar la cantidad de trabajo planeado contra lo que realmente se ha terminado, para evaluar el desempeño del proyecto, facilitar la toma de decisiones y determinar si el COSTO, el CRONOGRAMA y el TRABAJO REALIZADO se están llevando de acuerdo a lo planeado.

2.3.3. UTILIZACIÓN

Para determinar correctamente el estado y el rendimiento de un proyecto mediante la implementación de la Técnica del Valor Ganado EARNED VALUE se requiere según [18]:

- Definir claramente los objetivos del proyecto para planificar, programar y controlar su desarrollo.

[3] Establecimiento de comercio, especialmente el situado en país colonial. Diccionario de la Real Academia Española.
[4]Organización internacional sin fines de lucro que asocia a profesionales para la gestión de proyectos. http://www.pmi.org/
[5] Practice Standard for Earned Value Management by the Project Management Institute (2005 edition).
[6] A Guide to the Project Management Body of Knowledge (PMBOK® Guide), 2004 Edition by the Project Management Institute.

- Establecer la Estructura de Descomposición del Proyecto (WBS[7]), es decir, Identificar cada entregable del proyecto.
- Desarrollar un cronograma para la terminación de cada entregable del proyecto, a partir del WBS y el alcance del Proyecto
- Establecer compromisos de realización del proyecto y asignar responsabilidades, basados en el conocimiento de los recursos disponibles.
- Asignar un valor a cada entregable, es decir, cada tarea individual no gana valor sino hasta que esta haya finalizado.
- Reportar el trabajo ejecutado y asignar los costes reales.
- Realizar y analizar los cálculos de Valor Ganado para tomar decisiones e implementar acciones correctivas si se requieren.

2.3.4. MEDIDAS BÁSICAS E INDICADORES

La Técnica del Earned Value Analysis (EVA) maneja su propia simbología y conceptos, según [12] estos son:

MEDIDAS BÁSICAS

- ***Budgeted Cost of Work (BCW):*** Representa el esfuerzo estimado para cada tarea de trabajo.
- ***Budgeted Cost of Work Sheduled (BCWS)****:* Representa el costo presupuestado del trabajo programado. Responde a la pregunta: ¿Cuánto trabajo se debe haber terminado?
- ***Budgeted Cost of Work Performed (BCWP)***: Representa el costo presupuestado del trabajo realizado. Responde a la pregunta: ¿Cuánto trabajo se ha terminado realmente?
- ***Actual Cost of Work Performed (ACWP):*** Representa el costo real del trabajo realizado. Responde a la pregunta: ¿Cuánto hemos gastado?
- ***Budget at Completion (BAC):*** Presupuesto a la terminación. Es la suma de todos los presupuestos asignados a un proyecto (BCWS).
- ***Planned Value (PV):*** Valor planeado.

INDICADORES DE PROGRESO

- ***Earned Value (EV):*** Representa la suma de los PVs para todas las tareas realizadas. EV = BCWP/BAC
- ***Schedule Performance Index (SPI):*** Índice del rendimiento del cronograma. SPI = BCWP/BCWS
- ***Schedule Variance (SV):*** Variación del cronograma. SV = BCWP – BCWS.

[7] Método para lograr una descomposición lógica de un elemento largo o complejo.

- ***Cost Performance Index (CPI):*** Índice del rendimiento del costo. CPI = BCWP/ACWP
- ***Cost Variance (CV):*** Variación del costo. CV = BCWP – ACWP

ANÁLISIS DE INDICADORES

En la Tabla 2 encontramos los diferentes indicadores y la interpretación dada al indicador, teniendo en cuenta el resultado obtenido en el momento de aplicar la Técnica del *Earned Value Analysis (EVA)* en el proyecto.

Variable	Resultado	Interpretación
CV	**= 0**	*El proyecto está dentro del presupuesto.*
	POSITIVO	*El proyecto está por debajo del presupuesto.*
	NEGATIVO	*El proyecto está por encima del presupuesto.*
CPI	**= 1**	*El proyecto está dentro del presupuesto.*
	> 1	*El proyecto está por debajo del presupuesto.*
	< 1	*El proyecto está por encima del presupuesto.*
SV	**= 0**	*El proyecto está a tiempo.*
	POSITIVO	*El proyecto está adelantado con respecto al cronograma.*
	NEGATIVO	*El proyecto está retrasado con respecto al cronograma.*
SPI	**= 1**	*El proyecto está a tiempo.*
	> 1	*El proyecto está adelantado con respecto al cronograma.*
	< 1	*El proyecto está retrasado con respecto al cronograma.*

Tabla 2. Análisis de Indicadores.

2.3.5. VENTAJAS Y DESVENTAJAS

A continuación se presenta algunas ventajas y desventajas de la aplicación de la Técnica del *Earned Value Analysis (EVA)* en la gestión de proyectos:

VENTAJAS

- Técnica que permite llevar un registro real del desempeño del proyecto.
- Supervisión total de las variables del proyecto.
- Ayuda a identificar las formas de corregir o prevenir las situaciones no deseadas en el desarrollo del proyecto.
- Es una forma eficaz de comunicar a los interesados del proyecto el estado actual de su desarrollo.
- Ofrece medidas de eficiencia en relación al costo y al cronograma.
- Permite identificar los posibles riesgos y problemas en una fase temprana, permitiendo que se tomen acciones correctivas a tiempo.
- Reduce la necesidad de que todos los miembros del equipo estén realizando informes constantemente sobre el avance del proyecto.
- Existe uniformidad en el análisis de todas las fases de desarrollo del proyecto al aplicar los mismos criterios en cada uno de sus elementos.

DESVENTAJAS

- A medida que el proyecto avanza el valor ganado tiende al valor planeado, lo cual origina una pérdida de su capacidad predictiva a medida que nos acercamos al final del proyecto.
- La variación del cronograma (SV) a medida que nos acercamos al final del proyecto tenderá a cero mientras que el índice de rendimiento (SPI) tenderá a 1, independientemente de los retrasos o sobre costos, por lo que se ha desarrollado el concepto de programación ganada (ES, earned Schedule).
- Tanto el valor ganado como la programación ganado no tienen en cuenta el efecto de aprendizaje que se tiene a lo largo del ciclo de vida del proyecto.
- En una técnica que no tiene en cuenta la flexibilidad de la gestión, al no permitir establecer alternativas diferentes a las establecidas en el plan de proyecto inicial, teniendo en cuenta los posibles cambios o eventualidades que puedan presentarse durante su ejecución.
- Es una técnica que no considera el riesgo, pues a medida que avanza el proyecto muchas de sus actividades reducen o anulan su riesgo.

2.4. LENGUAJE DE PROGRAMACIÓN JAVA

2.4.1. DEFINICIÓN

Java es un lenguaje de programación orientado a objetos que ha sido desarrollado a partir de los años 90 por Sun Microsystems, con el objetivo de contar con un entorno de desarrollo que sea independiente de la plataforma, debido a la gran variedad de dispositivos y procesadores existentes en el mercado y sus continuas modificaciones. En el año 2006 y 2007, Sun Microsystems libera la mayor parte de sus tecnologías bajo la licencia GNU GPL de tal forma que prácticamente todo el Java de Sun es ahora Software Libre, a excepción de la biblioteca de clases que son requeridas para ejecutar las aplicaciones.

2.4.2. CARACTERISTICAS GENERALES

Según Sun Microsystems Java es considerado como un lenguaje simple, orientado a objetos, distribuido, robusto, seguro, de arquitectura neutra, portable, interpretado, de alto rendimiento, multitarea y dinámico. Es decir:

- **Simple:** Java ha sido creado teniendo en cuenta los lenguajes más utilizados por los desarrolladores que eran el C y el C++, eliminando una serie de características poco utilizadas y de difícil comprensión del C++, lo cual facilita su aprendizaje.

- **Orientado a objetos:** Este diseño se orienta hacia los datos (objetos), sus funciones e interrelaciones (métodos). Siguiendo los mismos criterios de C++.

- **Distribuido:** Este lenguaje incluye una amplia librería de rutinas que permiten trabajar fácilmente con los protocolos de TCP/IP como HTTP o FTP.

- **Robusto:** Se emplean referencias a objetos, los cuales son identificadores simbólicos. El gestor de memoria de Java lleva una contabilidad de las referencias a los objetos y cuando ya no existe una referencia a un objeto, éste se convierte en candidato para la recogida de basura (garbage collection).

- **Seguro:** Es un lenguaje que ha puesto mucho énfasis a la seguridad en cuanto a virus e intrusiones y autenticación, al estar orientado a entornos distribuidos en red.

- **Arquitectura neutra:** El código fuente una vez compilado funciona sobre cualquier equipo y sistema operativo, al estar diseñado para ser fácilmente interpretado por la máquina virtual de java.

- **Portable:** Su arquitectura neutra nos proporciona portabilidad.

- **Interpretado:** Java genera instrucciones bytecodes que se traducen en tiempo de ejecución a instrucciones de la máquina nativa, es decir, son interpretadas y no se almacenan en ningún lugar.

- **Alto rendimiento:** Además de la generación de los bytecodes que de por sí ya es eficiente, a esto se le aplican diversos procesos de optimización.

- **Multitarea:** Java permite construir aplicaciones con múltiples hilos de ejecución, lo que simplifica su uso y lo hace más robusto.

- **Dinámico:** Java se diseñó para adaptarse a un entorno cambiante, permite hacer comprobaciones de tipo en tiempo de ejecución y se puede confiar en los tipos en Java, permite añadir nuevos métodos e instancias sin tener ningún efecto sobre sus clientes.

2.4.3. PLATAFORMA JAVA

Una plataforma de desarrollo se define como el entorno hardware o software que se requiere para la ejecución de un programa, aunque la mayoría de las plataformas se pueden describir como una combinación de sistema operativo y hardware, la plataforma para Java se diferencia de las otras en que se compone de una plataforma software que funciona sobre otras plataformas (GNU/Linux, Windows, Macintosh, Solaris etc.) y se conforma por dos componentes esenciales:

- **La Máquina virtual (MV):** Como ya se ha mencionado, una de las principales características que proporciona Java es la independencia de la plataforma hardware: una vez compilado el código fuente, los programas se deben poder ejecutar en cualquier plataforma.
- **El Application programming interface (API):** El API de Java es una gran colección de software ya desarrollado que proporciona múltiples capacidades como entornos gráficos, comunicaciones, multiproceso, etc. Está organizado en librerías de clases relacionadas e interfaces, estas Las librerías reciben el nombre de packages.

2.4.4. ENTORNOS DE DESARROLLO

Para desarrollar una aplicación en java, Sun Microsystems[8] distribuye de forma gratuita el Java Development Kit (JDK), el cual es un conjunto de programas y librerías que

[8] Empresa informática comprada recientemente Oracle Corporation.

permiten el desarrollo, compilación y ejecución de estas aplicaciones, pero también encontramos otros entornos de desarrollo que vale la pena destacar como son:

- **El proyecto Eclipse** que sigue una filosofía de código abierto y se encuentra disponible en su página de descarga: http://www.eclipse.org.
- **Jcreator,** que además de desarrollar una versión comercial, dispone de una versión limitada, de fácil manejo y se encuentra disponible para su descarga en http://www.jcreator.com.
- **BlueJ,** es un sencillo entorno de programación diseñado para la enseñanza y el aprendizaje de Java, es un proyecto que nace de un grupo de investigación universitario integrado por miembros británicos y australianos. Se encuentra disponible para su descarga en: http://www.bluej.org/.
- **NetBeans,** es un proyecto de código abierto desarrollado por la compañía Sun Microsystems y puede descargarse e instalarse en un mismo paquete con el kit de desarrollo JDK. También lo encontramos disponible para su descarga en: http://www.netbeans.org/.

2.4.5. EL API (Applications programming interface) de JAVA

La multitud de bibliotecas y funciones que proporciona el mismo lenguaje es uno de los aspectos más importantes de Java; bibliotecas que están bien documentadas y organizadas en paquetes y que funcionan correctamente en las diferentes plataformas. Este conjunto de bibliotecas se encuentra definido en el API de Java y entre las principales clases encontramos:

Paquete	Descripción
java.lang	*Clases fundamentales para el lenguaje como la clase String y otras.*
Java.io	*Clases para la entrada y salida a través de flujos de datos, y ficheros del sistema.*
Java.util	*Clases de utilidad como colecciones de datos, modelo de eventos, facilidades horarias, generación aleatoria de números, y otras.*
Java.math	*Clase que agrupa todas las funciones matemáticas.*
Java.applet	*Clase con utilidades para crear applets y clases que las applets utilizan para comunicarse con su contexto.*
Java.awt	*Clases que permiten la creación de interfaces gráficas con el usuario, y dibujar imágenes y gráficos.*
Javax.swing	*Clases con componentes gráficos que funcionan igual en*

	todas las plataformas Java.
Java.security	*Clases responsables de la seguridad en Java (encriptación, etc.).*
java.net	*Clases con funciones para aplicaciones en red.*
Java.sql	*Clase que incorpora el JDBC para la conexión de Java con bases de datos.*

Tabla 3. Principales bibliotecas del API de JAVA

2.5. ECLIPSE

2.5.1. DEFINICIÓN

Eclipse [5] es un entorno de desarrollo de código abierto y multiplataforma para el desarrollo de "Aplicaciones de Cliente Enriquecido"[9], originalmente fue desarrollado por IBM como sucesor de las herramientas para VisualAge y actualmente es desarrollado por la Fundación Eclipse[10].En definitiva, Eclipse puede ser considerado como una plataforma universal para la integración de herramientas de desarrollo, como se muestra en la Figura 2:

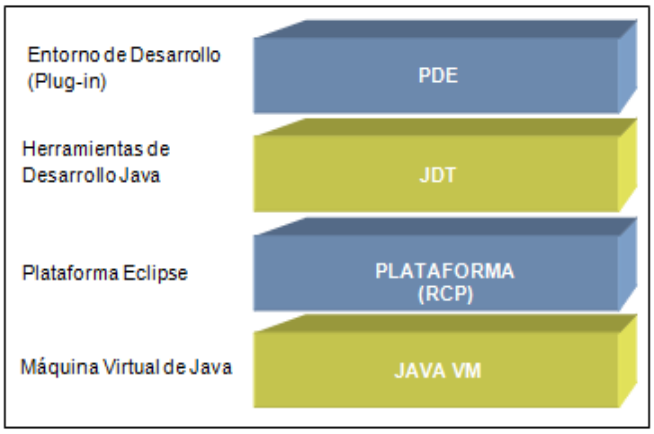

Figura 2. Plataforma Eclipse.

[9] Cliente enriquecido es un término medio entre cliente liviano (aplicación que se accede por una interfaz Web) y el cliente pesado (aplicación que se ejecuta en el sistema operativo del usuario) que proporciona una interfaz gráfica con una sintaxis basada en XML.

[10] La Fundación Eclipse es una organización independiente y sin ánimo de lucro que fomenta el Software Libre con ayuda de una amplia comunidad de usuarios, que extienden constantemente las áreas de aplicación que han sido cubiertas.

2.5.2. ARQUITECTURA

La arquitectura de la plataforma Eclipse es una arquitectura abierta, extensible basada en componentes añadidos (plug-ins), que a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite o no el usuario. La base de esta arquitectura es la Plataforma de Cliente Enriquecido (del Inglés Rich Client Platform RCP) que tiene como objetivo principal cargar y ejecutar los complementos, entre los principales componentes que constituyen esta la plataforma son:

- **Plataforma principal** - inicio de Eclipse, ejecución de plug-ins.
- **OSGi** - una plataforma para bundling estándar.
- **El Standard Widget Toolkit (SWT)** - Un widget toolkit portable.
- **JFace** - manejo de archivos, manejo de texto, editores de texto
- **El Workbench de Eclipse** - vistas, editores, perspectivas, asistentes

Estos componentes se pueden observar en la Figura 3:

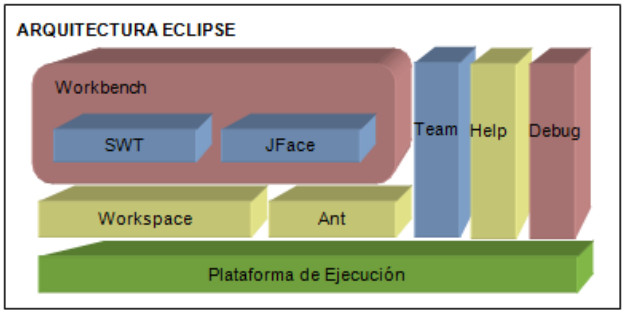

Figura 3. Arquitectura de Eclipse.

2.5.3. DESARROLLO DE COMPLEMENTOS

Al desarrollar un complemento para la plataforma eclipse este necesita obligatoriamente de un archivo de manifiesto llamado plugin.xml, donde se describe la configuración y su integración con la plataforma, también suele incluir archivos binarios en Java, un índice de contenidos de ayuda, iconos, esquemas y otros recursos. Es importante tener en cuenta que toda la Información relacionada con el plug-ins está contenida en los siguientes 3 ficheros: "build.properties", "MANIFEST.MF" y "plugin.xml".

La Figura Nº 4 nos muestra la arquitectita de complementos que se maneja para el entorno de desarrollo Eclipse:

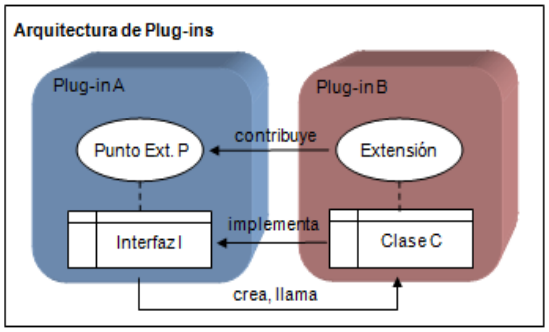

Figura 4. Arquitectura de Plug-ins en Eclipse.

Disposición Básica

Un desarrollador o fabricante proporciona una herramienta independiente como un plug-in que permite llevar acabo determinada funcionalidad y la comunicación entre plug-ins se lleva a cabo mediante los puntos de extensión. Es decir:

1. Plug-in A:
Declara un punto de extensión P.
Declara la interfaz I correspondiente a P.

2. Plug-in B:
Implementa la interfaz I en la clase C.
Define clase C como una contribución a P.

3. El plug-in A instancia C y llama a sus I-métodos.

2.6. LIBRERÍA JFREECHART

JFreeChart [14] es una librería para gráficos escrita totalmente en Java que facilita crear y mostrar gráficos de calidad profesional en nuestras aplicaciones. Entre las características principales de esta biblioteca tenemos:

- Un API consistente y bien documentado con soporte para un amplio rango de tipos de gráficas (en este proyecto, se han utilizado únicamente las gráficas de línea).

- Un diseño flexible, fácilmente extendible, y con la posibilidad de ser usado tanto en tecnologías de servidor (aplicaciones Web) y de cliente (SWT, por ejemplo).

- Soporte para varios tipos de salida, incluyendo componentes Swing o SWT, archivos de imagen como PNG y JPEG, y formatos gráficos de vectores (incluyendo PDF, EPS y SVG).

- JFreeChart es open source y éste está distribuido bajo la licencia LGPL, que permite el uso en aplicaciones propietarias.

- El código y la especificación del mismo es gratuita, sin embargo, la guía del desarrollador de JFreeChart, así como el acceso a algunos ejemplos asociados, se pueden adquirir pagando en la página oficial: http://www.jfree.org/jfreechart/

Un ejemplo de los diferentes gráficos que se pueden conseguir utilizando JFreeChart, se pueden observar en la Figura 5:

Figura 5. Gráficos con JFreeChart

2.7. METODOLOGÍAS DE DESARROLLO

2.7.1. DEFINICIÓN

El papel de las metodologías de desarrollo es sin duda un factor esencial en la Ingeniería del Software, al permitir estructurar, planificar y controlar el desarrollo de un proyecto con el fin de garantizar la consecución de los objetivos propuestos en el tiempo establecido.

2.7.2. TIPOS DE METODOLOGÍAS

En la actualidad es posible diferenciar estas metodologías en dos grandes enfoques: las metodologías tradicionales y las metodologías ágiles.

Las metodologías tradicionales centran su atención en llevar una documentación exhaustiva del proceso y en el cumplimiento de un plan de proyecto que ha sido definido en su fase inicial, lo que dificultaba la implementación de nuevos requisitos y origina altos costos en el proyecto. Entre las principales metodologías tradicionales tenemos:

- **Modelo en Cascada:** Esta metodología es un proceso lineal secuencial donde el desarrollo de software comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. Se hace hincapié en la planificación, los horarios, fec mhas, presupuestos y ejecución de todo el sistema de una sola vez.

- **Prototipado:** Esta metodología propone la creación de un prototipo que represente el sistema en base los requerimientos dados inicialmente por el cliente, aunque no es un sistema completo debe poseer las características del sistema final o parte de ellas y medida que el proyecto avanza y el producto se fortalece hasta abarcar cada vez más las características del producto final.

- **Rational Unified Process (RUP):** Metodología que ha sido desarrollada por Rational Software para es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales, respetando un cronograma y presupuesto previamente establecido.

- **Microsoft Solution Framework (MSF)**[11]: Metodología que contiene las mejores prácticas en cuanto a administración de proyectos se refiere, más que una metodología rígida de administración de proyectos es una serie de modelos que puede adaptarse a cualquier proyecto de tecnología de información.

Las metodologías agiles surgen como una reacción a las metodologías tradicionales que hacían más extenso el proceso de desarrollo de software en su intento de minimizar los riesgos, pero al enfrentamos diariamente a desarrollos de software que requieren de lapsos de tiempo más cortos y resultados a corto plazo, estas metodologías proponen una mayor implicación del cliente en el proyecto y da mayor importancia la capacidad de respuesta para afrontar un cambio que al seguimiento

[11] Microsoft Solution Framework, (en línea), disponible en http://www.gpicr.com/msf.aspx

estricto de un plan. Como resultado de este nuevo enfoque se crea el Manifiesto Ágil[12]. Entre las principales metodologías agiles tenemos:

- **eXtreme Programming (XP):** Esta metodología ha sido formulada por Kent Beck y es la más destacada dentro de los procesos agiles de desarrollo, esta metodología establece el cambio de los requisitos de un proyecto como un aspecto natural e inevitable durante su desarrollo, por lo tanto, es una mejor aproximación quizá la más próxima a la forma de organizar y participar en proyectos de software libre., que intentar definir todos los requisitos al inicio del proyecto e invertir esfuerzo para evitar modificaciones.

- **SCRUM:** Es un enfoque creado para la gestión y desarrollo de software basado en un proceso iterativo e incremental, está conformado por un conjunto de prácticas y roles que pueden tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto.

- **Crystal Methodologies:** Es un conjunto de metodologías para el desarrollo de software que han sido creadas por Alistair Cockburn y se caracterizan por estar centradas en las personas que conforman el equipo de trabajo, reduciendo al máximo el número de artefactos producidos. Las políticas de trabajo definidas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros).

- **Adaptive Software Development (ASD):** Esta metodología ha sido desarrollado Cutter Consortium hacia el año 2000. Sus principales características son: iterativo, orientado a los componentes de software más que a las actividades o tareas y es tolerante a los cambios. El ciclo de vida que propone tiene tres fases esenciales: especulación, colaboración y aprendizaje. La estrategia de esta metodología se basa en el concepto de emergencia, una propiedad de los sistemas adaptativos complejos que describe la forma en que la interacción de las partes genera una propiedad que no puede ser explicada en función de los componentes individuales.

