Universidad de las Ciencias Informáticas

Facultad 5

Título:

Propuesta de herramienta CASE para los proyectos del Centro de Desarrollo de Informática Industrial (CEDIN).

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

Autores:
Elizabeth De la Cruz Rodríguez
Daliamny Guzmán Hernández

Tutor:
Ing. Maikel Pérez Javier

Co-tutor:
Ing. Antonio Cedeño Pozo

Ciudad de La Habana, Cuba
Junio, 2010
"Año 52 de la Revolución"

Aquella teoría que no encuentre aplicación práctica en la vida, es una acrobacia del pensamiento.

Swami Vivekananda

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

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Daliamny Guzmán Hernández

_____

Firma del Autor

Elizabeth De la Cruz Rodríguez

_____

Firma del Autor

Maikel Pérez Javier

_____

Firma del Tutor

Antonio Cedeño Pozo

_____

Firma del Co-Tutor

AGRADECIMIENTOS

Especialmente a:

Mis padres por ser lo más grande que me ha dado la vida, por todo su apoyo y comprensión.

Mis tíos Amable y Amparo por ser mis segundos padres, por tratarme desde que nací como una hija.

Mi novio por cada ayuda que me brindó, por estar a mi lado en las buenas y en las malas, por su paciencia y por hacer suyo este trabajo.

Maray y a Michel por toda su ayuda y cariño.

Mis amigas Elizabeth, Yeny, Isandra y Oadis, por aguantarme todo este tiempo y por estar siempre presentes para mí.

Mi tutor y Co -Tutor por la ayuda brindada.

A la profe Yadira Ramírez por su ayuda incondicional.

A todos los que de alguna forma u otra me han ayudado a llegar a este día.

Daliamny

Especialmente a:

Mi familia por estar siempre a mi lado, a mis padres por ser la verdadera razón de mi existencia y darme a cada paso de mi vida su alma entera.

Mis amigas Yeni, Daliamny, Oadis e Isandra por brindarme en todo momento su apoyo incondicional.

A mí cuñado Valentín por su constante apoyo y comprensión.

Mi tutor y Co -Tutor por la ayuda brindada en la realización de este trabajo.

A la profe Yadira Ramírez por su ayuda incondicional.

Todas mis amistades que de una forma u otra tuvieron participación en todos los momentos vividos a lo largo de la carrera.

Elizabeth

DEDICATORIA

A mis padres por el esfuerzo de toda una vida, por su confianza y su cariño, por ser un ejemplo de sacrificio y por querer siempre lo mejor para mí.

A mis tíos por cuidarme siempre.

A mi cosi, ¨Valentín¨, por ser para mí un hombre excepcional, mi guía a seguir, y el amor de mi vida.

A toda mi familia.

A todos los que de alguna forma ayudaron a la realización de este trabajo.

Daliamny

A mis padres María Elena y Francisco por todo su amor y dedicación, por tanto cariño y confianza depositada en mí, por apoyarme en mis decisiones, por hacer hasta lo imposible por la realización de mis sueños.

Gracias de todo corazón.

A mi abuelo Manuel por quererme tanto desde pequeña.

A mis tías Leonor y Esther por ayudarme tanto durante todos estos años.

A toda mi familia en general por alentarme y darme fuerzas para seguir adelante.

Elizabeth

RESUMEN

En la actualidad existe gran variedad de herramientas CASE, su objetivo es apoyar a los especialistas de software en el desarrollo del ciclo de vida de proyectos de software.

Los proyectos de software que pertenecen al Centro de Desarrollo de Informática Industrial (CEDIN), necesitan contar con una herramienta CASE que les permita hacer más eficiente y coherente el proceso de modelado de sus soluciones de software, ya que actualmente utilizan tres tecnologías cuyo uso simultáneo impide que se usen al máximo sus potencialidades, trayendo como consecuencia que se generen artefactos con incompatibilidad de formatos obstaculizando la generación de elementos persistentes.

Para la selección de la herramienta más conveniente se describe el estado del arte de las tecnologías más empleadas actualmente por grandes empresas desarrolladoras de software. A partir de las necesidades identificadas en el centro, se seleccionaron una serie de indicadores tecnológicos con sus respectivas variables de medición, que brindaron una forma cuantitativa de comparar dichas herramientas, llevando a cabo la evaluación de los mismos a partir de un modelo de decisión que facilitó la elección del NetBeans con plugins para UML como la herramienta más adecuada para el modelado de las soluciones de software en el centro. Se elaboró un caso de estudio donde se muestran las facilidades de diagramación de la herramienta y se propone un diseño con las posibles mejoras a partir de la facilidad de extensión que presenta la misma.

Por último, se valida la propuesta con la dirección técnica del Centro de Desarrollo de Informática Industrial y especialistas en el tema.

PALABRAS CLAVE

Herramientas CASE, indicadores tecnológicos, modelo de decisión, extensión.

ÍNDICE

AGRADECIMIENTOS IV
DEDICATORIA V
RESUMEN VI
INTRODUCCIÓN 9
1. **CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA** 13
1.1 Introducción. 13
1.2 Lenguaje Unificado de Modelado (UML). 13
1.2.1 Evolución de UML 13
1.3 Definición de herramienta CASE. 15
1.4 Evolución de las herramientas CASE. 16
1.5 Objetivos de las herramientas CASE. 16
1.6 Clasificación de las herramientas CASE. 16
1.7 Beneficios ofrecidos por la tecnología CASE. 17
1.8 Buenas prácticas de las herramientas CASE. 18
1.9 Tecnologías y tendencias actuales. 19
1.9.1 ArgoUML. 19
1.9.2 BOUML. 20
1.9.3 Visual Paradigm For UML. 20
1.9.4 Eclipse. 21
1.9.5 Umbrello UML Modeller 21
1.9.6 Rational Software Architect 22
1.9.7 NetBeans 22
1.9.8 StarUML 23
1.9.9 DIA 23
1.9.10 Rational Rose 24
1.9.11 Poseidon for UML 24
1.10 Conclusiones del capítulo. 25
2. **CAPÍTULO II. ANÁLISIS Y SELECCIÓN DE LA HERRAMIENTA CASE** 26
2.1 Introducción. 26
2.2 Estudio Comparativo Inicial. 26
2.3 Fase de Aplicación del Modelo de Decisión. 28
2.3.1 Modelo de Decisión. 29
2.3.2 Fase de resultados. 35
2.3.3 Análisis de los resultados. 36
2.4 Introducción al desarrollo de diagramas UML con NetBeans. 37
2.4.1 Creación de un proyecto de modelado UML. 38
2.5 Caso de Estudio. Sistema de Seguridad de SCADA. 41

2.5.1 Realización del Diagrama de Casos de Uso del Negocio. 42
2.5.2 Realización del Diagrama de Casos de Uso del Sistema. 43
2.5.3 Realización del Diagrama de Actividades. 43
2.5.4 Realización del Diagrama de Clases del Análisis. 44
2.5.5 Realización del Diagrama de Clases para aplicaciones de escritorio. 44
2.5.6 Realización del Diagrama Secuencia. 45
2.5.7 Realización del Diagrama de Colaboración. 46
2.5.8 Realización del Diagrama de Componentes a nivel de Subsistemas de Aplicación. 46
2.5.9 Realización del Diagrama de Componentes del Sistema. 47
2.5.10 Realización del Diagrama de Despliegue. 47
2.6 Entrevista a líderes de proyectos del CEDIN 48
2.7 Conclusiones del capítulo. 49

3. **CAPÍTULO III. VALIDACIÓN DE LA PROPUESTA** 50
3.1 Introducción. 50
3.2 NetBeans UML como propuesta de solución. 50
3.2.1 Arquitectura de NetBeans. 50
3.2.2 Configuración de los módulos de NetBeans. 51
3.2.3 ¿Cómo hacer un módulo NetBeans? 52
3.3 Análisis de la arquitectura del plugin UML. 62
3.3.1 Descripción del módulo org.netbeans.modules.uml.core. 64
3.3.2 Interfaz de usuario y elementos gráficos. 66
3.3.3 Estrategia de extensión. 68
3.3.4 Propuesta de modificación. 68
3.3.4.1 Ejemplo de modificación 69
3.4 Evaluación de la viabilidad de la propuesta, sobre la base del criterio de expertos. Método Delphi. 72
3.4.1 Elección de expertos. 73
3.4.2 Conformación del cuestionario, para validación de la propuesta. 73
3.4.3 Desarrollo práctico y explotación de resultados. 73
3.4.3.1 Concordancia de los expertos. 75
3.5 Conclusiones del capítulo 77
CONCLUSIONES 78
RECOMENDACIONES 79
REFERENCIAS BIBLIOGRÁFICAS 80
BIBLIOGRAFÍA 83
ANEXOS 85
GLOSARIO DE TÉRMINOS 95

INTRODUCCIÓN

En la actualidad el desarrollo del software en Cuba está llamado a convertirse en una significativa fuente de ingresos. Con el avance de las Tecnologías de la Informática y las Comunicaciones (TIC) se pretende desarrollar aún más los sectores económicos del país y la calidad de vida de los ciudadanos.

En su afán por lograr estos avances, surge al calor de la Batalla de Ideas la Universidad de las Ciencias Informáticas (UCI), con el objetivo de enfrentar los retos del mundo informático actual, desarrollando productos de software tanto para el ámbito nacional como internacional, a partir de la vinculación del estudio-trabajo-investigación como modelo de formación y asegurar la informatización de cada rincón del país donde sea necesario.

Con la finalidad de ofrecer solución a las diversas necesidades de automatización del país, surge el Centro de Desarrollo de Informática Industrial, el cual agrupa un conjunto de proyectos productivos encaminados al uso de las herramientas libres, donde se interrelacionan estudiantes y profesores, los cuales deben contar con un alto grado de destreza, experiencia y formación que les permita desarrollar las actividades productivas de su entorno, investigando, produciendo software y servicios informáticos para la sociedad. Para llevar a cabo un procedimiento eficaz para sus estrategias de negocio, este centro necesita herramientas que sean capaces de responder a los intereses del mismo.

Entre estas herramientas se encuentran las herramientas CASE (del inglés Computer Aided Software Engineering), que son un conjunto de técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información. Éstas han sido creadas con mucha precisión para beneficiar a los analistas y desarrolladores, pero en ocasiones se hace muy difícil encontrar una que satisfaga todas las necesidades en una empresa, es por esto que a la hora de ser implantadas se debe proceder con mucho cuidado, realizando un estudio previo de la situación actual de la institución, para obtener la más adecuada.

Los proyectos de software que pertenecen al Centro de Desarrollo de Informática Industrial (CEDIN), actualmente utilizan tres herramientas CASE para el modelado de sistemas o productos de software, éstas son Visual Paradigm, Rational Software Architect y NetBeans con la extensión UML. El uso simultáneo de dichas tecnologías impide el máximo aprovechamiento de sus potencialidades, así como la reutilización de los diferentes artefactos que se generan con estas herramientas, éste último aspecto tiene que ver con la incompatibilidad de formatos para la generación de elementos

persistentes. Los elementos mencionados evidencian que en el Centro de Desarrollo de Informática Industrial no existe una línea común para la realización de los procesos de Ingeniería de Software, dando lugar a demoras e insatisfacciones tanto por parte de los especialistas y desarrolladores como por parte de los clientes que requieren soluciones de diferentes líneas de componentes.

El Visual Paradigm y el Rational Software Architect son las herramientas más utilizadas en el centro, y tienen cualidades que satisfacen a medias el proceso, debido a que presentan un conjunto de características que no cumplen con las expectativas del centro; son software privativos, por lo que para adquirirlos es necesario pagar costosas licencias; existe la versión Visual Paradigm UML Community para proyectos de código abierto, libre de costos, que para obtenerlo existen severos requerimientos, tales como:

- Solo para proyectos con fines académicos.
- No válida para proyectos de uso comercial.

Esta versión del Visual Paradigm no incorpora un grupo de funcionalidades que aparecen en la versión comercial del producto.

A partir de la situación problemática planteada se define el siguiente **problema a resolver:** ¿Cómo facilitar el modelado de las soluciones de Software en el Centro de Desarrollo de Informática Industrial?

Para lograr este propósito se identificó como **objeto de estudio** las herramientas para el modelado de soluciones del Software.

Para resolver el problema se define como **objetivo general** elegir una herramienta CASE adecuada para el Centro de Desarrollo de Informática Industrial y proponer un diseño con las posibles mejoras.

El **campo de acción** de esta investigación, está centrado en las herramientas CASE basadas en tecnologías libres.

Como **idea a defender** de esta investigación se plantea: La selección de una herramienta CASE libre y extensible, así como la propuesta de personalización, lograrán facilitar el modelado de las soluciones del software y adaptar la herramienta teniendo en cuenta las necesidades del Centro de Desarrollo de Informática Industrial.

Fueron definidas las siguientes tareas de investigación:

- Estudio y análisis de los diferentes tipos de herramientas CASE existentes para fundamentar el estado del arte.
- Comparación de las herramientas en cuanto a funcionalidades y características de extensibilidad.
- Selección de la herramienta CASE más adecuada para usarla en el Centro de Desarrollo de Informática Industrial.
- Propuesta de un diseño donde se muestre como extender o modificar la herramienta seleccionada.
- Validación de la propuesta de solución con el grupo técnico del Centro de Desarrollo de Informática Industrial y especialistas en el tema.

Durante la investigación, los métodos científicos teóricos empleados fueron:

Métodos Teóricos:

Analítico – Sintético: Se utilizó para la selección de información sobre las herramientas CASE permitiendo la extracción de los elementos más importantes.

Histórico – Lógico: Se utilizó para especificar el desarrollo de las herramientas CASE y su evolución hasta la actualidad.

Método Empírico:

Entrevista: Se le aplicó a analistas y desarrolladores del centro con el objetivo de determinar los problemas reales, las necesidades y características del mismo.

Encuesta: Se le realizó a un grupo de expertos del CEDIN permitiendo la validación de la propuesta.

El presente trabajo está estructurado en tres capítulos, los cuales se muestran a continuación:

Capítulo 1: Fundamentación Teórica: Se realiza un estudio de los conceptos asociados a las herramientas CASE, donde se abordan sus características más importantes, objetivos, importancia en el desarrollo de software, así como su evolución hasta la actualidad. Además se identifican las herramientas más utilizadas en la actualidad por las grandes compañías productoras de software y se analiza cada una de estas herramientas.

Capítulo 2: Análisis y Selección de la Herramienta CASE: Se comparan las herramientas seleccionadas en cuanto a sus características y funcionalidades mediante un modelo de decisión, facilitando la selección de una herramienta candidata para el centro y se muestra mediante un caso de estudio las capacidades de diagramación de dicha herramienta.

Capítulo 3: Validación de la Propuesta: Se identifican las características de la herramienta

seleccionada que permiten su extensión, se propone un diseño donde se muestra como extenderla o modificarla y se valida la propuesta con la dirección técnica del Centro de Desarrollo de Informática Industrial.

CAPÍTULO I. FUNDAMENTACIÓN TEÓRICA

1.1 Introducción.

En el presente capítulo, se abordará el estado del arte de los conceptos asociados a las herramientas CASE en este caso las de soporte a UML, se realizará un análisis de las principales características de las mismas y se explicará los objetivos e importancia en el desarrollo de sistemas de software, efectuando un análisis más profundo en aquellas herramientas que más se han utilizado por proyectos de gran magnitud de todos los tiempos.

1.2 Lenguaje Unificado de Modelado (UML).

El UML o Lenguaje Unificado de Modelado es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de los sistemas de software y también para otros tipos de sistemas. Representa una colección de las mejores prácticas de ingeniería que han sido probadas con éxito en el modelado de sistemas grandes y complejos. Se convirtió en estándar del Object Management Group (OMG) en 1997, después de tres años de trabajo, como consecuencia de la llamada Guerra de las Metodologías. Es un lenguaje gráfico, que puede ser usado en todas las fases de desarrollo de software y que permite representar los sistemas con varios modelos parciales facilitando su entendimiento y comunicación. [1] [2] [3]

1.2.1 Evolución de UML

Impulsados por la necesidad de métodos más ágiles de desarrollo de software, UML 1.x evoluciona en UML 2.0, el cual ofrece a los proveedores de herramientas CASE las características necesarias que permitirán cumplir “la producción automática de programas basado en especificaciones del software”.

OMG UML 1.x está constituido por siete diagramas básicos y dos diagramas que constituyen variaciones de los básicos (ver figura 1):

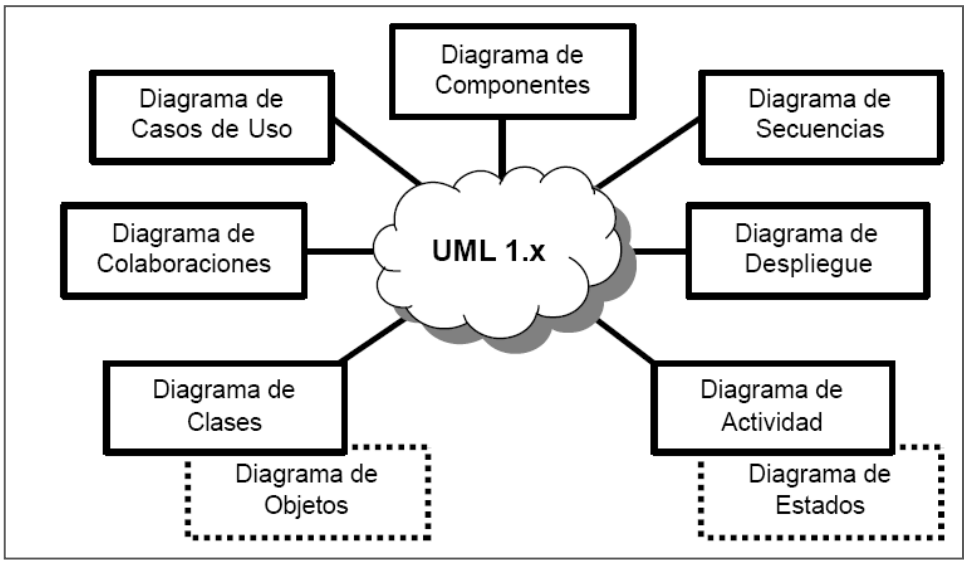


Figura 1. Diagramas de UML 1.x

En OMG UML 2.0 se definen una serie de diagramas adicionales a los establecidos en OMG UML1.x. El conjunto de diagramas se encuentra organizado en torno a dos categorías: diagramas estructurales (representados en gris claro) y diagramas dinámicos o de comportamiento (representados en celeste). Los diferentes diagramas son indicados en la figura 2:

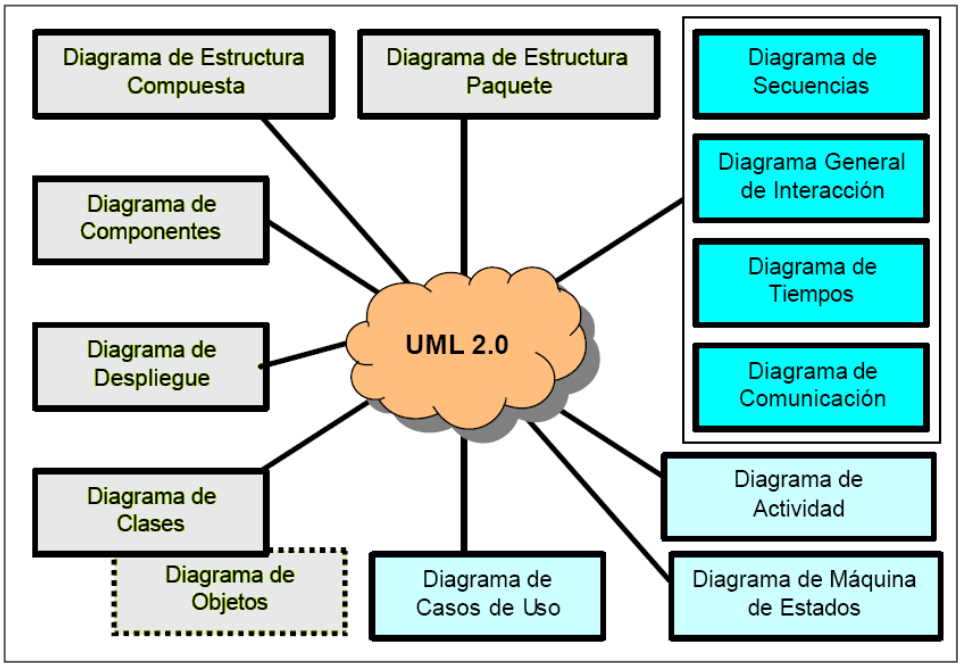


Figura 2. Diagramas de UML 2.0

Nótese al comparar las dos figuras anteriores que el diagrama de colaboración de OMG UML 1.x se ha transformado en el Diagrama de Comunicación en OMG UML 2.0. Adicionalmente se incorporan los siguientes diagramas:

- **Diagrama de Estructura Compuesta**. Se emplea para visualizar de manera gráfica las partes que definen la estructura interna de un clasificador. Cuando se utiliza en el marco de una clase, este diagrama permite elaborar un diagrama de clases donde se muestran los diferentes atributos (partes) y las clases, a partir de las cuales se definen los atributos, indicando principalmente las asociaciones de agregación o de composición de la clase a la que se le elabora el diagrama.
- **Diagrama General de Interacción**. Se emplea fundamentalmente para representar las interacciones, a través de diagramas o fragmentos de diagramas de secuencias, entre los actores y el sistema como una gran caja negra, y de diagramas de actividades en los que aparecen dichos fragmentos.
- **Diagramas de Tiempos**. Empleados para mostrar las interacciones donde el propósito fundamental consiste en razonar sobre la ocurrencia de eventos en el tiempo que provocan el cambio de estados de un elemento estructural (clase, componente, etc.).
- **Diagrama de Comunicación**. Equivalente al diagrama de colaboración del OMG UML 1.x. Permite especificar interacciones entre objetos que conforman la estructura interna de un clasificador. [4]

1.3 Definición de herramienta CASE.

Los siguientes autores definieron las herramientas CASE como:

Kendall & Kendall:

"Las Herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras". [5]

B.Terry & D.Logee:

"Herramientas individuales para ayudar al desarrollador de software o administrador de *proyecto durante una o más fases del desarrollo de software (o mantenimiento)".* [6]

Carma McClure

"Una combinación de herramientas de software y metodologías de desarrollo". [6]

Dadas las definiciones anteriores se puede concluir que las herramientas CASE son:

Un conjunto de métodos y ayudas que facilitan a los ingenieros de software y analistas, el desarrollo del software de principio a fin o en alguna de sus fases. Las mismas fueron creadas para automatizar estas etapas del producto y facilitar la organización de las tareas a cumplir durante su ciclo de vida. Aceleran el desarrollo de los sistemas y aumentan la calidad del software.

1.4 Evolución de las herramientas CASE.

A inicios de los 80's	Ayuda en la documentación por computadora. Diagramación asistida por computadora. Herramientas de análisis y diseño.
A mediado de los 80's	Diseño automático de análisis y pruebas. Repositorios automáticos de información de sistemas.
A final de los 80's	Generación automática de código desde especificaciones de diseño.
A inicios de los 90's	Metodología Inteligente. Interfaz de Usuario reusable como una metodología de desarrollo.

Tabla 1. Evolución de las herramientas CASE.

1.5 Objetivos de las herramientas CASE.

- Mejorar la productividad en el desarrollo y mantenimiento del software.
- Aumentar la calidad del software.
- Mejorar el tiempo, coste de desarrollo y mantenimiento de los sistemas informáticos.
- Mejorar la planificación de un proyecto.
- Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
- Ayuda a la reutilización del software, portabilidad y estandarización de la documentación.
- Gestión global en todas las fases de desarrollo de software con una misma herramienta.
- Facilitar el uso de las distintas metodologías propias de la ingeniería del software. [5]

1.6 Clasificación de las herramientas CASE.

Las herramientas CASE se pueden clasificar en base a los parámetros siguientes:

- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.

- Su funcionalidad.

La siguiente clasificación es la más habitual basada en las fases del ciclo de desarrollo que cubren:

- CASE de alto nivel: son aquellas herramientas que automatizan o apoyan las fases finales o superiores del ciclo de vida del desarrollo de sistemas como la planificación de sistemas, el análisis de sistemas y el diseño de sistemas.
- CASE de bajo nivel: son aquellas herramientas que automatizan o apoyan las fases finales o inferiores del ciclo de vida como el diseño detallado de sistemas, la implantación de sistemas y el soporte de sistemas.
- CASE cruzado de ciclo de vida: se aplica a aquellas herramientas que apoyan actividades que tienen lugar a lo largo de todo el ciclo de vida, se incluyen actividades como la gestión de proyectos y la estimación. [5]

1.7 Beneficios ofrecidos por la tecnología CASE.

- **Facilidad para la revisión de aplicaciones**:

  Las herramientas CASE proporcionan un beneficio sustancial para las organizaciones al facilitar la revisión de las aplicaciones. Contar con un depósito central agiliza el proceso de revisión ya que éste proporciona bases para las definiciones y estándares de los datos.

- **Soporte para el desarrollo de prototipos de sistemas**:

  El desarrollo de prototipos de aplicaciones toma varias formas. En ocasiones se desarrollan diseños para pantallas y reportes con la finalidad de mostrar la organización y composición de los datos, encabezados y mensajes. El objetivo principal es mostrar al usuario desde los momentos iniciales del diseño el aspecto que tendrá la aplicación una vez desarrollada. Ello facilitará la aplicación de los cambios que se consideren necesarios, todavía en la fase de diseño.

- **Generación de código**:

  Algunas herramientas CASE tienen la capacidad de producir el código fuente. La ventaja más visible de esta característica es la disminución del tiempo necesario para preparar un programa. Sin embargo, la generación del código también asegura una estructura estándar y consistente para el programa (lo que tiene gran influencia en el mantenimiento) y disminuye la ocurrencia de varios tipos de errores, mejorando de esta manera la calidad.

- **Mejora en la habilidad para satisfacer los requerimientos del usuario**:

  Satisfacer los requerimientos del usuario es una de las principales características en el desarrollo del producto, esto guarda relación con el éxito del sistema. Las herramientas CASE disminuyen el tiempo de desarrollo, una característica que es importante para los usuarios.

Las descripciones gráficas y los diagramas, así como los prototipos de reportes y la composición de las pantallas, contribuyen a un intercambio de ideas más efectivo entre el usuario y el especialista.

- **Soporte interactivo para el proceso de desarrollo**:
  Las herramientas CASE soportan pasos interactivos al eliminar el tedio manual de dibujar diagramas, elaborar catálogos y clasificar. Como resultado de esto, se anticipa que los analistas repasarán y revisarán los detalles del sistema con mayor frecuencia y en forma más consistente.
- **Generación de documentación técnica**:
  Las herramientas CASE producen la elaboración automática de los documentos de los proyectos a medida que éstos se van desarrollando. Es primordial contar con la documentación de todo el proceso de desarrollo, cuando no se conocían estas herramientas quedaban todos los documentos regados y a veces no se llegaban ni a completar, con la facilidad que nos brinda la herramienta de elaborar los documentos, se gana en tiempo y organización. [7]

1.8 Buenas prácticas de las herramientas CASE.

Control de Versiones. La herramienta debe reconocer las versiones de códigos que se ejecutan en los clientes y servidores, y asegurarse que sean consistentes. También debe ser capaz de controlar un gran número de tipos de objetos incluyendo texto, gráficos, mapas de bits, documentos complejos y objetos únicos, tales como definiciones de pantallas y de informes, archivos de objetos y datos de prueba y resultados. Debe mantener versiones de objetos con niveles arbitrarios de granularidad; por ejemplo, una única definición de datos o una agrupación de módulos.

Trabajar con una variedad de administradores de recursos. La herramienta debe adaptarse ella misma a los administradores de recursos que existen en varios servidores de la red; su interacción con los administradores de recursos debería ser negociable a tiempo de ejecución.

Soporte multiusuario. La herramienta debe permitir que varios diseñadores trabajen en una aplicación simultáneamente. Debe gestionarse los accesos concurrentes a la base de datos por diferentes usuarios, mediante el arbitrio y bloqueos de accesos a nivel de archivo o de registro.

Seguridad. Debe mantener contraseñas y permisos de acceso en distintos niveles para cada usuario. También debe facilitar la realización automática de copias de seguridad y recuperaciones de las mismas, así como el almacenamiento de grupos de información determinados, por ejemplo, por proyecto o aplicaciones.

Desarrollo en equipo, repositorio de librerías compartidas. La herramienta debe permitir que

grupos de programadores trabajen en un proyecto común; debe proporcionar además un mecanismo para compartir las librerías entre distintos realizadores y múltiples herramientas. Gestiona y controla el acceso multiusuario a los datos y bloquea los objetos para evitar que se pierdan modificaciones inadvertidamente cuando se realizan simultáneamente. [6]

1.9 Tecnologías y tendencias actuales.

La Ingeniería de Software recomienda el uso de herramientas CASE para llevar a cabo el Análisis y Diseño de cualquier Sistema de Información, en este caso las de soporte a UML. Se analizaron una serie de tecnologías que actualmente son empleadas por grandes empresas productoras de software en el mundo, con el objetivo de seleccionar una que permita suplir las necesidades existentes en CEDIN. A este grupo de herramientas se le realizó un análisis, destacando sus principales características, teniendo en cuenta los siguientes aspectos: plataformas que soporta, licencia, diagramas que se pueden realizar, generación de código, generación de documentación, ingeniería inversa, entre otros.

1.9.1 ArgoUML.

- **Plataforma:** Multiplataforma**.**
- **Licencia:** libre BSD (Berkeley Software Distribution).
- **Extensible:** sí
- **Código Abierto:** sí
- **Diagramas que se pueden realizar:** de clases, de estado, de actividad, de casos de uso, de colaboración, despliegue, implementación y de secuencia.
- **Generación de código:** Java, C++, C#, PHP4, PHP5 y SQL.
- **Generación de documentación:** no.
- **Ingeniería inversa:** no.
- **Exportación de diagramas:** GIF, PNG, PostScript, PGML y SVG.
- **XMI:** sí.
- **Ventajas:** Genera código automáticamente. Propone soluciones a algunos errores y contiene un panel de propiedades y de tareas pendientes bastante útil. Es una herramienta fácil de usar para el modelado de sistemas.
- **Desventajas:** Sin embargo, ArgoUML está incompleto. No es conforme completamente a los estándares UML y carece de soporte completo para algunos tipos de diagramas incluyendo los diagramas de secuencia y los de colaboración. Otro de sus inconvenientes es que consume muchos recursos**.** [8]. (Véase Anexo1).

1.9.2 BOUML.

- **Plataforma:** Multiplataforma.
- **Licencia:** GPL
- **Extensible:** sí.
- **Código Abierto:** sí.
- **Diagramas que se pueden realizar:** todos los diagramas UML estándares.
- **Generación de código:** no.
- **Generación de documentación:** sí, en varios formatos (HTML, XMI).
- **Ingeniería inversa:** sí.
- **Exportación de diagramas:** HTML
- **XMI:** sí.
- **Ventajas:** Es una herramienta sencilla de utilizar, rápida y que apenas consume memoria.
- **Desventajas:** No es muy intuitiva ni usable. [9]. (Véase Anexo2).

1.9.3 Visual Paradigm For UML.

- **Plataforma:** Multiplataforma.
- **Licencia:** Comercial.
- **Extensible:** sí.
- **Código Abierto:** sí.
- **Diagramas que se pueden realizar:** diagrama de paquetes, de clase, de objetos**,** de estructura compuesta, de componentes, de despliegue, de casos de uso, diagrama de actividades, de máquina de estados, de comunicación**,** de serie o secuencia**,** diagrama de tiempo y de interacción.
- **Generación de código:** Java**,** C++, CORBA IDL, PHP, XML Schema, Ada, Python. Además, apoya la generación del código C#, VB.NET, Object Definition Language (ODL), Flash ActionScript, Delphi, Perl y Ruby.
- **Generación de documentación:** sí, en varios formatos (PDF, HTML o MS Word).
- **Ingeniería inversa:** sí.
- **Exportación de diagramas:** JPG, PNG, SVG y EMF.
- **XMI:** sí.
- **Ventajas:** Visual Paradigm UML emplea una rápida respuesta con poca memoria utilizando moderadamente los tiempos del procesador, lo que le permite manejar grandes y complicadas estructuras de un proyecto en una forma muy eficiente y, que solo requiera de una configuración de escritorio.

- **Desventajas:** La versión llamada Community Edition no permite su uso en proyectos comerciales e incluye marca de agua recordando este hecho. [10]. (Véase Anexo 3).

1.9.4 Eclipse.

- **Plataforma:** Multiplataforma.
- **Licencia:** Pública
- **Extensible:** sí.
- **Código Abierto:** sí.
- **Diagramas que se pueden realizar:** Es capaz de trabajar con todos los tipos de diagramas UML: clases, componentes, colaboración, actividad, secuencia, estados, casos de uso.
- **Generación de código:** Java a partir de los diagramas UML.
- **Generación de documentación**: sí.
- **Ingeniería inversa:** sí.
- **Exportación de diagramas:** SVG.
- **XMI:** sí.
- **Ventajas:** Basa su funcionamiento en plugins, con lo que es ampliable para que haga prácticamente cualquier cosa.
- **Desventajas:** Exporta diagramas sólo como SVG. [11]. (Véase Anexo 4).

1.9.5 Umbrello UML Modeller

- **Plataforma:** Linux.
- **Licencia:** GPL (libre).
- **Extensible:** sí.
- **Código Abierto:** sí.
- **Diagramas que se pueden realizar:** de clases, de secuencia, de colaboración, de casos de uso, de estado, de actividad, de despliegue y de componentes.
- **Generación de código:** Java (advanced), PHP, JavaScript, ActionScript, SQL, Python, Ada, IDL, XML Schema, Perl, C++ (advanced).
- **Generación de documentación:** sí.
- **Ingeniería inversa:** sí, sólo desde código C++.
- **Exportación de diagramas:** SVG.
- **XMI:** sí.
- **Ventajas:** Es software libre.
- **Desventajas:** No realiza el diagrama de actividades completamente. [12]. (Véase Anexo 5).

1.9.6 Rational Software Architect

- **Plataforma:** Multiplataforma.
- **Licencia:** NodeLocked.
- **Extensible:** sí.
- **Código Abierto:** no.
- **Diagramas que se pueden realizar:** diagramas de casos de uso, clase, secuencia, actividad, estado, componentes y despliegue.
- **Generación de código:** sí, Java™/J2EE™, Servicios Web, SOA y C/C++.
- **Generación de documentación:** sí.
- **Ingeniería inversa:** sí.
- **Exportación de diagramas:** no.
- **XMI:** sí.
- **Ventajas**: Se puede usar fácilmente. Aprovecha al máximo la plataforma abierta Eclipse 3.2 extensible. Está diseñado para integrarse con el ciclo de vida de su software y con los procesos del equipo de desarrollo.
- **Desventajas**: Comercial. [13]. (Véase Anexo 6).

1.9.7 NetBeans

- **Plataforma: Multiplataforma.**
- **Licencia:** CDDL + GPLv2.
- **Extensible:** sí.
- **Código Abierto:** sí.
- **Diagramas que se pueden realizar:** Es capaz de trabajar con todos los tipos de diagramas UML: clases, componentes, colaboración, actividad, secuencia, estados, casos de uso, despliegue.
- **Generación de código:** sí.
- **Generación de documentación:** sí.
- **Ingeniería inversa:** sí.
- **Exportación de diagramas:** sí.
- **XMI:** sí.
- **Ventajas:** Creado por Sun Microsystems, de código abierto y cuenta con un ambiente de desarrollo empresarial con soporte total.
- **Desventajas:** Consume bastantes recursos. [14]. (Véase Anexo 7).

1.9.8 StarUML

- **Plataforma**: Win32.
- **Licencia:** GPL.
- **Extensible:** sí.
- **Código Abierto:** sí.
- **Diagramas que se pueden realizar:** de casos de uso, de clases, de secuencia, de colaboración, de estado, de actividades, de componentes, de despliegue.
- **Generación de código:** Java, C# y C++.
- **Generación de documentación:** sí, con Microsoft Office.
- **Ingeniería inversa:** sí.
- **Exportación de diagramas: -**
- **XMI:** sí.
- **Ventajas:** Puede generar documentación en formatos Word, Excel y PowerPoint sobre los diagramas. Usa patrones GoF (Gang of Four), EJB (Enterprise JavaBeans) y personalizados. Es un software libre.
- **Desventajas:** No es un software muy usado. No es posible crear diagramas de objetos ni de casos de uso del negocio. [15]. (Véase Anexo 8).

1.9.9 DIA

- **Plataforma:** Multiplataforma.
- **Licencia:** Libre.
- **Extensible:** sí.
- **Código Abierto:** sí.
- **Diagramas que se pueden realizar:** Realiza todos los diagramas UML menos los diagramas de actividades, de despliegue y de objetos.
- **Generación de código:** no.
- **Generación de documentación:** no.
- **Ingeniería inversa:** no.
- **Exportación de diagramas:** SVG, EPS, PostScript, JPEG, VDX.
- **XMI:** sí.
- **Ventajas:** Permite incorporar nuevas formas en formato vectorial SVG (que se pueden crear con casi cualquier programa vectorial de software libre).
- **Desventajas:** No realiza ingeniería inversa. Difícil de usar. [16]. (Véase Anexo 9).

1.9.10 Rational Rose

- **Plataforma:** Windows.
- **Licencia:** Comercial.
- **Extensible:** sí.
- **Código Abierto:** no.
- **Diagramas que se pueden realizar:** Diagrama de casos de uso, de interacción, de actividad, de clases, de estado, de interacción, de componentes y de despliegue.
- **Generación de código:** sí, Java, C++, Ada, Visual Basic, Corba, Oracle.
- **Generación de documentación:** sí.
- **Ingeniería inversa:** sí.
- **Exportación de diagramas:** sí.
- **XMI:** sí.
- **Ventajas:** Es una herramienta muy completa y estable como muy pocas herramientas que se han creado.
- **Desventajas:** Entorno gráfico no muy amigable para el usuario. No es un software libre. [17] [18]. (Véase Anexo 10).

1.9.11 Poseidon for UML

- **Plataforma:** Multiplataforma.
- **Licencia:** Comercial.
- **Extensible:** sí.
- **Código Abierto:** no.
- **Diagramas que se pueden realizar:** de clases, de estado, de actividad, de casos de uso, de colaboración, despliegue (combinación de diagramas de objetos/componentes/despliegue).
- **Generación de código:** JAVA, HTML.
- **Generación de documentación:** sí.
- **Ingeniería inversa:** sí.
- **Exportación de diagramas:** GIF, PostScript, EPS, PGML, SVG.
- **XMI:** sí.
- **Ventajas:** Poseidon permite generar etiquetas Java en el diseño de clases, que aparecerán en el código generado. Además, versiones de Poseidon más avanzadas permiten generar documentos HTML con la documentación y esquemas oportunos.

- **Desventajas:** Instalación costosa, poco amigable, difícil de empezar. [19] [20]. (Véase Anexo 11).

1.10 Conclusiones del capítulo.

En el presente capítulo se abordaron los conceptos asociados a las herramientas CASE, haciendo hincapié en su importancia para el desarrollo de software, también se analizaron algunas de las tendencias actuales, con el objetivo de identificar las herramientas que satisfacen las necesidades del centro para su posterior análisis. No se puede descartar, que alguna herramienta no evaluada en esta investigación pueda ser una buena herramienta CASE para el modelado UML.

CAPÍTULO II. ANÁLISIS Y SELECCIÓN DE LA HERRAMIENTA CASE

2.1 Introducción.

El proceso de selección de una herramienta CASE en una organización es un proceso muy complejo, en el presente capítulo se muestra el resultado de la aplicación de un modelo de decisión realizado a las herramientas que se describen en el capítulo anterior, con el objetivo de seleccionar la más adecuada y proponerla al centro para que solucione sus necesidades. Luego mediante un caso de estudio relacionado con las necesidades del centro, se demuestran las facilidades de utilización e integración de la herramienta propuesta.

2.2 Estudio Comparativo Inicial.

Para la selección de la herramienta es necesario realizar un estudio y determinar los problemas reales por los que está atravesando el centro en el uso de las herramientas CASE, además de las características particulares de la organización, buscando un conjunto de soluciones que satisfagan aunque solo sea en parte, sus problemas, necesidades y características, para luego establecer y aplicar un plan de implantación que gestione y asegure la correcta adopción de la herramienta.
Las siguientes características son indispensables a la hora de seleccionar la herramienta que logre solventar las necesidades del centro.

- Libre.
- Extensible.
- Código abierto.
- Multiplataforma o para GNU Linux.

Para elegir la herramienta se establecen una serie de comparaciones que ayudarán a su elección final, primeramente se realiza una comparación en cuanto a los parámetros mencionados anteriormente, donde la herramienta que no cumpla al menos uno de ellos se descarta de esta investigación, descartando también, aquellas que ni siquiera pretenden cubrir todos los tipos de diagramas básicos de UML. Estos parámetros son explicados seguidamente por su importancia en el proceso de selección.

Extensible: Permite ampliar las funcionalidades del software utilizando un mecanismo de extensión.
Libre: Se refiere a que una vez obtenido por el usuario, el mismo goza de libertad para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.
Código Abierto: Esta característica permite obtener el código fuente del software, para ser analizado o modificado.

Plataforma: Se refiere a los Sistemas Operativos en que puede ser ejecutado el software.

En la siguiente tabla se muestran las once herramientas candidatas, clasificadas en cuanto a los parámetros explicados previamente.

Herramientas	Extensible	Plataforma	Libre	Código Abierto
ArgoUML	sí	Multiplataforma	sí	sí
BOUML	sí	Multiplataforma	sí	sí
Visual Paradigm	sí	Multiplataforma	no	sí
Eclipse	sí	Multiplataforma	sí	sí
Umbrello	sí	Linux	sí	sí
Rational Software Architect	sí	Multiplataforma	no	no
NetBeans	sí	Multiplataforma	sí	sí
StarUML	sí	Win32	sí	sí
DIA	sí	Multiplataforma	sí	sí
Rational Rose	sí	Windows	no	no
Poseidon	sí	Multiplataforma	no	no

Tabla 2.Características básicas.

De la comparación anterior solo seis herramientas cumplen con las cuatro características excluyentes, éstas son: ArgoUml, BOUML, Eclipse, NetBeans, DIA y Umbrello. Las mismas fueron analizadas en la práctica y evaluadas en cuanto a la cantidad de diagramas que realizan, siendo descartada la herramienta DIA ya que no permite elaborar todos los tipos de diagramas básicos de UML.

La siguiente fase consiste en el desarrollo y aplicación de un modelo de decisión, que permita seleccionar la herramienta. Para la selección de dicho modelo se tuvieron en cuenta varias propuestas, entre ellas la del Departamento de Procesos y Sistemas, LISI, Universidad Simón Bolívar [21], también una evaluación comparativa de herramientas CASE para UML desarrollado por el Dpto. de Informática Universidad Carlos III de Madrid [22], y por último un modelo propuesto por un grupo de estudiantes de la Universidad Simón Bolívar [23].

En el primero se evalúan las herramientas desde el punto de vista organizacional, es decir, se comparan de acuerdo a las características propias de la organización que la adopta, no de las propiedades que ella posee. El segundo método se centra en evaluar las herramientas solo desde el punto de vista notacional, la notación proporciona un conjunto completo de elementos gráficos para modelar sistemas orientados a objetos, y establece las reglas de como estos elementos deben ser conectados y usados, esto es muy importante, pero atendiendo a las necesidades del centro, se decidió aplicar el último, porque además de seleccionar la herramienta por el diseño que propone, se apoya también en el alcance y solidez de la herramienta. También se decide aplicar por la cantidad de indicadores que permite analizar, por la serie de pasos por los que debe pasar la herramienta para ser seleccionada, es decir, es más detallado el proceso, y por la facilidad de aplicación del modelo.

2.3 Fase de Aplicación del Modelo de Decisión.

Luego de realizar una entrevista (ver Anexo 12) a los analistas y desarrolladores del centro, fueron identificados una serie de indicadores con sus respectivas variables de medición, que ayudaron a través del modelo de decisión utilizado, a evaluar analítica y cuantitativamente las herramientas candidatas.

Para el desarrollo del modelo, estos indicadores se agruparon en dos grupos**,** internos y externos. Los internos se refieren a aquellas características que tienen que ver con la estructura interna de la herramienta, éstos a su vez fueron clasificados en cuanto al alcance, ¿qué abarca la herramienta? y diseño, ¿Cómo es la herramienta? Los externos por su parte se refieren a los factores complementarios del entorno de la herramienta y su criterio es la solidez de la empresa desarrolladora. Cada uno de estos criterios, fueron tipificados en subcriterios y cada subcriterio tiene asociado un conjunto de indicadores.

En la siguiente tabla se muestran cada uno de los factores que fueron explicados anteriormente, los indicadores se encuentran acompañados por una abreviatura o código que les identifica, así como por su tipo (entre paréntesis).

Categoría	Criterio	Sub-Criterio	Indicadores
Internos	**Alcance**	**Componentes**	ALC1 (2)-Se analiza el total de diagramas que realiza la herramienta.
			ALC2 (3)-Se analiza si apoya el repositorio común.
			ALC3 (3)- Se analiza la facilidad de generación de gráficos.
			ALC4 (3)- Se analiza la generación de códigos de la herramienta.
			ALC5 (3)- Se analiza el proceso de ingeniería inversa de la herramienta.

			ALC6 (3)- Se analiza si la herramienta exporta en formato XMI (XML Metadata Interchange).
		Plataformas que soporta la herramienta.	ALC7 (2)- Se analiza la cantidad de plataformas que soporta la herramienta.
	Diseño	**Funcionalidad**	DIS1 (1)- Se analiza el completamiento del diagrama de casos de uso.
			DIS2 (1)- Se analiza el completamiento del diagrama de clases.
			DIS3 (1)- Se analiza el completamiento del diagrama de actividades.
			DIS4 (1)- Se analiza el completamiento del diagrama de clases del análisis.
			DIS 5 (1)- Se analiza el completamiento del diagrama de secuencia.
			DIS 6 (1)- Se analiza el completamiento del diagrama de colaboración.
			DIS 7 (1)- Se analiza el completamiento del diagrama de despliegue.
			DIS 8 (1)- Se analiza el completamiento del diagrama de componentes.
			DIS 9 (1)- Se analiza el completamiento del diagrama de objeto.
			DIS10 (2)- Se analizan la cantidad de lenguajes en que la herramienta genera códigos.
			DIS11 (1)-Se analiza el grado de completitud del código generado por la herramienta.
			DIS12 (2)-Se analizan la cantidad de lenguajes soportados por la ingeniería inversa.
			DIS13 (1)-Se mide la calidad del código generado por la herramienta.
			DIS14 (1)- Se mide la calidad del producto de la ingeniería inversa.
		Facilidad de Uso.	DIS15 (1)-Se analiza si la interfaz gráfica de la herramienta es amigable.
		Curva de aprendizaje.	DIS16 (3)-Se analiza si la herramienta es intuitiva.
			DIS17 (3)-Se analiza si la herramienta tiene tutorial.
			DIS18 (1)- Se mide el grado de satisfacción del servicio del tutorial.
		Ayuda en Línea	DIS19 (1)- Se analiza la facilidad del sistema de ayuda.
			DIS20 (1)-Se analiza la facilidad de uso de la herramienta.
Externos	**Solidez**	**Prestigio del desarrollador**	SOL1 (3)- Se analizan si existen o no publicaciones expertas positivas.
			SOL2 (3)- Se analiza si la herramienta posee certificaciones.