[12] Pires, Donald, "Manifiesto Ágil", UCLA, (en línea), disponible en http://www.manifiestoagile.com

3. MARCO METODOLÓGICO

La presente propuesta de investigación se basa principalmente en los conceptos de la Investigación Tecnológica Aplicada o Investigación Tecnológica, al estar orientada a la compresión y el análisis de una situación en particular, con el objetivo de complementar una solución software ya existente. Esta solución software, corresponde a la Herramienta agentTool Process Editor (APE), la cual implementa la Técnica del Earned Value Analysis o el Método del Valor Ganado, para la gestión de proyectos de Sistemas Multiagentes (SMA).

La nueva funcionalidad a desarrollar en la Herramienta agentTool Process Editor (APE), corresponde a la creación de una nueva vista gráfica que permitirá a los Directores de Proyecto, entender y analizar más fácilmente los resultados obtenidos de la aplicación de la Técnica *Earned Value Analysis (EVA)*, en el seguimiento y control de sus proyectos. En su desarrollo se implementarán los principios y conceptos básicos de la metodología eXtreme Programing o programación extrema, al ser una de las metodologías más destacadas en los procesos ágiles de desarrollo de software y al ser una de las metodologías que más se adapta a los entornos de desarrollo del Software Libre.

El proceso de investigación que se propone para el desarrollo de esta propuesta, se conforma por las siguientes fases:

3.1. FASE CONCEPTUAL

- Revisión y análisis de la literatura seleccionada para al desarrollo de la propuesta.
- Estudio y descripción de la situación actual.
- Formulación del problema de Investigación y de las preguntas a resolver.
- Estudio de la documentación que describe o especifica cada uno de los procesos a implementar.
- Estudio del impacto y la viabilidad de la propuesta.

3.2. FASE DE PLANIFICACIÓN

Esta fase inicia con la confección de las “historias de usuario”. Una historia de usuario es considerada como un escrito o descripción breve donde el cliente describe en forma

clara y concisa sobre la prestación de un proceso o servicio, y son utilizadas para la especificación de los requerimientos. Las actividades a desarrollar en esta fase son:

- Estudio y reestructuración de las historias obtenidas por parte del cliente.
- Identificación de los requisitos funcionales y no funcionales del sistema.
- Elaboración del diagrama de casos de uso.
- Definición de la arquitectura de desarrollo.
- Definición del entorno tecnológico para el desarrollo de la vista gráfica.

3.3. FASE DE DESARROLLO

Esta fase consiste en poner en marcha el plan de trabajo establecido en la fase de planificación, enfocados en cuatro atributos principales: la estructura del archivo de datos, la arquitectura del software, el detalle procedimental y la caracterización de la interfaz de usuario. Entre las actividades a desarrollar en esta fase son:

- Definición de los estándares de codificación.
- Elaboración del diagrama de clases.
- Implementación de los requisitos funcionales y no funcionales de la aplicación.
- Desarrollo de las pruebas de funcionalidad para validar los requerimientos establecidos.
- Presentación de los resultados obtenidos para la integración de las funcionalidades con el resto del sistema.
- Reportar y corregir errores de funcionalidad.
- Ajustar y mejorar funcionalidades del sistema.

3.4. FASE DE PRUEBAS

Las pruebas se integran dentro de las diferentes fases del desarrollo con el objetivo de verificar la calidad del producto software y el cumplimiento de los requerimientos establecidos, mediante un monitoreo sobre el proceso realizado que nos permita determinar:

- Los resultados del proceso de adopción e implantación de la nueva funcionalidad.
- Las dificultades presentadas durante el desarrollo del proyecto.
- Las recomendaciones o sugerencias para implementar nuevas mejoras, mediante la utilización del software para el cubrimiento de otros aspectos del proceso.

3.5. PRODUCTO FINAL

Contar con un adecuado y correcto funcionamiento de la Vista Gráfica en la herramienta agentTool Process Editor (APE).

3.6. ELABORACIÓN DE INFORME FINAL

Recopilación de los informes y resultados obtenidos de las diferentes fases del proceso de investigación, para la creación del informe final del desarrollo de la propuesta.

El proceso de investigación que se describe para el desarrollo de la vista gráfica en la herramienta agentTool Process Editor (APE) se puede observar en la Figura 6.

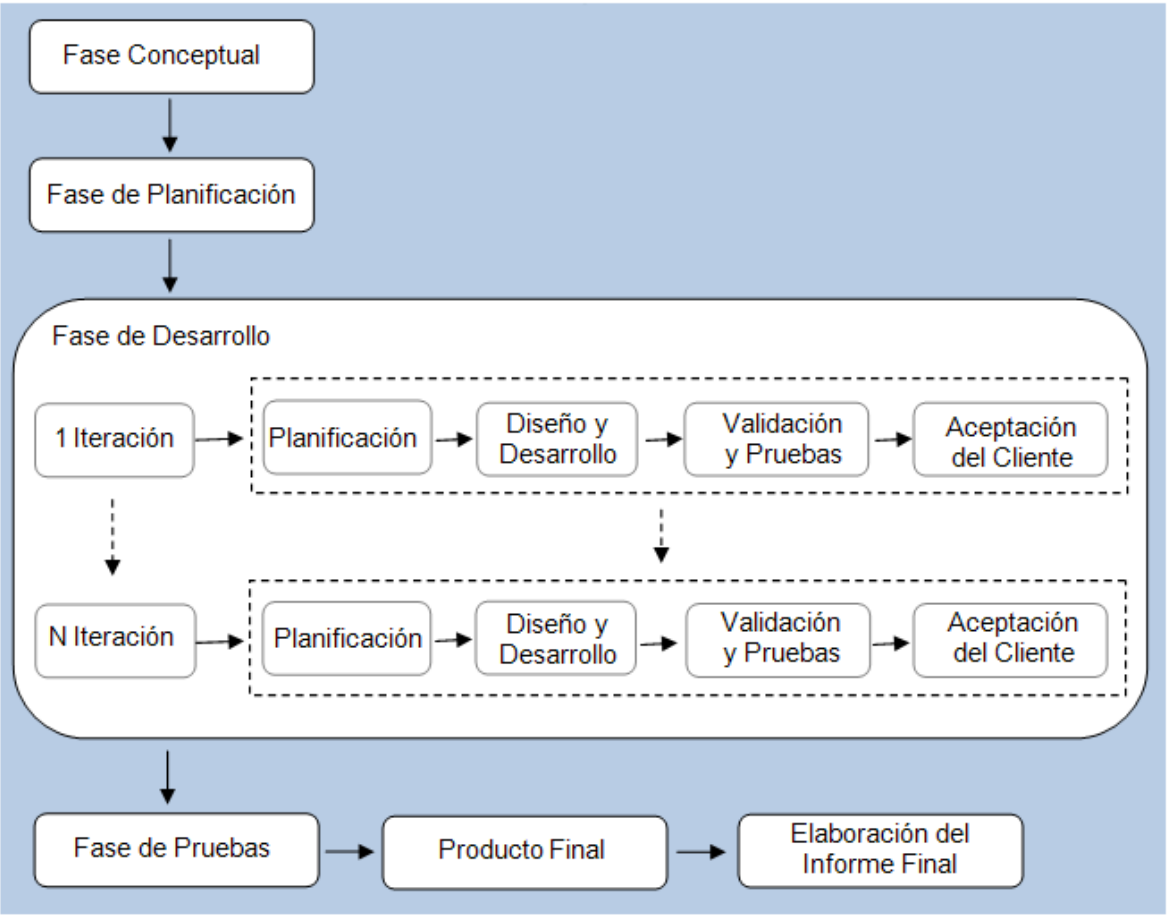

Figura 6. Proceso de Investigación

4. DESARROLLO DE LA VISTA GRÁFICA PARA APE

4.1. FASE CONCEPTUAL

Los resultados obtenidos en la fase conceptual están definidos en el Capítulo N°1 con la descripción de la Investigación, la definición de los antecedentes del proyecto, el planteamiento del problema, la justificación, el impacto y viabilidad de la propuesta. En el Capítulo N° 2 con la definición del Marco Teórico como resultado del estudio y análisis de la fundamentación teórica necesaria para el desarrollo de la vista gráfica en la herramienta Agent Tool Process Editor (APE).

4.2. FASE DE PLANIFICACIÓN

4.2.1. Definición de actores o roles

Un actor es una entidad externa que realiza algún tipo de interacción con la herramienta o el sistema que se está desarrollando. En el desarrollo de la vista gráfica para la herramienta Agent Tool Process Editor (APE) se han identificado dos actores: el primer actor corresponde al director de proyecto y el segundo actor se le denomina programador.

El director de proyecto corresponde a la persona encargada de hacer uso de la vista gráfica, para apoyar sus labores de control y seguimiento en la construcción de un Sistema Multiagente (SMA), y el actor programador corresponde a la empresa o persona encargada de realizar cambios o mejoras en la herramienta con el fin de adaptarla a sus necesidades, al ser considerada como una herramienta de Software Libre. En la Tabla 1 se especifican los actores establecidos para el manejo de la vista gráfica y su respectiva funcionalidad:

ACTOR	DESCRIPCIÓN	RESPONSABILIDAD (Papel que desempeña)	NECESIDAD (Utilización de la Vista Gráfica)
Director de Proyecto	*Representa a la persona encargada de realizar el proceso control y seguimiento de los diferentes proyectos.*	*Controlar el desarrollo y el avance de las diferentes actividades o tareas que conforman el proyecto.* *Registrar la entrega de actividades en la*	*Utiliza la vista gráfica para conocer y analizar más fácilmente el estado actual del desarrollo de un proyecto y determinar posibles atrasos o sobrecostos.*

		herramienta. *Analizar los resultados obtenidos de aplicar la técnica del Earned Value Analysis en el control y seguimiento de los proyectos.*	*Utiliza la vista para representar gráficamente los resultados obtenidos de aplicar la técnica del Earned Value Analysis en el control y seguimiento de los proyectos.*
Progra_ mador	*Representa a la persona encargada de modificar o incluir mejoras a la vista gráfica desarrollada.*	*Implementa nuevas funcionalidades para facilitar el control y seguimiento de los proyectos por parte de sus directores.*	*Estudia y analiza el código fuente de la vista gráfica para entender su correcto funcionamiento y poder implementar nuevas funcionalidades.*

Tabla 4. Definición de actores

4.2.2. Diagrama de Casos de Uso

Los diagramas de casos de uso son considerados como un diagrama de comportamiento, que nos indica cómo debería interactuar el sistema con el usuario o con otro tipo de sistema para conseguir los objetivos propuestos. En la Figura 7 podemos observar el diagrama de casos de uso que ha sido establecido para el desarrollo de la vista gráfica en APE.

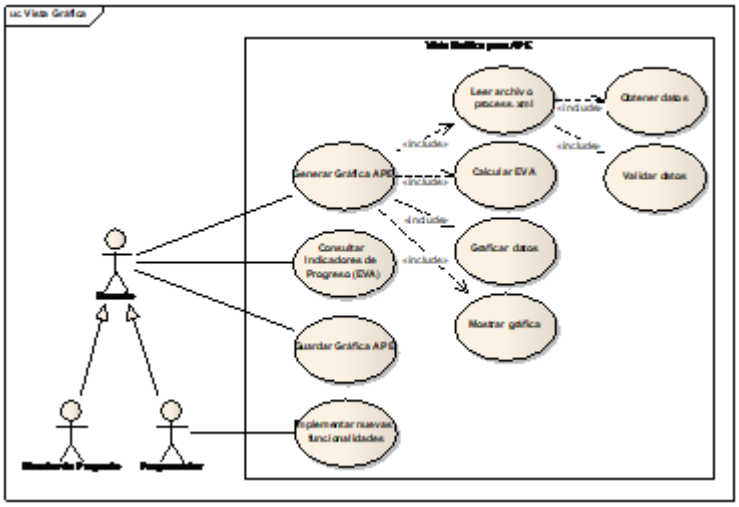

Figura 7. Diagrama de Casos de Uso

4.2.3. Especificación de Requisitos

La Especificación de Requisitos consiste en la descripción completa del comportamiento de la vista gráfica a desarrollar. Estos requisitos se clasificaron en requisitos de la interfaz de usuario y en requisitos funcionales y no funcionales.

- **Requisitos de Interfaz de Usuario**

La interfaz de usuario consiste en la creación de una nueva vista en la herramienta agent Tool Process Editor (APE), que estará conformada por las siguientes dos secciones: la primera sección mostrará los indicadores de progreso obtenidos de aplicar la técnica del Earned Value Analysis (EVA) en el proyecto y por un botón que permitirá realizar la función de spline sobre la gráfica generada, la segunda sección será la encargada de mostrar la gráfica generada para el análisis del costo y el avance del proyecto.

- **Requisitos Funcionales**

Los requisitos funcionales son los que determinan la funcionalidad que debe proporcionar al usuario la vista gráfica a desarrollar. Estos requisitos son:

REQF1: Verificar la existencia del archivo process.xml para el proyecto en revisión.
REQF2: Informar al usuario si no es posible acceder al archivo process.xml del proyecto.
REQF3: Leer los datos del archivo process.xml.
REQF4: Verificar que el tipo de dato para los valores estimados y reales de las actividades, corresponde con el tipo de dato requerido para el proceso.
REQF5: Calcular las medidas básicas e indicadores de progreso según la técnica del Earned Value Analysis (EVA).
REQF6: Determinar los valores a graficar para conocer el comportamiento del proyecto en relación a las variables de análisis BCWS, BCWP y ACWP.
REQF7: Graficar los valores establecidos en función del tiempo para el análisis de las variables BCWS, BCWP y ACWP.
REQF8: Mostrar al usuario el valor de los indicadores de progreso.
REQF9: Mostrar al usuario la gráfica obtenida para el análisis de las variables BCWS, BCWP y ACWP.
REQF10: Permitir al usuario visualizar los puntos graficados en cada variable de análisis.
REQF11: Realizar la interpolación de datos para suavizar la grafica generada.
REQF12: Permitir al usuario guardar la gráfica generada para el proyecto en revisión.
REQF13: Permitir al usuario ejecutar la vista gráfica desde la vista Process Management.

- **Requisitos No Funcionales**

Los requisitos no funcionales, son aquellos que describen las facilidades que debe proporcionar la vista gráfica, estos requisitos son:

REQNF1: Facilidad de uso por parte de los usuarios.
REQNF2: Ejecución de todos sus procesos de manera satisfactoria sin presentar interrupciones de manera abrupta.
REQNF3: Portabilidad. Gracias al lenguaje de programación utilizado, es portable a cualquier plataforma.
REQNF4: Tiempo de respuesta. Está determinado por la capacidad del procesador donde se ejecuta las funciones de la librería y el número de actividades que conformen el proyecto.
REQNF5: Escalabilidad. La vista gráfica debe estar en capacidad de permitir incluir, modificar o eliminar nuevas funcionalidades después de su construcción y puesta en marcha inicial.
REQNF6: El código fuente debe estar debidamente documentado.
REQNF7: Disponibilidad de ejecución de la vista gráfica a partir de la vista Process Management.

4.2.4. Arquitectura de Desarrollo

La arquitectura seleccionada para el desarrollo de la vista gráfica es la denominada arquitectura de 3 capas y un nivel. Es una arquitectura que tiene como objetivo primordial diferenciar la capa de presentación, con la capa de la lógica del negocio y la capa de datos; y se denomina de 1 nivel porque la solución de 3 capas reside en un sólo ordenador. (Ver Figura 8).

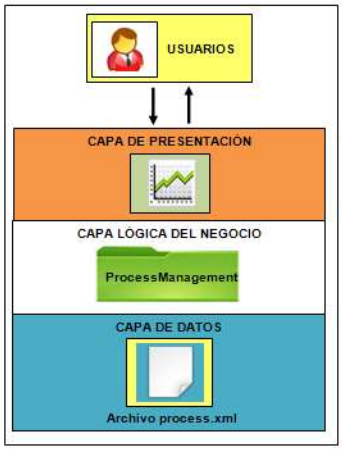

Figura 8. Arquitectura de 3 capas

La capa de presentación, es la capa que ve el usuario y por la cual se le comunica o se le captura información, también se le conoce como interfaz gráfica y se comunica únicamente con la capa del negocio.

La capa de la lógica del negocio, es donde se establecen las reglas que deben cumplir para dar solución a las necesidades o requisitos planteados. En esta capa se reciben las peticiones del usuario y se envían las respuestas tras la ejecución de los procesos, es una capa que se comunica con la capa de presentación y la capa de datos.

La capa de datos, es donde residen o se almacenan los datos. Es la capa que se reciben las solicitudes de almacenamiento o recuperación de la información desde la capa del negocio.

Es importante tener en cuenta que las capas definidas en la arquitectura son simplemente agrupaciones lógicas de los componentes de software que conforman la aplicación. Ayudan a diferenciar entre los distintos tipos de tareas que realizan los componentes o clases, facilitando el diseño de la reutilización en la solución.

4.2.5. Definición del Entorno Tecnológico

A continuación se presentan las diferentes tecnologías software seleccionadas para el desarrollo de la vista gráfica en la herramienta APE.

- Java Rutine Environment (JRE).
- Entorno de programación Eclipse.
- Plug-in agentTool III (aT3).
- Herramienta agentTool Process Editor (APE).
- Lenguaje de programación JAVA.
- Kit de herramientas SWT (Standard Widget Toolkit)
- Librería O-MASE y JFreeChart.
- Suite Ofimática OpenOffice.org.
- Compresor y descompresor de archivos 7 zip.
- Navegador Web Mozilla Firefox.

4.3. FASE DE DESARROLLO

4.3.1. Instalación y Configuración del Entorno de Desarrollo

4.3.1.1. Instalación del Java Rutine Environment (JRE)

El JRE es un conjunto de utilidades y componentes que permiten la ejecución de aplicaciones Java, su descarga e instalación es muy sencilla y gratuita. Para su descarga e instalación podemos acceder al enlace http://www.java.com/es/ realizando clic en el botón ***Descarga gratuita de Java***, luego aparecerá un cuadro de diálogo que permitirá ***ejecutar*** o ***guardar*** el archivo descargado. Para iniciar su instalación realizamos clic en el botón ***Ejecutar*** de lo contrario realizamos clic en el botón ***Guardar*** y después realizamos su instalación.

Para comprobar que Java se ha instalado y funciona correctamente en el equipo ejecutamos el ***applet de prueba*** que se encuentra disponible en el siguiente enlace: http://www.java.com/es/download/help/windows_manual_download.xml, este applet nos indicará si el proceso se ha realizado satisfactoriamente mediante la visualización de la siguiente imagen:

Figura 9. Comprobación de instalación del JRE

4.3.1.2. Instalación de la Plataforma Eclipse Ganymede

La instalación del entorno de desarrollo en su versión 3.4, es un proceso fácil de realizar, sólo basta con acceder al enlace http://www.eclipse.org/ y realizar la descargar de la versión de nuestro interés, para nuestro caso accedemos al siguiente enlace: http://www.eclipse.org/ganymede/. Una vez elegida la versión, elegimos el lugar donde queremos realizar la descarga y guardamos el archivo en nuestro disco duro. Ahora sólo resta descomprimir el archivo y acceder al ejecutable (*eclipse o elcipse.exe*) para ejecutar nuestro entorno de desarrollo. (Ver Figura 10).

Componentes del workbench de Eclipse.

- ***SWT (Estándar Widget Toolkit):*** Es el conjunto de elementos gráficos a bajo nivel.
- ***JFace:*** Es el armazón básico para las funcionalidades básicas UI.
- ***Workbench:*** Es el aspecto UI de la plataforma Eclipse.

Terminología del workbench de Eclipse. (Ver Figura 10).

1. Barra de Menús
2. Barra de Herramientas
3. Barra de Perspectiva.
4. Vista de Recursos
5. Editor de texto
6. Vista de Esquema
7. Vista de Tareas
8. Área de Mensajes

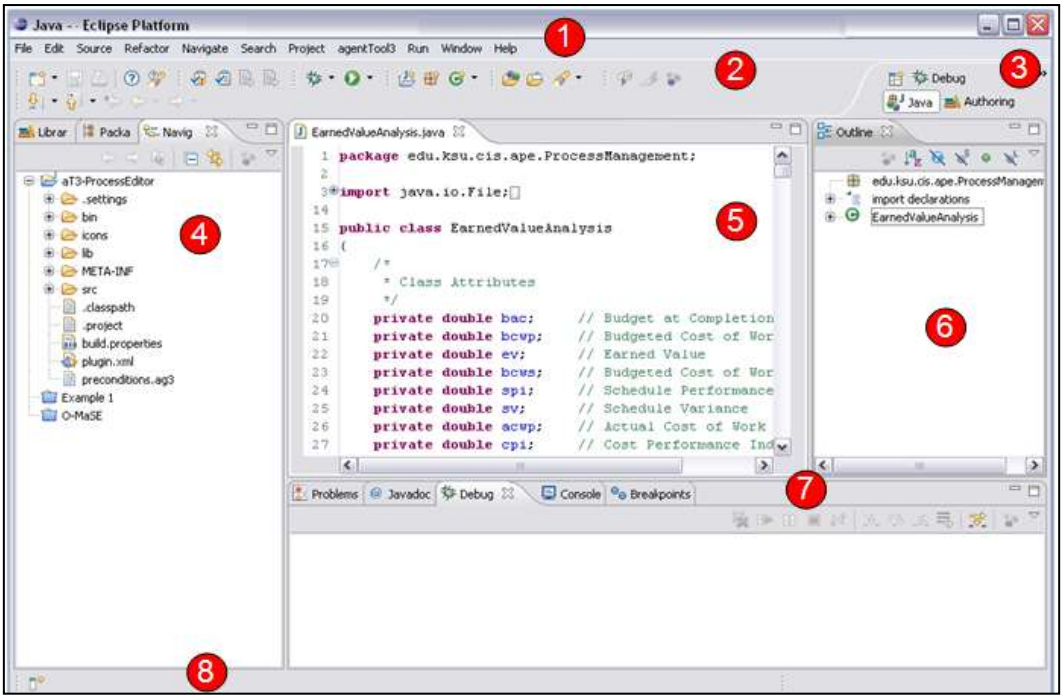

Figura 10. Terminología del workbench de Eclipse

4.3.1.3. Instalación de plug-in agentTool III (aT3)

El proceso de instalación de aT3 en la plataforma Eclipse la realizamos desde un servidor remoto, realizando las siguientes instrucciones:

- Ejecutar la plataforma Eclipse Ganymede.
- Ingresar al menú *Help/Software Updates/Avalible Software/Add Site* para especificar la dirección del servidor remoto de descarga. Sitio web de descarga: http://agenttool.cis.ksu.edu/update/. (Ver Figura 11).
- Realizar clic el botón *Refresh*, seleccionar los items a instalar en la plataforma y realizar clic en el botón *Install*. (Ver Figura 12).
- Reiniciar la plataforma Eclipse una vez se ha finalizado el proceso de instalación de aT3.