Tabla 3. Grafo Semántico de Indicadores.

2.3.1 Modelo de Decisión.

Las cinco herramientas a evaluar fueron sometidas a un grupo de pasos, que permitieron la selección

de la herramienta más apropiada para suplir las necesidades del Centro de Desarrollo de Informática Industrial. Seguidamente son detallados los procedimientos que conforman el modelo de decisión.

Paso N° 1. Cálculo de los valores de los indicadores por herramienta CASE.

Mediante la evaluación de las herramientas se obtuvieron los valores de los indicadores tecnológicos. En la mayoría de los casos el valor estaba sujeto a la valoración de la herramienta en la práctica, y en otros a la información que existía en su página oficial. En este caso como el modelo de decisión tiene en cuenta solo resultados cuantitativos para comparar las herramientas, no se consideró necesario documentar los resultados de este paso.

Paso N° 2.Clasificación de los resultados por tipo de indicador.

Obtenidos todos los valores de los indicadores por herramienta, se procede a clasificarlos según el tipo de dominio que posee cada uno, tal como lo muestra la Tabla No 4.

Tipo de Indicador	Dominio que posee
(1) Grado	Un valor continuo entre 0 y 1, donde 0 significa ninguna satisfacción y 1 alta satisfacción.
(2) (N)	Un valor N positivo cualquiera menor que 30.
(3) Flag	Un valor que representa una bandera (0 = no, 1 = sí).

Tabla 4. Correspondencia Tipo de Indicador y Dominio.

En la siguiente tabla se muestran los indicadores de cada herramienta clasificados según el tipo de domino.

Indicador	Tipo de Indicador	ArgoUML	BOUML	Eclipse	NetBeans	Umbrello
ALC 1	(N)	7	7	11	8	9
ALC 2	Flag	1	1	1	1	1
ALC 3	Flag	1	1	1	1	1
ALC 4	Flag	1	1	1	1	1
ALC 5	Flag	0	1	1	1	1
ALC 6	Flag	1	1	0	1	1
ALC 7	(N)	2	3	2	2	1
DIS 1	Grado	0.5	0.5	0.5	0.5	0.5
DIS 2	Grado	0.9	0.9	0.9	0.9	0.9
DIS 3	Grado	0.2	0.8	0.9	0.9	0.6
DIS 4	Grado	0.3	0.3	0.3	0.3	0.3
DIS 5	Grado	0.2	0.5	0.9	0.9	0.7
DIS 6	Grado	0.3	0.5	0.7	0.7	0.6
DIS 7	Grado	0.7	0.7	0.7	0.7	0.7
DIS 8	Grado	0	0.6	0.8	0.8	0.4
DIS 9	Grado	0	0.5	0.5	0	0
DIS 10	(N)	5	0	1	1	18
DIS 11	Grado	0.8	0	0.9	0.9	0.7
DIS 12	(N)	0	3	1	1	6
DIS 13	Grado	0	0	0.8	0.8	0.8
DIS 14	Grado	0.9	0.1	0.9	0.9	0.9
DIS 15	Flag	1	0	1	1	1
DIS 16	Flag	1	0	1	1	1

DIS 17	Grado	0.1	0	0.7	0.9	0.8
DIS 18	Grado	0	0	0.7	0.9	0.8
DIS 19	Grado	0.8	0.1	0.7	0.9	0.8
DIS 20	Grado	0.5	0.4	0.7	0.9	0.8
Sol 1	Flag	1	1	1	1	1
Sol 2	Flag	1	1	1	1	1

Tabla 5. Resultados por tipo de indicador.

Paso N° 3. Transformación de cada tipo.

Luego de ser clasificados todos los indicadores por tipo, se transformó el valor del indicador, es decir, estos fueron llevados a una escala del 1 al 5, de acuerdo con su correspondiente tipo, con el fin de homogenizar todos los valores.

Las fórmulas utilizadas para realizar las equivalencias fueron las siguientes:

Indicadores de Grado: Esta fórmula se aplica a todos los indicadores de grado.

Si (V <= 0.2) $\Rightarrow$ Transformación = 1

Si (V <= 0.4) $\Rightarrow$ Transformación = 2

Si (V <= 0.6) $\Rightarrow$ Transformación = 3

Si (V <= 0.8) $\Rightarrow$ Transformación = 4

Si (V <= 1) $\Rightarrow$ Transformación = 5

Donde:

V $\in$ {[0,1]}

Transformación $\in$ {1, 2, 3, 4,5}

En la siguiente tabla se muestran los valores de los indicadores de tipo grado con las transformaciones que se detallan en la fórmula mostrada anteriormente.

Indicador Grado	ArgoUML	BOUML	Eclipse	NetBeans	Umbrello
DIS 1	3	3	3	3	3
DIS 2	5	5	5	5	5
DIS 3	1	4	5	5	3
DIS 4	2	2	2	2	2
DIS 5	1	3	5	5	4
DIS 6	2	3	4	4	3
DIS 7	4	4	4	4	4
DIS 8	1	3	4	4	2
DIS 9	1	3	3	1	1
DIS 11	4	1	5	5	4
DIS 13	1	1	4	4	4
DIS 14	5	1	5	5	5
DIS 17	1	1	4	5	4
DIS 18	1	1	4	5	4
DIS 19	4	1	4	5	4
DIS 20	3	2	4	5	4

Tabla 6. Transformación de los indicadores de Tipo 1.

Indicadores (N): Esta fórmula se aplica para todos los indicadores tipo N.

Si (V = 0) $\Rightarrow$ Transformación = 0

Si (V <= 5) $\Rightarrow$ Transformación = 3

Si (V <= 15) $\Rightarrow$ Transformación = 4

Si (V <= 20) $\Rightarrow$ Transformación = 5

Donde:

V $\in$ { [0, 20]}

Transformación $\in$ {0, 3, 4,5}

En la siguiente tabla se muestran los valores de los indicadores de tipo (N) con las transformaciones que se detallan en la fórmula mostrada anteriormente.

Indicador (N)	ArgoUML	BOUML	Eclipse	NetBeans	Umbrello
ALC 1	4	4	4	4	4
ALC 7	3	3	3	3	3
DIS 10	3	0	3	3	5
DIS 12	0	3	3	3	4

Tabla 7. Transformación de los indicadores de Tipo 2.

Indicadores Flag: Esta fórmula se aplica para todos los indicadores tipo Flag (0,1).

Si (V = 0) $\Rightarrow$ Transformación = 1

Si (V = 1) $\Rightarrow$ Transformación = 5

Donde:

V $\in$ {0,1}

Transformación $\in$ {1,5}

En la siguiente tabla se muestran los valores de los indicadores de tipo flag con las transformaciones que se detallan en la fórmula mostrada anteriormente.

Indicador Flag	ArgoUML	BOUML	Eclipse	NetBeans	Umbrello
ALC 2	5	5	5	5	5
ALC 3	5	5	5	5	5
ALC 4	5	5	5	5	5
ALC 5	1	5	5	5	5
ALC 6	5	5	1	5	5
DIS 15	5	1	5	5	5
DIS 16	5	1	5	5	5
Sol 1	5	5	5	5	5
Sol 2	5	5	5	5	5

Tabla 8. Transformación de los indicadores de Tipo 3.

Realizadas las clasificaciones, se muestran en la siguiente gráfica las herramientas con sus respectivos valores por indicador.

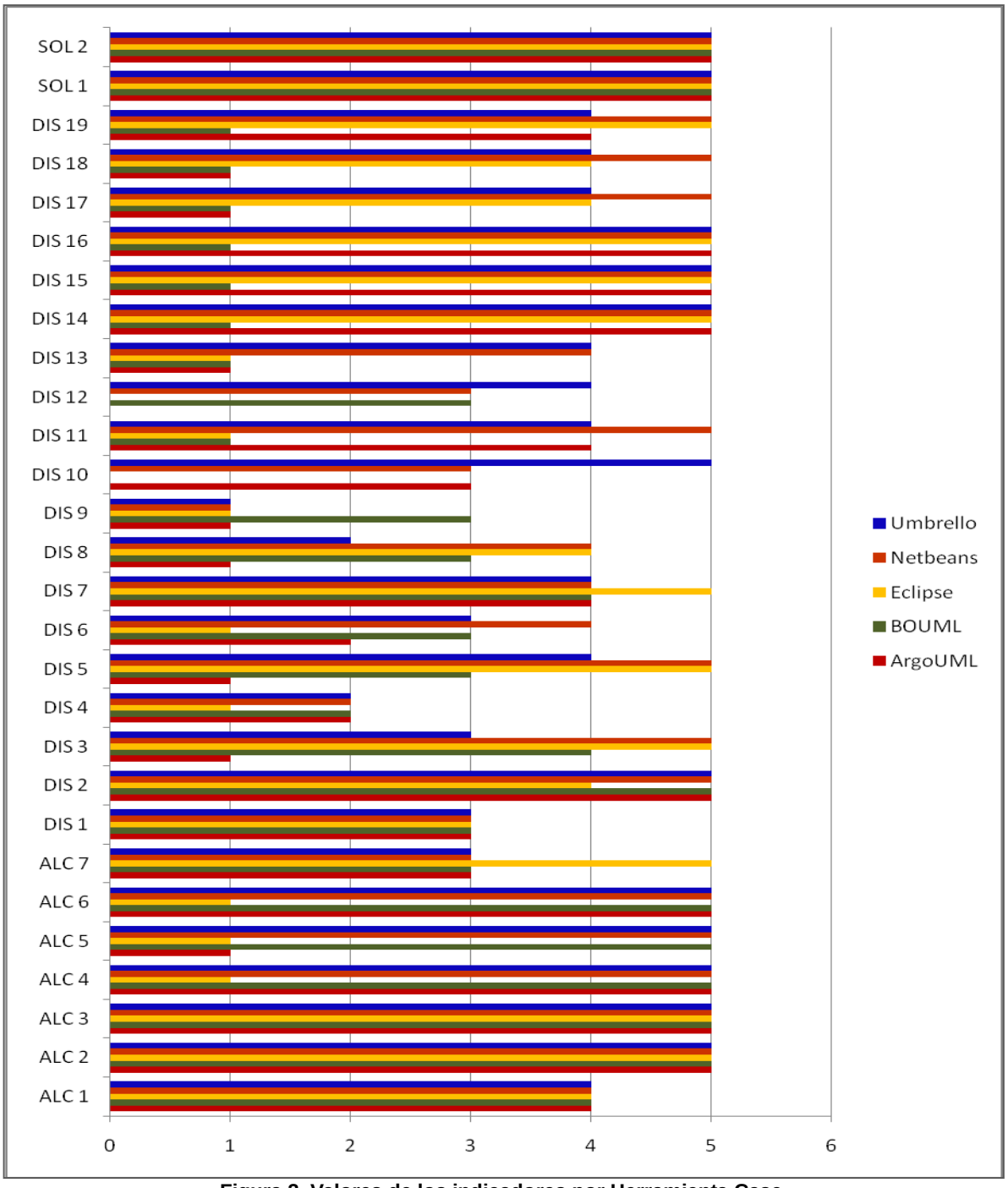


Figura 3. Valores de los indicadores por Herramienta Case.

Al analizar la figura anterior se puede observar que el NetBeans, el Eclipse y el Umbrello son las herramientas que mayor cantidad de indicadores presentan alcanzando la puntuación más satisfactoria, lo que puede beneficiar su selección en los próximos pasos.

Paso N° 4. Totalización de los resultados por herramienta CASE.

Luego de tener todas las fórmulas aplicadas y las transformaciones realizadas, se totalizaron los resultados parciales por herramienta CASE, sumando el contenido de las columnas correspondientes a las transformaciones realizadas a cada indicador.

$$\text{Totalización} = \sum_{i=0}^{29} \text{Transformación};$$

Donde: Totalización ∈ [0, Suma].

En la siguiente figura se muestran los valores totales de las herramientas luego de aplicarle la fórmula anterior.

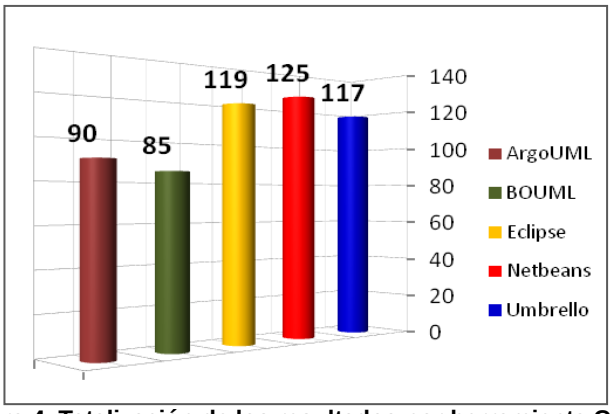


Figura 4. Totalización de los resultados por herramienta CASE

2.3.2 Fase de resultados.

Finalmente, luego de haber aplicado los pasos descritos anteriormente, pertenecientes al modelo de decisión, se procede a asignarles una clasificación final a las herramientas de acuerdo con los totales obtenidos mostrados en la gráfica anterior. Esta clasificación se detalla en la siguiente tabla.

Clasificación	Puntaje
Óptima	>= 120
Sub-Óptima	>= 100 < 120
Analizable	>= 85 < 100
Poco Recomendable	< 85

Tabla 9. Clasificación de las herramientas.

En la siguiente figura se muestran las herramientas con las equivalencias realizadas y los cálculos descritos anteriormente para las cinco herramientas analizadas.

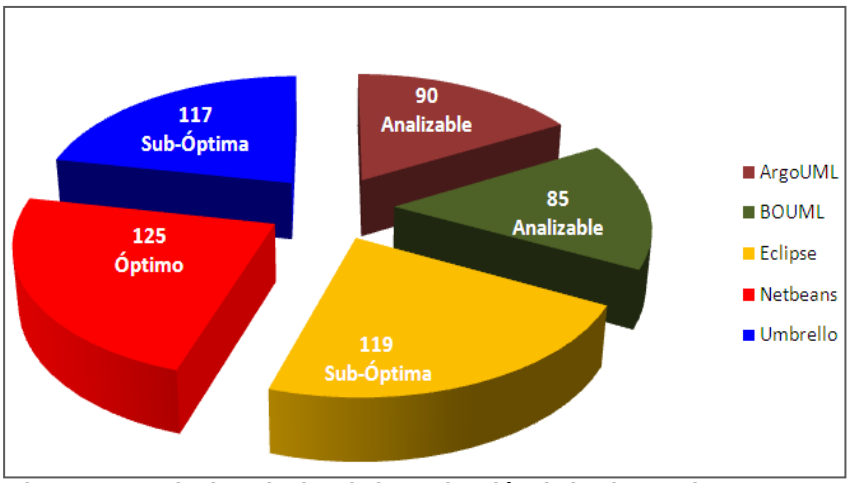


Figura 5. Resultados Finales de la Evaluación de las herramientas CASE.

2.3.3 Análisis de los resultados.

Teniendo en cuenta los resultados obtenidos anteriormente, se propuso la utilización del **NetBeans,** en particular la **versión 6.1**, la cual a diferencia de las demás versiones incluye el módulo UML, esto da a entender la estabilidad del UML dentro del IDE. En otras versiones más avanzadas como la **6.7.1** al añadirle el plugin de UML se comprobó que la cantidad de diagramas que realiza no es la suficiente como para satisfacer las necesidades del centro.

Figura 6. Pantalla de Bienvenida de NetBeans 6.1.

Otras características que condujeron a su selección fueron las siguientes:

Extensibilidad:

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos, estas aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. [24]

Código Abierto, patrocinado por SUN:

NetBeans viene con las ventajas de haber sido creado, respaldado y convertido en código abierto por Sun Microsystems. La compañía ha seguido aportando un patrocinio que asegura que NetBeans sea un entorno de desarrollo de clase empresarial con un soporte completo. [24]

Facilidad de uso:

Es una herramienta muy intuitiva lo que permite aún más su facilidad de uso, además posee un sistema de ayuda detallada y completa que posibilita satisfacer las dudas que se presenten en cualquier situación.

Total de diagramas que realiza:

Realiza un total de ocho diagramas entre ellos se encuentran el de: clases, actividades, secuencia, componentes, despliegue, casos de uso, estado y colaboración, contando con la mayoría de los estereotipos para la realización de los mismos.

Implementa funcionalidades básicas de UML2: principalmente en los diagramas de secuencia, de clases y de despliegue, esto equivale a un mejoramiento en los mismos.

2.4 Introducción al desarrollo de diagramas UML con NetBeans.

NetBeans es un IDE de desarrollo escrito empleando tecnología Java, por lo que se ejecuta en cualquier plataforma. Los beneficios de ser una herramienta libre permiten a los usuarios modificar el IDE si así lo desean, esto ha permitido que sea una de las herramientas más utilizadas por toda la comunidad de desarrolladores.

Es una herramienta que mediante la incorporación del plugin UML posibilita el modelado de software mejorando así la productividad del desarrollador, también proporciona edición en ambos sentidos, permitiendo que el código fuente cambie conjuntamente con los cambios en el modelo y permite a los desarrolladores evitar tener que consultar constantemente los comentarios del código. Permite además la conexión a repositorios lo que posibilita mayor organización y evita que los equipos de desarrollo pierdan tiempo en revisiones agotadoras para actualizar la información.

Las características de NetBeans, como su extensibilidad hacia varios lenguajes de desarrollo y su conformidad con UML, aseguran que se satisfarán los requerimientos específicos del sistema que se está modelando. [24]

En la siguiente figura se muestra como está estructurado el entorno de trabajo de NetBeans.

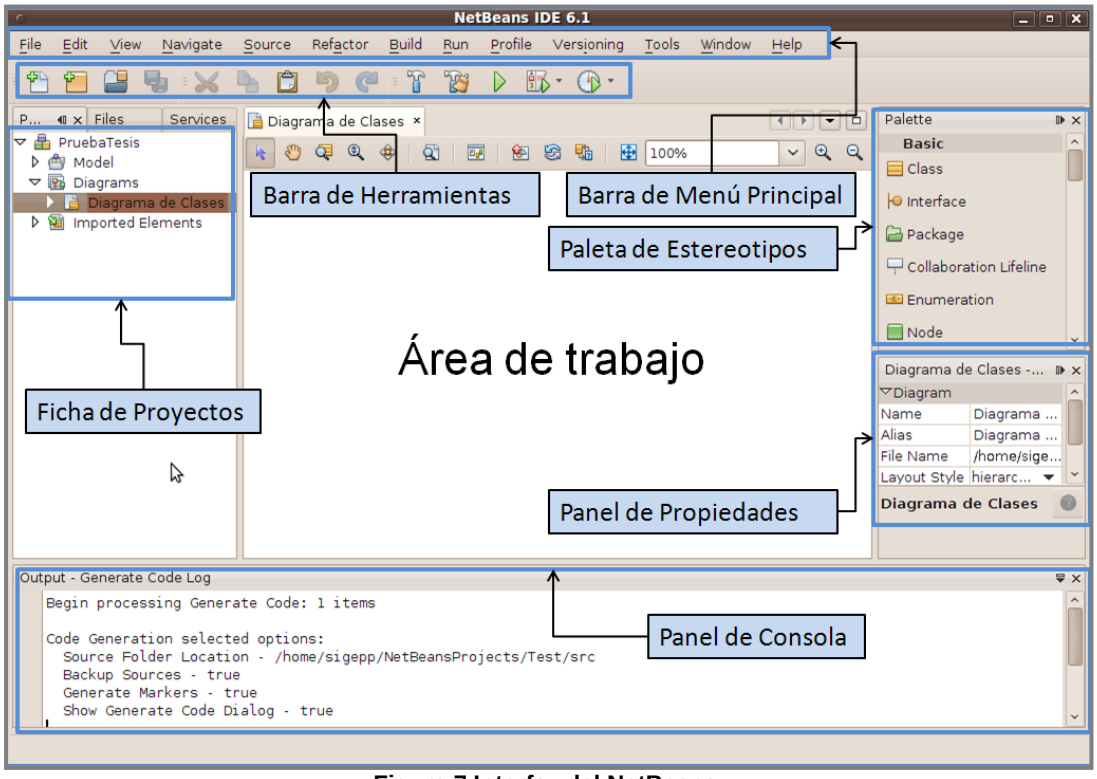


Figura 7.Interfaz del NetBeans.

A continuación se describen las áreas señaladas en la figura 7.

Área de Trabajo: En esta zona de la herramienta se diseñan los diagramas**,** utilizando los componentes visuales de la paleta de estereotipos.

Barra de Menú Principal: Se muestran todas las opciones para gestionar el NetBeans.

Barra de Herramientas: Atajos para las principales funcionalidades que ofrece la herramienta.

Ficha de Proyectos: Muestra todos los proyectos de NetBeans creados utilizando una estructura de árbol.

Panel de propiedades: Muestra las propiedades de los objetos seleccionados.

Panel de Consola: Muestra la salida de los procesos que se realizan.

Paleta de Estereotipos: Ofrece los estereotipos y objetos de diseño para la creación de los diagramas.

2.4.1 Creación de un proyecto de modelado UML.

NetBeans define su propia estructura de directorios para sus proyectos, en el caso de los proyectos

de modelado UML, cuando se crea un nuevo proyecto se crea automáticamente un directorio con el nombre del proyecto creado, el cual contendrá toda la información referente al proyecto además de los archivos de aquellos diagramas y objetos de diseño creados.

Para crear un nuevo proyecto UML, dirigirse al menú **File** y seleccionar **New Proyect**. Automáticamente se muestra un formulario que guiará el proceso de creación del nuevo proyecto.

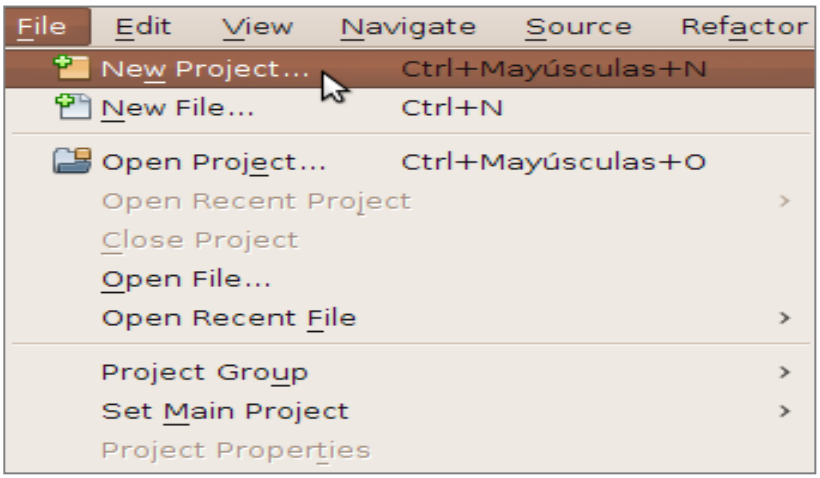


Figura 8.Crear Espacio de Trabajo.

Luego se selecciona dentro de las categorías de proyectos mostradas la opción **UML** y la modalidad del proyecto a crear, en este caso se selecciona **Plataform-Independent Model** y presionando el botón siguiente (**NEXT)** se pasa al siguiente paso del formulario.

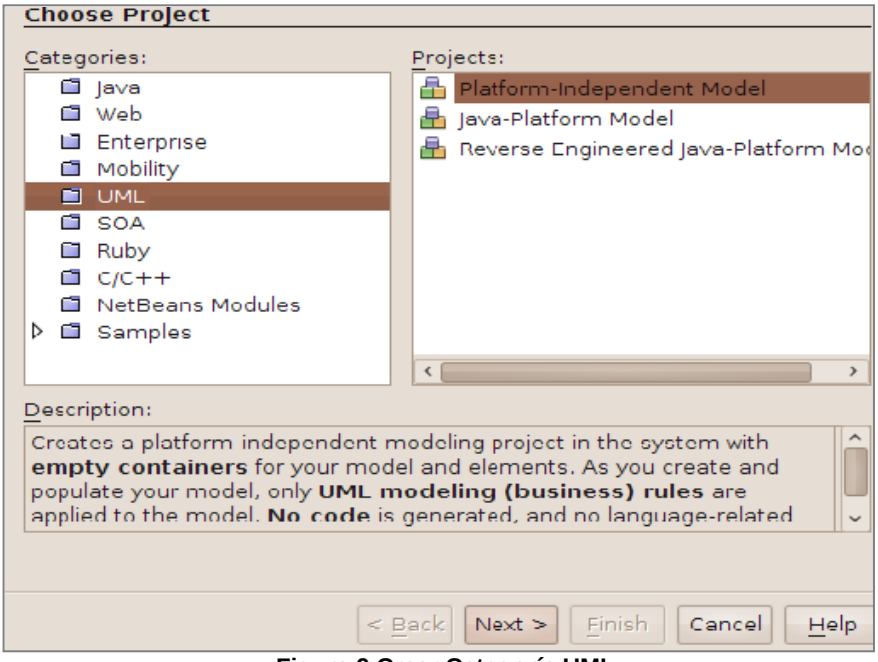


Figura 9.Crear Categoría UML.

El formulario mostrado se completa con el nombre del proyecto UML y la ruta del directorio donde se desea establecer el nuevo proyecto.

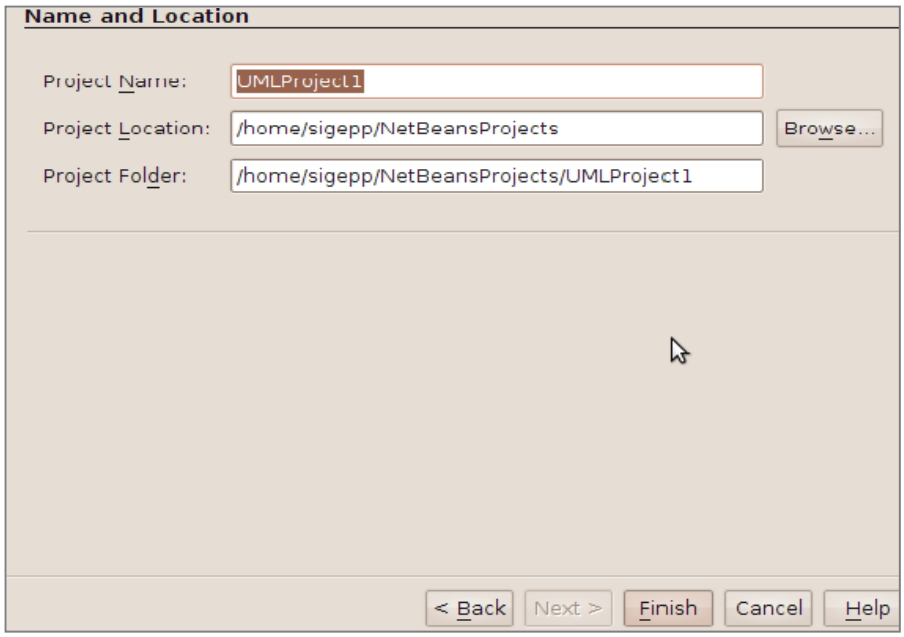


Figura 10.Crear Módulo UML.

Para finalizar la creación del proyecto se selecciona el tipo de diagrama que se creará inicialmente, completando el nombre del mismo.

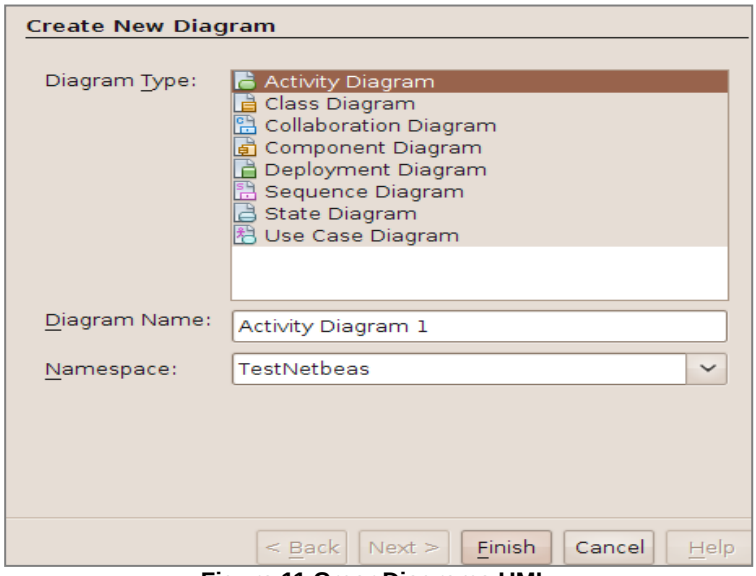


Figura 11.Crear Diagrama UML.

En este formulario se muestran todos los tipos de diagramas que se pueden realizar en la herramienta.

Luego de creado el proyecto se puede apreciar que tiene una estructura de árbol con tres ítems en

los que se almacenarán los modelos. El ítem **Model**, que es el encargado de almacenar y organizar todos los elementos agregados. Se debe tener en cuenta que no pueden repetirse los nombres, ejemplo, Clases, Actores, etc., ya que lo almacena todo en el mismo lugar, si en otro momento se necesita en otro modelo, bastará solo con tomarlos y arrastrarlos al área de trabajo. Dentro del ítem **Diagrams** se guardarán las referencias de los diagramas realizados. En la siguiente figura se muestra la estructura de un proyecto UML de NetBeans.

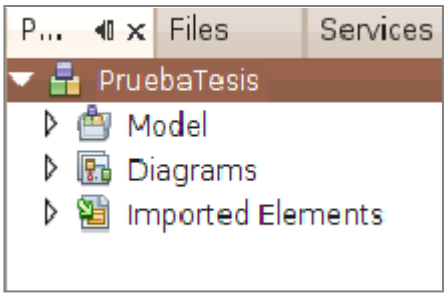


Figura 12. Módulo Creado.

En el siguiente apartado se mostrarán las características de diagramación de la herramienta mediante la realización de un caso de estudio relacionado con el Sistema de Seguridad del proyecto SCADA, y se detallarán las desventajas que presenta cada uno de sus diagramas.

2.5 Caso de Estudio. Sistema de Seguridad de SCADA.

El SCADA desea contar con un Sistema de Seguridad capaz de proporcionar principalmente servicios de identificación-autenticación de los usuarios que en él operan, auditan y trabajan en general, de forma que se pudiera mantener una diferenciación en cuanto a su capacidad de acceder a los distintos recursos del sistema, llevando un control de quiénes y cómo accedían a operaciones sobre los recursos del sistema.

Existen varios posibles factores para determinar la autenticidad de una persona, dispositivo o sistema, incluyendo algo que se conozca, algo que tenga o algo que sea. En general, mientras más factores se usen en el proceso de autenticación, más sólido será el proceso.

Es por ello que se quiere contar con un módulo de Seguridad del SCADA Nacional capaz de proveer al mismo de las funcionalidades necesarias para garantizar el trabajo autorizado por usuarios y módulos, además de brindar las herramientas necesarias para la protección contra ataques maliciosos o involuntarios al sistema por parte de personas o recursos tales como fallas de corriente, problemas de red o servidores, etc. Hoy en día los ataques a sistemas críticos tal como el SCADA son cada vez más frecuentes y la mayoría de las veces son llevados a cabo por personas que trabajan dentro de las propias empresas, estas personas dado que pueden moverse con facilidad dentro del centro pues tienen acceso a atacar los recursos de más importancia, de allí la necesidad de tener un

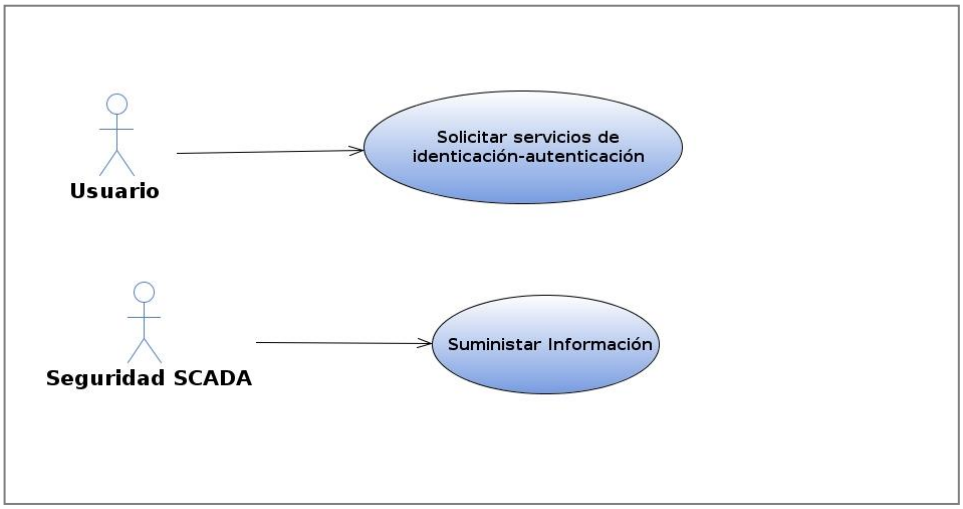
sistema que de alguna manera minimice esos riesgos, y es por eso que el sistema de seguridad estará orientado a usuarios y recursos.

También se desea contar con un subsistema Biométrico , encargado de capturar, procesar, almacenar y comparar huellas digitales de usuario, sirviendo así como motor principal para el desarrollo de las funcionalidades Autenticar por Biometría y Cambiar Huella Digital. Pero además tener el diseño de un sistema de detección de Intrusos con el objetivo de detectar posibles ataques por parte de usuarios al sistema, evaluando violaciones en procesos de autenticación y control de acceso con la emisión de alarmas en caso de violación de cualquier tipo.

Este subsistema viene equipado también de una aplicación que corresponde a la herramienta a ser utilizada por administradores de seguridad en el sistema de manera que éstos puedan visualizar y tomar acciones sobre alarmas, sesiones de usuarios y usuarios.

2.5.1 Realización del Diagrama de Casos de Uso del Negocio.

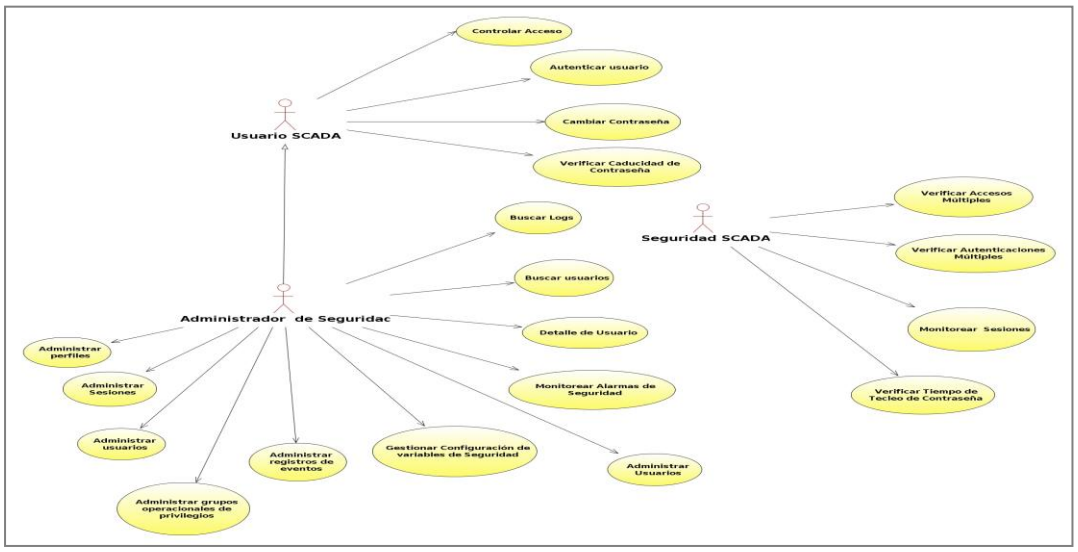
A continuación se muestra el diagrama de casos de uso del negocio del sistema de seguridad del SCADA.



Desventaja: La herramienta no contiene el estereotipo para la realización de dicho diagrama por lo que se propone el color azul para diferenciarlo de los diagramas de casos de uso del sistema.

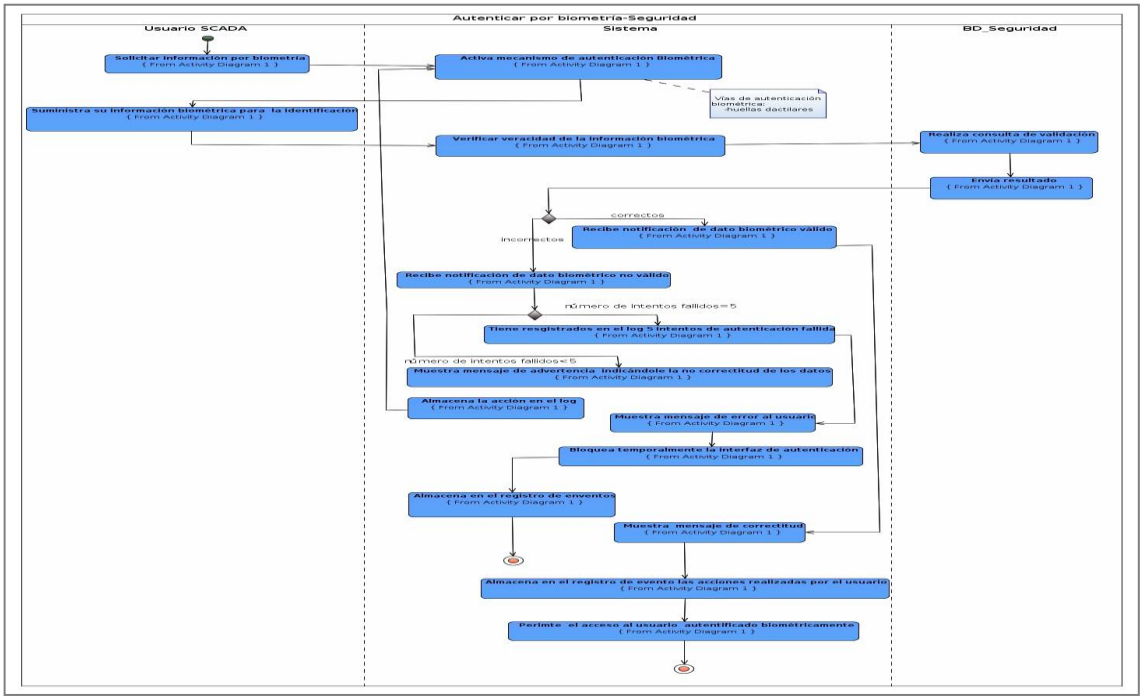
2.5.2 Realización del Diagrama de Casos de Uso del Sistema.

A continuación se muestra el diagrama de Casos de Uso del Sistema (DCUS) perteneciente al módulo de seguridad de SCADA Nacional.



2.5.3 Realización del Diagrama de Actividades.

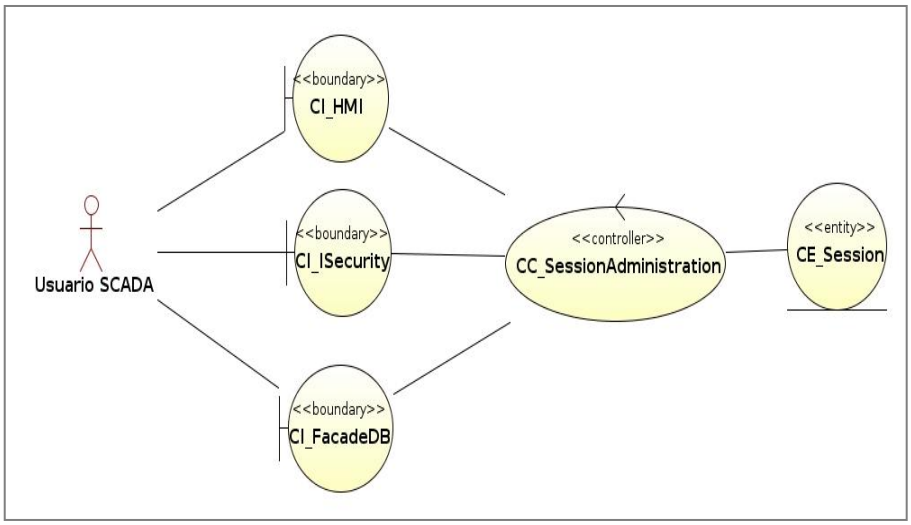
A continuación se muestra el diagrama de Actividades del Caso de uso Autenticar por Biometría.



Desventaja: No es posible representar una entidad en este diagrama, puesto que no existe un estereotipo que represente la misma.

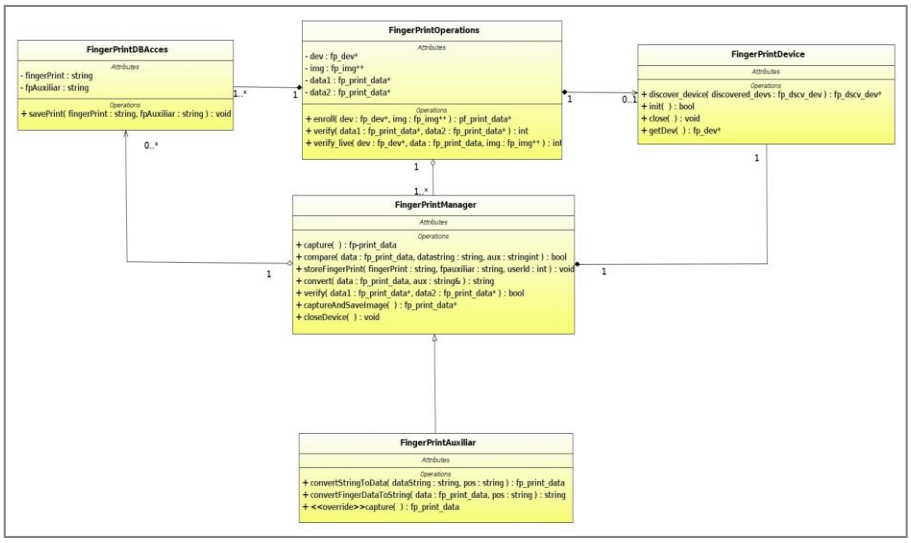
2.5.4 Realización del Diagrama de Clases del Análisis.

En la siguiente imagen se muestra el diagrama de clases del análisis representando algunas de las clases interfaz, controladoras y entidades que intervienen en el proceso de autenticación de un usuario.



2.5.5 Realización del Diagrama de Clases para aplicaciones de escritorio.

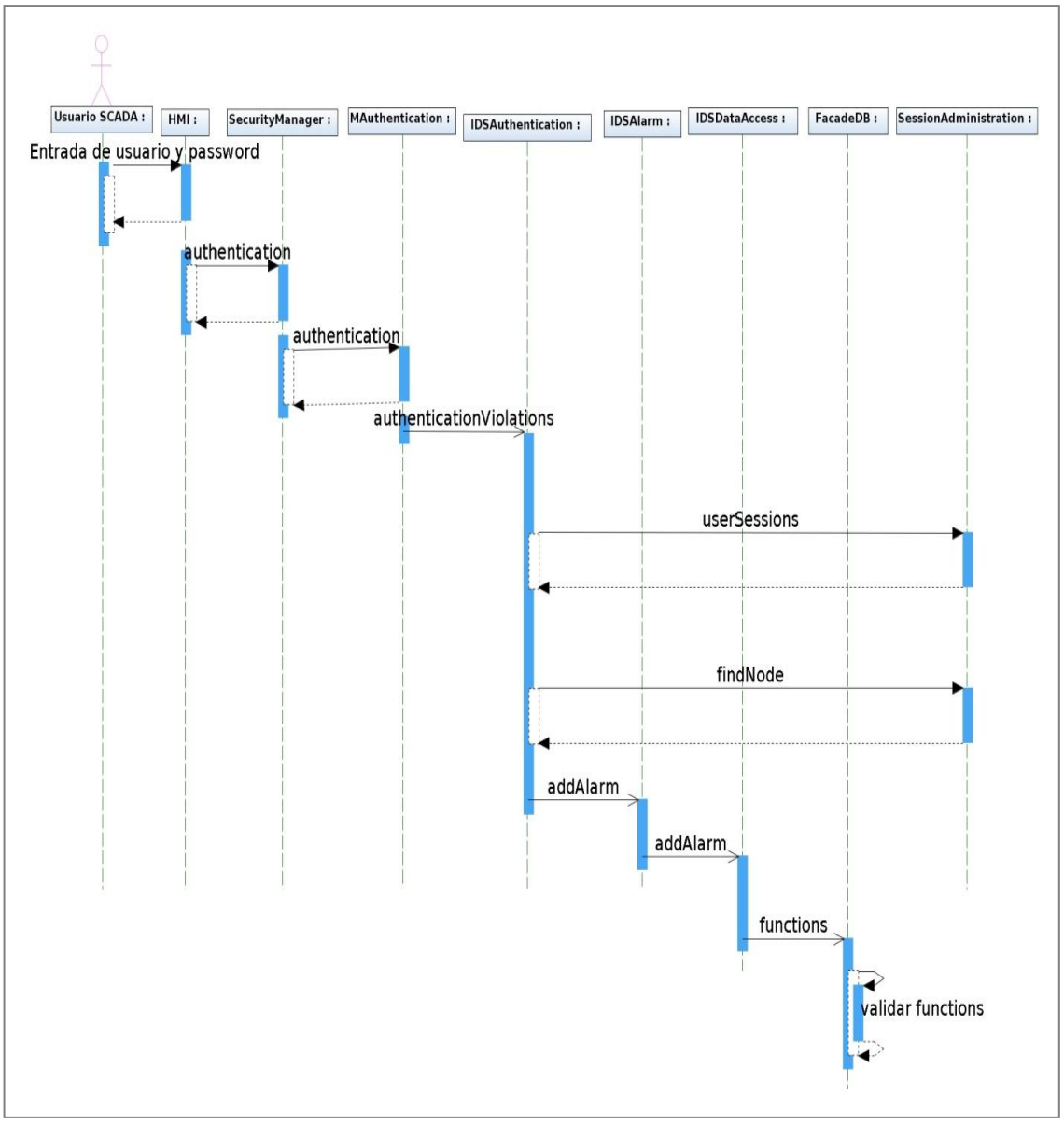
El siguiente diagrama muestra las clases que intervienen en el Subsistema Biométrico.



Desventaja: Para la realización de este diagrama la herramienta presenta dificultades con la ventana de propiedades de un atributo u operación, ya que los tipos de datos de los mismos se deben insertar manualmente.

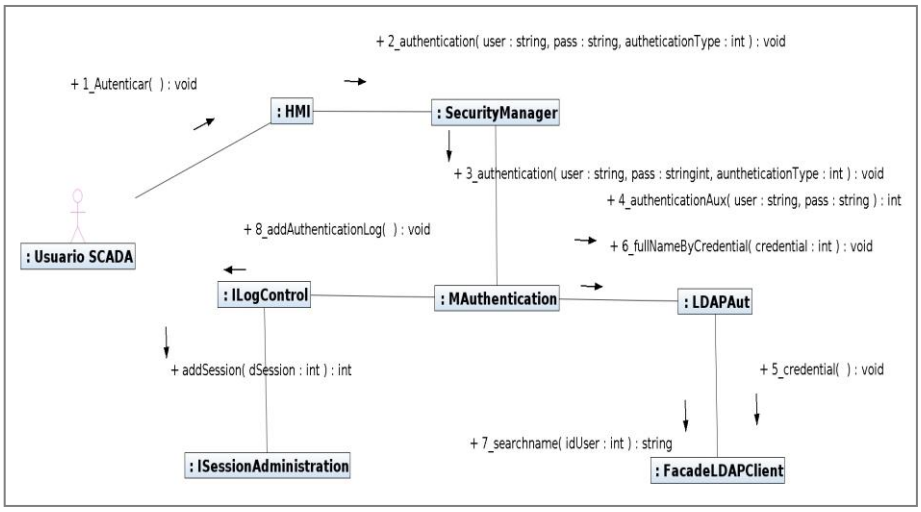
2.5.6 Realización del Diagrama Secuencia.