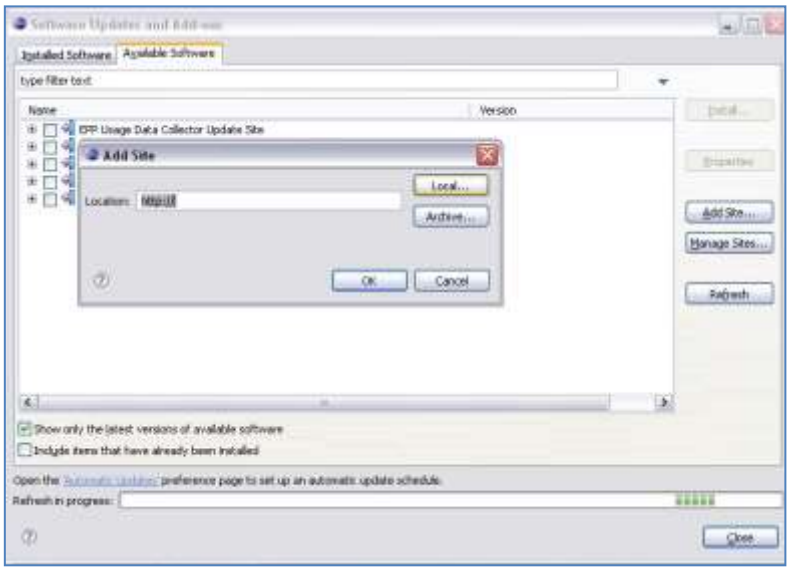

Figura 11. Especificación del sitio de descarga para APE

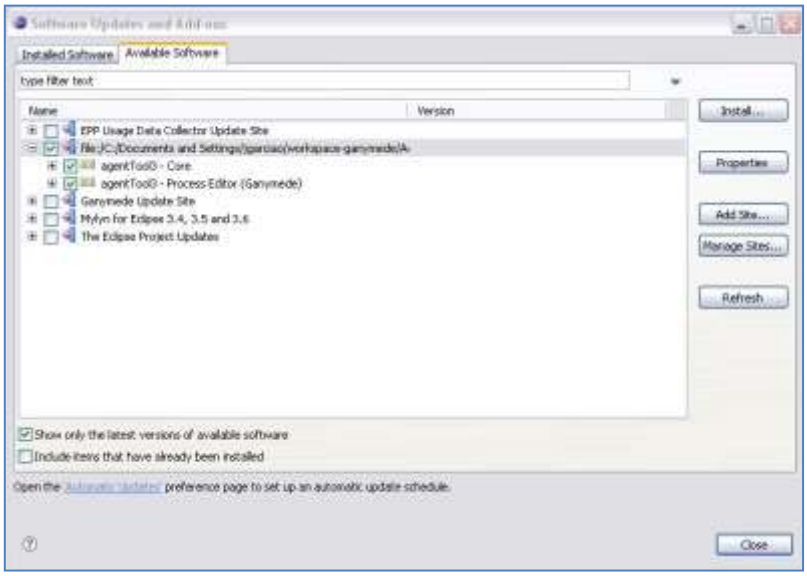

Figura 12. Componentes de instalación de aT3

4.3.1.4. Importar el proyecto agentTool Process Editor (APE)

Para importar el proyecto agentTool Process al workspace de la plataforma Eclipse se realiza clic en el menú *File/Import/Existing Projects*, como se muestra en la Figura 13:

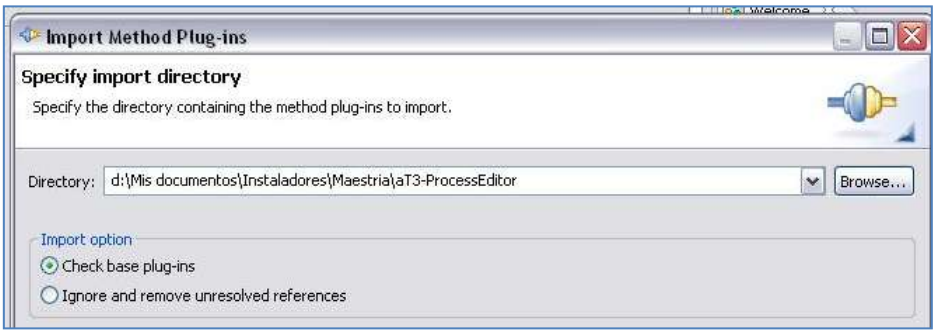

Figura 13. Importar proyecto APE

Una vez importado el proyecto APE al entorno de desarrollo es posible visualizar su estructura de paquetes, como se visualiza en la Figura 14:

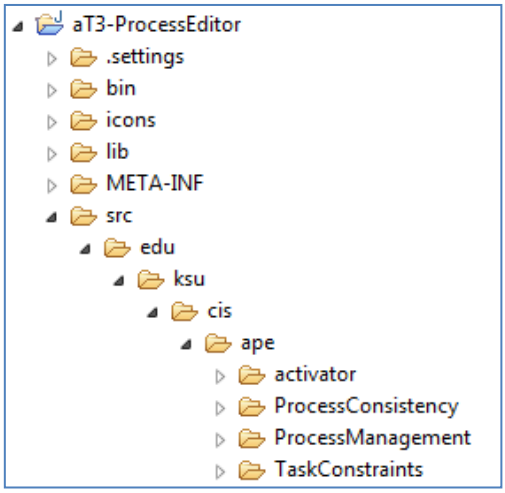

Figura 14. Estructura del proyecto APE

4.3.1.5. Relacionar librería O-MASE

La librería O-MASE se define como un conjunto de funciones que implementan la metodología O-MASE para facilitar la construcción de un Sistema Multiagente (SMA). Esta librería nos proporciona un proyecto de Ejemplo con el cual podemos realizar las respectivas pruebas que garanticen el correcto funcionamiento de la vista gráfica en la herramienta APE. Para relacionar la librería en la plataforma Eclipse realizamos las siguientes instrucciones: (Ver Figura 15).

- Abrir la vista library mediante el menú: ***Windows/Show view/other/Method Authoring/library*.**

- Relacionar librería mediante el menú: ***File/Open/Method Library/(ruta de la librería O-MASE)***.

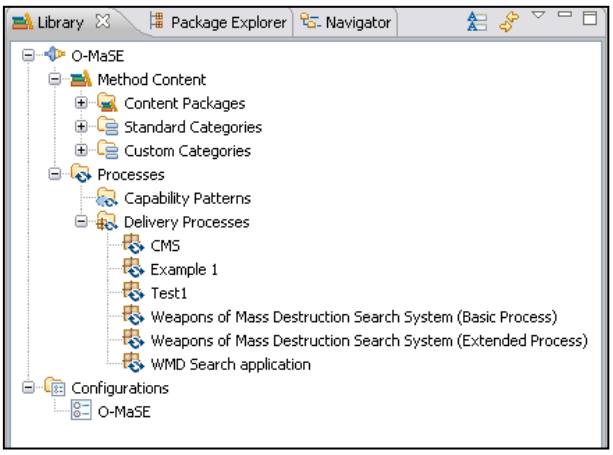

Figura 15. Relacionar Librería O-MASE

4.3.1.6. Relacionar librería Jfreechart y Jcommon.

Para trabajar con JFreeChart necesitamos descargar la librería jfreechart, la cual incluye y requiere de la librería JCommon. Desempaquetamos el .zip y sólo necesitamos los .jar de ***jfreechart-1.0.13*** y ***jcommon-1.0.16*** (o las versiones descargadas). Estos .jar se encuentran en el subdirectorio ***lib*** y procedemos a relacionar las librerías en el archivo ***MANIFEST.MF*** del plug-ins, en la pestaña ***Runtime***, como se muestra en la Figura 16:

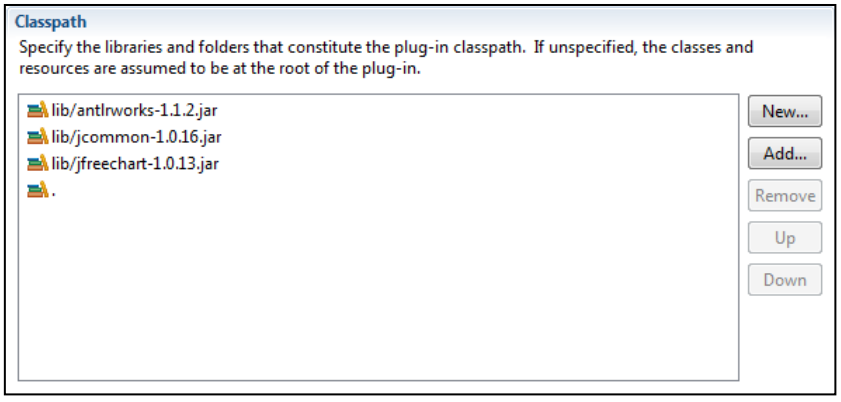

Figura 16. Relacionar librería Jfreechart y Jcommon en plug-ins

Posteriormente relacionamos las librerías en el proyecto aT3 Process Editor, seleccionando el proyecto en la vista ***Navigator***, realizando clic derecho y seleccionando la opción ***Properties/Java Build Path/Libraries***. (Ver Figura 17).

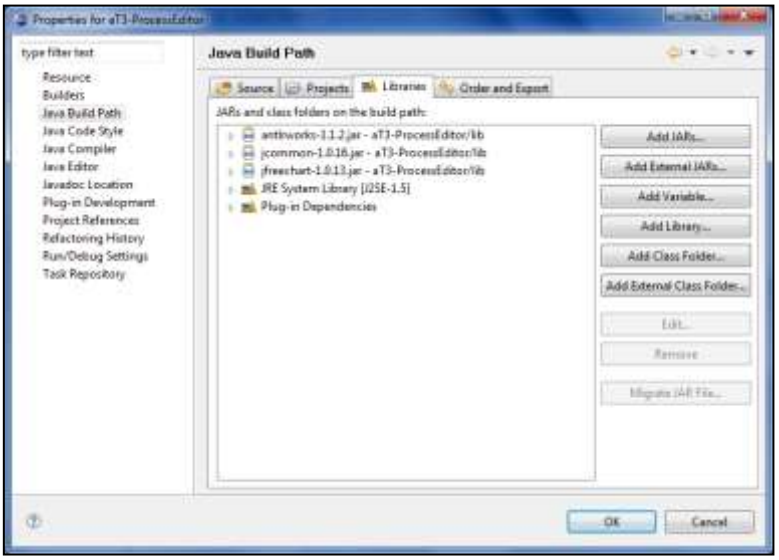

Figura 17. Relacionar librería Jfreechart y Jcommon en el proyecto aT3 Process Editor.

Posteriormente relacionamos las librerías en el proyecto aT3 Process Editor, seleccionando el proyecto en la vista ***Navigator***, realizando clic derecho y seleccionando la opción ***Properties/Java Build Path/Libraries***. (Ver Figura 19).

4.3.1.7. Crear una configuración para ejecutar y depurar la aplicación.

La configuración a realizar permitirá detectar y corregir errores en el proyecto aT3-ProcessEditor, así como verificar el correcto funcionamiento de los cambios o modificaciones realizados en el código fuente. Para crear esta configuración seleccionamos el menú ***Run/Debug Configuration/*** y en la ventana emergente que se nos muestra por defecto realizamos doble clic en la opción ***Eclipse Aplication*** e ingresamos un nombre para la configuración, especificamos el JRE con el cual vamos a ejecutar o depurar la aplicación, y en la pestaña ***Common*** en la opción ***Display in favorites menú*** seleccionamos la opción ***Debug,*** y para finalizar seleccionamos el botón ***Apply*** y ***Close***. (Ver Figura 18).

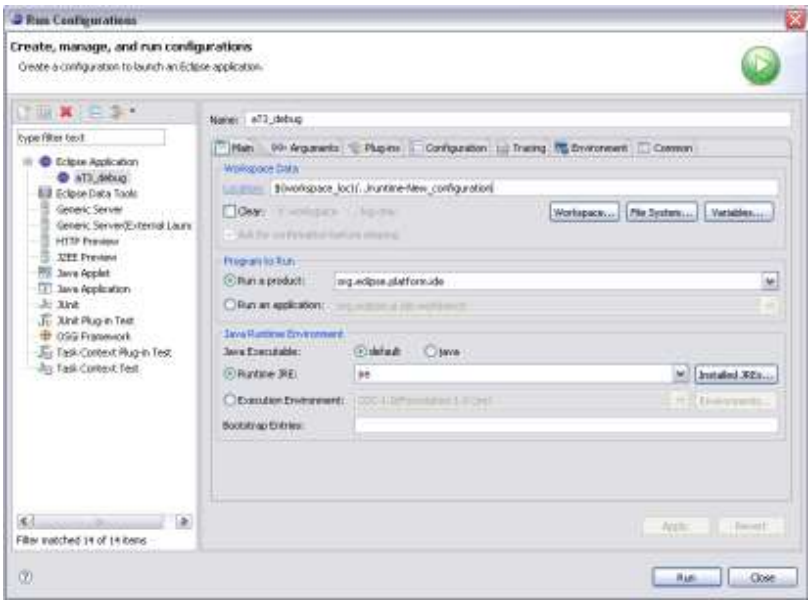

Figura 18. Configuración para ejecutar y depurar la aplicación.

4.3.2. Estándar de codificación

El estándar de codificación utilizado para el desarrollo de la vista gráfica en la herramienta agentTool Process Editor (APE) es el ***Java Language Specification*** establecido por Sun Microsystems para el lenguaje de programación JAVA, el cual puede ser consultado en el siguiente enlace: http://java.sun.com/docs/books/jls/second_edition/html/jTOC.doc.html.

4.3.3. Diagrama de clases

El diagrama de clases es un diseño que captura la estructura lógica del sistema y el comportamiento que tiene la implementación a realizar, es un diagrama que nos describe las clases a implementar con sus atributos, métodos y relaciones. En la Figura 19 podemos observar el diagrama de clases que ha sido definido para el desarrollo de la vista gráfica en APE:

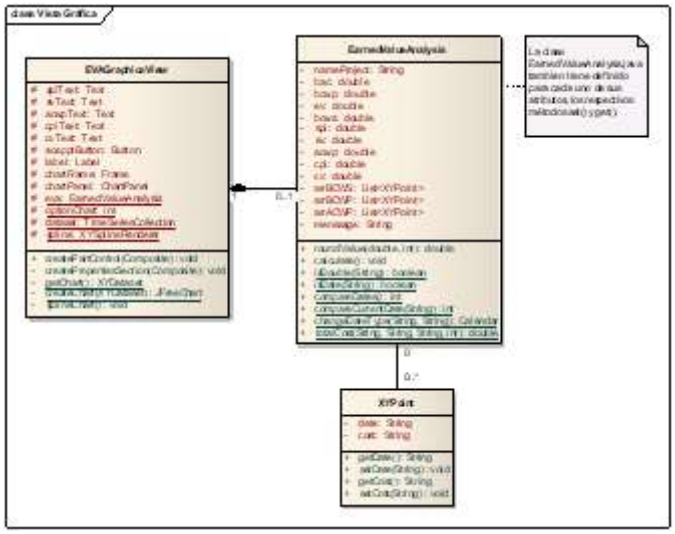

Figura 19. Diagrama de clases.

En el diagrama de clases la relación de *agregación* es un tipo de asociación que nos indica que un elemento contiene o está compuesto por otros elementos, es decir, la clase EVAGraphicsView está compuesta por un elemento de la clase EarnedValueAnalysis. La relación de asociación entre la clase EarnedValueAnalysis y la clase XYPoint nos indica que los dos elementos del modelo tienen una relación, usualmente implementada como una variable de instancia en una clase.

4.3.4. Implementación de Requisitos

4.3.4.1. Adicionar nueva vista al plug-ins

Las vistas son elementos fundamentales de la ventana de trabajo de eclipse, que se basan en la clase abstracta Viewer. Para adicionar una nueva vista al plug-in debemos realizar:

✓ **Adicionar una extensión de la vista en el archivo plugin.xml.**

Para adicionar una vista en el archivo plugin.xml, se deben especificar los siguientes atributos básicos:

- **category:** para organizar las vistas
- **id:** cada vista necesita un identificador único
- **name:** nombre con el que aparecerá la vista
- **class:** clase que implementa la vista
- **icon:** icono de la vista

El siguiente código XML nos muestra la adición de la nueva vista en el plug-ins:

```
<view name="APE - EVA Graphics"
      icon="icons/Task.png"
      class="edu.ksu.cis.ape.ProcessManagement.EVAGraphicsView"
      id="edu.ksu.cis.ape.ProcessManagement.EVAGraphicsView">
</view>
```

✓ **Definir una clase para la vista de extensión en el Plug-in.**

Ahora tenemos que definir una clase que implemente la mayor parte del comportamiento predeterminado de una vista, al heredar de la clase abstracta Viewer. Esta clase debe implementar del método createPartControl el cual será ejecutado por la plataforma y donde se podrán crear cualquier número de controles SWT para cumplir con los requisitos establecidos para la interfaz de usuario. La clase definida para la vista de extensión es la clase EVAGraphicsView, la cual se puede observar en el diagrama de clases. (Ver Figura 18).

4.3.4.2. Creación de widgets de SWT

SWT (Standard Widget Toolkit) [21] es un kit de herramientas gráficas para trabajar bajo la plataforma de Java desarrollado por IBM y actualmente es mantenido por la fundación Eclipse. Los programas que usan SWT son portables, pero la implementación del kit de herramientas, a pesar de estar escrita en Java, es diferente para cada plataforma. Algunos widgets de SWT son:

Figura 20. SWT Widgets

4.3.4.3. Creación de la gráfica con JFreeChart

Para la creación de una gráfica con JFreeChart, se deben realizar los siguientes pasos básicos:

- **Crear un dataset que contenga los datos a mostrar en las gráficas**

Para crear el dataset con los datos a graficar realizamos el siguiente proceso:

✓ Establecer para cada variable de análisis su respectiva serie de datos (X, Y) para graficar.

```
TimeSeries serieBCWS = new TimeSeries("BCWS");
```

✓ Añadir la pareja de datos (X, Y) a la serie respectiva.

```
serieBCWS.addOrUpdate(new Day(day,month,year), cost);
```

✓ En la librería JFreeChart, se tiene la siguiente clase que representa el conjunto de datos.

```
TimeSeriesCollection dataset = new TimeSeriesCollection();
```

✓ Añadir en el dataset las diferentes series de datos a graficar.

```
dataset.addSeries(serieBCWS);
```

- **Crear una instancia de la clase JFreeChart que será la responsable de dibujar las gráficas.**

✓ El objeto que representa a una gráfica en JFreeChart, se crea instanciando una clase de tipo JFreeChart, de la siguiente manera:

```
JFreeChart chart = ChartFactory.createTimeSeriesChart(
                   "Project: "+nameProject, // Title
                   "TIME (t)",              // x-axis label
                   "COST ($)",              // y-axis label
                   dataset,                 // data
                   true,                    // Create legend?
                   true,                    // Generate tooltips?
                   false                    // Generate URLs?
);
```

✓ Para modificar el cuadro de la gráfica generado, se obtiene un objeto de la clase CategoryPlot.

```
XYPlot plot = (XYPlot) chart.getPlot();
```

✓ JFreeChart nos ofrece una clase específica para modificar el estilo de las gráficas de línea. Esta clase es uno de los atributos de nuestro objeto plot:

```
XYItemRenderer r = plot.getRenderer();
if (r instanceof XYLineAndShapeRenderer)
{
    XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer) r;
    renderer.setBaseShapesVisible(true);
    renderer.setBaseShapesFilled(true);
    renderer.setSeriesPaint(0, Color.red);
    renderer.setSeriesPaint(1, Color.blue);
    renderer.setSeriesPaint(2, Color.black);
}
```

- **Dibujar o imprimir la gráfica sobre algún objeto de la vista que lo permita visualizar.**

```
final Composite chartComposite = new Composite(chartSection, SWT.NONE | SWT.EMBEDDED);
chartComposite.setLayout(new GridLayout(1, false));
chartSection.setClient(chartComposite);
chartFrame = SWT_AWT.new_Frame(chartComposite);
chartPanel = new ChartPanel(chart);
chartFrame.add(chartPanel);
```

4.3.4.4. Ejecución de la vista gráfica a partir de vista Process Management

Para la ejecución de la vista gráfica a partir de la partir de vista Process Management re realizaron los siguientes procesos:

- Creación de un icono en la barra de herramientas de la vista Process Management, para la ejecución de la vista gráfica, y creación del mismo icono para ser mostrado en el momento de seleccionarse el proyecto en revisión y realizar clic derecho con el mouse. (Ver Figura 21).

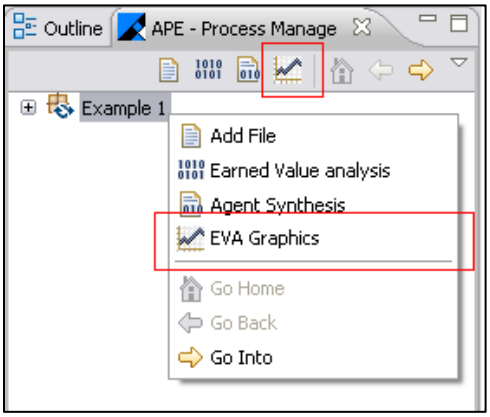

Figura 21. Vista APE - Process Management

- Implementación del método para la captura del evento de realizar clic sobre el nuevo icono creado, y para la ejecución del proceso que permite ejecutar la vista gráfica de APE.

```
action4 = new Action(){
    public void run(){
        IWorkbench workbench = PlatformUI.getWorkbench();
        IWorkbenchWindow workbenchWindow = workbench.getActiveWorkbenchWindow();
        try {
            workbenchWindow.getActivePage().showView("edu.ksu.cis.ape.ProcessManagement.EVAGraphicsView");
        } catch (Exception e) {
            System.out.println("Error en action4 run(): "+e);
        }
    }
};
```

4.3.4.5. Clases y principales métodos implementados en el paquete ProcessManagement.

- **Clase EVAGraphicsView.java**

EVAGraphicsView.java	
Método:	*createPartControl(Composite parent)*
Descripción:	*Método que crea la instancia intermedia denominada Composite, la cual permitirá organizar los componentes que harán parte de la vista gráfica.*
Parámetros:	*Recibe una instancia composite.*
Valor devuelto:	*Ninguno.*
Método:	*createPropertiesSection(**final** Composite parent)*
Descripción:	*Método que crea los diferentes componentes que harán parte de la vista gráfica.*
Parámetros:	*Recibe una instancia composite.*
Valor Devuelto:	*Ninguno.*
Método:	*getChart()*
Descripción:	*Método que permite obtener la gráfica que ha sido creada para el control y seguimiento del proyecto.*
Parámetros:	*Ninguno.*
Valor Devuelto:	*JFreeChart.*
Método:	*createDataset()*
Descripción:	*Método que determina el conjunto de datos X,Y que se deben graficar para análisis de las variables BCWS, BCWP y ACWP.*
Parámetros:	*Ninguno*
Valor Devuelto:	*XYDataset.*
Método:	*createChart(XYDataset dataset)*
Descripción:	*Método que grafica el conjunto de datos establecidos para análisis de las variables BCWS, BCWP y ACWP.*
Parámetros:	*Recibe el XYDataset*
Valor Devuelto:	*JFreeChart*
Método:	*splineChart()*
Descripción:	*Método que permite suavizar la gráfica generada mediante la interpolación de los datos.*
Parámetros:	*Ninguno.*
Valor Devuelto:	*Ninguno.*

Tabla 5. Clase EVAGraphicsView.java

- **Clase EarnedValueAnalysis**

EarnedValueAnalysis.java	
Método:	*calculate()*
Descripción:	*Método para realizar el cálculo de las medidas básicas e indicadores de la Técnica Earned Value Analysis y asigna los resultados a los atributos de la clase por medio de los métodos set allí definidos.*
Parámetros:	*Ninguno.*
Valor devuelto:	*Ninguno.*
Método:	*isDouble(String value)*
Descripción:	*Método para verificar si el valor recibido como parámetro corresponde a un valor tipo double.*
Parámetros:	*Recibe el valor a verificar.*
Valor Devuelto:	*Boolean. Es true si la verificación es correcta y false si la verificación no lo es, o si el método presentó errores durante su ejecución.*
Método:	*isDate (String value)*
Descripción:	*Método para verificar si el valor recibido como parámetro corresponde a un valor tipo date.*
Parámetros:	*Recibe el valor a verificar.*
Valor Devuelto:	*Boolean. Es true si la verificación es correcta y false si la verificación no lo es, o si el método presentó errores durante su ejecución.*
Método:	*compareDates(String date, String dateTask)*
Descripción:	*Método para comparar dos fechas y determinar si la primer fecha recibida como parámetro es mayor, menor o igual que la segunda fecha recibida como parámetro.*
Parámetros:	*Recibe las fechas a comparar.*
Valor Devuelto:	*1 Si la fecha es mayor, -1 si la fecha es menor y 0 si las fechas son iguales.*
Método:	*compareCurrentDate(String date)*
Descripción:	*Método para comparar la fecha recibida como parámetro con la fecha actual.*
Parámetros:	*Recibe la fecha a comparar.*
Valor Devuelto:	*1 Si la fecha es mayor, -1 si la fecha es menor y 0 si las fechas son iguales.*
Método:	*changeDateType(String date, String optionMonth)*
Descripción:	*Método para modificar el formato de la fecha recibida como*

	parámetro.
Parámetros:	*Recibe la fecha a modificar*
Valor Devuelto:	*Calendar con la fecha modificada.*
Método:	*totalCost(String date, String cost, String option,* ***int*** *itemNumber)*
Descripción:	*Método para calcular el valor del costo total acumulado hasta la fecha real de entrega de la actividad, recibida como parámetro.*
Parámetros:	*Recibe la fecha real de entrega de la actividad, el costo de la actividad y el número que identifica la posición de la actividad en el archivo process.xml.*
Valor Devuelto:	*Double.*

Tabla 6. Clase EarnedValueAnalysis.java

- **Clase XYPoint**

La clase XYPoint está conformada por los atributos date (fecha) y cost (costo), los cuales representan a la pareja de puntos (X, Y) a graficar, para el análisis de las variables BCWS, BCWP y ACWP para conocer el avance y estado actual del desarrollo de un Sistema Multiagente (SMA). Para asignar y acceder a los valores establecidos en la pareja de puntos (X, Y) se utilizarán los respectivos métodos set() y get() establecidos en la clase.