El siguiente diagrama de secuencia se refiere a las violaciones cometidas en el proceso de autenticación en cuanto a la cantidad máxima de usuarios que puede sostener un usuario, así como la duplicación de nodos. Este chequeo es invocado tras cada intento de autenticación en el sistema, cada violación de este tipo generará una alarma o actualizará el estado en una ya existente.



2.5.7 Realización del Diagrama de Colaboración.

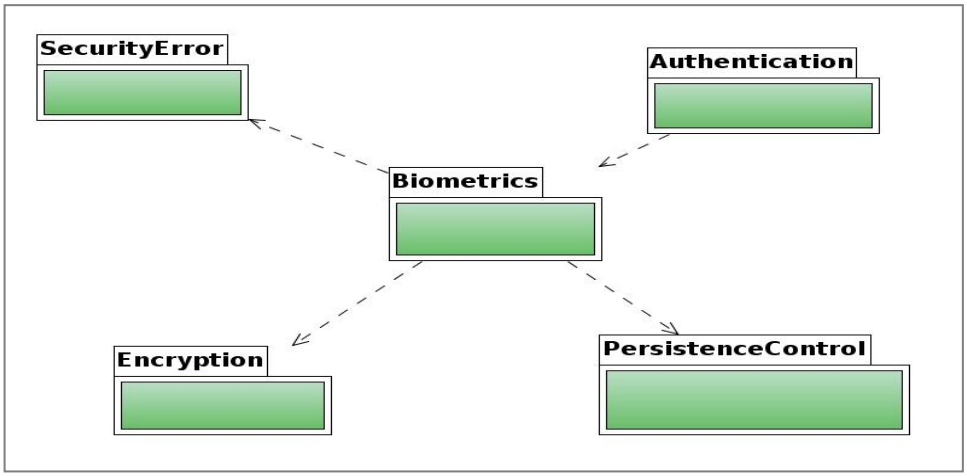
En la siguiente imagen se muestra el diagrama de colaboración del caso de uso Autenticar LDAP.



Desventaja: La herramienta en la realización de este diagrama no contiene la enumeración de los mensajes que se envían entre LIfeline, este número demuestra el orden en que se van desarrollando los mensajes, por lo que el usuario debe adicionarlo de forma manual.

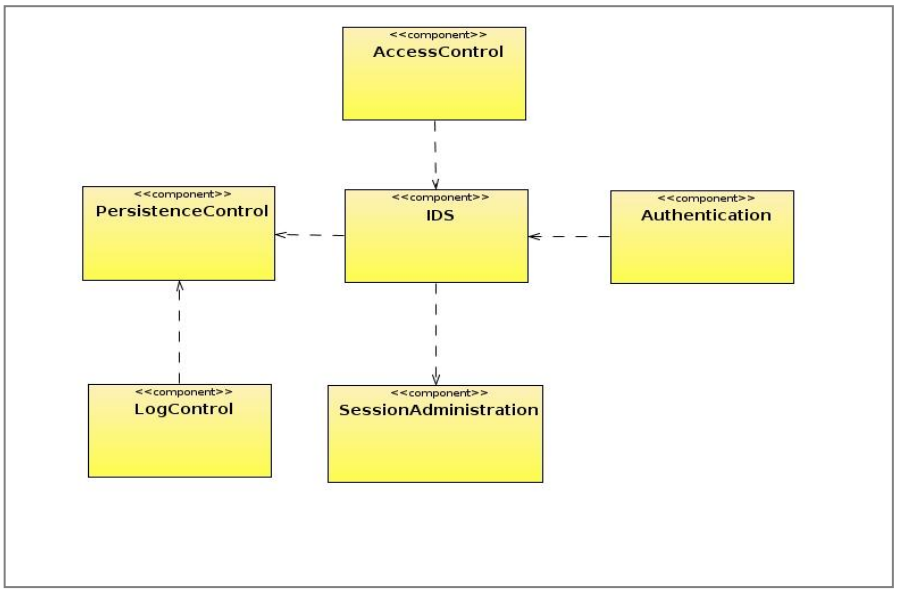
2.5.8 Realización del Diagrama de Componentes a nivel de Subsistemas de Aplicación.

El siguiente diagrama de componentes muestra la relación del subsistema de Biometría con los subsistemas: Autenticación, Encriptación, Acceso a Datos y SecurityError.



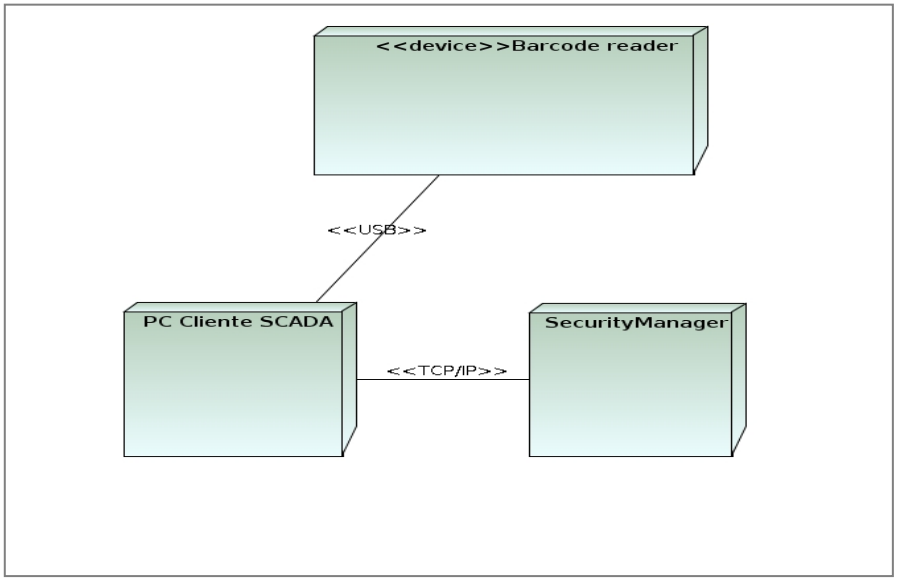
2.5.9 Realización del Diagrama de Componentes del Sistema.

En la siguiente imagen se muestra el diagrama de componentes del sistema, en el cual se representa la integración y dependencia del sistema de Detección de Intrusos para con el sistema en general.



2.5.10 Realización del Diagrama de Despliegue.

A continuación se muestra el diagrama de Despliegue del sistema de seguridad del SCADA.



2.6 Entrevista a líderes de proyectos del CEDIN

Con el objetivo de identificar los diagramas que se utilizan para la modelación y verificar la realización de los mismos en la herramienta se realizó una entrevista (ver Anexo 13) a los líderes de algunos de los proyectos pertenecientes al CEDIN. En la tabla 10 se muestra la relación de los diagramas que se utilizan en cada uno de los proyectos productivos analizados.

- Diagrama de Clases (DCL).
- Diagrama Clases del Análisis (DCA).
- Diagrama de Secuencia (DS).
- Diagrama de Despliegue (DD).
- Diagrama de Componentes (D_Comp).
- Diagrama de Caso de Uso del Sistema (DCS).
- Diagrama de Casos de Uso del Negocio (DCN).
- Diagrama de Colaboración (D_Colab).
- Diagrama de Actividades (DActv).
- Diagrama de Comunicación (DC).
- Diagrama de Estructura Compuesta (DEC).
- Diagrama General de Interacción (DGI).
- Diagramas de Tiempos (DT).

Nombre del Proyecto	DCN	DCS	DCA	DActv.	DD	D_Comp.	D_Colab.	DS	DCL	DEC	DGI	DT	DC
TETSCADA								x	x				
SCADA Meteorología	x	x			x	x			x				
Grupo para Desarrollo de Videojuegos (GDEVI)		x			x	x			x				
Paseos Virtuales		x	x		x	x		x	x				
Herramienta de Desarrollo para Sistemas de Visión Estereoscópica (HDSVE)		x			x	x							

Tabla 10. Diagramas que se utilizan en los proyectos entrevistados.

Luego de analizar los resultados de la tabla anterior se puede constatar que la herramienta propuesta

satisface cada una de las necesidades de estos proyectos en cuanto a modelado se refiere, ya que no se identificaron proyectos con necesidades de usar los cuatro diagramas que adiciona UML2, los cuales el NetBeans no proporciona momentáneamente. También se pudo verificar que los diagramas de actividades y colaboración no son de mucha importancia para los proyectos desarrollados en el centro, por lo que no se le prestará mucha atención a la modificación de éstos con vistas a eliminar las desventajas que presentan.

2.7 Conclusiones del capítulo.

Como se ha podido comprobar en el presente capítulo, el proceso de selección de una herramienta CASE es bastante complejo. Las herramientas fueron evaluadas en varias fases, que constituyeron el modelo de decisión aplicado, dando como resultado el entorno de desarrollo NetBeans con el plugin UML, como propuesta de herramienta de modelado UML para el Centro de Desarrollo de Informática Industrial. Se describió el entrono de trabajo y se mostró a través de un caso de estudio relacionado con el centro las capacidades de diagramación de la herramienta, así como las desventajas que presenta para la realización de algunos diagramas. Se demostró además a través de una entrevista realizada a los líderes de algunos proyectos del centro que la herramienta realiza todos los diagramas que los mismos necesitan. Con ésta propuesta se pretende suplir las necesidades planteadas por los desarrolladores y analistas de dicha organización.

CAPÍTULO III. VALIDACIÓN DE LA PROPUESTA

3.1 Introducción.

En el presente capítulo se profundiza sobre la herramienta propuesta en el capítulo anterior, realizando una descripción de su arquitectura, donde se identifican las características que permiten su extensión, se muestra como se crea un módulo y se describe una propuesta donde se muestra como modificarla a través del código fuente. Por último, se valida la propuesta con especialistas del Centro de Desarrollo de Informática Industrial.

3.2 NetBeans UML como propuesta de solución.

En la actualidad existen disponibles dos productos: el NetBeans IDE y el NetBeans Plataform.

El NetBeans IDE es un entorno de desarrollo, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para otro lenguaje de programación. Existe además un número importante de módulos para extender el IDE NetBeans.

También disponible está el NetBeans Platform; que es una base modular y extensible usada como una estructura de integración para crear aplicaciones de escritorio grandes. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

Ambos productos son de códigos abiertos y gratuitos para el uso tanto comercial, como no comercial. El código fuente está disponible para su reutilización de acuerdo con la *Common Development and Distribution License (*CDDL), una licencia basada en la *Mozilla Public License* (MPL). [25]

3.2.1 Arquitectura de NetBeans.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

Utiliza una plataforma RCP (Rich-Client Platform) que proporciona el potente mecanismo de módulos que usa el propio NetBeans para ser usado dentro de las aplicaciones. Esto permitirá definir la arquitectura de la aplicación en base a módulos desacoplados, se tendrá el sistema de actualización

remota de módulos de NetBeans a disposición, gestión de versiones entre módulos, classloaders independientes por módulo. Además, proporciona otras características muy útiles para el desarrollo de aplicaciones de escritorio como gestor de ventanas, API de acciones, API para la creación de diálogos y wizards, integración con java help o generación de distribución Java Web Start. [26]

3.2.2 Configuración de los módulos de NetBeans.

Un módulo es un conjunto de clases Java que proveen a una aplicación (u otros módulos) con servicios en especial. Estas clases utilizan un manifest.mf para declarar el módulo y un layer.xml de configuración para registrar su funcionalidad en el “System Filesystem”.

El NetBeans IDE provee “wizards” que ayudan bastante con esta labor “System Filesystem”, es el registro general de la plataforma NetBeans.

Además del archivo manifest.mf, posee también otra serie de archivos de soporte como:

- build.xml: contiene el ant para compilar y ejecutar la aplicación.
- bundle.properties: contiene las parejas de (propiedad, valor) como el nombre del módulo en NetBeans.
- layer.xml: Registra el módulo en el System Filesystem.
- project.xml: Contiene metadata del proyecto, como sus dependencias.
- <class_name>Settings.xml: Indica las interfaces del módulo.
- <class_name> Wstcref.xml: especifica una referencia al módulo.

Muchos de estos archivos son complejos de manejar y para editarlos es necesario entender su schema y el significado de los tags, pero NetBeans provee wizards que son mucho más explícitos para manejar estos archivos de soporte. Por esta razón, el NetBeans IDE es casi un requisito para el desarrollo de los módulos sobre esta plataforma.

Módulo Library Wrapper.

Cualquier .jar puede convertirse en un módulo de NetBeans por medio de un Library Wrapper Module. Éstos no tienen código, sólo una referencia a un .jar. Todas las protecciones del classloader de NetBeans se aplican a este módulo y por lo tanto también al .jar.

Estructura del Módulo Library Wrapper.

Posee un manifiesto donde indica:

Manifest-Version: 1.0

OpenIDE-Module: net.java.dev.colorchooser

OpenIDE-Module-Localizing-Bundle: net/java/dev/colorchooser/Bundle.properties

OpenIDE-Module-Specification-Versión: 1.0. [27]

3.2.3 ¿Cómo hacer un módulo NetBeans?

Construir un módulo para que se ejecuten de manera armoniosa y transparente dentro de NetBeans es muy fácil, al proporcionar un api simple y muy bien documentado para la creación de éstos. A continuación se detallan los pasos por los cuales atravesar para la construcción de un módulo bastante sencillo, demostrando la facilidad de extensión de la herramienta. [28]

1er Paso: Crear Nuevo Módulo.

Para crear un Nuevo proyecto, dirigirse al menú **File** y seleccionar **New Proyect.**

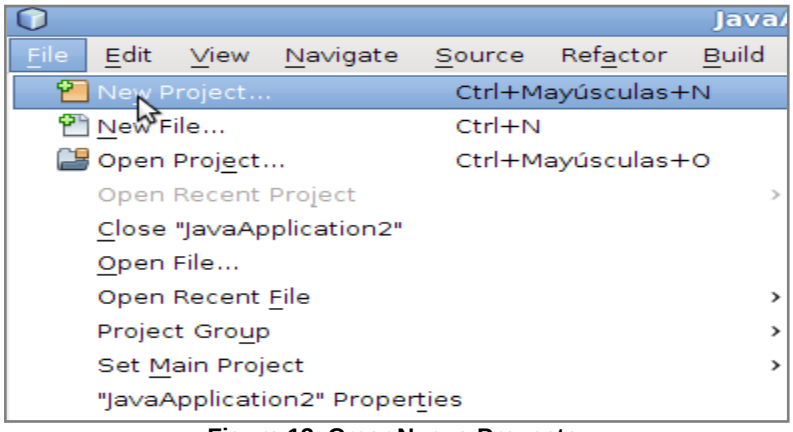


Figura 13. Crear Nuevo Proyecto.

En el siguiente formulario dentro de las categorías mostradas se selecciona **NetBeans Module** y en la modalidad del proyecto a crear se selecciona **Module**.

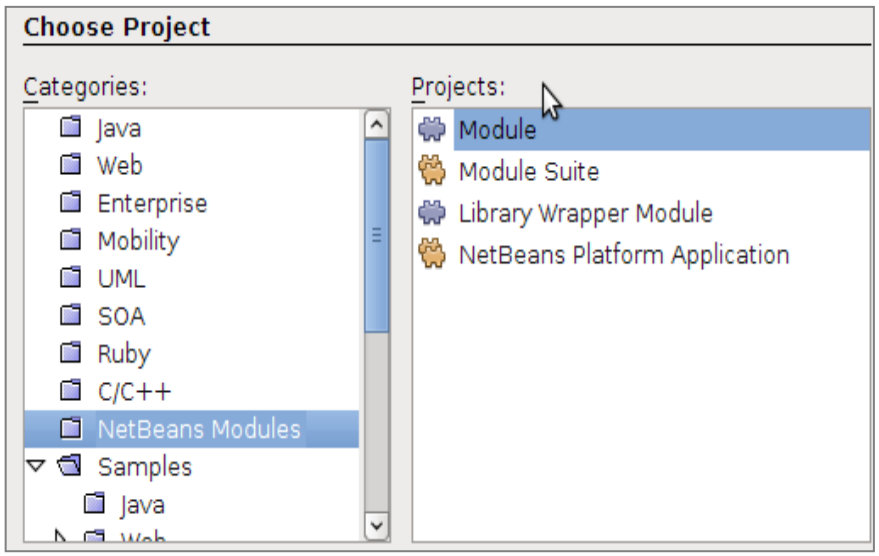


Figura 14. Crear Nuevo Módulo.

El siguiente formulario se completa con el nombre del Módulo y la ruta del directorio donde será guardado el mismo. Luego se selecciona entre dos opciones:

- Standalone Module: Permite crear plugins para cualquier aplicación (incluido el IDE).

- Add to Module Suite: Permite agregar el módulo creado a una Suite de módulos.

Como lo que se quiere es crear un plugin para el IDE se selecciona la primera opción y como plataforma de destino la actual (NetBeans IDE 6.1).

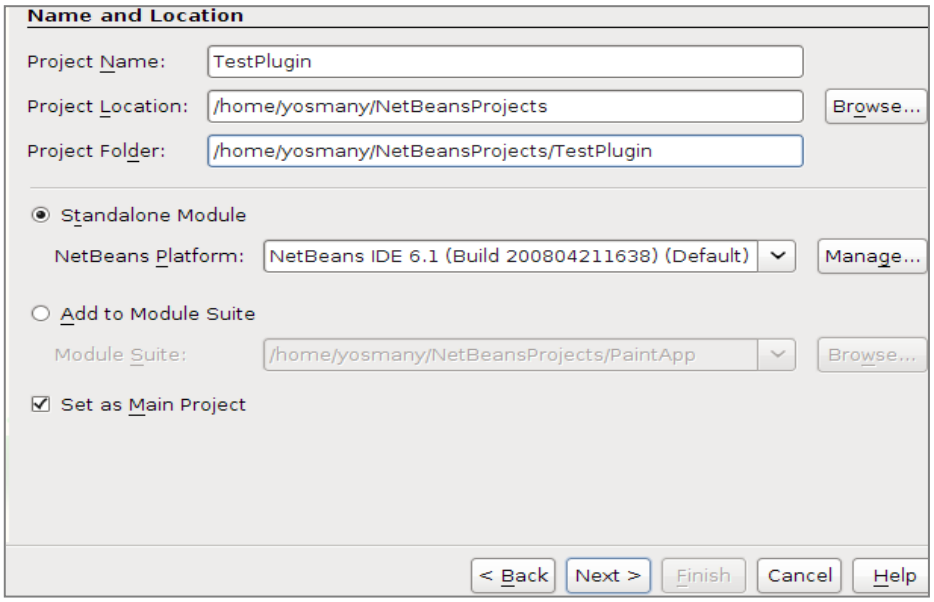


Figura 15. Nombrar el Módulo.

En la siguiente pantalla se procede a crear el nombre base del paquete.

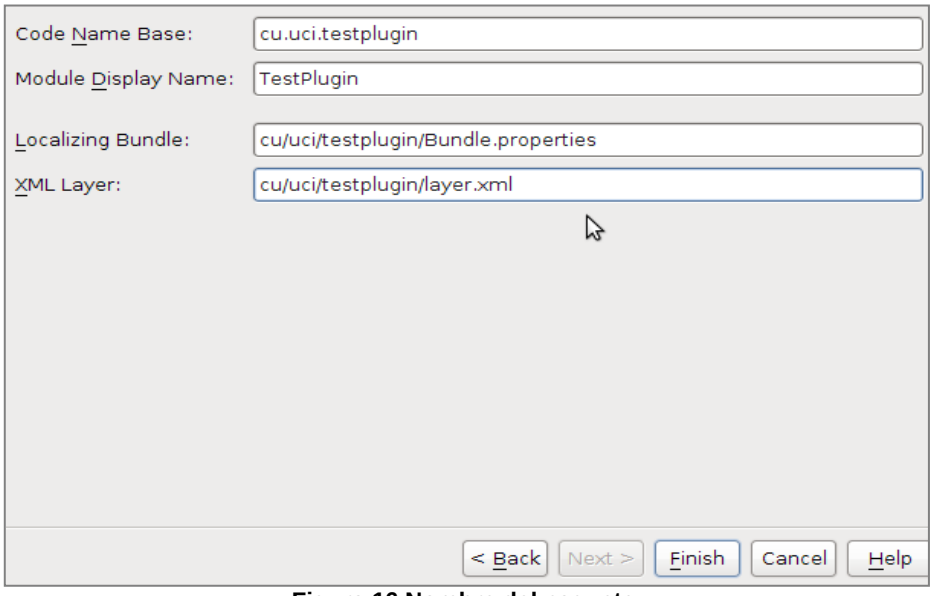


Figura 16.Nombre del paquete.

El segundo paso consiste en crear una acción.

2º Paso: Crear Acción.

Se presiona clic derecho sobre el módulo creado y luego en **New** seleccionar **Action**.

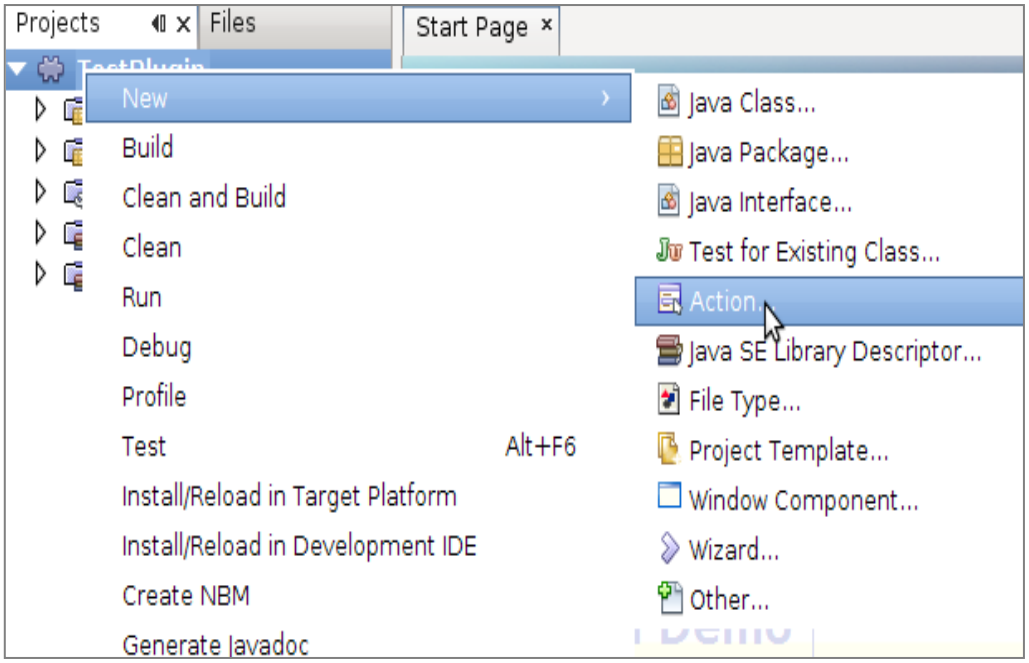


Figura 17. Adicionar Acción.

En el siguiente formulario se especifica el tipo de acción. Por defecto brinda la opción **Always Enabled** que es la adecuada, se presiona **Next** para continuar.

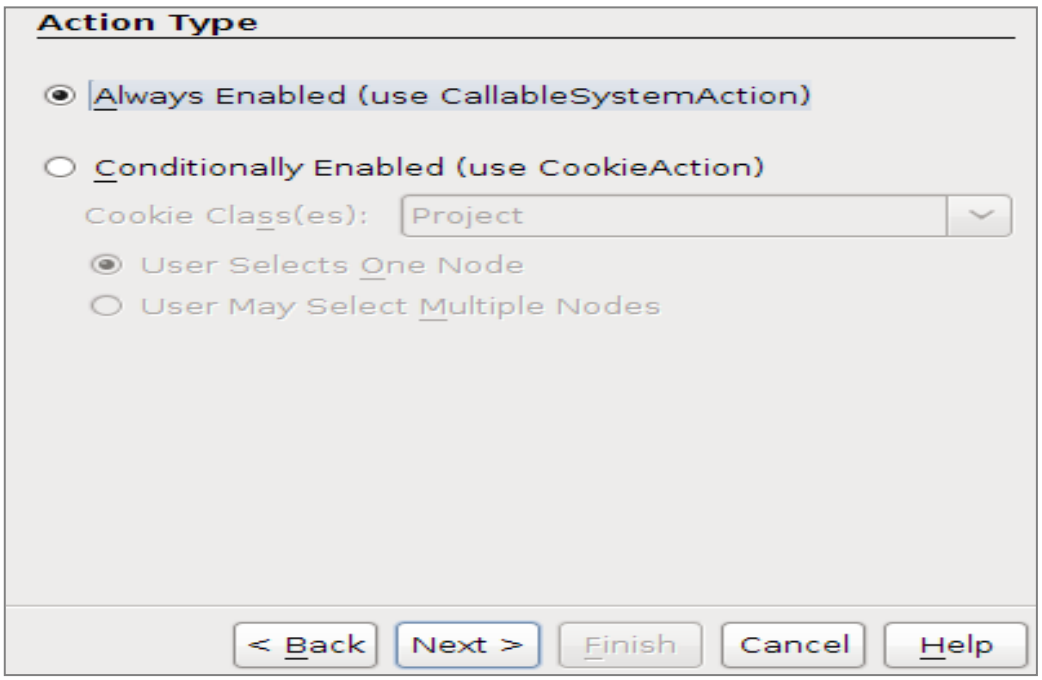


Figura 18. Elegir tipo de Acción.

En la pantalla de **GUI Registration** se define donde aparecerá la acción creada como un elemento del menú. Se elige el menú de Archivos y se selecciona **Global Toolbar Button**. En **Toolbar** se selecciona **Build** y en **Position** se selecciona el **Profile Main Project.**

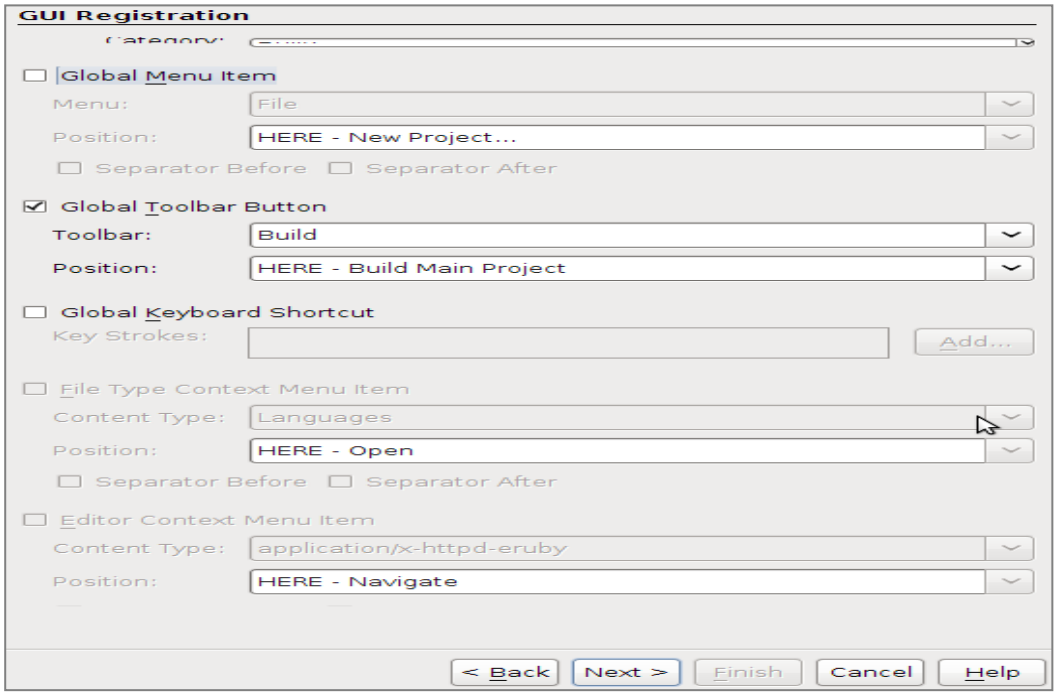


Figura 19. Registración GUI.

En la siguiente pantalla se configura el nombre del plugin en pantalla, el ícono y la ubicación. Las dimensiones del ícono no deben ser mayores de 24 x 24 y su extensión debe ser .png.

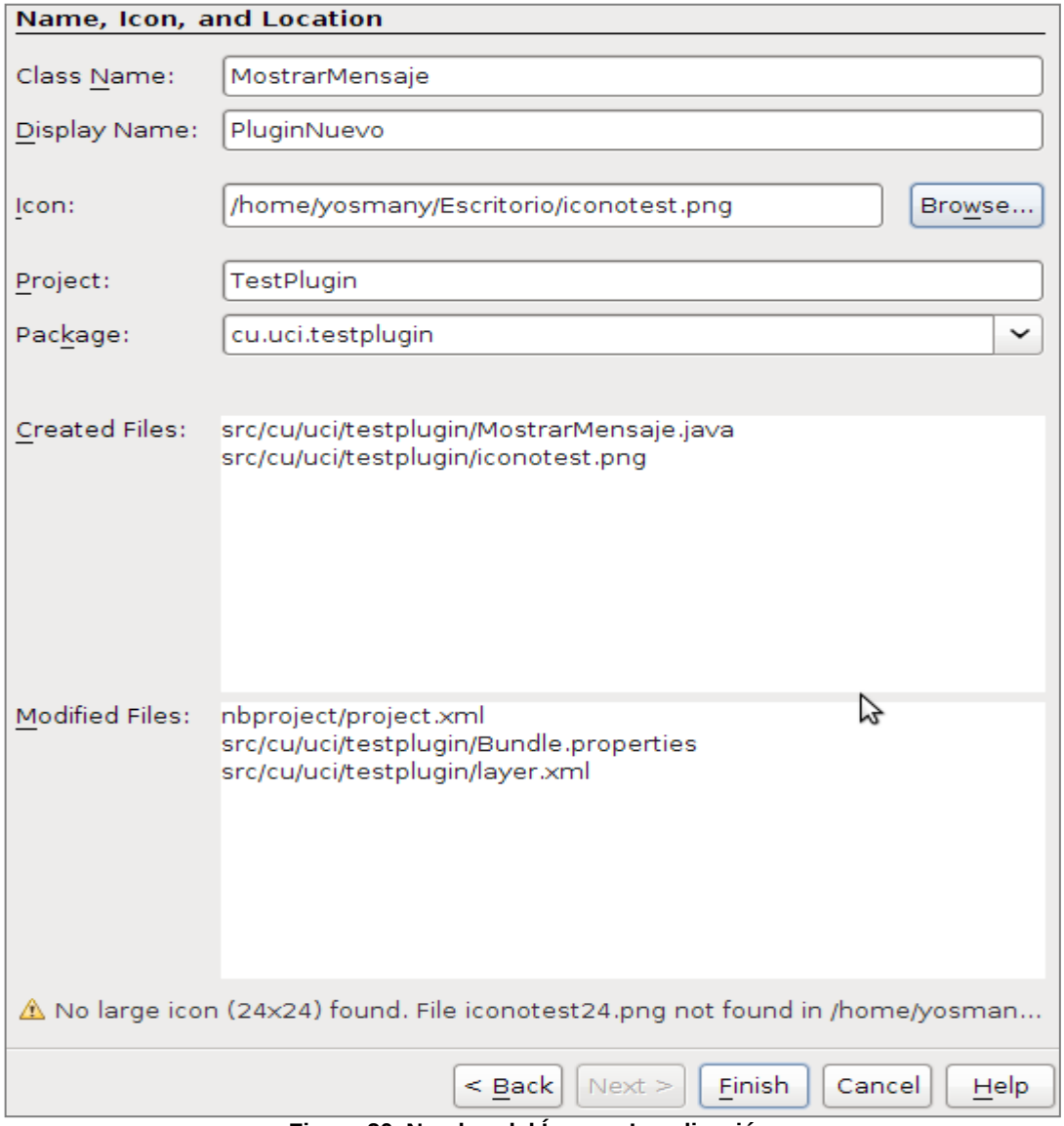


Figura 20. Nombre del Ícono y Localización.

Luego el proyecto luce así.

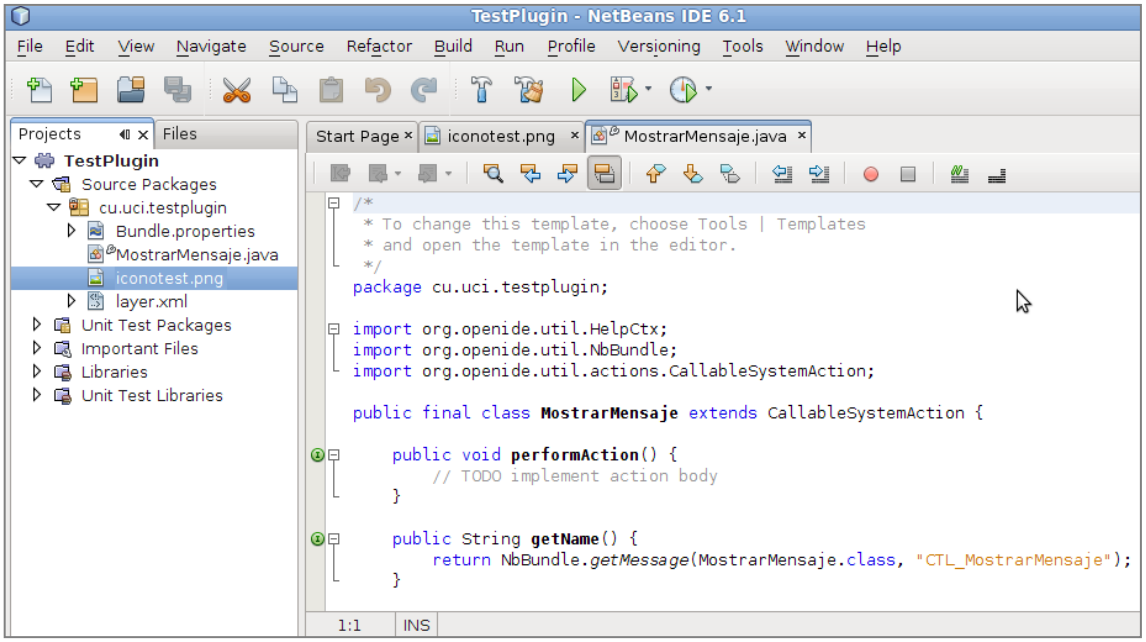


Figura 21. Acción Vacía.

Para probar el módulo se presiona clic derecho en **TestPlugin** y se selecciona **Install/Reload in Target Platform**. Esto abrirá una nueva instancia del NetBeans con el módulo instalado.

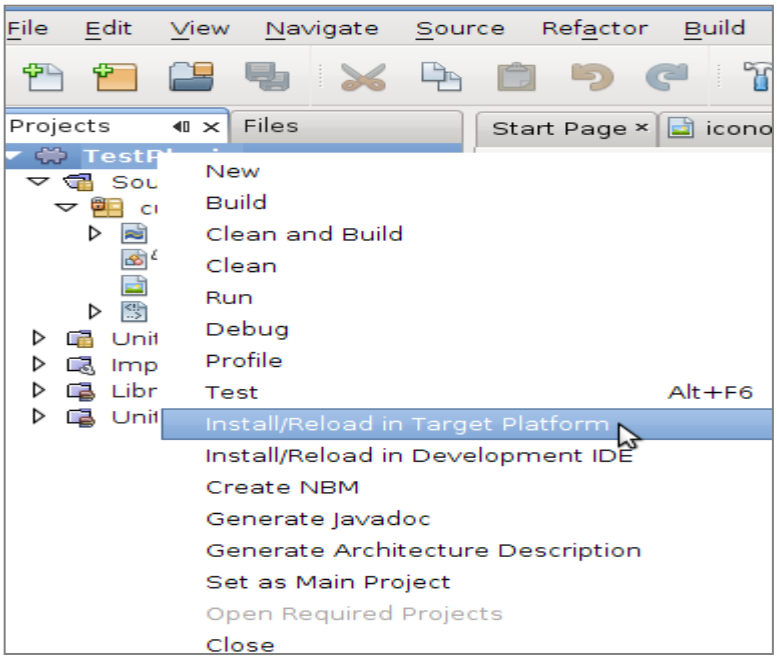


Figura 22. Instalar y Actualizar en plataforma de destino.

El ícono se ve insertado en la barra de herramientas de esta forma.

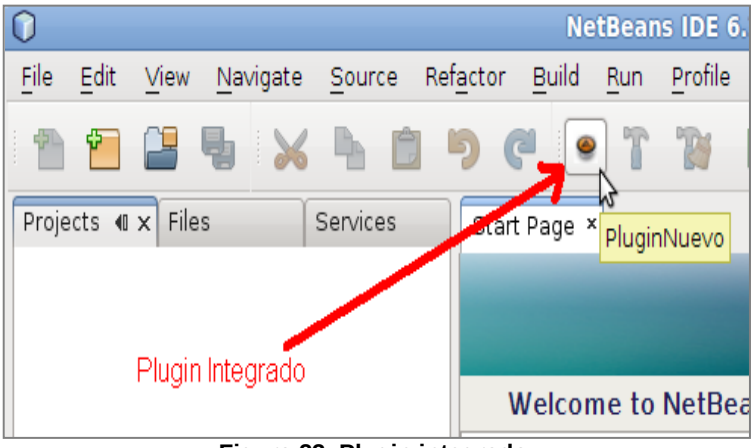


Figura 23. Plugin integrado.

Al presionar el ícono no se obtiene resultado alguno, para ello se crean las dependencias.

3º Paso: Crear Dependencias.

Un módulo puede tener dependencias de otros módulos, al intentar instalarlos NetBeans preguntará acerca de sus dependencias.

Para configurar las dependencias se presiona clic derecho sobre **TestPlugin** y luego **Propiedades**.

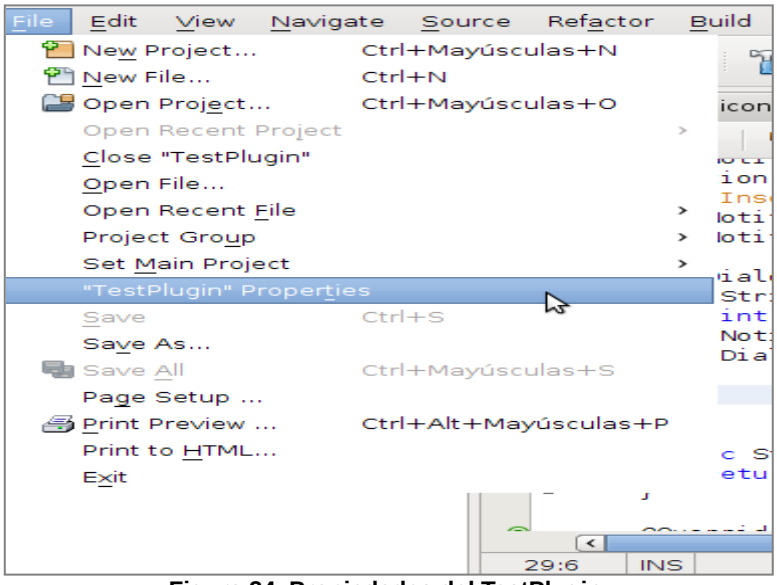


Figura 24. Propiedades del TestPlugin.

En la pantalla de Propiedades dentro de categorías se selecciona bibliotecas (Libraries). Luego se visualizan una lista de las plataformas java, NetBeans y todas las dependencias del módulo. Por defecto se verá la dependencia de servicios públicos de la API del módulo.

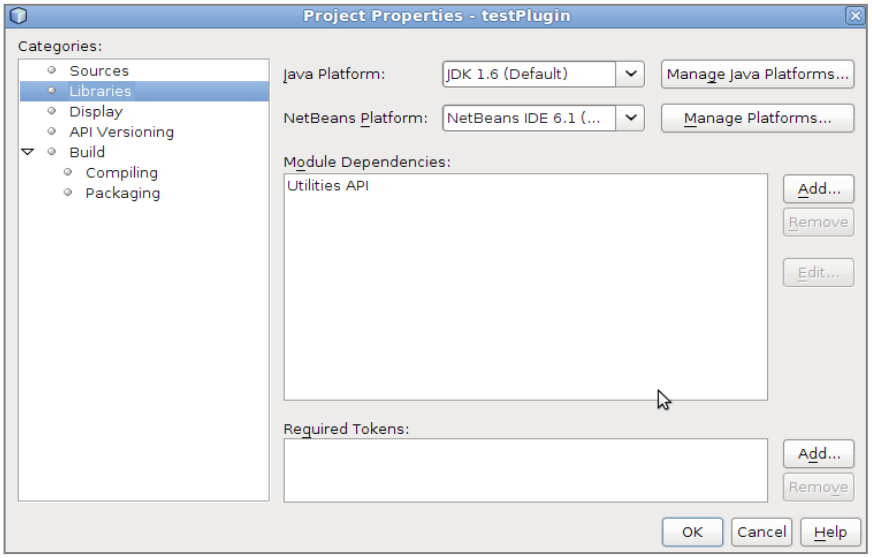


Figura 25. Dependencias del Módulo.

Al presionar clic en el botón **Add**, se verá la pantalla Agregar módulo de dependencia donde se enumeran todos los módulos de NetBeans.

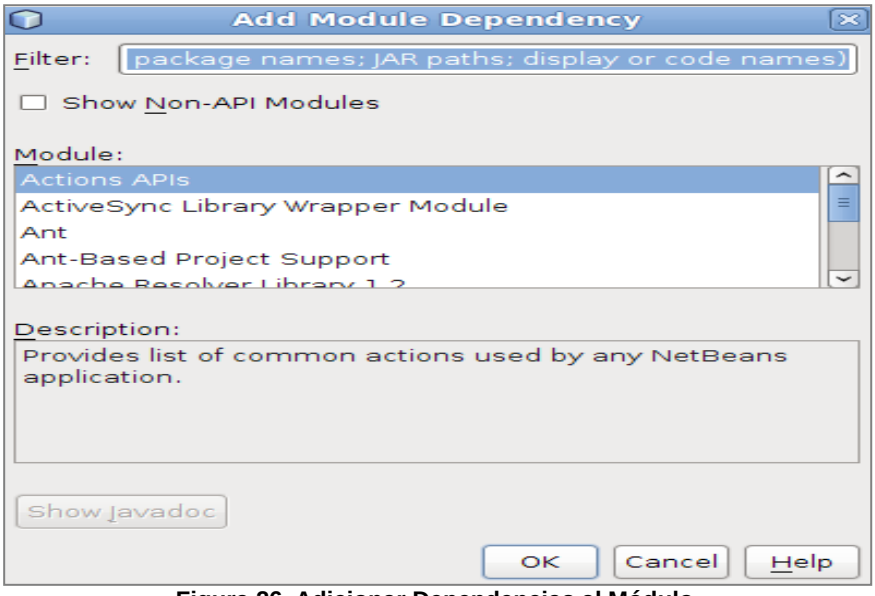


Figura 26. Adicionar Dependencias al Módulo.

Se agrega una dependencia para la API de cuadros de diálogo. Tipo de "diálogos" en el campo de texto del filtro.

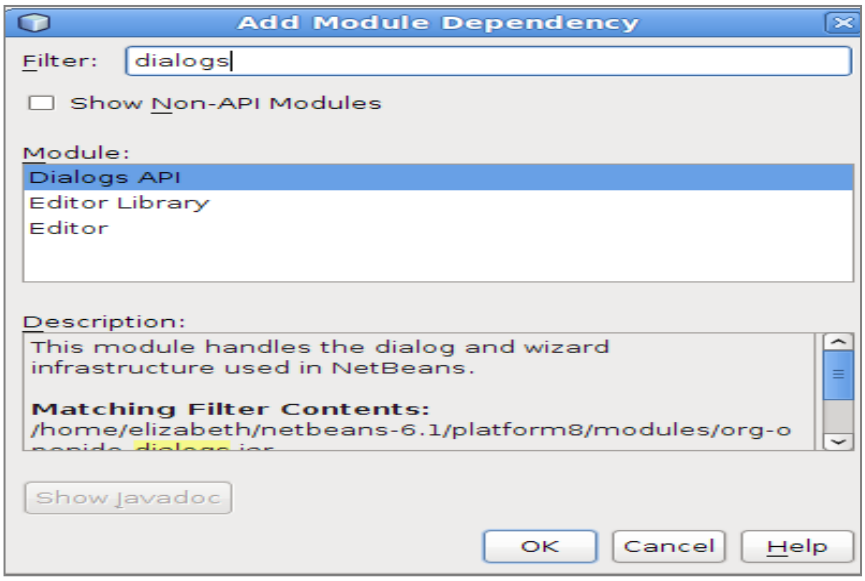


Figura 27. Adicionar dependencia Dialogos API.

Se selecciona el módulo **Dialogs API** y el módulo cuenta con dos dependencias Dialogs API y Utilities API.

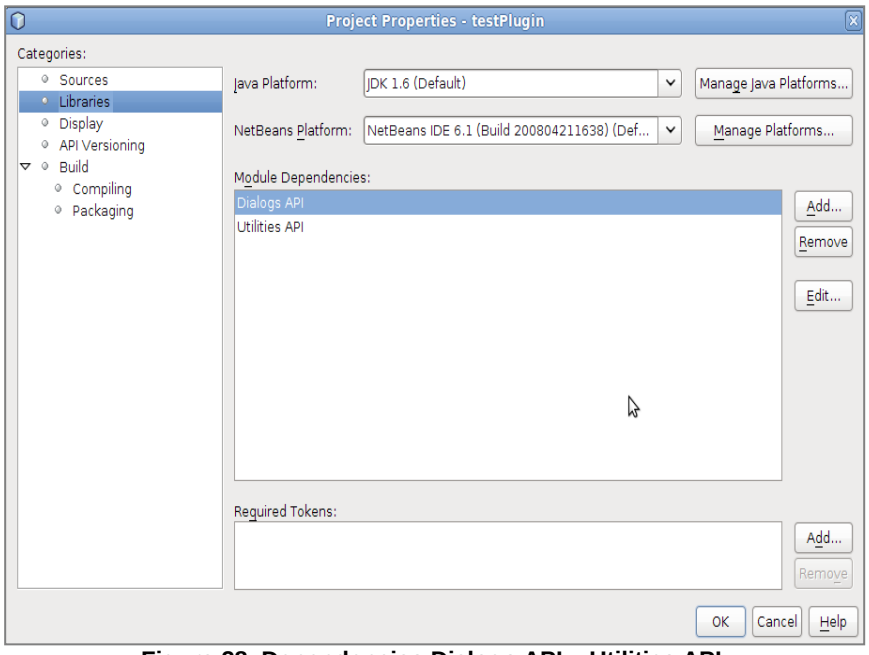


Figura 28. Dependencias Dialogs API y Utilities API.

Se procede a reescribir el método actionPerformed() en la clase MostrarMensaje.java. Utilizando NotifyDescriptor.InputLine se creará un cuadro de diálogo de entrada de línea, recibe el nombre de usuario y al presionar aceptar se mostrará el mensaje. EL siguiente fragmento de código fue el que se utilizó para lograr el funcionamiento del módulo.

```
public final class MostrarMensaje extends CallableSystemAction {

   public void performAction()
{
   NotifyDescriptor.InputLine question;
   question = new NotifyDescriptor.InputLine("Nombre:",
      "Inserte su nombre?",
      NotifyDescriptor.OK_CANCEL_OPTION,
      NotifyDescriptor.QUESTION_MESSAGE);
      if (DialogDisplayer.getDefault().notify(question) == NotifyDescriptor.OK_OPTION)
       {
         String msg = "Hola "+question.getInputText()+"!!!."+"\n"+"El plugin de Netbean funciona correctamente.";
         int msgType = NotifyDescriptor.WARNING_MESSAGE;
         NotifyDescriptor d = new NotifyDescriptor.Message(msg, msgType);
         DialogDisplayer.getDefault().notify(d);
       }
    }
```

Nuevamente presionar clic derecho en **TestPlugin** se selecciona **Install/Reload in Target Platform**. Se abre una nueva instancia de NetBeans, se presiona sobre el ícono y se muestra el siguiente formulario.