4.3.5. Pruebas de Funcionalidad.

Las pruebas de funcionalidad realizadas durante la fase de desarrollo de la vista gráfica son:

- Verificación de la existencia del archivo process.xml previamente a la lectura del archivo.
- Creación de métodos para la verificación del tipo de dato registrado en el archivo process.xml, para garantizar que corresponda a un tipo de dato válido para ser representado gráficamente.
- Aplicar la Técnica del Earned Value Analysis (EVA) de manera manual al proyecto de Ejemplo para comparar las medidas básicas e indicadores obtenidos, con los resultados generados mediante la ejecución de la vista gráfica.
- Modificar los datos registrados en el archivo process.xml para el proyecto de Ejemplo, con datos que no correspondan al tipo de dato establecido y verificar que los datos introducidos sean excluidos en la representación gráfica.

- Verificar que la vista gráfica informa al usuario mediante un mensaje los errores presentados durante la lectura del archivo process.xml.

4.3.6. Pruebas de Integración.

Las pruebas de integración a realizar durante la fase de desarrollo de la vista gráfica consistieron en la ejecución de la vista gráfica con proyectos de ejemplo para garantizar su correcto funcionamiento a partir de la vista Process Management y como parte integral de la herramienta Agent Tool Process Editor (APE).

4.4. FASE DE PRUEBAS

Las pruebas realizadas involucran principalmente las operaciones ejecutadas por la vista gráfica bajo ciertos datos y condiciones controladas, con el objetivo de evaluar los resultados obtenidos. Entre las diferentes pruebas realizadas tenemos:

Las pruebas de funcionalidad que se integraron dentro de las diferentes fases del desarrollo, con el objetivo de verificar la calidad del producto software y el cumplimiento de los requisitos establecidos mediante un monitoreo sobre el proceso realizado.

Las pruebas de integración se realizaron la final del desarrollo de la vista gráfica, las cuales garantizaron su correcto funcionamiento a partir de los diferentes puntos de ejecución establecidos.

Las pruebas de aceptación fueron las realizadas con el cliente para presentar y verificar las funcionalidades implementadas en la vista gráfica, mediante el análisis de los resultados y datos presentados, y mediante el cumplimiento de los objetivos propuestos.

4.5. PRODUCTO FINAL

El producto final que se obtiene con el desarrollo de esta propuesta, consiste en contar con un adecuado y correcto funcionamiento de la vista gráfica EVA – Graphics en la herramienta agentTool Process Editor (APE), que garantiza el cumplimiento de los requisitos establecidos en la fase de planificación.

Para ejecutar la vista gráfica de APE podemos realizar una de las siguientes opciones:

- Seleccionar el proyecto *en la vista APE - Process Management* y realizar clic sobre el icono ***EVA Graphics*** que se encuentra en la barra de herramientas de la vista *APE - Process Management.* (Ver Figura 22).

- *Seleccionar el proyecto en la vista APE - Process Management y realizar clic derecho con el mouse y seleccionar* la opción del menú ***EVA Graphics.*** (Ver Figura 22).

- *Seleccionar el proyecto en la vista APE - Process Management y* acceder a la vista ***EVA Graphics*** mediante el menú ***Windows/Show View/Other*** y desplegar el paquete ***Other*** para seleccionar la vista ***APE – EVA Graphics***, como se muestra en la Figura 23.

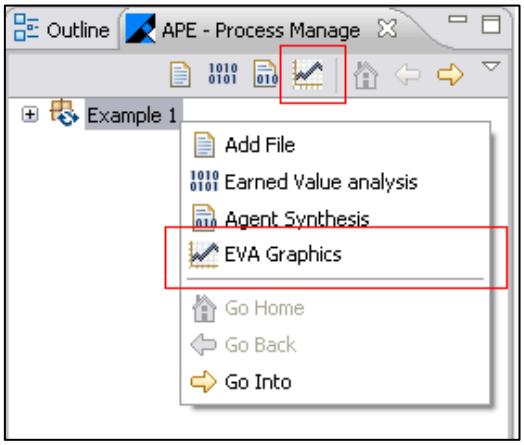

Figura 22. Vista APE - Process Management

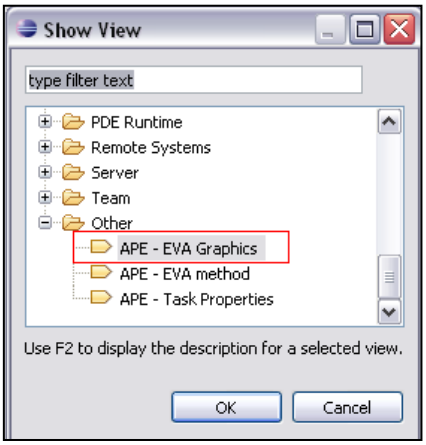

Figura 23. Show View: APE - Eva Graphics

Al realizar cualquiera de las opciones anteriores para la ejecución de la vista gráfica de APE, se podrá visualizar la siguiente información: una sección con los valores obtenidos para los indicadores de progreso al aplicar la Técnica del *Earned Value Analysis (EVA)* en el proyecto y un botón con el nombre *Spline* el cual permitirá suavizar la gráfica

generada, y una segunda sección donde se visualizará la gráfica generada para el proyecto, , la cual podremos guardar o imprimir al ubicarnos sobre la gráfica y realizar clic derecho con el mouse y seleccionar la opción de nuestro interés.

En la Figura 24 podemos observar una captura de pantalla, que nos ilustra el correcto funcionamiento de la vista gráfica en la herramienta APE:

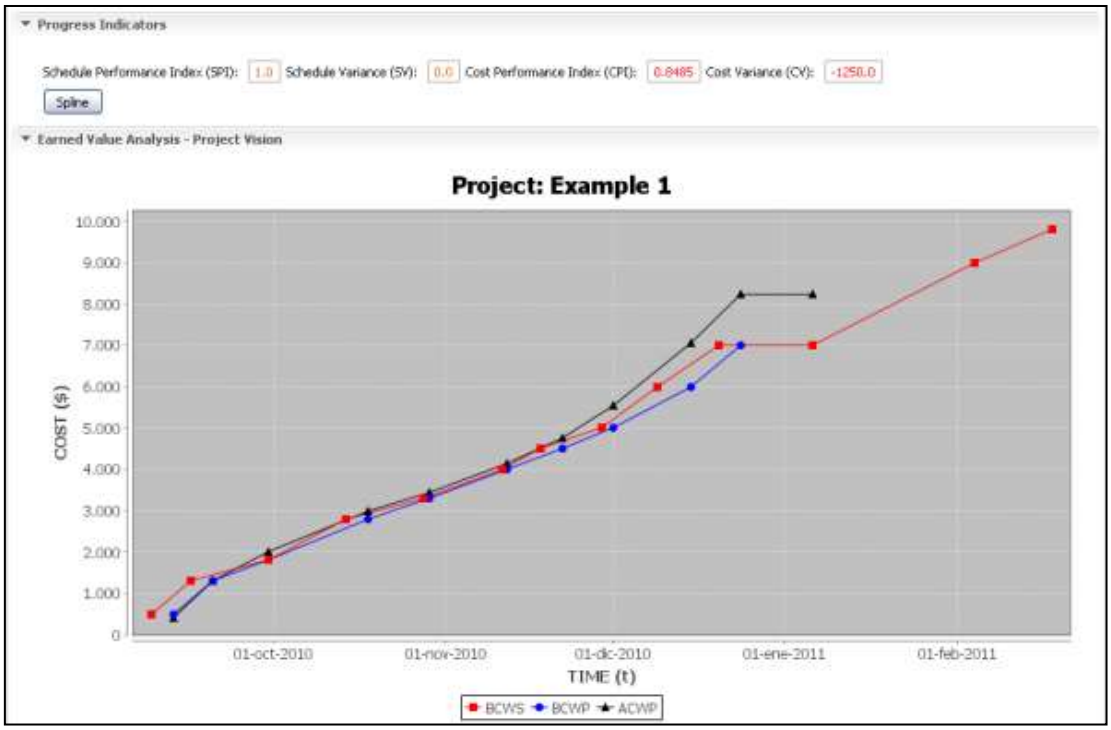

Figura 24. Vista Gráfica EVA - Graphics

4.6. ELABORACIÓN DEL INFORME FINAL

Esta fase finaliza con la elaboración del presente documento, el cual describe el proceso de desarrollo de la vista gráfica en la herramienta ***agentTool Process Editor (APE).***

5. TRABAJOS RELACIONADOS

En la actualidad existen herramientas CASE[13] o soluciones informáticas que apoyan la construcción de Sistemas Multiagente (SMA), teniendo como base el marco de trabajo que ha sido establecido por la metodología, esto con el fin de garantizar que el desarrollo de un producto software cumpla con los objetivos propuestos, el cronograma y los costos establecidos.

Entre las herramientas existentes encontramos:

- **Herramienta AgentTool [20]**

Esta herramienta soporta la mayoría de las fases que conforman el proceso de desarrollo de un Sistema Multiagente, mediante la metodología MaSE. En la fase de análisis de MaSE, se definen tres pasos básicos: la definición de objetivos, la especificación de casos de uso y el refinamiento de roles; en la fase de diseño se definen cuatro pasos: crear clases de agentes, construir conversaciones, ensamblar clases de agentes y el diseño del sistema. Mediante la herramienta también es posible generar el código fuente a partir de las especificaciones de diseño y se incorporan utilidades que permiten verificar la corrección de los protocolos que utilizan los agentes.

- **ZEUS [3]**

ZEUS también está conformado por una herramienta de apoyo y una metodología que propone el desarrollo de un sistema multiagente en cuatro etapas: el análisis del dominio, el diseño de los agentes, la realización de los agentes y soporte en tiempo de ejecución. La herramienta está constituida fundamentalmente por tres grupos funcionales: biblioteca de componentes de agentes, programas de construcción y agentes de utilidad, que permiten trasladar los conceptos de diseño para obtener como resultado un modelo ejecutable del sistema. (Ver Figura 22).

[13] Es un acrónimo para Computer-Aided Software Engineering. Esencialmente es una herramienta que ayuda al ingeniero de software desarrollar y mantener software.

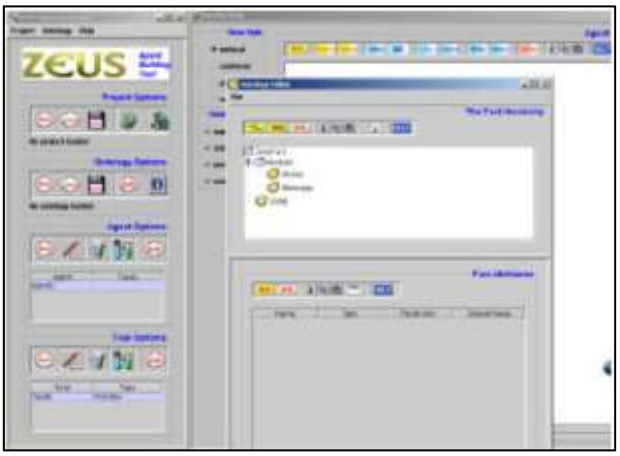

Figura 25. Entorno de desarrollo ZEUS

- **Entorno de desarrollo INGENIAS [6]**

La metodología INGENIAS y su IDE han sido desarrollados a partir de los resultados obtenidos en MESSAGE[14]. MESSAGE es una metodología que propone un lenguaje de especificación de sistemas multiagente mediante meta-modelos[15] y lenguaje natural, los cuales se integran en el ciclo de vida del proyecto mediante un conjunto de reglas y actividades definidas. (Ver Figura 23).

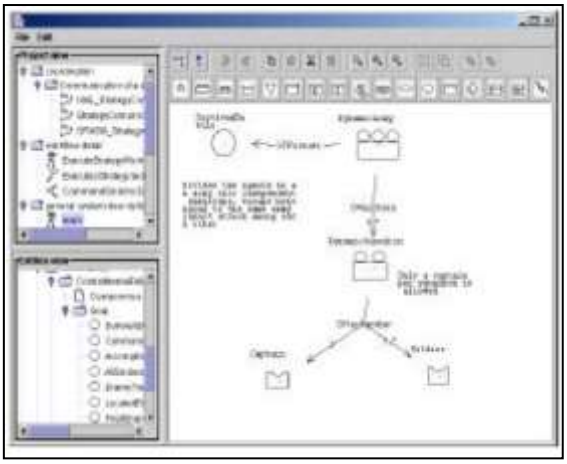

Figura 26. Herramienta de soporte para INGENIAS

[14] MESSAGE (Methodology for Engineering System of Software Agents) fue un proyecto que tuvo como objetivo extender las metodologías del software orientado a objetos para la realización de software orientado a agentes y como resultado del proyecto se elaboró esta nueva metodología, orientada al desarrollo de sistemas industriales de media o gran escala.

[15] Un meta-modelo es una representación de los tipos de entidades que pueden existir en un modelo, sus relaciones y restricciones de aplicación.

Aunque INGENIAS aborda con mayor profundidad los aspectos definidos por esta metodología, el IDE de INGENIAS incorpora una herramienta para la especificación y una para la generación de código o documentación, la herramienta de especificación se ha construido con la herramienta METAEDIT+ que procesa los meta-modelos que define INGENIAS y la generación de código se basa en la sustitución avanzada de texto, variables, repeticiones, etc.

- **Entorno de desarrollo PDT (Prometheus Design Tool)**

PDT [19] es una herramienta posee un ambiente de desarrollo muy completo y sencillo para la construcción de Sistemas Multi-agente, siguiendo la metodología Prometheus. Esta metodología ha sido creada por LIn Padgham y Michael Winikoff, en la cual definen un lenguaje de modelado que forma parte de los fundamentos de AUML 16 junto con otras metodologías. La metodología se puede simplificar en tres fases principales: Especificación del Sistema (proceso iterativo), Diseño de la Arquitectura, y Diseño Detallado del sistema. La herramienta soporta el desarrollo de agentes que contengan creencias, objetivos, planes y eventos, cuenta son diferentes artefactos de diseño para representar esquemas y diseñar múltiples niveles de abstracción.

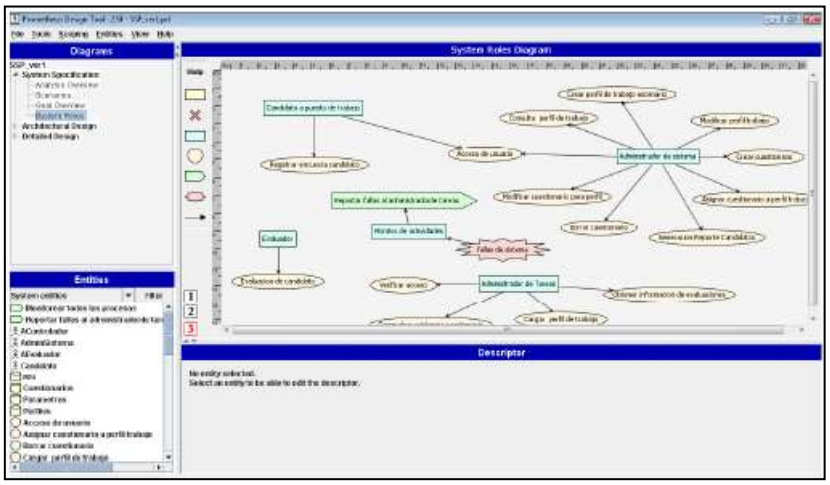

Figura 27. Entorno de desarrollo PDT

Otras herramientas que también nos sirven de apoyo en la construcción de Sistemas Multiagente (SMA) son el **Entorno de Desarrollo TAOM4E** y el **Entorno de Desarrollo APTK (**Ver Capítulo N° 2, Sección 2.2.4, página 20).

[16] AUML(AGENT UNIFIED MODELING LANGUAGE). Modelo unificado basado en agentes. http://www.auml.org/

Estas herramientas al igual que las mencionadas anteriormente, tienen como referencia un marco de trabajo que ha sido establecido por la metodología, pues ésta parte de un concepto distinto de agente y se especializa en áreas concretas de trabajo, facilitando la estructuración y planificación del proceso de desarrollo, pero no proporcionan ninguna funcionalidad que facilite a los directores de proyecto sus labores de control y seguimiento en cualquier fase del desarrollo de un proyecto.

En respuesta a estos inconvenientes se desarrolla la herramienta agentTool Process Editor (APE) para implementar la técnica del Earned Value Analysis (EVA) y así, facilitar las labores de control y seguimiento por parte de los directores de proyecto. Actualmente esta herramienta también pone a disposición de los directores de proyecto, una nueva funcionalidad producto del desarrollo de esta propuesta, la cual consiste en la creación de una nueva vista que permitirá representar gráficamente toda la información relacionada con el desarrollo de un Sistema Multiagente (SMA) en función del alcance, tiempo y costo del proyecto.

La vista gráfica permitirá a los directores de proyecto:

- Identificar de manera oportuna, los posibles cambios o variaciones que pueden afectar el normal desarrollo de un Sistema Multiagente (SMA).
- Implementar medidas correctivas o preventivas de manera oportuna.
- Identificar el avance del proyecto en cualquier fase de su desarrollo.
- Medir y analizar resultados de manera eficiente.
- Ajustar el proyecto con el presupuesto y tiempo establecido inicialmente.

Las funcionalidades que ofrece la herramienta agentTool Process Editor (APE) se integran fácilmente al entorno de desarrollo agentTool III (aT3), obteniéndose como resultado un conjunto de herramientas CASE que apoyan el proceso de desarrollo de software orientado a agentes.

6. CONCLUSIONES Y TRABAJO FUTURO

El desarrollo de la vista gráfica en la herramienta agentTool Process Editor (APE) ha permitido dar una visión diferente a los resultados obtenidos durante el proceso de control y seguimiento de un Sistema Multiagente (SMA) mediante de la técnica *Earned Value Analysis (EVA),* al implementar mecanismos que permiten representar gráficamente la gran variedad de información que gira alrededor del desarrollo de este tipo de proyectos, haciendo que su gestión, control y seguimiento sea más eficiente.

La representación gráfica de la información permite a los Directores de Proyecto analizar y comprender de forma sencilla, la información generada en la construcción de un Sistema Multiagente (SMA) y controlar más fácilmente las actividades y tareas a realizar, así como los productos a generar, en función de las características propias del proyecto.

Para conocer el avance de un proyecto mediante la utilización de la vista gráfica de APE, es necesario determinar previamente el producto, el cronograma y el costo de nuestro proyecto, para realizar un adecuado análisis de estas variables en cualquier fase o periodo de evaluación, para esto es necesario que los Directores de Proyecto, sean consientes de la importancia que tiene el registrar oportunamente la información relacionada con la planificación y finalización de las actividades del proyecto.

También es importante impulsar el desarrollo de nuevos proyectos que permitan implementar nuevas funcionalidades, mejoras y actualizaciones en la herramienta agent Tool Process Editor (APE), como por ejemplo:

- Contar con un historial histórico que permita guardar y consultar las diferentes representaciones gráficas que han sido generadas durante el control y seguimiento del proyecto, con el objetivo de realizar una retroalimentación del proceso para tener en cuenta futuros desarrollos.

- Realizar representaciones gráficas sobre el comportamiento de los indicadores de progreso en función del tiempo.

- Representar gráficamente la información generada, mediante el tipo de gráfica seleccionado por el usuario, como por ejemplo, un diagrama de barras, de dispersión, circular o de sectores.

Algunos de los beneficios ofrecidos por la vista gráfica en la herramienta agentTool Process Editor (APE) son:

- Facilitar la toma de decisiones y representar de forma sencilla la información generada.
- Realizar fácilmente un análisis del proyecto en función de los costos y tiempos estipulados durante en cualquier fase o periodo de evaluación del proyecto.
- Controlar fácilmente las actividades y tareas a realizar, así como los productos a generar, en función de las características propias del proyecto.
- Identificar de manera rápida y oportuna los posibles cambios y variaciones que pueden afectar el normal desarrollo del proyecto.
- Facilitar a los directores de proyecto la implementación de acciones correctivas o preventivas de manera oportuna.

Finalmente con los resultados obtenidos en el desarrollo de esta propuesta, es posible evidenciar la importancia y utilidad que tienen las representaciones gráficas en los procesos de control y análisis de la información, por lo tanto, es importante promover e impulsar el uso y los beneficios que ofrecen estas herramientas CASE en la construcción de un Sistema Multiagente (SMA), permitiendo hacer de esta labor un proceso cada vez más fácil y cómoda para los desarrolladores de software.

7. BIBLIOGRAFÍA

[1] Alistair Cockbun. Agile Software Development. Addison-Wesley. 2001.

[2] Bertoline, D., Novikau, A., Sus,i A., and Perini, A. TAOM4E: an Eclipse ready tool for Agent-Oriented Modeling. ITC-irst. Italy

[3] Collis, J., Ndumu D. The Role Modelling Guide. Applied Research and Technology, BT Labs.1999.

[4] Daum Berthold. Profesional Eclipse 3 para desarrolladores JAVA. Madrid: Anaya Multimedia, 2005.

[5] Eclipse Foundation. Project Eclipse. Disponible en: http://www.eclipse.org/

[6] Entorno de desarrollo INGENIAS. Disponible en: http://grasia.fdi.ucm.es/main/es/node/61

[7] Ferber, J. Multi-Agent System: An Introduction to Distributed Artificial Intelligence.Addison-Wesley.1999.

[8] Free Software Foundation. "Free software definición". [Online]. Disponible en: http://www.gnu.org/philosophy/free-sw.html.

[9] Garcia Ojeda, J. C., DeLoach S.A., and Robby. agentTool III: From Process Definition to Code Generation. In Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'09). Demo Session. Budapest - Hungary.

[10] Garcia Ojeda, J. C., DeLoach S. A., Robby and Oyenan, W.H. agentTool Process Editor: Supporting the Design of Tailored Agent-based Processes. In Proceedings of the 24th Annual Symposium on Applied Computing. Technical Track on Agent-Oriented Software Engineering Methodologies and Systems (AOMS@SAC'09).