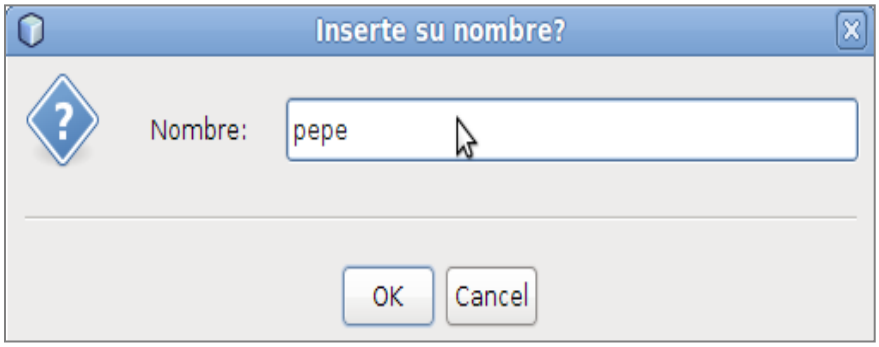


Figura 29. Formulario para agregar Nombre.

Luego de agregar el nombre se muestra el siguiente mensaje.

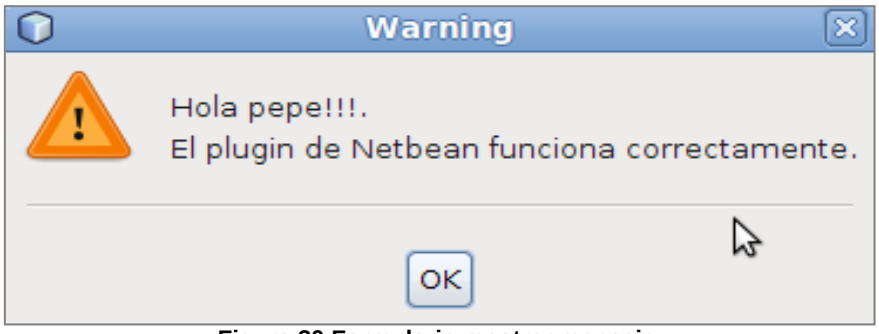


Figura 30.Formulario mostrar mensaje.

Para crear un archivo .nbm para compartirlo con otros desarrolladores se presiona clic derecho sobre **TestPlugin** y se selecciona **CreateNBM** como se muestra en la siguiente figura.

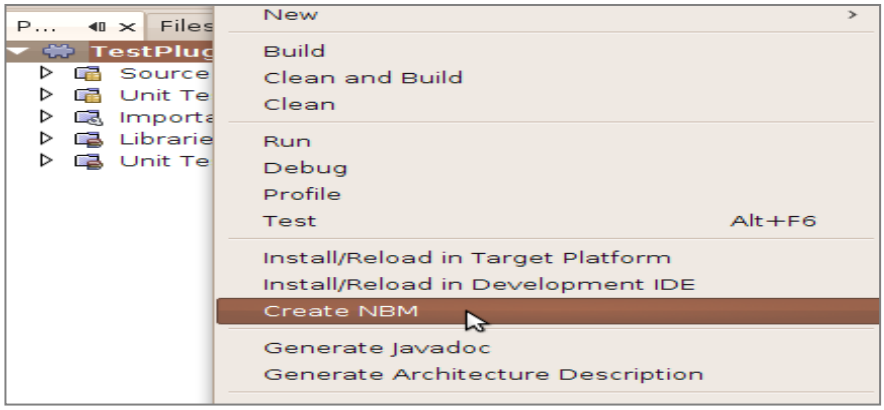


Figura 31. Crear archivo .nbm.

El archivo que se creó se generó dentro del directorio build del proyecto, con el nombre **cu-uci-testmodule.nbm.**

Como se ha podido observar NetBeans es una herramienta fácil de extender. Una de las características que más se agradecen es la manera de obtener los plugins, que cada día se parece más a los repositorios de software de las distribuciones Linux, lo cual hace comodísimo tener el IDE perfectamente adaptado a la tarea que se está realizando. Con solo un clic se activa y se desactivan módulos de modo que los módulos que no se están usando no se cargan, aligerando así los recursos consumidos. [28] [29]

3.3 Análisis de la arquitectura del plugin UML.

La arquitectura del plugin UML de NetBeans es modular o basada en componentes, lo cual quiere decir que fue desarrollado a partir de módulos que agrupan funcionalidades comunes y que dependen directamente de un módulo principal o núcleo. Existe un módulo que no implementa ninguna funcionalidad nombrado **org.nebeans.modules.uml.kit** el cual se encarga de agrupar todos los restantes módulos y brindar la descripción general del plugin.

En la siguiente imagen se muestran los módulos que componen el plugin UML.

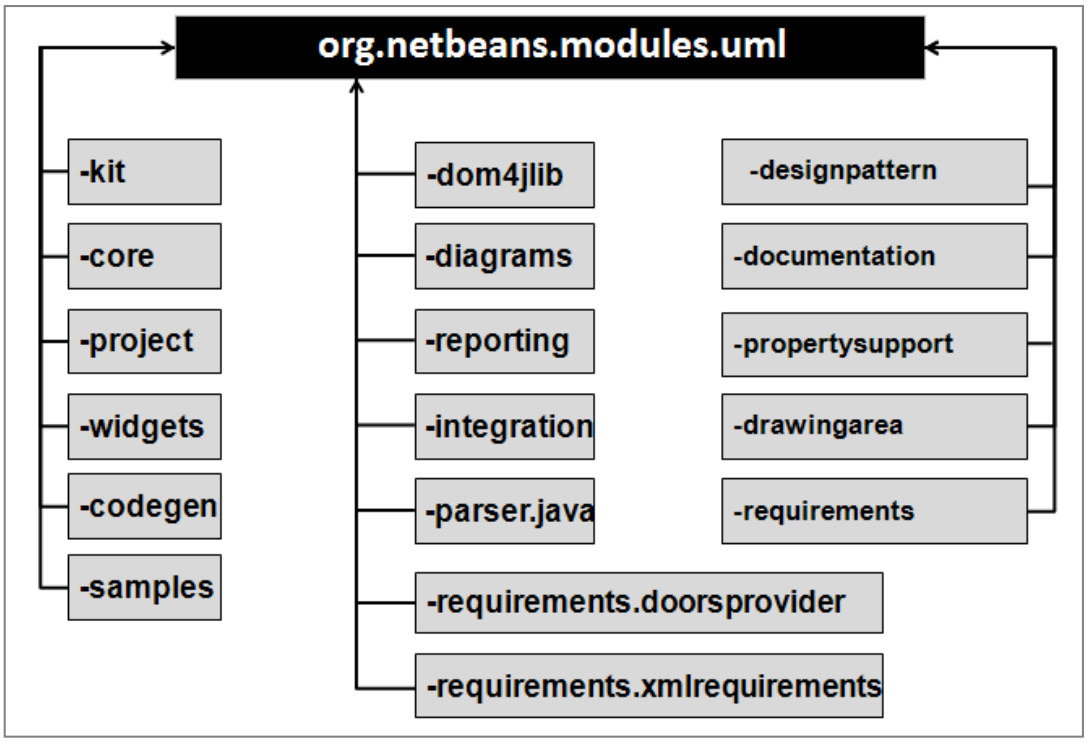


Figura 32.Módulos del plugin UML del NetBeans.

A continuación se muestra una breve descripción de los principales módulos que integran el plugin UML.

- **org.netbeans.modules.uml.core**: Constituye el núcleo del NetBeans UML, en este módulo son implementadas las funcionalidades básicas de un proyecto UML y comunes a otros componentes. Es definido el modelo de datos (metadatos) a utilizar y las clases encargadas de manejar los mismos, también son definidas las funcionalidades de los principales componentes visuales que dan paso a la creación de los diagramas y la gestión del espacio de trabajo utilizado.
- **org.netbeans.modules.uml**.**proyect:** En este módulo se define el concepto de proyecto tipo UML, estableciendo todos los recursos visuales necesarios para su identificación dentro del IDE NetBeans. Se crean los elementos de interfaz de usuarios (wizzards, formularios, etc.) necesarios para la creación de los proyectos y diagramas.
- **org.netbeans.modules.uml.codegeneration:** Componente que implementa la capacidad de generación de código a partir de los elementos modelados.
- **org.netbeans.modules.uml.designcenter:** Componente que define los tipos de diagramas creados e implementa sus funcionalidades. También se encarga de la organización y estructuración de la paleta de herramientas de modelado por cada diagrama.

- **org.netbeans.modules.uml.drawingarea:** Componente que permite el control del área de modelado. En él son implementados todos los efectos visuales que intervienen en el modelado.
- **org.netbeans.modules.uml.dom4dom4jlib:** Este módulo no encapsula ninguna funcionalidad, su objetivo es únicamente el de proveer las librerías necesarias (org-dom4j, Tidy, xsdlib, jaxen, etc.) para la implementación de las funcionalidades de los módulos. Estas librerías permiten la manipulación de ficheros XML, la persistencia de objetos Java en archivos XML y entre otras cosas la incorporación de nuevos tipos de datos necesarios para complementar las funcionalidades del API UML.

Los componentes o módulos pueden tener dependencia directa entre ellos y con las APIs de desarrollo que brinda la plataforma de desarrollo de extensiones del NetBeans, éstas posibilitan la utilización de componentes visuales como ventanas, formularios, cuadros de diálogos, botones, barras de menús etc. y el acceso a las variables de configuración del IDE. Algunas de las APIs más usadas por los componentes del plugin UML son:

- org.netbeans.api.visual
- org.openide.actions
- org.openide.awt
- org.openide.dialogs
- org.openide.explorer
- org.openide.filesystems
- org.openide.modules
- org.openide.nodes
- org.openide.util
- org.openide.windows
- org.netbeans.spi.palette
- org.netbeans.spi.navigator
- org.netbeans.modules.projectuiapi
- org.netbeans.modules.projectapi
- org.netbeans.modules.editor

3.3.1 Descripción del módulo org.netbeans.modules.uml.core.

Como se había explicado anteriormente el módulo **org.netbeans.modules.core** o núcleo del plugin UML define la línea base de la arquitectura del plugin y el modelo de datos a utilizar así como la definición de los conceptos a partir de éstos. El modelado de las entidades de dominio para representar la información de los artefactos, diagramas y otros elementos del modelado se hizo sobre la idea de un grafo que agrupa una serie de nodos, los cuales tienen una jerarquía de nodos padres e hijos, representando de esta forma las relaciones que existen entre estos elementos. Para representar estos nodos de forma jerárquica y su persistencia en documentos XML fue utilizada la librería Dom4j que de forma optimizada realiza la lectura y escritura de archivos XML, para comprender su funcionamiento hay que conocer la estructura de los documentos XML. No tienen

mucha complicación son documentos cuya información está estructurada en forma de árbol, es decir, cada elemento o nodo contiene sus elementos y/o sus atributos.

En la figura 33 se puede observar la estructura de paquetes que tiene el código fuente del módulo.

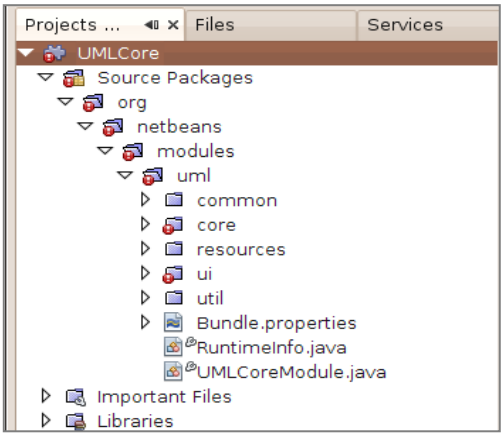


Figura 33. Estructura de paquetes del módulo org.netbeans.modules.uml.core.

En el paquete ***org.netbeans.modules.uml.common*** son agrupadas las clases comunes que puedan ser invocadas por otras clases en otros paquetes de código y clases de interfaz de usuario como ventanas de diálogo y ventanas de notificación de mensajes. También en este paquete se encuentran los archivos de internacionalización del módulo y otras clases útiles que facilitan el trabajo con los tipos de datos primitivos del lenguaje java.

En el paquete ***org.netbeans.modules.uml.core*** se encuentran las clases que representan el modelo de datos y la implementación de la lógica de negocio. El modelo de datos, su estructura, concepto y comportamiento se encuentran ubicados en el paquete de código ***org.netbeans.modules.uml.core.metamodel***.

La clase **org.netbeans.modules.uml.core.Application** es el punto de entrada del plugin UML, se encarga de implementar funcionalidades relacionadas con espacio de trabajo, y los proyectos de tipo UML. En la siguiente figura se muestran algunos de los métodos que implementa la clase Application.

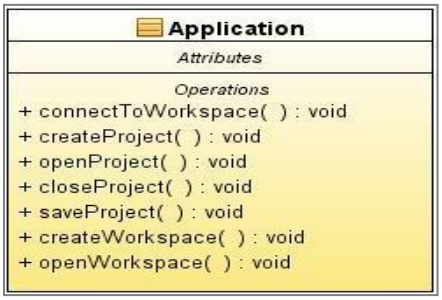


Figura 34.Métodos de la clase Application.

Las imágenes utilizadas para la representación de los artefactos de modelado, los íconos de las funcionalidades de la interfaz de usuario y otros recursos se encuentran ubicados en el paquete de código ***org.netbeans.modules.uml.core.resources***.

El núcleo es de gran importancia si se desea incorporar una nueva funcionalidad al plugin, este permite acceder a los proyectos UML creados y a toda su información representada de forma objetual. Para esto es necesario conocer el modelo de datos del plugin UML.
A continuación se describen algunas de las clases más importantes que integran el modelo de datos.

- **org.netbeans.modules.uml.project.UMLProject:** Esta clase representa un proyecto UML por lo que contiene toda la información del mismo, como el nombre, propiedades de configuración, listado de elementos asociados (diagramas, actores, clases, etc.) entre otros atributos. Desde cualquier componente que tenga como dependencia el módulo **org.netbeans.modules.uml.project** se podrá acceder a las propiedades del proyecto.
- **org.netbeans.modules.uml.core.metamodel.core.foundation.Elemen**t: Esta clase es la representación genérica de los elementos de modelado (actor, clase, caso de uso, etc.), a través de ella se puede acceder a la jerarquía de nodos del elemento que representa.

3.3.2 Interfaz de usuario y elementos gráficos.

Los objetos de interfaz de usuario como formularios, cajas de texto, botones entre otros son implementados haciendo uso de la librería **Swing (javax.swing.*)** que empaqueta una serie de componentes GUI (Graphical User Interface).
A diferencia de los componentes AWT (Abstract Window Toolkit), que están asociados a los recursos de pantalla nativa, los componentes Swing tienen su propia estrategia de pintado (en lugar de utilizar la API nativa como DirectX en Windows) en la pantalla. Esto da lugar a una ejecución más lenta, pero una aplicación Swing se ve igual en todas las plataformas. La generación de elementos gráficos es realizada mediante el uso del API **Java2D** que comprende el renderizado, la definición de figuras geométricas, el uso de fuentes de letras, la manipulación de imágenes y el enriquecimiento en la definición del color. También permite la creación de bibliotecas personalizadas de gráficos avanzados o de efectos especiales de imagen e incluso puede ser usada para el desarrollo de animaciones u otras presentaciones multimedia al combinarla con otras APIs de Java. El renderizado es realizado mediante la clase **java.awt.Graphics2D,** lo que proporciona un control más potente sobre la presentación de texto, imágenes o figuras geométricas. Un objeto Graphics (que es una clase abstracta) representa el lienzo abstracto y el contexto en el que puede dibujarse cualquier cosa, este lienzo puede estar enlazado con un área física de un monitor, o representar una imagen en memoria que solo se desea manipular y no tiene representación directa durante este proceso [30] [31].

Las figuras geométricas que representan los elementos de modelado UML son creadas a partir de las clases que provee el API Java2D, pertenecientes a los paquetes java.awt y java.awt.geom, que definen figuras geométricas simples, tales como puntos, líneas, curvas y rectángulos. Usando las clases geométricas, es posible definir y manipular virtualmente cualquier objeto bidimensional de una manera sencilla. Las clases e interfaces a utilizar para este fin son las siguientes:

- **Interfaces:**
  - Pathelterator: Define métodos para iterar sobre los distintos segmentos o subtrazos que conforman el contorno de una figura o rango tipográfico.
  - Shape: proporciona un conjunto básico de métodos para describir y generar contornos de objetos geométricos. Es implementada por generalPath y una multitud de clases geométricas.
- **Clases**
  - Área: Representa un área geométrica (con la forma que sea) que soporta operaciones de intersección, unión, etc. para obtener nuevas áreas con formas diferentes.
  - FlatteningPathlteratore: ha comentado que Pathlterator proporciona los subtrazos de un contorno, estos subtrazos pueden ser segmentos o curvas de escasa complejidad. La clase FlatteningPathlterator es igual a Pathlterator pero siempre devuelve segmentos, de ahí lo de *Flattening* (aplanar en español).
  - GeneralPath: implementa a Shape. Representa el trazo de un objeto geométrico construido a partir de líneas y curvas cuadráticas y cúbicas (curvas que pueden expresarse matemáticamente con escasa complejidad).
  - RectangularShape: proporciona la base de un gran número de figuras (Shape) que se dibujan enmarcadas en un rectángulo.
  - Figuras geométricas:
    - Arc2D: representa un arco.
    - CubicCurve2D: representa un segmento curvo.
    - Ellipse2D: representa una elipse, y extiende a la clase RectangularShape.
    - Line2D: representa un segmento de línea, e implementa a Shape.
    - Point2D: es un punto que representa una localización en un sistema de coordenadas.
    - QuadCurve2D: representa un segmento de curva cuadrática, e implementa a Shape.
    - Rectangule2D: representa un rectángulo y extiende a RectangularShape.
    - Roundrectangle2D: representa un rectángulo con los vértices redondeados, y extiende también a RectangularShape.

Las medidas de todas estas figuras pueden especificarse tanto en formato double como float, para ello solo es necesario añadir la extensión .Double o .FLoat al nombre del constructor de la figura que se quiere crear, como por ejemplo Arc2D.Float (). [31]

3.3.3 Estrategia de extensión.

Existen dos posibles vías para lograr extender las funcionalidades del plugin UML, ambas dependen del conocimiento que se tenga de las tecnologías usadas en su implementación.

La primera sería modificar directamente un componente, lo que incluye añadir, eliminar o mejorar sus funcionalidades, respetando la arquitectura definida en la que cada componente es experto en implementar un conjunto de funcionalidades, es decir no incluir nuevas funcionalidades en módulos que no son expertos en ese ámbito.

Una segunda opción consiste en implementar un nuevo conjunto de funcionalidades agrupadas en un nuevo módulo, el cual debe tener como dependencia al módulo ***org.netbeans.modules.uml.core*** (u otro módulo del cual se necesiten sus funcionalidades) para poder hacer uso del modelo de datos y otras clases necesitadas. Esta vía se aplicó anteriormente cuando se detalló cómo realizar un plugin en el epígrafe 3.2.3, solo que las dependencias aplicadas no fueron las del UML para poder ver su funcionalidad.

3.3.4 Propuesta de modificación.

Como se ha explicado anteriormente existen diagramas que carecen de algunos estereotipos o funcionalidades de suma importancia para lograr un buen modelado. A partir de las entrevistas realizadas a los analistas, desarrolladores y líderes de los proyectos pertenecientes al CEDIN, se pudieron identificar las funcionalidades de la herramienta en general a las que se les debe prestar mayor atención a la hora de modificarla para adecuarla al centro. A continuación se muestran las modificaciones inmediatas a realizar para satisfacer las necesidades de cada uno de los proyectos pertenecientes al CEDIN.

- Diagrama Casos de Uso del Negocio.
  - Caso uso de negocio.
  - Actor del negocio.
- Diagrama Clases.
  - Funcionalidades de la ventana de propiedades.
- Generación de código en C++.
- Diagrama de clases de diseño con estereotipos web.
  - Client Page.
  - Form.
  - Server Page.
- Opción cargar estereotipos mediante imágenes.

3.3.4.1 Ejemplo de modificación

Como fue explicado en el epígrafe 2.5.1 Realización del Diagrama de Casos de Uso del Negocio, la herramienta no cuenta con el estereotipo indispensable que represente al caso de uso del negocio, por lo que no cumple con los estándares necesarios, ocasionando inconformidades en la completitud del modelado.

Momentáneamente se propuso que el diagrama de casos de uso del negocio se representara como se muestra en la siguiente figura, que a diferencia de los casos de uso del sistema se muestra de color azul.

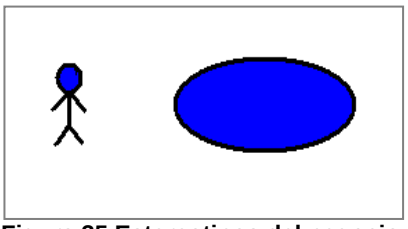


Figura 35.Estereotipos del negocio.

En la siguiente figura se muestran los estereotipos ideales para la realización del diagrama de casos de uso del negocio.

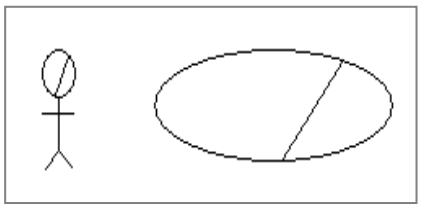


Figura 36.Estereotipos ideales del negocio.

Como propuesta para la implementación de estas modificaciones primeramente se deben identificar los módulos que tengan responsabilidades asociadas al cambio que se quiere efectuar y luego de localizados realizar los cambios en cada uno de ellos.

Para darle solución a la limitación que tienen los diagramas de caso de uso de negocio en cuanto a la carencia de los estereotipos; actor de negocio y caso de uso de negocio, se identificó primeramente el módulo del plugin UML que se encarga de representar estos elementos de modelado. En este caso el módulo en cuestión es el **org.netbeans.modules.uml.diagrams.** De todo el flujo que tiene la creación o modificación de una funcionalidad, en este caso solamente el cambio estará limitado a la representación visual de los estereotipos de modelado necesitados.

Un estereotipo de modelado es representado a través de una relación jerárquica entre la clase que representa al estereotipo y la clase **org.netbeans.api.visual.widget.Widget** propia del API de desarrollo del NetBeans. Es decir, un estereotipo hereda las propiedades y funcionalidades de un

componente widget que no es más que una pequeña pieza reutilizable de una escena que define la apariencia del elemento representado. Los widgets que representan al caso de uso del sistema y al actor del sistema son el **UseCaseWidget** y **ActorSymbolWidget** respectivamente, ambas clases se encuentran en el paquete de código **org.netbeans.modules.uml.diagrams.nodes**.