[11] Garcia Ojeda, J. C., DeLoach, S.A., Robby and Oyenan W.H., and Valenzuela J. O-MaSE: A Customizable Approach to Developing Multiagent Development Processes. In Agent Oriented Software Engineering VIII, Proceedings of AOSE 2007. Luck and Padgham (Eds.) Springer Berlin / Heidelberg, 2008.

[12] Gustafson, D. Shaum's Outline Series. Theory and Problems of Software Engineering. McGraw-Hill. New York, NY, 2002.

[13] Highsmith, J., Orr, K. Adaptive Software Development: A Collaborative Approach to Managing Complex Systems. Dorset House. 2000.

[14] Librería JFreeChar. Disponible en: http://www.jfree.org/jfreechart/.

[15] Massimo Cossentino, Valeria Seidita. Composition of a New Process to Meet Agile Needs Using Method Engineering. Italy.

[16] Multiagent & Cooperative Robotics (MACR) Laboratory at Kansas State University, The agentTool III Project. Disponible en: http://agenttool.cis.ksu.edu/index.php?option=com_content&task=view&id=2&Itemid=2

[17] Pérez, D. A., Ginesta, M. G., Hernández, M. y Hernández J. M. Ingeniería del Software en entornos de Software Libre. Segunda Edición. España: Fundación para la Universidad Oberta de Catalunya, 2007.

[18] PMI, Practice Standard For Earned Value Management. Project Management Institute, Inc. 2005.

[19] Prometheus Design Tool (PDT). Disponible en: http://www.cs.rmit.edu.au/agents/pdt/index.shtml.

[20] Scott A. DeLoach. Scott A. DeLoach. Analysis and Design using MaSE and agentTool. Actas de conferencia. Proceedings of the 12th Midwest Artificial Intelligence and Congnitive Science Conference (MAICS) .2001.

[21] SWT (Standard Widget Toolkit). Disponible en: http://www.eclipse.org/swt/

[22] TAOM4E (Tool for Agent Oriented visual Modeling). Disponible en: http://selab.fbk.eu/taom/

[23] Woodridge, M., Jennings, N. R. "Agent Theories, Architectures, and Languages: a Survey," Intelligent Agents, Wooldridge and Jennings Eds, Springer-Verlag, Berlin, 1-22, 1995.

ANEXO A. MANUAL DE USUARIO

Introducción

La herramienta ***agentTool Process Editor (APE)*** ha sido desarrollada como un complemento para el entorno de desarrollo eclipse, con el objetivo de facilitar a los Directores de Proyecto sus labores de control y seguimiento durante la construcción de un ***Sistema Multiagente (SMA)***, al implementar la Técnica del ***Earned Value Analysis (EVA)***. Esta herramienta es complementada con el desarrollo de la vista gráfica **EVA – Graphics**, la cual permite representar gráficamente toda la información relacionada con el alcance, tiempo y costo del proyecto, haciendo que su control y seguimiento sea más eficiente.

El siguiente manual nos describe la manera es que puede ser utilizada la vista gráfica de la herramienta APE.

1. Definición del Entorno Tecnológico

A continuación se presentan las diferentes tecnologías software que se requieren para el correcto funcionamiento de la vista gráfica en la herramienta APE.

- ***Eclipse Ganymede:*** Es un entorno de desarrollo de código abierto y multiplataforma para el desarrollo de "Aplicaciones de Cliente Enriquecido"[17], considerado como una plataforma universal para la integración de herramientas de desarrollo.

- ***agentTool III (aT3):*** aT3 es un entorno de desarrollo que soporta los procesos de análisis y diseño para la construcción de un Sistema Multiagente (SMA) mediante la implementación de la metodología O-MaSE. Este entono de desarrollo ha sido desarrollado como un plug-ins para la plataforma Eclipse.

- ***agentTool Process Editor (APE):*** APE es una herramienta que ha sido desarrollada con el objetivo de facilitar el control y seguimiento de las actividades o tareas que conforman el proceso de desarrollo de un Sistema Multiagente (SMA).

[17] Cliente enriquecido es un término medio entre cliente liviano (aplicación que se accede por una interfaz Web) y el cliente pesado (aplicación que se ejecuta en el sistema operativo del usuario) que proporciona una interfaz gráfica con una sintaxis basada en XML.

2. Instalación y Configuración del Entorno de Desarrollo

2.1. Instalación de la Plataforma Eclipse Ganymede

La instalación del entorno de desarrollo en su versión 3.4 es un proceso fácil de realizar, sólo basta con acceder al enlace http://www.eclipse.org/ y realizar la descargar de la versión de nuestro interés, para nuestro caso accedemos al siguiente enlace: http://www.eclipse.org/ganymede/. Una vez elegida la versión de eclipse, especificamos el lugar donde queremos realizar la descarga y guardamos el archivo en nuestro disco duro. Ahora sólo resta descomprimir el archivo y acceder al ejecutable (eclipse o elcipse.exe) para ejecutar nuestro entorno de desarrollo. (Ver Figura 1).

Terminología del workbench de Eclipse. (Ver Figura 1).

1. Barra de Menús
2. Barra de Herramientas
3. Barra de Perspectiva.
4. Vista de Recursos
5. Editor de texto
6. Vista de Esquema
7. Vista de Tareas
8. Área de Mensajes

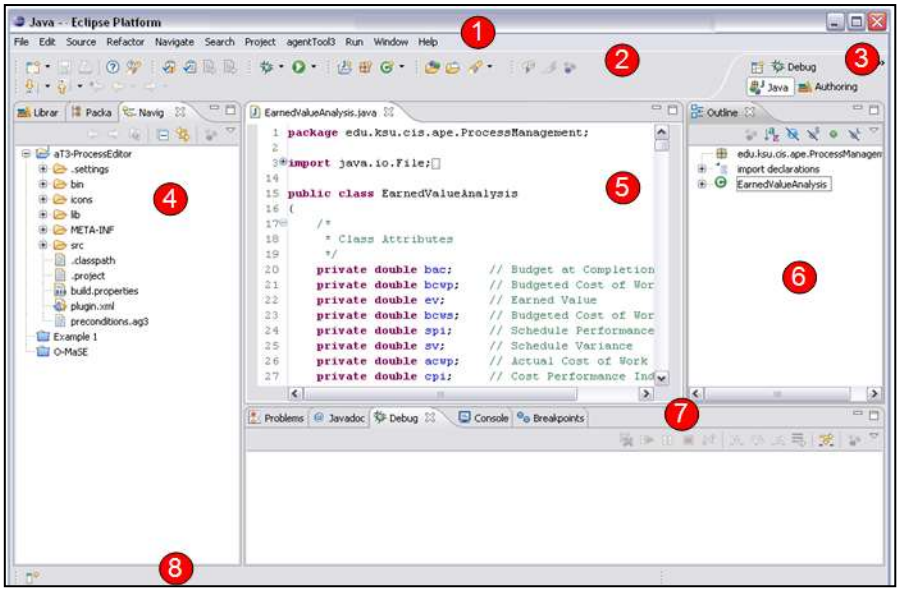

Figura 28. Eclipse Ganymede

2.2. Instalación de plug-in agentTool III (aT3)

El proceso de instalación de aT3 en la plataforma Eclipse la realizamos desde un servidor remoto, realizando las siguientes instrucciones:

- Ejecutar la plataforma Eclipse Ganymede.
- Ingresar al menú ***Help/Software Updates/Avalible Software/Add Site*** para especificar la dirección del servidor remoto de descarga. Sitio web de descarga: http://agenttool.cis.ksu.edu/update/. (Ver Figura 2).

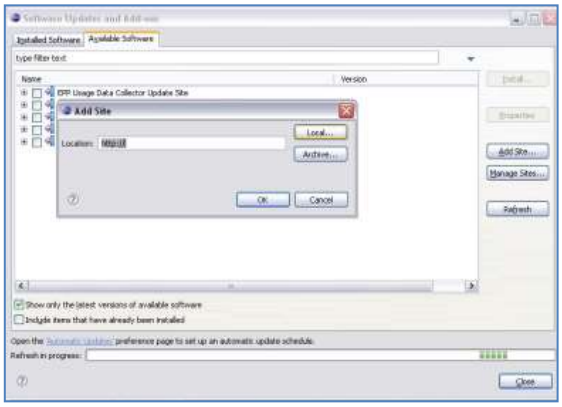

Figura 29. Especificación del sitio de descarga para APE

- Realizar clic el botón ***Refresh*** para seleccionar los items a instalar en la plataforma eclipse y para finalizar realizamos clic en el botón ***Install***. (Ver Figura 3).

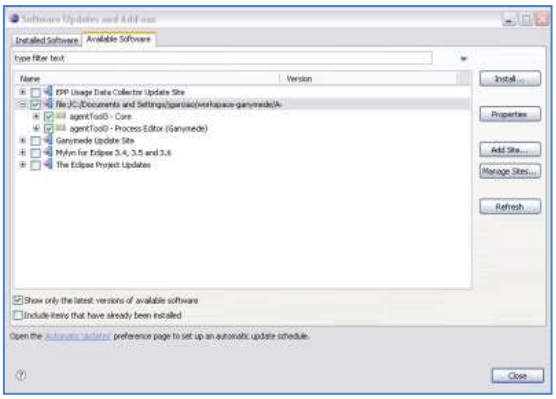

Figura 30. Componentes de instalación de aT3

- Reiniciar la plataforma Eclipse una vez se ha finalizado el proceso de instalación de aT3.

2.3. Relacionar librería O-MASE

Para relacionar la librería O_MASE en la plataforma Eclipse (Ver Figura 4) realizamos las siguientes instrucciones:

- Abrir la vista library mediante el menú: ***Windows/Show view/other/Method Authoring/library***.
- Relacionar librería mediante el menú: ***File/Open/Method Library/ (ruta a la librería O-MASE)*.**

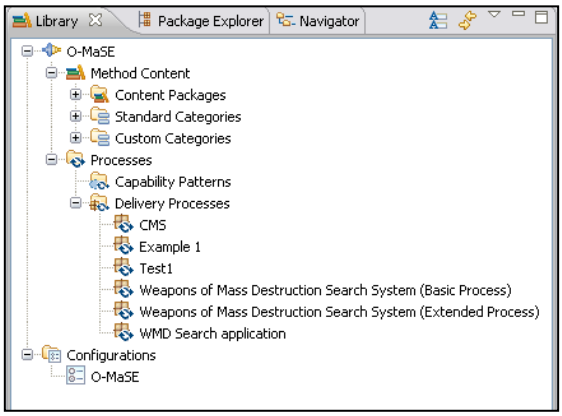

Figura 31. Relacionar Librería O-MASE

3. Ejecución de la Vista Gráfica EVA – Graphics

Para conocer el estado actual o el avance de un proyecto mediante la utilización de la vista gráfica de la herramienta APE, es necesario determinar previamente el producto, el cronograma y el costo de nuestro proyecto, para realizar un adecuado análisis de estas variables durante cualquier fase o periodo de evaluación, para esto es necesario que los Directores de Proyecto, sean consientes de la importancia que tiene el registrar oportunamente la información relacionada con la planificación y finalización de las actividades del proyecto.

Para conocer más en detalle sobre el desarrollo de estas actividades puede consultar la documentación y los tutoriales disponibles en la página oficial del proyecto: http://agenttool.cis.ksu.edu/index.php?option=com_content&task=view&id=5&Itemid=5, donde encontrará información relacionada con la creación y la gestión de proceso de desarrollo personalizado en APE, verificación de la consistencia de un proceso de desarrollo personalizado y la especificación de restricciones para cada tarea.

Una vez el Director de Proyecto, cuenta con un proyecto para realizar sus labores de control y seguimiento se procede a ejecutar la vista gráfica de la herramienta APE. Para el siguiente ejemplo haremos uso del proyecto **Example 1** que se encuentra en la librería O-MASE. (Ver Figura 5).

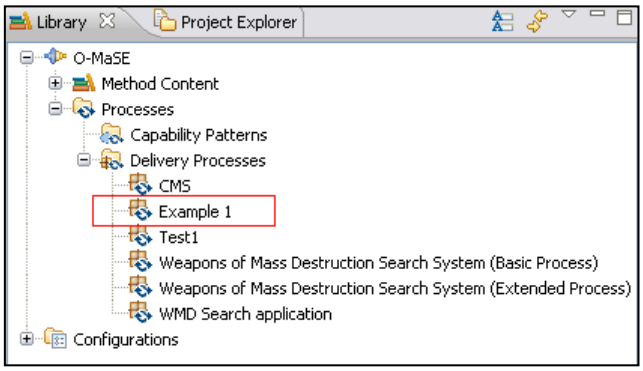

Figura 32. Library O-MASE: Proyecto Example 1

Una vez visualizamos nuestro proyecto de ejemplo ***Example 1*** en la vista ***Library***, accedemos a la vista ***APE - Process Management*** mediante el menú ***Windows/Show View/Other,*** seleccionamos la vista *APE-Process Management* y seleccionamos el proyecto ***Example 1*** en la vista *Library,* lo cual permitirá que el proyecto se visualice en la vista *APE - Process Management.* (Ver Figura 6).

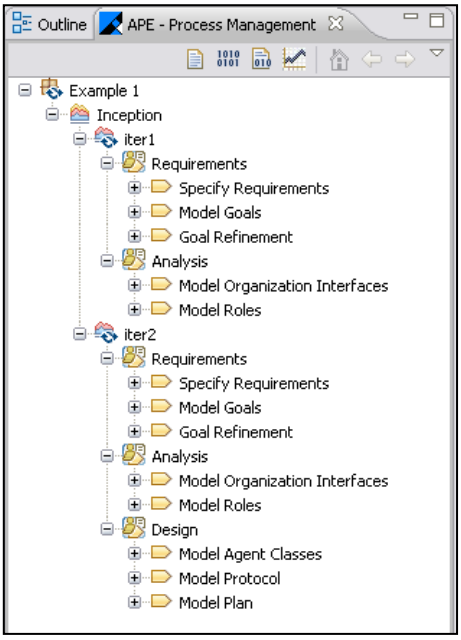

Figura 33. Estructura del Proyecto Example 1

Para ejecutar la vista gráfica de APE podemos realizar una de las siguientes opciones:

- Seleccionar el proyecto *en la vista APE - Process Management* y realizar clic sobre el icono ***EVA Graphics*** que se encuentra en la barra de herramientas de la vista *APE - Process Management.* (Ver Figura 7).

- *Seleccionar el proyecto en la vista APE - Process Management y realizar clic derecho con el mouse y seleccionar* la opción del menú ***EVA Graphics.*** (Ver Figura 7).

- *Seleccionar el proyecto en la vista APE - Process Management y* acceder a la vista ***EVA Graphics*** mediante el menú ***Windows/Show View/Other*** y desplegar el paquete ***Other*** para seleccionar la vista ***APE – EVA Graphics***, como se muestra en la Figura 8.

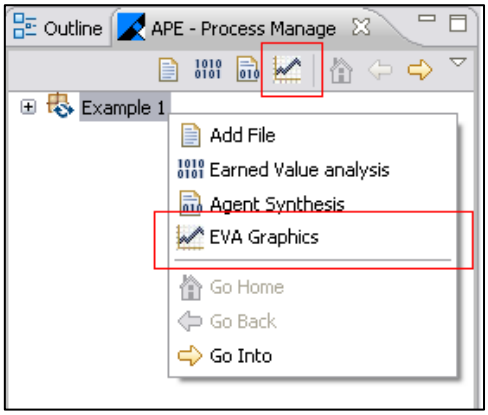

Figura 34. Vista APE - Process Management

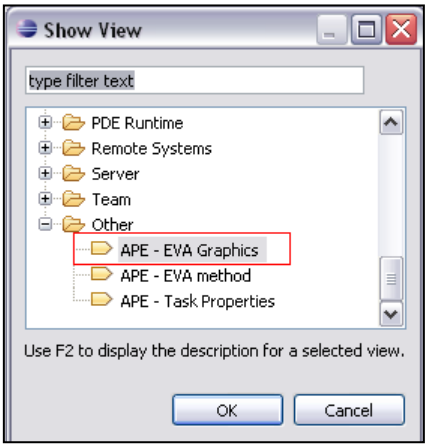

Figura 35. Show View: APE - Eva Graphics

Al realizar cualquiera de las opciones anteriores para la ejecución de la vista gráfica de APE, se podrá visualizar la siguiente información: una sección con los valores obtenidos para los indicadores de progreso al aplicar la Técnica del *Earned Value Analysis (EVA)* en el proyecto y un botón con el nombre *Spline* el cual permitirá suavizar la gráfica generada, y una segunda sección donde se visualizará la gráfica generada para el proyecto, , la cual podremos guardar o imprimir al ubicarnos sobre la gráfica y realizar clic derecho con el mouse y seleccionar la opción de nuestro interés.

Un ejemplo del correcto funcionamiento de la vista gráfica *EVA – Graphics* en la herramienta agentTool Process Editor (APE) se puede observar en la Figura 9.

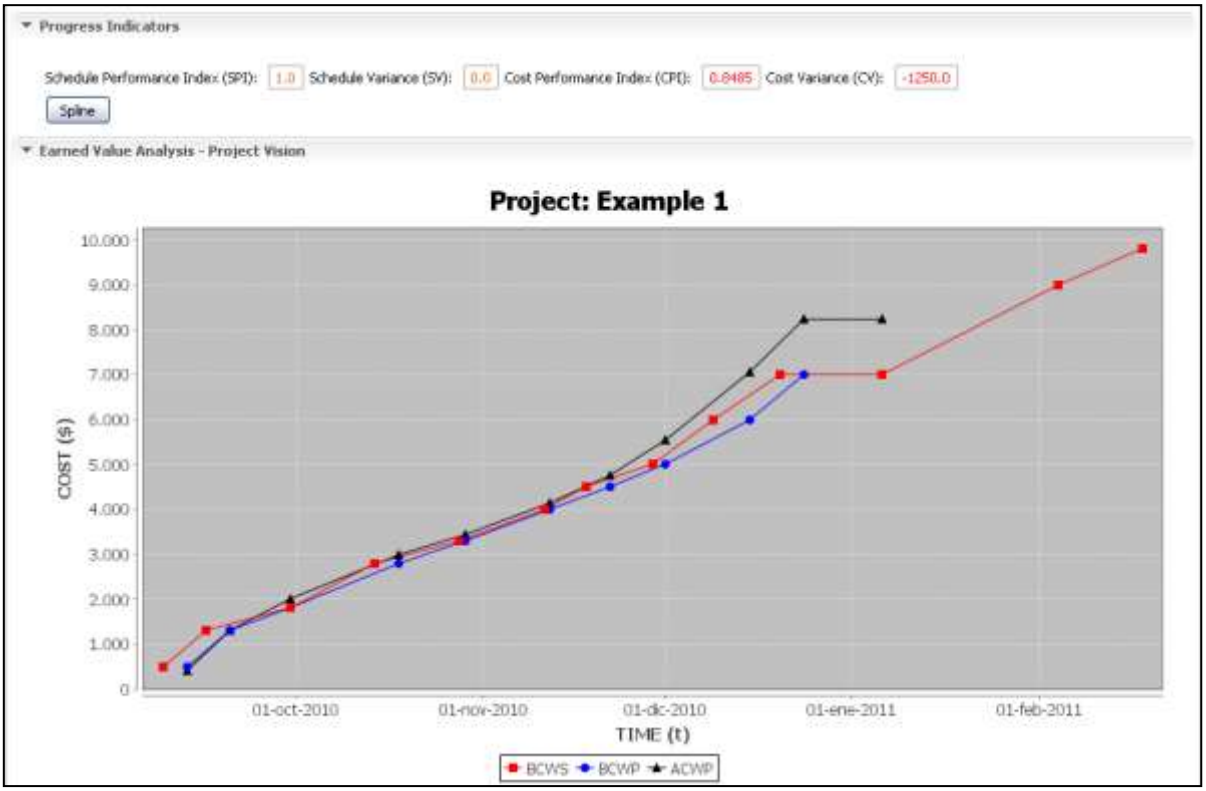

Figura 36. Vista Gráfica EVA – Graphics

ANEXO B. MANUAL TECNICO

1. Introducción

La herramienta ***agentTool Process Editor (APE)*** ha sido desarrollada como un complemento para el entorno de desarrollo eclipse, con el objetivo de facilitar a los Directores de Proyecto sus labores de control y seguimiento durante la construcción de un ***Sistema Multiagente (SMA)***, al implementar la Técnica del ***Earned Value Analysis (EVA)***. Esta herramienta es complementada con el desarrollo de la vista gráfica **EVA – Graphics**, la cual permite representar gráficamente toda la información relacionada con el alcance, tiempo y costo del proyecto, haciendo que su control y seguimiento sea más eficiente.

El siguiente manual describe los conceptos básicos y los requerimientos software, que han sido utilizados para garantizar el correcto un funcionamiento de la vista gráfica en la herramienta APE, y para facilitar a futuros desarrolladores de software la implementación de mejoras en la herramienta o en la vista gráfica desarrollada.

2. Marco Metodológico

El desarrollo de la vista gráfica en APE se basa principalmente en los conceptos de la ***Investigación Tecnológica Aplicada o Investigación Tecnológica***, al ser un desarrollo orientado a la compresión y análisis de una situación en particular, con el objetivo de complementar una solución software ya existente. En este desarrollo se implementaron los principios y conceptos básicos de la metodología ***eXtreme Programing*** o ***Programación Extrema***, al ser una de las metodologías más destacadas en los procesos ágiles de desarrollo de software y al ser una de las metodologías que más se adapta a los entornos de desarrollo de ***Software Libre***.

3. Arquitectura de Desarrollo

La arquitectura seleccionada para el desarrollo de la vista gráfica es la denominada ***arquitectura de 3 capas y un nivel***. Esta arquitectura tiene como objetivo primordial diferenciar la capa de presentación, con la capa de la lógica del negocio y la capa de datos; y se denomina de 1 nivel porque la solución de 3 capas reside en un sólo ordenador. (Ver Figura 1).

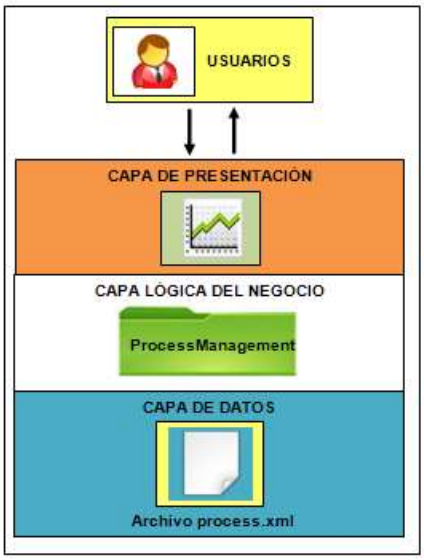

Figura 37. Arquitectura de 3 capas

La capa de presentación, es la capa que ve el usuario y por la cual se le comunica o se le captura información, también se le conoce como interfaz gráfica y se comunica únicamente con la capa del negocio. La capa de la lógica del negocio, es donde se establecen las reglas que deben cumplir para dar solución a las necesidades o requisitos planteados, es la capa que recibe las peticiones del usuario y envía las respuestas tras la ejecución de los procesos y se comunica con la capa de presentación y la capa de datos. La capa de datos es donde residen o se almacenan los datos, es una capa que recibe las solicitudes de almacenamiento o recuperación de la información desde la capa del negocio.

Es importante tener en cuenta que las capas definidas en la arquitectura son simplemente agrupaciones lógicas de los componentes de software que conforman la aplicación. Ayudan a diferenciar entre los distintos tipos de tareas que realizan los componentes o clases, facilitando el diseño de la reutilización en la solución.

4. Estándar de Codificación

El estándar de codificación utilizado para el desarrollo de la vista gráfica en la herramienta agentTool Process Editor (APE) es el ***Java Language Specification*** establecido por Sun Microsystems para el lenguaje de programación JAVA, el cual puede ser consultado en el siguiente enlace: http://java.sun.com/docs/books/jls/second_edition/html/jTOC.doc.html.

5. Definición del Entorno Tecnológico

A continuación se presentan las diferentes tecnologías software seleccionadas para el desarrollo de la vista gráfica en la herramienta APE.

- ***Java Rutine Environment (JRE):*** El JRE es un conjunto de utilidades y componentes que permiten la ejecución de aplicaciones Java.

- ***Eclipse Ganymede:*** Es un entorno de desarrollo de código abierto y multiplataforma para el desarrollo de "Aplicaciones de Cliente Enriquecido"[18], considerado como una plataforma universal para la integración de herramientas de desarrollo.

- ***agentTool III (aT3):*** aT3 es un entorno de desarrollo que soporta los procesos de análisis y diseño para la construcción de un Sistema Multiagente (SMA) mediante la implementación de la metodología O-MaSE. Este entono de desarrollo que ha sido desarrollado como un plug-ins para la plataforma Eclipse.

- ***agentTool Process Editor (APE):*** APE es una herramienta que ha sido desarrollada como un complemento para el entorno de desarrollo Eclipse, con el objetivo de facilitar el control y seguimiento de las actividades o tareas que conforman el proceso de desarrollo de un Sistema Multiagente (SMA).

- ***Lenguaje de programación JAVA:*** Java es un lenguaje de programación orientado a objetos que ha sido desarrollado por Sun Microsystems, con el objetivo de contar con un entorno de desarrollo que sea independiente de la plataforma.

- ***SWT (Standard Widget Toolkit):*** SWT es un kit de herramientas gráficas para trabajar bajo la plataforma de Java.

- ***Librería O-MASE:*** Esta librería está conformada un conjunto de funciones que implementan la metodología O-MASE, para facilitar la construcción de un Sistema Multiagente (SMA). Esta librería nos proporciona un proyecto de Ejemplo con el cual podemos realizar las respectivas pruebas, que garantizan el correcto funcionamiento de la vista gráfica en la herramienta APE.

- ***Librería JFreeChart:*** JFreeChart es una librería para gráficos escrita totalmente en Java, que facilita crear y mostrar gráficos de calidad profesional en nuestras aplicaciones.

[18] Cliente enriquecido es un término medio entre cliente liviano (aplicación que se accede por una interfaz Web) y el cliente pesado (aplicación que se ejecuta en el sistema operativo del usuario) que proporciona una interfaz gráfica con una sintaxis basada en XML.

6. Definición de Actores

En el desarrollo de la vista gráfica para APE se han identificado dos actores[19]. El primer actor corresponde al director de proyecto y el segundo actor se le denomina programador, el director de proyecto corresponde a la persona encargada de hacer uso de la vista gráfica, para apoyar sus labores de control y seguimiento en la construcción de un Sistema Multiagente (SMA), y el actor programador corresponde a la empresa o persona encargada de realizar cambios o mejoras en la herramienta con el fin de adaptarla a sus necesidades, al ser considerada como una herramienta de Software Libre.

7. Diagrama de Casos de Uso

El diagrama de casos de uso es considerado como un diagrama de comportamiento, que nos indica cómo debe interactuar el sistema con el usuario o con otro tipo de sistema. En la Figura 2 podemos observar el diagrama de casos de uso establecido para el desarrollo de la vista gráfica en APE.

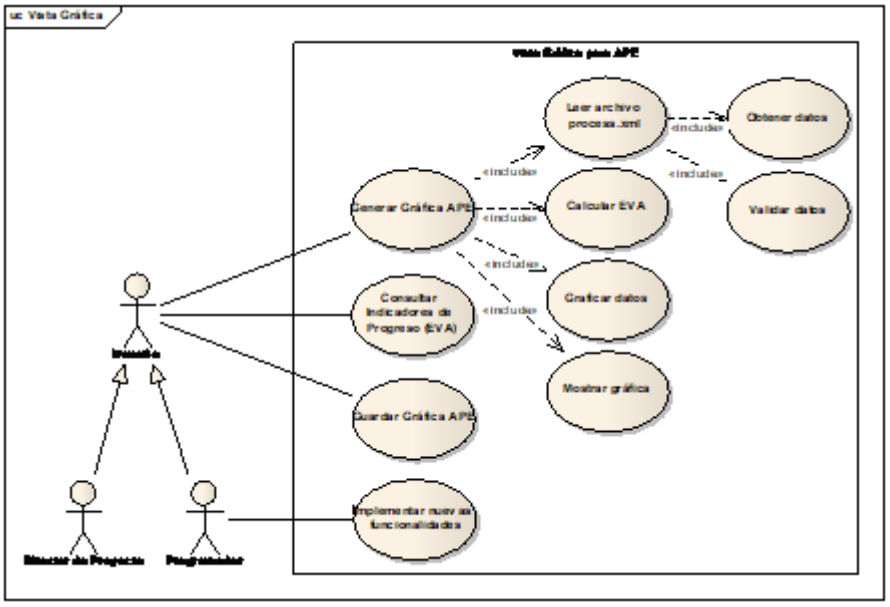

Figura 38. Diagrama de Casos de Uso

[19] Un actor es una entidad externa que realiza algún tipo de interacción con la herramienta o el sistema que se está desarrollando

8. Especificación de Requisitos

La Especificación de Requisitos consiste en la descripción completa del comportamiento de la vista gráfica desarrollada. Estos requisitos se clasificaron en requisitos de la interfaz de usuario y en requisitos funcionales y no funcionales.

- **Requisitos de Interfaz de Usuario**

La interfaz de usuario consiste en la creación de una nueva vista en la herramienta agent Tool Process Editor (APE), que está conformada por dos secciones: la primera sección mostrará los indicadores de progreso obtenidos de aplicar la técnica del *Earned Value Analysis (EVA)* en el proyecto y por un botón que permitirá realizar la función de *spline* sobre la gráfica generada, la segunda sección es la encargada de mostrar la gráfica generada para el análisis del costo y el avance del proyecto.

- **Requisitos Funcionales**

Los requisitos funcionales son los que determinan la funcionalidad que debe proporcionar al usuario la vista gráfica desarrollada. Estos requisitos son:

REQF1: Verificar la existencia del archivo process.xml para el proyecto en revisión.

REQF2: Informar al usuario si no es posible acceder al archivo process.xml del proyecto.

REQF3: Leer los datos del archivo process.xml.

REQF4: Verificar que el tipo de dato para los valores estimados y reales de las actividades, corresponde con el tipo de dato requerido para el proceso.

REQF5: Calcular las medidas básicas e indicadores de progreso según la técnica del Earned Value Analysis (EVA).

REQF6: Determinar los valores a graficar para conocer el comportamiento del proyecto en relación a las variables de análisis BCWS, BCWP y ACWP.

REQF7: Graficar los valores establecidos en función del tiempo para el análisis de las variables BCWS, BCWP y ACWP.

REQF8: Mostrar al usuario el valor de los indicadores de progreso.

REQF9: Mostrar al usuario la gráfica obtenida para el análisis de las variables BCWS, BCWP y ACWP.

REQF10: Permitir al usuario visualizar los puntos graficados en cada variable de análisis.

REQF11: Realizar la interpolación de datos para suavizar la grafica generada.

REQF12: Permitir al usuario guardar la gráfica generada para el proyecto en revisión.

REQF13: Permitir al usuario ejecutar la vista gráfica desde la vista Process Management.

- **Requisitos No Funcionales**

Los requisitos no funcionales, son aquellos que describen las facilidades que debe proporcionar la vista gráfica, estos requisitos son:

REQNF1: Facilidad de uso por parte de los usuarios.

REQNF2: Ejecución de todos sus procesos de manera satisfactoria sin presentar interrupciones de manera abrupta.

REQNF3: Portabilidad. Gracias al lenguaje de programación utilizado, es portable a cualquier plataforma.

REQNF4: Tiempo de respuesta. Está determinado por la capacidad del procesador donde se ejecuta las funciones de la librería y el número de actividades que conformen el proyecto.

REQNF5: Escalabilidad. La vista gráfica debe estar en capacidad de permitir incluir, modificar o eliminar nuevas funcionalidades después de su construcción y puesta en marcha inicial.

REQNF6: El código fuente debe estar debidamente documentado.

REQNF7: Disponibilidad de ejecución de la vista gráfica a partir de la vista Process Management.

9. Diagrama de clases

El diagrama de clases es un diseño que captura la estructura lógica del sistema y el comportamiento que tiene la implementación realizada, es un diagrama que nos describe las clases implementadas con sus atributos, métodos y relaciones. La Figura 3 nos muestra el diagrama de clases que ha sido definido en el desarrollo de la vista gráfica.

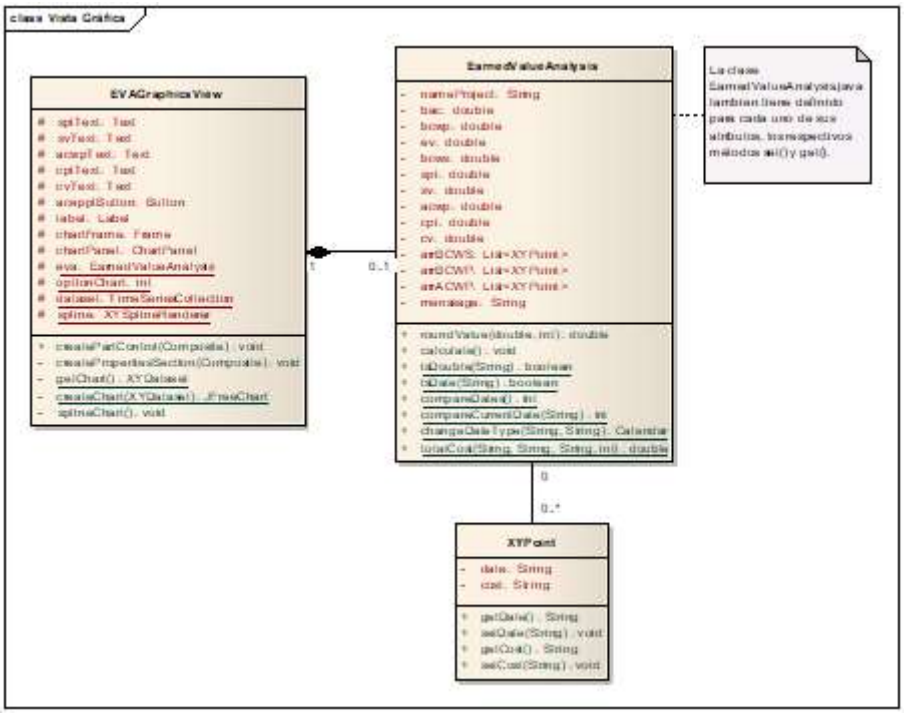

Figura 39. Diagrama de clases

10. Instalación y Configuración del Entorno de Desarrollo

10.1. Instalación del Java Rutine Environment (JRE)

El JRE lo podemos descargar de forma gratuita en el siguiente enlace http://www.java.com/es/ realizando clic en el botón *Descarga gratuita de Java,* luego aparecerá un cuadro de diálogo que permitirá *ejecutar* o *guardar* el archivo descargado. Para comprobar que Java se ha instalado y funciona correctamente en el equipo ejecutamos el ***applet de prueba*** que se encuentra disponible en:

http://www.java.com/es/download/help/windows_manual_download.xml, el cual nos indicará si el proceso se ha realizado satisfactoriamente mediante la visualización de la siguiente imagen:

Figura 40. Comprobación de instalación del JRE

10.2. Instalación de la Plataforma Eclipse Ganymede

La instalación del entorno de desarrollo en su versión 3.4 es un proceso fácil de realizar, sólo basta con acceder al enlace http://www.eclipse.org/ y realizar la descargar de la versión de nuestro interés, para nuestro caso accedemos al siguiente enlace: http://www.eclipse.org/ganymede/. Una vez elegida la versión de eclipse, especificamos el lugar donde queremos realizar la descarga y guardamos el archivo en nuestro disco duro. Ahora sólo resta descomprimir el archivo y acceder al ejecutable (eclipse o elcipse.exe) para ejecutar nuestro entorno de desarrollo. (Ver Figura 5).

Terminología del workbench de Eclipse. (Ver Figura 5).

9. Barra de Menús
10. Barra de Herramientas
11. Barra de Perspectiva.
12. Vista de Recursos
13. Editor de texto
14. Vista de Esquema
15. Vista de Tareas
16. Área de Mensajes

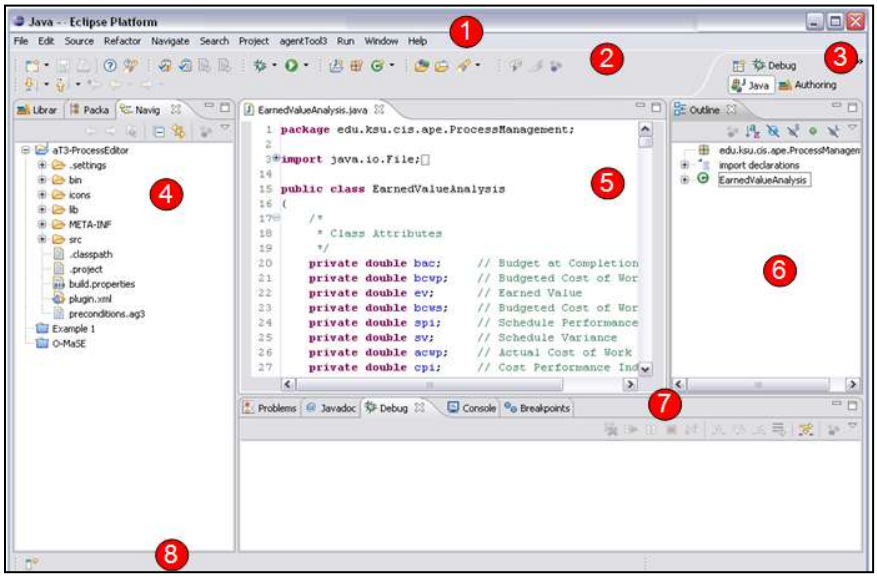

Figura 41. Eclipse Ganymede

10.3. Instalación de plug-in agentTool III (aT3)

El proceso de instalación de aT3 en la plataforma Eclipse la realizamos desde un servidor remoto, realizando las siguientes instrucciones:

- Ejecutar la plataforma Eclipse Ganymede.
- Ingresar al menú ***Help/Software Updates/Avalible Software/Add Site*** para especificar la dirección del servidor remoto de descarga. Sitio web de descarga: http://agenttool.cis.ksu.edu/update/. (Ver Figura 6).

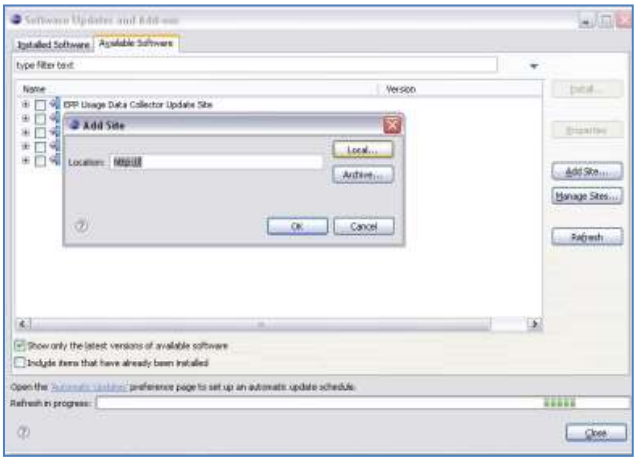

Figura 42. Especificación del sitio de descarga para APE

- Realizar clic el botón ***Refresh***, seleccionar los items a instalar en la plataforma eclipse y realizamos clic en el botón ***Install***. (Ver Figura 7).

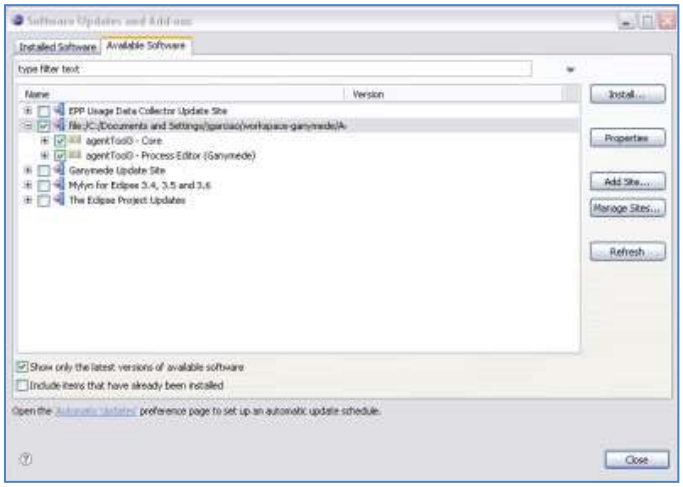

Figura 43. Componentes de instalación de aT3

- Reiniciar la plataforma Eclipse una vez se ha finalizado el proceso de instalación de aT3.

10.4. Importar el proyecto agentTool Process Editor (APE)

Para importar el proyecto agentTool Process al workspace de la plataforma Eclipse, realizamos clic en el menú ***File/Import/Existing Projects*.** (Ver Figura 8).

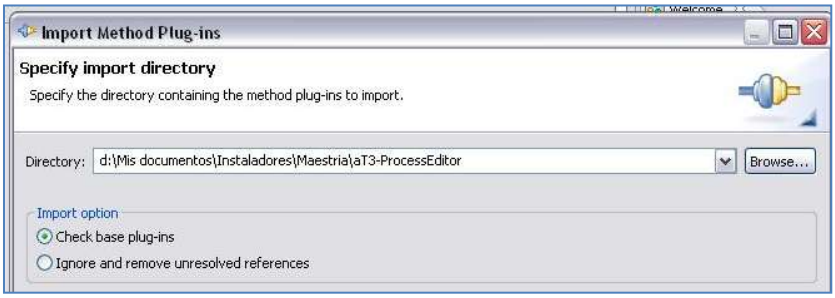

Figura 44. Importar proyecto APE

Una vez es importado el proyecto APE al entorno de desarrollo eclipse es posible visualizar su estructura de paquetes y las clases implementadas en el paquete ***ProcessManagement*** para la creación de la vista gráfica en APE, como se muestra en la Figura 9.

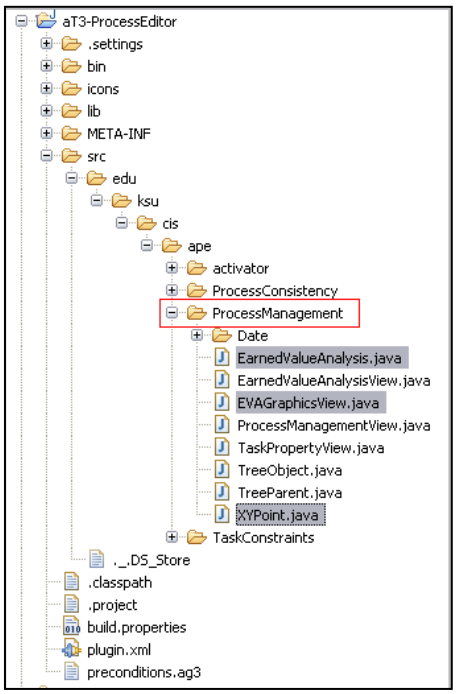

Figura 45. Estructura de paquetes Proyecto aT3-ProcessEditor

10.5. Relacionar librería O-MASE

Para relacionar la librería O_MASE en la plataforma Eclipse (Ver Figura 10) realizamos las siguientes instrucciones:

- Abrir la vista library mediante el menú: ***Windows/Show view/other/Method Authoring/library***.
- Relacionar librería mediante el menú: ***File/Open/Method Library/ (ruta a la librería O-MASE)***.

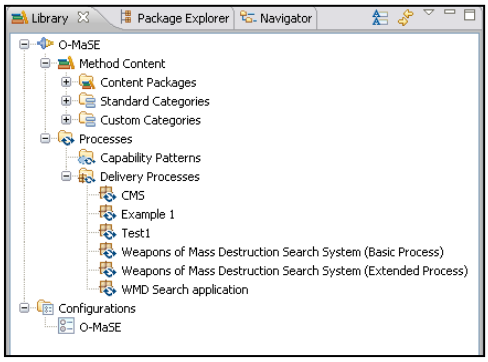

Figura 46. Relacionar Librería O-MASE

10.6. Crear una configuración para ejecutar y depurar la aplicación

La configuración a realizar permitirá detectar y corregir errores en el proyecto aT3-ProcessEditor, así como verificar el correcto funcionamiento de los cambios o modificaciones realizados en el código fuente. Para crear esta configuración seleccionamos el menú ***Run/Debug Configuration/*** y en la ventana emergente que se nos muestra por defecto realizamos doble clic en la opción ***Eclipse Aplication*** e ingresamos un nombre para la configuración, especificamos el JRE con el cual vamos a ejecutar o depurar la aplicación, y en la pestaña ***Common*** en la opción ***Display in favorites menú*** seleccionamos la opción ***Debug***, y para finalizar seleccionamos el botón ***Apply*** y ***Close***. (Ver Figura 11).

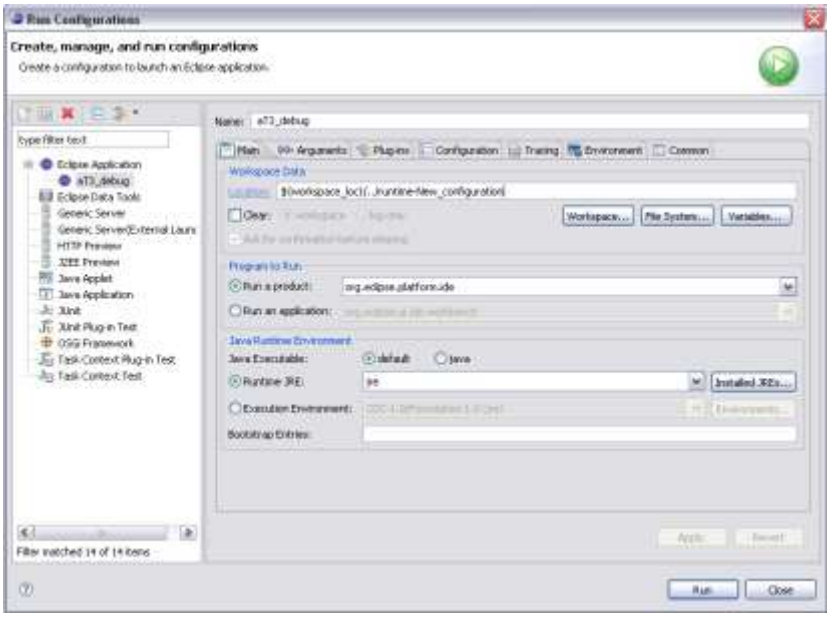

Figura 47. Configuración para ejecutar y depurar la aplicación

11. Ejecución de la Vista Gráfica EVA - Graphics

Para ejecutar el proyecto ***aT3-ProcessEditor*** y verificar su correcto funcionamiento, realizamos clic en la configuración creada en el numeral 10.6. (Ver Figura 12).

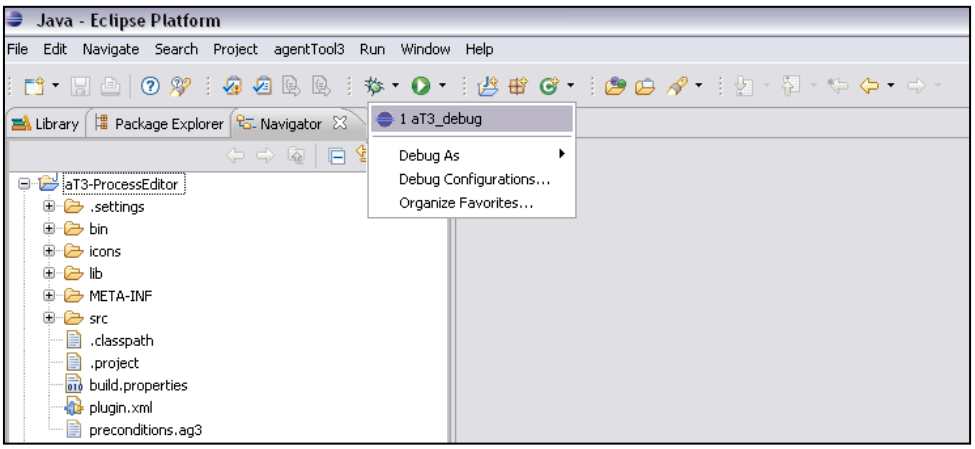

Figura 48. Ejecutar o depurar proyecto aT3-ProcessEditor

El resultado de ejecutar esta configuración nos permitirá visualizar un nuevo entorno de desarrollo Eclipse donde se realizarán las pruebas de ufncionamiento. Al ubicarnos en el nuevo entorno de desarrollo, realizamos clic en el menú ***Windows/Show View/Other*** para abrir las vistas ***APE - Process Management*** y ***APE–Task Properties*** respectivamente.

Después de seleccionar la vista ***APE-Process Management*** nos ubicamos en la vista ***Library*** para seleccionar el proyecto ***Example 1*** (Ver Figura 13), lo cual permitirá que el proyecto se visualice en la vista *APE - Process Management* (Ver Figura 14) y así especificar algunos datos de prueba para el proyecto de ejemplo, por medio de la vista ***APE – Task Properties***, como se muestra en la Figura 15.

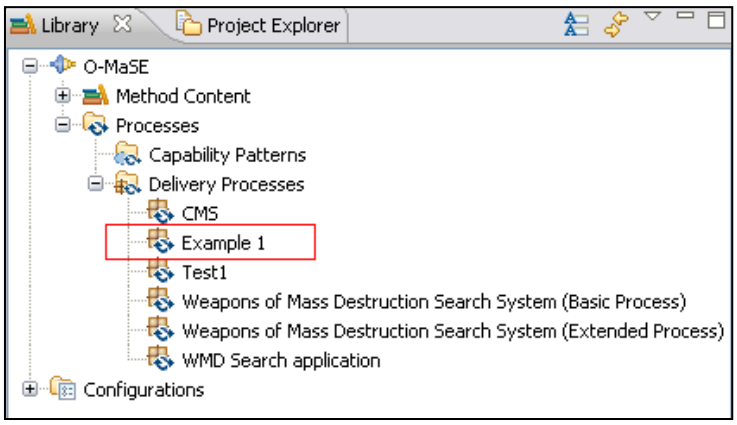

Figura 49. Proyecto Example 1

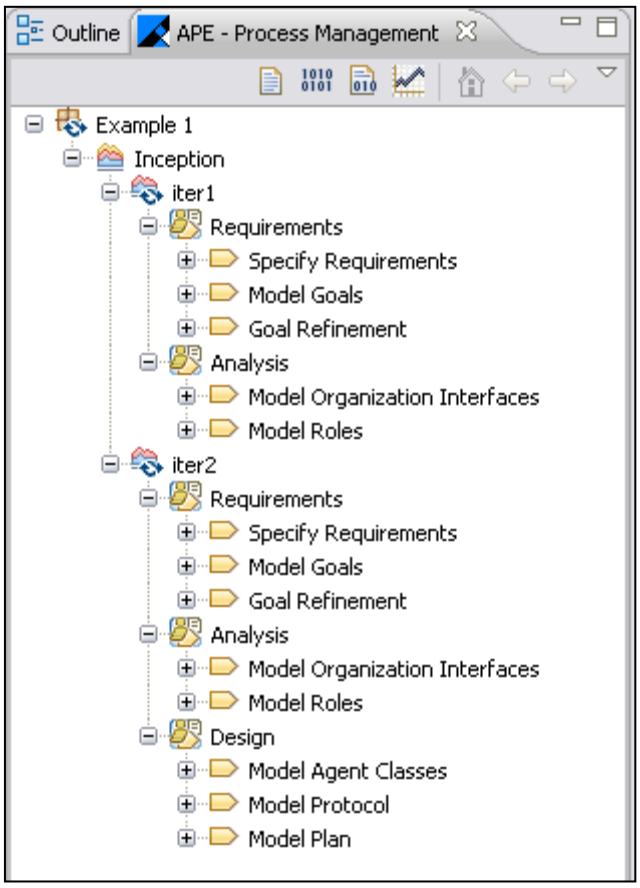

Figura 50. Vista APE – Process Management

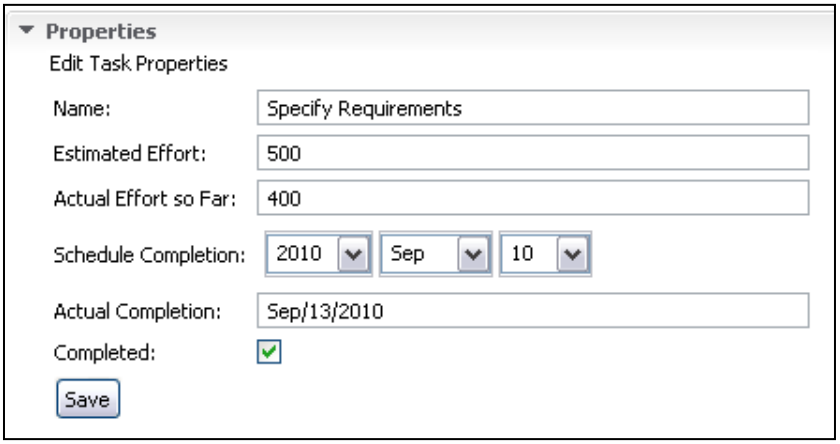

Figura 51. Vista APE - Task Properties

Una vez se han especificado y guardado algunos datos de prueba para el proyecto de *Example 1*, ejecutamos la vista gráfica de APE a partir de la vista *APE-Process Management* seleccionando el icono ***EVA Graphics*** o realizando clic derecho sobre el proyecto y seleccionando la opción del menú ***EVA Graphics*****,** tal y como se muestra en la Figura 16.

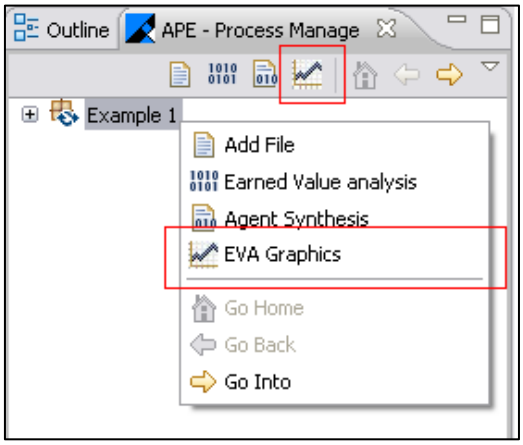

Figura 52. Vista APE - Process Management

Otra manera de poder ejecutar la vista gráfica de la herramienta APE es ingresando al menú ***Windows/Show View/Other*** y al desplegar el paquete ***Other*** seleccionamos la vista ***APE – EVA Graphics***, como se muestra en la Figura 17.

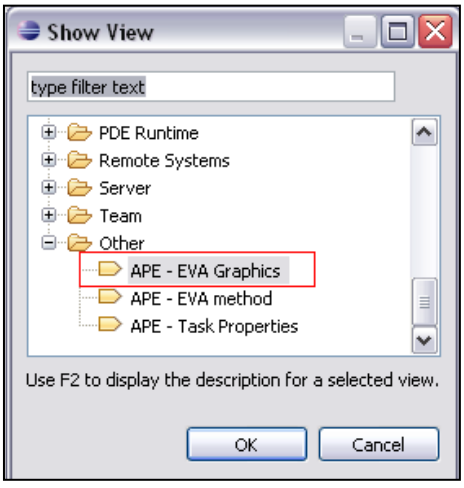

Figura 53. Show View: APE - Eva Graphics

Al realizar cualquiera de las opciones anteriores para la ejecución de la vista gráfica de APE, se podrá visualizar la siguiente información: una sección con los valores obtenidos para los indicadores de progreso al aplicar la Técnica

del *Earned Value Analysis (EVA)* en el proyecto y un botón con el nombre *Spline* el cual permitirá suavizar la gráfica generada, y una segunda sección donde se visualizará la gráfica generada para el proyecto, la cual podremos guardar o imprimir al ubicarnos sobre la gráfica y realizar clic derecho con el mouse y seleccionar la opción de nuestro interés.

Un ejemplo del correcto funcionamiento de la vista gráfica *EVA – Graphics* en la herramienta agentTool Process Editor (APE) se puede observar en la Figura 18.

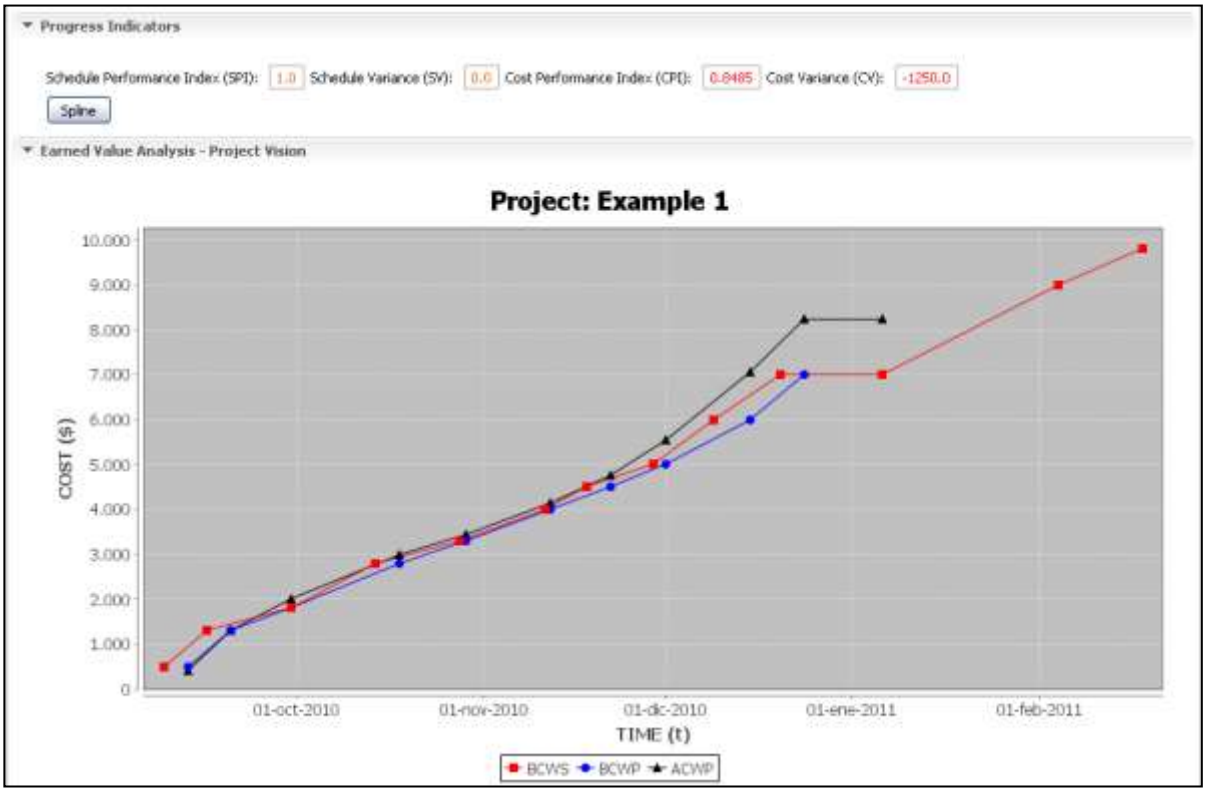

Figura 54. Vista Gráfica EVA – Graphics

ANEXO C. CODIGO FUENTE

El código fuente de una aplicación es el conjunto de instrucciones que deben ser ejecutadas por la computadora para ofrecer al usuario de la herramienta, un conjunto de funcionalidades para la cual ha sido desarrollada.

El código fuente que ha sido generado en la construcción de la vista gráfica para la herramienta agentTool Process Editor (APE), se encuentra en el paquete ProcessManagement dentro de las siguientes clases:

1. Clase EarnedValueAnalysis
2. Clase EVAGraphicsView
3. Clase XYPoint
4. ProcessManagementView

A continuación se presenta el código fuente que ha sido definido en cada una de las clases:

1. Clase EarnedValueAnalysis

```
package edu.ksu.cis.ape.ProcessManagement;
import java.io.File;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.GregorianCalendar;
import java.util.List;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import org.eclipse.core.runtime.Platform;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

/**
*
* @author Andrea P. García Prada
*
*/
/**
* EarnedValueAnalysis made the calculation of basic measures and indicators of
* progress by technical Earned Value Analysis (EVA).
*/
public class EarnedValueAnalysis
{
		/*
		 * Class Attributes
		 */
		private String nameProject ;  // Project Name
		private double bac;           // Budget at Completion
		private double bcwp;          // Budgeted Cost of Work Performed
		private double ev;            // Earned Value
		private double bcws;          // Budgeted Cost of Work Scheduled
```

```
private double spi;             // Schedule Performance Index
private double sv;              // Schedule Variance
private double acwp;            // Actual Cost of Work Performed
private double cpi;             // Cost Performance Index
private double cv;              // Cost Variance
private List<XYPoint> arrBCWS;
private List<XYPoint> arrBCWP;
private List<XYPoint> arrACWP;
private String menssage;

/**
 * Constructor class
 */
public EarnedValueAnalysis()
{
        arrBCWS = new ArrayList<XYPoint>();
        arrBCWP = new ArrayList<XYPoint>();
        arrACWP = new ArrayList<XYPoint>();
        nameProject = "";
        menssage = “”;
        bac = bcwp = ev = bcws = spi = sv = acwp = cpi = cv = 0;
}

/**
 * Getters and setters
 */
public String getNameProject() {
        return nameProject;
}

public void setNameProject(String nameProject) {
        this.nameProject = nameProject;
}

public double getBAC() {
        return bac;
}

public void setBAC(double bac) {
        this.bac = bac;
}

public double getBCWP() {
        return bcwp;
}

public void setBCWP(double bcwp) {
        this.bcwp = bcwp;
}

public double getEV() {
        return ev;
}

public void setEV(double ev) {
        this.ev = ev;
}

public double getBCWS() {
        return bcws;
}

public void setBCWS(double bcws) {
        this.bcws = bcws;
```

```
}

public double getSPI() {
	return spi;
}

public void setSPI(double spi) {
	this.spi = spi;
}

public double getSV() {
	return sv;
}

public void setSV(double sv) {
	this.sv = sv;
}

public double getACWP() {
	return acwp;
}

public void setACWP(double acwp) {
	this.acwp = acwp;
}

public double getCPI() {
	return cpi;
}

public void setCPI(double cpi) {
	this.cpi = cpi;
}

public double getCV() {
	return cv;
}

public void setCV(double cv) {
	this.cv = cv;
}

public String getMenssage() {
	return menssage;
}

public void setMenssage(String menssage) {
	this.menssage = menssage;
}

public List<XYPoint> getArrBCWS() {
	return arrBCWS;
}

public void setArrBCWS(List<XYPoint> arrBCWS) {
	this.arrBCWS = arrBCWS;
}

public List<XYPoint> getArrBCWP() {
	return arrBCWP;
}

public void setArrBCWP(List<XYPoint> arrBCWP) {
	this.arrBCWP = arrBCWP;
```

```
	}

	public List<XYPoint> getArrACWP() {
		return arrACWP;
	}

	public void setArrACWP(List<XYPoint> arrACWP) {
		this.arrACWP = arrACWP;
	}

	/**
	 * Method to round a number.
	 * @param number
	 * @param decimalNumber
	 * @return
	 */
	public double roundValue(double number, int decimalNumber) {
		double value = 0;
		double newValue = 0;
		double newNumber = 0;
		try {
			value = (double) Math.pow (10, decimalNumber);
			number = number * value;
			newValue = Math.round(number);
		   newNumber = (double) newValue/value;
		}
		catch (Exception e) {
			System.out.println("Error in EarnedValueAnalysis.java - roundValue() "+ e);
		}
		return newNumber;
	}

	/**
	 * Method for calculating the basic measures and indicators.
	 */
	public void calculate() {
		XYPoint xyPoint  = new XYPoint();
		boolean isDate   = false;
		boolean isDouble = false;
		int decimalNumber = 4;
		int compare = -2;
		try {
			File file = new
File(Platform.getLocation()+"/"+edu.ksu.cis.ape.ProcessManagement.ProcessManagementView.getProcessComponentImpl().getNa
me().toString()+"//"+"process.xml");
			if (file.exists()) {
				DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
				DocumentBuilder db                  = dbf.newDocumentBuilder();
				Document doc                        = db.parse(file);

				doc.getDocumentElement().normalize();
				NodeList nodeLst = doc.getElementsByTagName("Element");
				for (int i = 0; i < nodeLst.getLength(); i++) {
					Element node = (Element) nodeLst.item(i);

					if (node.getAttribute("Type").indexOf("DeliveryProcessImpl") != -1) nameProject =
node.getAttribute("name");
					if (node.getAttribute("Type").indexOf("TaskDescriptorImpl") != -1) {
						// INITIALIZE VARS
						isDate = false; isDouble = false; compare = -2;
						// CHECK ESTIMATED VALUES
						isDate = isDate(node.getAttribute("schCom"));
						isDouble = isDouble(node.getAttribute("estEff"));
						// VALUE FOR BAC - BCWS
```

```
                                                if (isDate == true && isDouble == true) {
                                                        compare =
compareCurrentDate(node.getAttribute("schCom"));
                                                        if((compare == -1)||(compare == 0))     bcws = bcws +
Double.parseDouble(node.getAttribute("estEff"));
                                                        bac = bac + Double.parseDouble(node.getAttribute("estEff"));
                                                }
                                                // VALUE FOR GRAPH BCWS
                                                if (isDate == true && isDouble == true) {
                                                        xyPoint = new XYPoint();
                                                        double totalCost = totalCost(node.getAttribute("schCom"),
node.getAttribute("estEff"), "BCWS", i);
                                                        xyPoint.setDate(node.getAttribute("schCom"));
                                                        xyPoint.setCost(String.valueOf(totalCost));
                                                        arrBCWS.add(xyPoint);
                                                }
                                                // INITIALIZE VAR
                                                compare = -2;
                                                // VALUE FOR BCWP
                                                if (isDate == true && isDouble == true) {
                                                        if(node.getAttribute("Com").equals("true")) {
                                                                compare =
compareCurrentDate(node.getAttribute("schCom"));
                                                                if((compare == -1)||(compare == 0)) {
                                                                        // VALUE FOR GRAPH BCWP
                                                                        xyPoint = new XYPoint();
                                                                        double totalCost =
totalCost(node.getAttribute("schCom"), node.getAttribute("estEff"), "BCWP", i);
        xyPoint.setDate(node.getAttribute("actCom"));
                                                                        xyPoint.setCost(String.valueOf(totalCost));
                                                                        arrBCWP.add(xyPoint);
                                                                        // VALUE FOR BCWP
                                                                        if(bcwp < totalCost) bcwp = totalCost;
                                                                }
                                                        }
                                                }

                                                // INITIALIZE VARS
                                                isDate = false; isDouble = false;
                                                // CHECK REAL VALUES
                                                isDate = isDate(node.getAttribute("actCom"));
                                                isDouble = isDouble(node.getAttribute("actEff"));
                                                // INITIALIZE VAR
                                                compare = -2;
                                                // VALUE FOR ACWP
                                                if (isDate == true && isDouble == true) {
                                                        if(node.getAttribute("Com").equals("true")) {
                                                                compare =
compareCurrentDate(node.getAttribute("actCom"));
                                                                if((compare == -1)||(compare == 0)) {
                                                                        // VALUE FOR GRAPH ACWP
                                                                        xyPoint = new XYPoint();
                                                                        double totalCost =
totalCost(node.getAttribute("actCom"), node.getAttribute("actEff"), "ACWP", i);
        xyPoint.setDate(node.getAttribute("actCom"));
                                                                        xyPoint.setCost(String.valueOf(totalCost));
                                                                        arrACWP.add(xyPoint);
                                                                        // VALUE FOR ACWP
                                                                        if(acwp < totalCost) acwp = totalCost;
                                                                }
                                                        }
                                                }
```

```
                                        }//if (node.getAttribute("Type").indexOf("TaskDescriptorImpl") != -1) {
                                }//for (int s = 0; s < nodeLst.getLength(); s++) {

                                ev          = roundValue(bcwp/bac, decimalNumber);
                                spi = roundValue(bcwp/bcws, decimalNumber);
                                sv  = roundValue(bcwp-bcws, decimalNumber);
                                cpi = roundValue(bcwp/acwp, decimalNumber);
                                cv  = roundValue(bcwp-acwp, decimalNumber);
                        }//if (file.exists())
                }
                catch (Exception e) {
                        menssage = "Can not access the file process.xml";
                        System.out.println("Error in EarnedValueAnalysis.java - calculate() "+ e);
                }
}

/**
 * Method to determine if the value is type double.
 * @param value
 * @return boolean
 */
public static boolean isDouble(String value) {
        boolean isNumber = false;
        double number = 0;
        try {
                number = Double.parseDouble(value);
                if((number > 0)||(number <= 0)) isNumber = true;
        }
        catch (Exception e) {
                System.out.println("Error in EarnedValueAnalysis.java - isDouble() "+ e);
        }
        return isNumber;
}

/**
 * Method to determine if the value is type date.
 * @param value
 * @return isDate
 */
public static boolean isDate(String value) {
        boolean isDate = false;
        int month = 0;
        int day = 0;
        int year = 0;
        Calendar cal = GregorianCalendar.getInstance();
        try {
                String[] result = value.split("/");
                if(result.length == 3) {
                        month = edu.ksu.cis.ape.ProcessManagement.Date.Date.getMonthN(result[0]);
                        day   = Integer.parseInt(result[1]);
                        year  = Integer.parseInt(result[2]);
                        if((month > 0)&&(month <= 12)) {
                                if((day >= 1)&&(day <= 31)) {
                                        if((year >= 1900)&&(year <= 9999)) {
                                                cal.set(Calendar.YEAR, year);
                                                cal.set(Calendar.MONTH, month);
                                                cal.set(Calendar.DAY_OF_MONTH, day);
                                                isDate = true;
                                        }
                                }
                        }
                }
        }
        catch (Exception e) {
```

```
                System.out.println("Error in EarnedValueAnalysis.java - isDate() "+ e);
        }
        return isDate;
}

/**
 * Method to compare two dates.
 * @param date
 * @param dateTask
 * @return int
 * a.compareTo(b) = 1(a > b); = -1(a < b); = 0(a = b)
 */
public static int compareDates(String date, String dateTask){
        Calendar date1 = Calendar.getInstance();
        Calendar date2 = Calendar.getInstance();
        String optionMonth = "MMM";
        int compare = -2;
        try{
                date1 = changeDateType(date, optionMonth);
                date2 = changeDateType(dateTask, optionMonth);
                compare = date1.compareTo(date2);
        }
        catch (Exception e) {
                System.out.println("Error in EarnedValueAnalysis.java - compareDates() "+ e);
        }
        return compare;
}

/**
 * Method to compare a date with the current date
 * @param date
 * @return int
 * a.compareTo(b) = 1(a > b); = -1(a < b); = 0(a = b)
 */
public static int compareCurrentDate(String date) {
        java.util.Date time = new java.util.Date();
        SimpleDateFormat formatDate = new SimpleDateFormat("MM/dd/yyyy");
        Calendar date1 = Calendar.getInstance();
        Calendar date2 = Calendar.getInstance();
        String currentDate = "";
        int compare = -2;
        try{
                currentDate = formatDate.format(time);
                date1 = changeDateType(date, "MMM");
                date2 = changeDateType(currentDate, "MM");
                compare = date1.compareTo(date2);
        }
        catch (Exception e) {
                System.out.println("Error in EarnedValueAnalysis.java - compareCurrentDate() "+ e);
        }
        return compare;
}

/**
 * Method to convert String to Calendar.
 * @param date
 * @param optionMonth
 * @return Calendar
 */
public static Calendar changeDateType(String date, String optionMonth) {
        Calendar newDate = Calendar.getInstance();
        try {
                String[] resultDate = date.split("/");
                if(resultDate.length == 3) {
```

```
                                        if(optionMonth.equals("MMM")) newDate.set(Calendar.MONTH,
edu.ksu.cis.ape.ProcessManagement.Date.Date.getMonthN(resultDate[0])+1);
                                        if(optionMonth.equals("MM"))  newDate.set(Calendar.MONTH,
Integer.parseInt(resultDate[0]));
                                        newDate.set(Calendar.DAY_OF_MONTH, Integer.parseInt(resultDate[1]));
                                        newDate.set(Calendar.YEAR, Integer.parseInt(resultDate[2]));
                                }
                        }
                        catch (Exception e) {
                                System.out.println("Error in EarnedValueAnalysis.java - changeDateType() "+ e);
                        }
                        return newDate;
                }

                /**
                 * Method to calculate the cumulative cost for the activity.
                 * @param date
                 * @param cost
                 * @param option
                 * @param itemNumber
                 * @return
                 */
                public static double totalCost(String date, String cost, String option, int itemNumber) {
                        String dateTask = "";
                        String costTask = "";
                        int compare = -2;
                        double totalCost = 0;
                        try {
                                File file = new
File(Platform.getLocation()+"/"+edu.ksu.cis.ape.ProcessManagement.ProcessManagementView.getProcessComponentImpl().getNa
me().toString()+"//"+"process.xml");
                                if (file.exists()) {
                                        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
                                        DocumentBuilder db                      = dbf.newDocumentBuilder();
                                        Document doc                            = db.parse(file);

                                        doc.getDocumentElement().normalize();
                                        NodeList nodeLst = doc.getElementsByTagName("Element");
                                        for (int i = 0; i < nodeLst.getLength(); i++) {
                                                Element node = (Element) nodeLst.item(i);
                                                if (node.getAttribute("Type").indexOf("TaskDescriptorImpl") != -1) {
                                                        // INITIALIZE VARS
                                                        dateTask = ""; costTask = ""; compare = -2;
                                                        if(itemNumber != i) {
                                                                if(option.equals("BCWS")) {
                                                                        dateTask = node.getAttribute("schCom");
                                                                        costTask = node.getAttribute("estEff");
                                                                        // COMPARE DATES
                                                                        compare = compareDates(date, dateTask);
                                                                        // TOTAL ESTIMATED COST
                                                                        if((compare == 0)||(compare == 1)) totalCost =
totalCost + Double.parseDouble(costTask);
                                                                }
                                                                // INITIALIZE VAR
                                                                compare = -2;
                                                                // ESTIMATED VALUES
                                                                if(option.equals("BCWP")) {
                                                                        if(node.getAttribute("Com").equals("true")) {
                                                                                dateTask = node.getAttribute("schCom");

                                                                                costTask = node.getAttribute("estEff");
                                                                                // COMPARE WITH CURRENT DATE
                                                                                compare =
compareCurrentDate(dateTask);
```

```
                                                                        // CALCULATE TOTAL COST
                                                                        if((compare == -1)||(compare == 0)) {
                                                                                compare = -2;
                                                                                compare = compareDates(date,
dateTask);

                                                                                if((compare == 0)||(compare ==
1)) totalCost = totalCost + Double.parseDouble(costTask);
                                                                        }
                                                                }
                                                        }
                                                        // INITIALIZE VAR
                                                        compare = -2;
                                                        // REAL VALUES
                                                        if(option.equals("ACWP")) {
                                                                if(node.getAttribute("Com").equals("true")) {
                                                                        dateTask = node.getAttribute("actCom");

                                                                        costTask = node.getAttribute("actEff");
                                                                        // COMPARE WITH CURRENT DATE
                                                                        compare =
compareCurrentDate(dateTask);
                                                                        // CALCULATE TOTAL COST
                                                                        if((compare == -1)||(compare == 0)) {
                                                                                compare = compareDates(date,
dateTask);

                                                                                if((compare == 0)||(compare ==
1)) totalCost = totalCost + Double.parseDouble(costTask);
                                                                        }
                                                                }
                                                        }
                                                }//if(itemNumber != i)
                                        }
                                }//for (int i = 0; i < nodeLst.getLength(); i++)
                                totalCost = totalCost + Double.parseDouble(cost);
                        }//if (file.exists())
                }
                catch (Exception e) {
                        System.out.println("Error in EarnedValueAnalysis.java - totalCost() "+ e);
                }
                return totalCost;
        }
}
```

2. Clase EVAGraphicsView

```
package edu.ksu.cis.ape.ProcessManagement;
import java.awt.Color;
import java.awt.Frame;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.List;
import org.eclipse.swt.SWT;
import org.eclipse.swt.awt.SWT_AWT;
import org.eclipse.swt.events.SelectionAdapter;
import org.eclipse.swt.events.SelectionEvent;
import org.eclipse.swt.layout.GridData;
import org.eclipse.swt.layout.GridLayout;
import org.eclipse.swt.widgets.Button;
import org.eclipse.swt.widgets.Composite;
import org.eclipse.swt.widgets.Label;
import org.eclipse.swt.widgets.Text;
```

```
import org.eclipse.ui.forms.widgets.Section;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.DateAxis;
import org.jfree.chart.plot.XYPlot;
import org.jfree.chart.renderer.xy.XYItemRenderer;
import org.jfree.chart.renderer.xy.XYLineAndShapeRenderer;
import org.jfree.chart.renderer.xy.XYSplineRenderer;
import org.jfree.data.time.Day;
import org.jfree.data.time.TimeSeries;
import org.jfree.data.time.TimeSeriesCollection;
import org.jfree.data.xy.XYDataset;
import org.jfree.ui.RectangleInsets;
import com.ibm.icu.util.Calendar;
import edu.ksu.cis.agenttool.core.editor.AbstractPropertiesView;

/**
 *
 * @author Andrea P. García Prada
 *
 */

/**
 *  EVAGraphicsView create the properties of the graphical view, for which information is displayed to the user.
 */
public class EVAGraphicsView extends AbstractPropertiesView {

	/**
	 * Class Attributes
	 */
	protected Text spiText, svText, acwpText, cpiText, cvText;
	protected Button acepptButton;
	protected Label label;
	protected Frame chartFrame;
	protected ChartPanel chartPanel;
	protected static String nameProject ;
	protected static int optionChart ;
	protected static TimeSeriesCollection dataset;
	protected static XYSplineRenderer spline = new XYSplineRenderer();

	/**
	 * Constructor class
	 */
	public EVAGraphicsView() {
		nameProject = "";
		optionChart = 0;
	}

	/**
	 * @see org.eclipse.ui.part.WorkbenchPart#createPartControl(Composite)
	 *
	 * @param parent
	 */
	@Override
	public void createPartControl(Composite parent) {
		super.createPartControl(parent);
		Composite form = toolkit.createComposite(parent);
		form.setLayout(new GridLayout(1, true));
		// Create a section for properties
		createPropertiesSection(form);
	}

	/**
```

```
	 * Creates a new properties section in the given composite.
	 *
	 * @param parent
	 */
	private void createPropertiesSection(final Composite parent) {
		EarnedValueAnalysis eva = new EarnedValueAnalysis();
		eva.calculate();
		nameProject = eva.getNameProject();

		// Properties Section N° 1
		Section propertiesSection = toolkit.createSection(parent, SWT.DEFAULT);
		propertiesSection.setText("Progress Indicators");
		propertiesSection.setLayout(new GridLayout());
		propertiesSection.setLayoutData(new GridData(SWT.FILL, SWT.FILL, true, false));

		Composite propertiesComp = toolkit.createComposite(propertiesSection);
		propertiesComp.setLayout(new GridLayout(8, false));
		propertiesSection.setClient(propertiesComp);

		toolkit.createLabel(propertiesComp, "Schedule Performance Index (SPI): ");
		spiText = toolkit.createText(propertiesComp, "", SWT.SINGLE);
		spiText.setEditable(false);
		spiText.setText(Double.toString(eva.getSPI()));
		if(eva.getSPI() < 1)  spiText.setForeground(new org.eclipse.swt.graphics.Color(null,255,0,0));
		if(eva.getSPI() == 1) spiText.setForeground(new org.eclipse.swt.graphics.Color(null,238,96,0));
		if(eva.getSPI() > 1)  spiText.setForeground(new org.eclipse.swt.graphics.Color(null,0,102,0));

		toolkit.createLabel(propertiesComp, "Schedule Variance (SV): ");
		svText = toolkit.createText(propertiesComp, "", SWT.SINGLE);
		svText.setEditable(false);
		svText.setText(Double.toString(eva.getSV()));
		if(eva.getSV() < 0)  svText.setForeground(new org.eclipse.swt.graphics.Color(null,255,0,0));
		if(eva.getSV() == 0) svText.setForeground(new org.eclipse.swt.graphics.Color(null,238,96,0));
		if(eva.getSV() > 0)  svText.setForeground(new org.eclipse.swt.graphics.Color(null,0,102,0));

		toolkit.createLabel(propertiesComp, "Cost Performance Index (CPI): ");
		cpiText = toolkit.createText(propertiesComp, "", SWT.SINGLE);
		cpiText.setEditable(false);
		cpiText.setText(Double.toString(eva.getCPI()));
		if(eva.getCPI() < 1)  cpiText.setForeground(new org.eclipse.swt.graphics.Color(null,255,0,0));
		if(eva.getCPI() == 1) cpiText.setForeground(new org.eclipse.swt.graphics.Color(null,238,96,0));
		if(eva.getCPI() > 1)  cpiText.setForeground(new org.eclipse.swt.graphics.Color(null,0,102,0));

		toolkit.createLabel(propertiesComp, "Cost Variance (CV): ");
		cvText = toolkit.createText(propertiesComp, "", SWT.SINGLE);
		cvText.setEditable(false);
		cvText.setText(Double.toString(eva.getCV()));
		if(eva.getCV() < 0)  cvText.setForeground(new org.eclipse.swt.graphics.Color(null,255,0,0));
		if(eva.getCV() == 0) cvText.setForeground(new org.eclipse.swt.graphics.Color(null,238,96,0));
		if(eva.getCV() > 0)  cvText.setForeground(new org.eclipse.swt.graphics.Color(null,0,102,0));

		// Properties Section N° 2
		final Section chartSection = toolkit.createSection(parent, SWT.DEFAULT);
		chartSection.setText("Earned Value Analysis - Project Vision");
		chartSection.setLayout(new GridLayout());
		chartSection.setLayoutData(new GridData(SWT.FILL, SWT.FILL, true, true));

		if(!eva.getMenssage().equals("")) {
			chartSection.setLayoutData(new GridData(SWT.FILL, SWT.FILL, true, false));
			Composite chartComposite = toolkit.createComposite(chartSection);
			chartComposite.setLayout(new GridLayout(1, false));
			chartSection.setClient(chartComposite);
			toolkit.createLabel(chartComposite, eva.getMenssage()).setForeground(new
org.eclipse.swt.graphics.Color(null,255,0,0));
```

```
		}
		else
		{
			JFreeChart chart = getChart(eva);
			optionChart = 0;

			final Composite chartComposite = new Composite(chartSection, SWT.NONE | SWT.EMBEDDED);
			chartComposite.setLayout(new GridLayout(1, false));
			chartSection.setClient(chartComposite);
			chartFrame = SWT_AWT.new_Frame(chartComposite);
			chartPanel = new ChartPanel(chart);
			chartFrame.add(chartPanel);

			acepptButton = new Button(propertiesComp, SWT.PUSH);
			acepptButton.setText("  Spline  ");
			acepptButton.addSelectionListener(new SelectionAdapter() {
				public void widgetSelected(SelectionEvent e){
					if(optionChart == 0) optionChart = 1;
					else
					{
						if(optionChart == 1) optionChart = 2;
						else if(optionChart == 2) optionChart = 1;
					}
					splineChart();
				}
			});
		}
		toolkit.createLabel(propertiesComp, "");
	}

	/**
	 * Method to get the graph created.
	 *
	 * @param eva Is an Instance of the class EarnedValueAnalysis
	 * @return chart
	 */
	private static JFreeChart getChart(EarnedValueAnalysis eva) {
		XYDataset dataset = createDataset(eva);
		JFreeChart chart = createChart(dataset);
		return chart;
	}

	/**
	 * Creates the dataset, with the series to graph.
	 *
	 * @param eva Is an Instance of the class EarnedValueAnalysis
	 * @return dataset
	 */
	//private static XYDataset createDataset()
	private static XYDataset createDataset(EarnedValueAnalysis eva) {
		dataset = new TimeSeriesCollection();
		XYPoint xyPoint 	= new XYPoint();
		List<XYPoint> arrBCWS = new ArrayList<XYPoint>();
		List<XYPoint> arrBCWP = new ArrayList<XYPoint>();
		List<XYPoint> arrACWP = new ArrayList<XYPoint>();
		Calendar calen = Calendar.getInstance();
		double lastValue = 0;

		try {
			arrBCWS = eva.getArrBCWS();
			arrBCWP = eva.getArrBCWP();
			arrACWP = eva.getArrACWP();
			// CREATE SERIE BCWS
			TimeSeries serieBCWS = new TimeSeries("BCWS");
```

```
for (int i = 0; i < arrBCWS.size(); i++) {
		xyPoint = new XYPoint();
		xyPoint = (XYPoint) arrBCWS.get(i);
		String[] result = xyPoint.getDate().split("/");
		int day   = Integer.parseInt(result[1]);
		int month = edu.ksu.cis.ape.ProcessManagement.Date.Date.getMonthN(result[0])+1;
		int year  = Integer.parseInt(result[2]);
		double cost = Double.parseDouble(xyPoint.getCost());
		int compare = EarnedValueAnalysis.compareCurrentDate(xyPoint.getDate());
		if((compare == -1)||(compare == 0)) {
			if(lastValue < cost) lastValue = cost;
		}
		serieBCWS.addOrUpdate(new Day(day,month,year), cost);
}

serieBCWS.addOrUpdate(new
Day(calen.get(Calendar.DAY_OF_MONTH),calen.get(Calendar.MONTH)+1,calen.get(Calendar.YEAR)), lastValue);
lastValue = 0;

// CREATE SERIE BCWS
TimeSeries serieBCWP = new TimeSeries("BCWP");
for (int i = 0; i < arrBCWP.size(); i++) {
		xyPoint = new XYPoint();
		xyPoint = (XYPoint) arrBCWP.get(i);
		String[] result = xyPoint.getDate().split("/");
		int day   = Integer.parseInt(result[1]);
		int month = edu.ksu.cis.ape.ProcessManagement.Date.Date.getMonthN(result[0])+1;
		int year  = Integer.parseInt(result[2]);
		double cost = Double.parseDouble(xyPoint.getCost());
		int compare = EarnedValueAnalysis.compareCurrentDate(xyPoint.getDate());
		if((compare == -1)||(compare == 0)) {
			if(lastValue < cost) lastValue = cost;
		}

		serieBCWP.addOrUpdate(new Day(day,month,year), cost);
}

serieBCWP.addOrUpdate(new
Day(calen.get(Calendar.DAY_OF_MONTH),calen.get(Calendar.MONTH)+1,calen.get(Calendar.YEAR)), lastValue);
lastValue = 0;

// CREATE SERIE ACWP
TimeSeries serieACWP = new TimeSeries("ACWP");
for (int i = 0; i < arrACWP.size(); i++)
{
		xyPoint = new XYPoint();
		xyPoint = (XYPoint) arrACWP.get(i);
		String[] result = xyPoint.getDate().split("/");
		int day   = Integer.parseInt(result[1]);
		int month = edu.ksu.cis.ape.ProcessManagement.Date.Date.getMonthN(result[0])+1;
		int year  = Integer.parseInt(result[2]);
		double cost = Double.parseDouble(xyPoint.getCost());
		int compare = EarnedValueAnalysis.compareCurrentDate(xyPoint.getDate());
		if((compare == -1)||(compare == 0)) {
			if(lastValue < cost) lastValue = cost;
		}
		serieACWP.addOrUpdate(new Day(day,month,year),
Double.parseDouble(xyPoint.getCost()));
		lastValue = Double.parseDouble(xyPoint.getCost());
}

serieACWP.addOrUpdate(new
Day(calen.get(Calendar.DAY_OF_MONTH),calen.get(Calendar.MONTH)+1,calen.get(Calendar.YEAR)), lastValue);
```

```
                    // ASSING THE DATASET SERIES
                    dataset.addSeries(serieBCWS);
                    dataset.addSeries(serieBCWP);
                    dataset.addSeries(serieACWP);
            }
            catch (Exception e) {
                    System.out.println("Error in EVAGraphicsView.java - createDataset() "+ e);
            }

            return dataset;
}

/**
 * Create a chart.
 *
 * @param dataset
 * @return A chart.
 */
private static JFreeChart createChart(XYDataset dataset) {
            JFreeChart chart = ChartFactory.createTimeSeriesChart(
                                                "Project: "+nameProject,      // Title
                                                "TIME (t)",                   // x-axis label
                                                "COST ($)",                   // y-axis label
                                                dataset,                      // data
                                                true,                         // Create legend?
                                                true,                         // Generate tooltips?
                                                false                         // Generate URLs?
            );

            // CHART PROPERTIES
            chart.setBackgroundPaint(Color.white);
            XYPlot plot = (XYPlot) chart.getPlot();
            plot.setBackgroundPaint(Color.lightGray);
            plot.setDomainGridlinePaint(Color.white);
            plot.setRangeGridlinePaint(Color.white);
            plot.setAxisOffset(new RectangleInsets(5.0, 5.0, 5.0, 5.0));
            plot.setDomainCrosshairVisible(true);
            plot.setRangeCrosshairVisible(true);

            // DEFINE GRAPHING POINTS AND COLORS
            XYItemRenderer r = plot.getRenderer();
            if (r instanceof XYLineAndShapeRenderer) {
                    XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer) r;
                    renderer.setBaseShapesVisible(true);
                    renderer.setBaseShapesFilled(true);
                    renderer.setSeriesPaint(0, Color.red);
                    renderer.setSeriesPaint(1, Color.blue);
                    renderer.setSeriesPaint(2, Color.black);
            }

            // DATE FORMAT
            DateAxis axis = (DateAxis) plot.getDomainAxis();
            axis.setVisible(true);
            axis.setDateFormatOverride(new SimpleDateFormat("dd-MMM-yyyy"));

            return chart;
}

/**
 * Method to smooth the generated graph.
 */
public void splineChart() {
            JFreeChart chart = createChart(dataset);
            if(optionChart == 1) {
```

```
                                acepptButton.setText("  Line  ");
                                XYPlot plot = (XYPlot) chart.getPlot();
                                plot.setRenderer(spline);
                                XYItemRenderer r = plot.getRenderer();
                                if (r instanceof XYLineAndShapeRenderer) {
                                        XYLineAndShapeRenderer renderer = (XYLineAndShapeRenderer) r;
                                        renderer.setBaseShapesVisible(true);
                                        renderer.setBaseShapesFilled(true);
                                        renderer.setSeriesPaint(0, Color.red);
                                        renderer.setSeriesPaint(1, Color.blue);
                                        renderer.setSeriesPaint(2, Color.black);
                                }
                        }
                        else acepptButton.setText("  Spline  ");
                        chartPanel.setChart(chart);
                }
        }
```

3. Clase XYPoint

```
package edu.ksu.cis.ape.ProcessManagement;

/**
 *
 * @author Andrea P. García Prada
 *
 */

/**
 * Class to save the values of the points x,y to plot.
 */
public class XYPoint {
        /**
         * Class Attributes
         */
        private String date;
        private String cost;

        /**
         * Constructor class
         */
        public XYPoint(){

        }

        /**
         * Getters and setters
         */
        public String getDate() {
                return date;
        }
        public void setDate(String date) {
                this.date = date;
        }
        public String getCost() {
                return cost;
        }
        public void setCost(String cost) {
                this.cost = cost;
        }
}
```

4. Clase ProcessManagementView

En la clase ProcessManagementView.java se realizan las siguientes modificaciones:

- Incluir el siguiente código fuente:

  Línea 102: `private Action action4;`

  Línea 379: `manager.add(action4);`

  Línea 386: `manager.add(action4);`

  Línea 397: `manager.add(action4);`

  Línea 520:

```
action4 = new Action(){
                public void run(){
                        IWorkbench workbench = PlatformUI.getWorkbench();
            IWorkbenchWindow workbenchWindow = workbench.getActiveWorkbenchWindow();
                        try {

    workbenchWindow.getActivePage().showView("edu.ksu.cis.ape.ProcessManagement.EVAGraphi
csView");
                        } catch (Exception e) {
                                System.out.println("Error en action4 run(): "+e);
                        }
                }
        };

        action4.setText("EVA Graphics");
        action4.setToolTipText("EVA Graphics");
        descriptor = Activator.getImageDescriptor("chart1.jpg");
        action4.setImageDescriptor(descriptor);
```

- Modificar la acción para el botón N° 2, al reemplazar el código fuente que se encontraba especificado a partir de la línea 460 hasta la línea 513, por el siguiente código fuente:

```
EarnedValueAnalysis eva = new EarnedValueAnalysis();
eva.calculate();
System.out.println("Budget at Completion             (BAC) :="+eva.getBAC());
System.out.println("Budgeted Cost of Work Performed (BCWP) :="+eva.getBCWP());
System.out.println("Earned Value                      (EV) :="+eva.getEV());
System.out.println("Budgeted Cost of Work Scheduled (BCWS) :="+eva.getBCWS());
System.out.println("Schedule Performance Index       (SPI) :="+eva.getSPI());
System.out.println("Schedule Variance                 (SV) :="+eva.getSV());
System.out.println("Actual Cost of Work Performed   (ACWP) :="+eva.getACWP());
System.out.println("Cost Performance Index           (CPI) :="+eva.getCPI())
System.out.println("Cost Variance                     (CV) :="+eva.getCV());
```

5. Clase EarnedValueAnalysisView

En la clase EarnedValueAnalysisView.java se modificó el método ***public void populateFields(String pname)*** al reemplazar el código fuente que se encontraba especificado a partir de la línea 75 hasta la línea 129, por el siguiente código fuente:

```
EarnedValueAnalysis eva = new EarnedValueAnalysis();
eva.calculate();
bacText.setText(String.valueOf(eva.getBAC()));
bcwpText.setText(String.valueOf(eva.getBCWP()));
evText.setText(String.valueOf(eva.getEV()));
bcwsText.setText(String.valueOf(eva.getBCWS()));
spiText.setText(String.valueOf(eva.getSPI()));
svText.setText(String.valueOf(eva.getSV()));
acwpText.setText(String.valueOf(eva.getACWP()));
cpiText.setText(String.valueOf(eva.getCPI()));
cvText.setText(String.valueOf(eva.getCV()));
```