Para crear el nuevo estereotipo que represente a un caso de uso de negocio se propone crear una clase que puede ser nombrada **BusinessUseCaseWidget** a la cual se le deben implementar los métodos necesarios para que se muestre gráficamente el estereotipo. Por la similitud de la figura geométrica que existe entre un caso de uso de negocio y de sistema, se recomienda utilizar los mismos métodos de la clase **UseCaseWidget** para su representación gráfica, modificando únicamente el método **paintWidget** que se encarga de pintar el elemento visual. En el siguiente fragmento de código se muestra la implementación del método mencionado anteriormente, con las modificaciones necesarias para que su apariencia luzca tal como se mostró en la figura 36.

```
@Override
      protected void paintWidget()
      {
         Rectangle bounds = ovalWidget.getClientArea();
         Point ovalScreenLocation =
ovalWidget.convertLocalToScene(ovalWidget.getClientArea()).getLocation();
         Point myScreenLocation = convertLocalToScene(getClientArea()).getLocation();
         Ellipse2D.Float ellipse = new Ellipse2D.Float(ovalScreenLocation.x - myScreenLocation.x,
                                          ovalScreenLocation.y - myScreenLocation.y,
                                          bounds.width,
                                          bounds.height);
         Graphics2D graphics = getGraphics();
         Shape curClip = graphics.getClip();
         graphics.setClip(ellipse);
         //modificacion realizada
          graphics.draw(new Line2D.Double(ellipse.getCenterX(), ellipse.getCenterY() +
          ellipse.getHeight() / 2,
                                  ellipse.getCenterX() + ellipse.getWidth() / 2, ellipse.getCenterY()));
         //fin de modificaion realizada
         super.paintWidget();
         graphics.setClip(curClip);
      }
   }
```

La modificación al método **paintWidget** consistió en añadir a la elipse que representa al estereotipo un segmento de recta que une los puntos A y B tal como se muestra en la figura 37. El trazado de la recta depende directamente de la posición y las dimensiones con que fue creada la elipse.

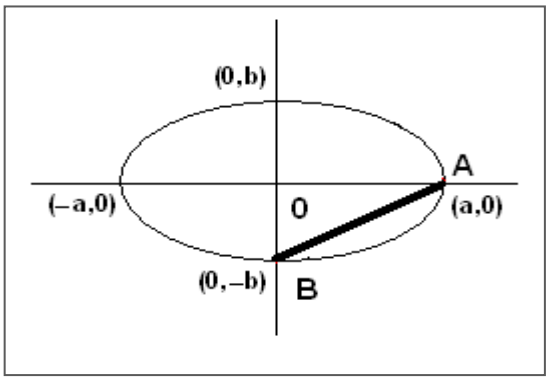


Figura 37.Estrategia para modificar el caso de uso.

Igualmente para crear el estereotipo que representa al actor de negocio se recomienda crear una clase que puede ser nombrada **BusinesActorWidget** que implemente las mismas funcionalidades de la clase **ActorSymbolWidget** modificando solamente el método **paintWidget**, dicho cambio se muestra en el siguiente fragmento de código.

```
@Override
   protected void paintWidget() {
      Rectangle rec=getClientArea();
      Graphics2D gr = getGraphics ();
      int x_c=rec.x+rec.width/2;
      int r=Math.min(rec.width/2, rec.height/8);
      Paint paint =null;
      if(paint==null)paint=getBackground();
      gr.setPaint(paint);
      gr.fillOval(x_c-r, rec.y, 2*r, 2*r);
      Color color = null;
      if(color==null)color=getForeground();
      gr.setColor(color);
      gr.drawOval(x_c-r, rec.y, 2*r, 2*r);
      //modificacion
      int p1x=x_c-r;
      gr.draw(new Line2D.Double(p1x+r,rec.y+2*r,p1x+2*r,rec.y+r));
      // fin de modificacion
      int h_free=rec.height-2*r;
      int hand_width=Math.min(rec.width/2, rec.height/4);
      gr.drawLine(x_c-hand_width, rec.y+2*r+h_free/8, x_c+hand_width, rec.y+2*r+h_free/8);
      gr.drawLine(x_c, rec.y+2*r, x_c, rec.y+2*r+h_free/2);
      gr.drawLine(x_c, rec.y+2*r+h_free/2, x_c-hand_width, rec.y+2*r+h_free);
      gr.drawLine(x_c, rec.y+2*r+h_free/2, x_c+hand_width, rec.y+2*r+h_free);
   }
```

Para la modificación del método **paintWidget** en este caso se procedió de la misma manera como se hizo con el ejemplo anterior, ya que lo que se modificó fue la elipse que identifica la cabeza del actor, esto se muestra en la siguiente figura.

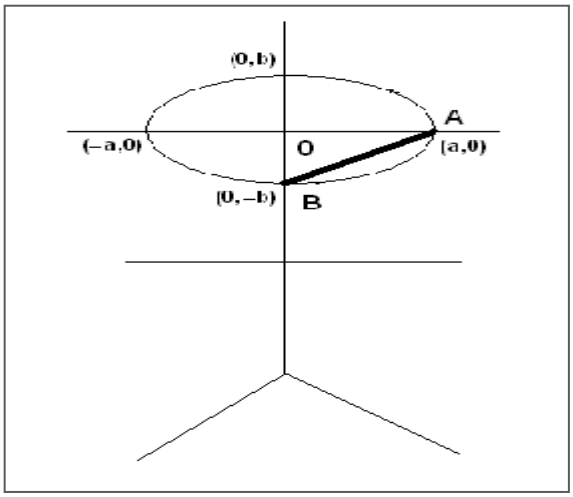


Figura 38. Estrategia para modificar el actor.

Para darle la funcionalidad adecuada a estos nuevos elementos, se deben modificar los demás módulos asociados, es decir, los que se identifiquen con el problema en cuestión, analizando cada una de sus clases y realizando los cambios pertinentes en las que lo necesiten.

3.4 Evaluación de la viabilidad de la propuesta, sobre la base del criterio de expertos. Método Delphi.

El Método Delphi es una técnica que permite llegar a opiniones de consenso en un grupo, sobre cierto asunto específico. Consiste en una serie de preguntas repetidas, por lo general utilizando encuestas o cuestionarios, sobre el tema que se investiga a personas que se considera que conocen el tema.

Esta técnica proporciona recoger el conocimiento del grupo de expertos sobre el tema que se ha escogido. Permite la información de consenso en un grupo y es útil como herramienta exploratoria para el pronóstico tecnológico.

Consiste en la aplicación de un cuestionario a cada uno de los posibles expertos donde se mide su coeficiente de competencia obteniendo los que presentan aptitudes para dar respuesta a cada una de las interrogantes. Las conclusiones del análisis de las respuestas se traducen en un segundo cuestionario, que de nuevo se remite al grupo de expertos. [32]

Para aplicar el método y garantizar la calidad de los resultados se siguieron tres etapas fundamentales:

- Elección de expertos.
- Conformación del cuestionario, para validación de la propuesta.

- Desarrollo práctico y explotación de resultados.

3.4.1 Elección de expertos.

El experto se define como un individuo o grupo de personas u organizaciones capaces de ofrecer valoraciones conclusivas de un problema en cuestión y hacer recomendaciones respecto a sus momentos fundamentales con un máximo de competencia. [33]

La selección del posible equipo de experto se realizó bajo los siguientes criterios:

- Graduado de nivel superior.
- Conocimientos sobre las herramientas CASE para modelado UML de distribución libre.
- Conocimientos y habilidades para el uso de estas herramientas.

Para la validación se seleccionaron 7 especialistas en la UCI, a los mismos se les aplicó una encuesta para determinar su coeficiente de competencia en la temática ¨herramientas CASE para modelado UML¨. (Ver Anexo 14)

3.4.2 Conformación del cuestionario, para validación de la propuesta.

Una vez obtenidos los expertos se procede a aplicarles un cuestionario (Ver Anexo 15) con el propósito de que valoren la propuesta y que brinden sus recomendaciones para sus posibles mejoras teniendo en cuenta los siguientes indicadores de evaluación.

- IE-1: Calidad de la propuesta.
- IE-2: Novedad científica de la propuesta.
- IE-3: Necesidad de aplicación de la propuesta.
- IE-4: Aporte social.
- IE-5: Posibilidad de aplicación de la propuesta.
- IE-6: Influencia de la propuesta en el mejoramiento de las soluciones del Software en el CEDIN.
- IE-7: Satisfacción de las necesidades de modelado del CEDIN.
- IE-8: Preparación del CEDIN para la adopción de la herramienta propuesta.

3.4.3 Desarrollo práctico y explotación de resultados.

Luego de realizadas las encuestas se procede a determinar el coeficiente de competencia de cada experto.

La competencia de los expertos se determina por el coeficiente **K**, el cual se calcula de acuerdo con la opinión del candidato sobre su nivel de conocimiento acerca del problema que se está resolviendo, y respecto a las fuentes que le permiten argumentar sus criterios. El coeficiente **K** se calcula para cada experto mediante la siguiente expresión:

K = 1/2 (**Kc** + **Ka**)

Donde: **Kc** = Coeficiente de conocimiento que tiene el experto sobre el tema en cuestión.

Ka = Coeficiente de argumentación o fundamentación de los criterios del experto.

El coeficiente de conocimiento (**Kc**) está en dependencia de la autovaloración del experto en relación a su nivel de conocimientos, que marca en una tabla. El número resultante se multiplica por 0,1.Se muestra el resultado de los cálculos en la tabla.

Expertos	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Kc	0.9	0.9	0.8	0.7	0.8	0.7	0.6

Tabla 11: Coeficiente de conocimiento que tiene cada experto.

El coeficiente de argumentación (**Ka**) se obtiene como resultado de la suma de los puntos alcanzados, a partir de la siguiente tabla patrón:

Fuentes de Argumentación	**Grado de influencia de cada fuente**		
	Alto	**Medio**	**Bajo**
Análisis teóricos realizados sobre el tema.	0,3	0.2	0.1
Experiencia.	0.5	0.4	0.3
Trabajos de autores nacionales.	0.05	0.05	0.05
Trabajos de autores extranjeros.	0.05	0.05	0.05
Conocimiento en el trabajo con la propuesta.	0.05	0.05	0.05

Tabla 12: Tabla patrón para el cálculo del coeficiente de argumentación

Esta tabla patrón se le presenta en la encuesta al experto sin cifras para que marque con una (x) el grado de influencia de las fuentes, de acuerdo con los niveles ALTO, MEDIO y BAJO. Se muestran los resultados en la siguiente tabla:

Expertos	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Ka	0.9	0.8	0.7	0.9	0.8	0.6	0.6

Tabla 13: Coeficiente de argumentación que tiene cada experto.

Finalmente, el coeficiente de competencia para cada experto es:

Expertos	1	2	3	4	5	6	7
K	0.90	0.85	0.75	0.80	0.80	0.65	0.60
Grado	alto	alto	medio	alto	alto	medio	medio

Tabla 14: Grado de influencia del coeficiente de competencia

Una vez calculado K, si:

0.8 ≤ **K** ≤ 1 K es alto, el Experto (i) tiene competencia alta.

0.5 ≤ **K** < 0.8 K medio, el Experto (i) tiene competencia media.

0 ≤ **K** < 0.5 K es bajo, el Experto (i) tiene competencia baja.

Como se puede percibir, la mayoría de los especialistas tienen un coeficiente de competencia alto, a excepción de tres especialistas que tienen su coeficiente K medio, por lo que los 7 candidatos pueden formar parte del panel de expertos que serán encargados de evaluar la propuesta. [34]

3.4.3.1 Concordancia de los expertos.

En la prueba estadística el Coeficiente de Concordancia de Kendall (W), ofrece el valor que posibilita decidir el nivel de concordancia entre los expertos.

- Si W = 0 (Significa que no existe concordancia en la evaluación emitida)
- Si W = 1 (Significa unidad de concordancia en la evaluación emitida)

La tendencia a 1 es lo deseado pudiéndose realizar nuevas rondas si en la primera no es alcanzada significación en la concordancia. Para alcanzar este propósito, se procede a calcular el coeficiente de concordancia de Kendall, mediante la siguiente expresión:

$$W = \frac{S}{\frac{1}{12}k^2\left(N^3 - N\right) - k\sum_T T} \qquad S = \sum_{j=1}^{N}\left(R_j - \frac{\sum_{j=1}^{N} R_j}{N}\right)^2 \qquad T = \frac{\sum\left(t^3 - t\right)}{12}$$

Donde:

W: Coeficiente de concordancia.

K: Cantidad de expertos.

N: Cantidad de variables.

T: Representa el resultado de los rangos iguales.

Rj: Suma de los rangos asignados a cada variable.

S: Suma de los cuadrados de las desviaciones.

t: número de observaciones dentro de cada uno de los grupos para el experto “i”.

Expertos/ Indicadores	IE-1	IE-2	IE-3	IE-4	IE-5	IE-6	IE-7	IE-8
E-1	4	4	5	4	4	4	4	4
E-2	4	4	5	3	4	5	5	3
E-3	3	4	4	4	3	4	4	4
E-4	4	3	4	3	4	4	4	4
E-5	4	4	5	5	5	5	4	4
E-6	4	3	5	4	4	4	4	3
E-7	4	4	5	4	3	4	4	3

Tabla 15: Valores de las preguntas concedidas por cada experto.

Expertos/ Indicadores	IE-1	IE-2	IE-3	IE-4	IE-5	IE-6	IE-7	IE-8
E-1	3,5	3,5	8	3,5	3,5	3,5	3,5	3,5
E-2	4	4	7	1,33	4	7	7	1,33
E-3	1,33	5,5	5,5	5,5	1,33	5,5	5,5	5,5
E-4	5,5	1,33	5,5	1,33	5,5	5,5	5,5	5,5
E-5	2,5	2,5	6,5	6,5	6,5	6,5	2,5	2,5
E-6	5	1,33	8	5	5	5	5	1,33
E-7	5	5	8	5	1,33	5	5	1,33

Tabla 16: Rangos de puntaje ligados.

T	t1	t2	t3	t4	t5	t6	t7
803,5	28	48,5	210,5	210,5	65	120,5	120,5

Tabla 17: Cálculo de las observaciones dentro de cada uno de los grupos por experto.

El coeficiente de Kendall calculado fue de 0.36. Luego se aplica la prueba de significación de hipótesis, planteándose la hipótesis nula y la alternativa de la siguiente forma:

H0: no existe comunidad de preferencia entre los expertos, w= 0.

H1: existe comunidad de preferencia entre los expertos, w≠0.

Para el estadígrafo de prueba se deben ver dos casos ya que en este caso depende del tamaño de N. Luego para:

- Muestras pequeñas: (N ≤ 7) se usa como estadígrafo el numerador del coeficiente de Kendall (S) para su valor crítico, donde W sea de significación de 0.05 y 0.01 en sus niveles.

- Muestras grandes: (N > 7) se usa como estadígrafo: Chi-cuadrado.

Calculado como:

$X^2 = K (N-1) W$

$X^2 = 17.64$

Por otra parte, se busca el Chi-cuadrado tabulado en la tabla del percentil de la distribución Chi cuadrado (Anexo 16) con un nivel de significación $\partial = 0.05$ que presenta un 95% de confianza y N-1 grados de libertad $X^2 (\partial, N-1)$.

$X^2 (\partial, N-1) = 2.17$

Se compara X^2 calculado y X^2 tabular, si se obtiene que X^2 calculado $>$ X^2 tabular entonces se rechaza H0 y se infiere que existe concordancia de criterios preferenciales entre los expertos al considerar válida la hipótesis alternativa H1, por lo que se concluye que existe concordancia en la aceptación de la propuesta por parte de los especialistas.

Por último, se resume básicamente en qué aspectos están de acuerdo los expertos como resultado del procesamiento y análisis de los criterios ofrecidos por ellos.

Los especialistas coincidieron en que el NetBeans con plugins para UML como propuesta de modelado para el CEDIN presenta la calidad necesaria para solventar sus necesidades, además, opinan que la herramienta teniendo en cuenta la soberanía tecnológica llevada a cabo en el país, es un buen aporte social, creyendo efectiva la aplicación de la propuesta. [34]

3.5 Conclusiones del capítulo

En el presente capítulo se describieron las características de extensión que posee NetBeans, realizando un estudio de la arquitectura de plugins que presenta, y desarrollando una breve descripción de los principales componentes por los cuales está compuesto el módulo UML. También se determinaron las mejoras inmediatas a realizar en la herramienta a partir de las desventajas expuestas en el modelado del caso de estudio y de las entrevistas realizadas a los líderes de proyecto. Además, se identificaron las dos posibles vías para extender el módulo UML de NetBeans, la primera modificando el código fuente y la segunda implementando un nuevo módulo con nuevas funcionalidades, siempre respetando la estructura y dependencias que deben tener los módulos para pertenecer a la arquitectura UML. Por último, se valida la propuesta con los especialistas del centro utilizando el método delphi, concluyendo que existe concordancia en la aceptación de la propuesta.

CONCLUSIONES

Luego de culminada la presente investigación se concluye que:

- Se identificaron y caracterizaron una serie de herramientas CASE de código abierto, extensibles, multiplataforma o para GNU Linux, y libres obteniendo un resumen importante que puede ser usado en otras investigaciones.
- Se obtuvo a partir del modelo de decisión aplicado en la selección de la herramienta, el IDE NetBeans con extensión UML como la herramienta con mejores prestaciones para ser usada en el modelado de las soluciones de software en el CEDIN.
- Se evaluó la herramienta seleccionada en la práctica, evidenciándose la facilidad de diagramación que presenta, y se identificaron las desventajas para sus posibles mejoras.
- Se propuso un diseño de mejora a raíz de las desventajas encontradas.
- Se validó la propuesta con siete especialistas con conocimientos sobre el tema obteniéndose de forma general opiniones muy positivas.

RECOMENDACIONES

- Hacer uso de la herramienta en todos los proyectos del CEDIN.

- Crear un grupo multidisciplinario encargado de incorporar nuevas funcionalidades a la herramienta, así como modificar otras existentes con vistas a obtener un mejor rendimiento durante la utilización de la misma. Entre estas funcionalidades se tienen:

  - Diagrama de Casos de Uso del Negocio.
    - Caso uso de negocio.
    - Actor del negocio.
  - Diagrama de Clases.
    - Funcionalidades de la ventana de propiedades.
  - Diagrama de clases de diseño con estereotipos web.
    - Client Page.
    - Form.
    - Server Page.
  - Generación de código en C++.
  - Opción cargar estereotipos personalizados mediante imágenes.

REFERENCIAS BIBLIOGRÁFICAS

[1].Jacobson, Ivar; Booch, Grady; Rumbaugh, James. El Proceso Unificado de Desarrollo de Software. Editorial Félix Varela, 2004.

[2].Booch, Grady. UML in Action. Communications of the ACM. October 1999. Vol. 42, No. 10.

[3].Pérez Lovelle, Sonia. Automatización de la Arquitectura de Componentes Genéricos usando UML.CUJAE, CUBA.

[4]. Jimmy Armando Muñoz Martínez. Diferencias entre Versiones de UML. BOGOTÁ, 27 de FEBRERO de 2007. [cited 26/10/2009]; Available from: http://www.google.com.cu/url?sa=t&source=web&cd=1&ved=0CBYQFjAA&url=http%3A%2F%2Fwww.e-groupsa.com%2Fusers%2Fjamm%2Fmaestria%2FNivelatorio%2FFundamentos%2520de%2520Ingenieria%2520de%2520Software%2FDiferencias%2520UML%252015-20.docx&rct=j&q=cantidad+de+diagramas+con+UML1&ei=SPUHTLCaFMS88gaO3sTDAQ&usg=AFQjCNGbVkxwB2tIYWrBqjVE4WxraElCEg

[5].Herramientas Case. **2005** [cited 22/10/2009]; Available from: http://www.pol.una.py/archivos/IngeInfo/ingeSoftI/MaterialPrimeraC.pdf

[6].Introducción a Herramientas CASE y System Architect. 2002 [cited 22/10/2009]; Available from: http://users.dsic.upv.es/asignaturas/eui/mtp/doc-practicas/intro_case_SA.pdf

[7].Herramientas CASE (Computer Aided Software Engineering). 2004 [cited 2/11/2009]; Available from: http://html.rincondelvago.com/herramientas-case.html

[8]. Welcome to ArgoUML. 2008 [cited 12/12/2009]; Available from: http://argouml.tigris.org/

[9].BOUML. 2010. [cited 15/12/2009]; Available from: http://bouml.free.fr/

[10].Visual Paradigm For UML. 2010 [cited15/12/2009]; Available from: http://www.visual-paradigm.com/

[11].Eclipse.2010 [cited 11/1/2010]; Available from: http://www.eclipse.org/modeling/mdt/?project=uml2

[12].Umbrello UML Modeler. 2008 [cited 13/1/2010]; Available from: http://uml.sourceforge.net/

[13]. Rational Software Architect. 2010. [cited 14/1/2010]; Available from: http://www-01.ibm.com/software/awdtools/architect/swarchitect/

[14].Cerda, Felipe. NetBeans el único IDE que necesitas. [cited 15/1/2010]; Available from: http://www.techbloog.com/talks/NetBeans65es_cl.pdf

[15]. Welcome to StarUML.2006 [cited 15/1/2010]; Available from: http://staruml.sourceforge.net/en/

[16].DIA. 2009. [cited 17/1/2010]; Available from: http://live.gnome.org/Dia

[17].Rational Rose. 2010 [cited 17/1/2010]; Available from: http://www-01.ibm.com/software/awdtools/developer/rose/

[18].Introducción al Rational Rose. Agostoy 2008. [cited 17/1/2010]; Available from: http://sistemaronald.blogspot.com/2008/08/introduccion-rational-rose.html

[19].New Poseidon For UML. 2010 [cited 19/1/2010]; Available from: http://www.gentleware.com/

[20].Diseño de Modelos. Poseidon for UML. 2003-2004. [cited 19/1/2010]; Available from: http://www.info-ab.uclm.es/asignaturas/42579/pdf/Practica3.pdf

[21]. Departamento de Procesos y Sistemas – LISI, Universidad Simón Bolívar. Indicadores Organizacionales para comparación de Herramientas Case en Venezuela.2001

[22]. Dpto. de Informática Universidad Carlos III de Madrid. Evaluación comparativa de herramientas Case para UML. 2006

[23]. Colectivo de Autores. Modelo de decisión para soportar la selección de herramientas Case. 2001

[24]. NetBeans IDE para Programación en Java. [cited 19/3/2010]; Available from: http://webcodigo.com/post/NetBeans-ide-para-programacion-en-java-applet-NetBeans.html

[25]. ¿Qué es NetBeans? [cited 2/4/2010]; Available from: http://netbeans.org/index_es.html

[26]. La Arquitectura de NetBeans V2 [cited 7/4/2010]; Available from: http://www.slideshare.net/ralphkui/la-arquitectura-de-NetBeans-v2

[27]. Conociendo a NetBeans Platform: Introducción [cited 7/4/2010]; Available from: http://wiki.netbeans.org/ConociendoNetbeansPlatformIntroduccion

[28]. NetBeans: your first plugin [cited 28/4/2010]; Available from: http://silveiraneto.net/2008/02/24/NetBeans-your-first-plugin

[29]. Conociendo a NetBeans Platform: Introducción [cited 2/5/2010]; Available from: http://amap.cantabria.es/confluence/display/DEV/NetBeans?labelsString=

[30]. Swing (javax.swing.*) [cited 12/5/2010]; Available from: http://www.esus.com/javaindex/j2se/jdk1.2/javaxswing.html

[31]. Java 2D. Dpto. de Lenguajes y Ciencias de la Computación E.T.S de Ingeniería Informática. Universidad de Málaga. [cited 27/5/2010]; Available from: http://books.google.com.cu/books?id=M6reV4TGyIQC&pg=PA1&lpg=PA1&dq=java.awt.Graphics2D+descripcion&source=bl&ots=jcuQlaavEN&sig=ltRk7i91Walr5Pki23d3seqawGc&hl=es&ei=EvL1S8H9NoP68AbpquD0Cg&sa=X&oi=book_result&ct=result&resnum=6&ved=0CCwQ6AEwBQ#v=onepage&q&f=false

[32]. El método Delphi. 2005-2006. [cited 2/6/2010]; Available from: http://www.monografias.com/trabajos-pdf/pronostico-delphi/pronostico-delphi.pdf

[33]. (Valdés, 1999) (Moráguez, 2001). [cited 2/6/2010]; Available from: http://www.gestiopolis.com/canales6/eco/metodo-delphi-estadistica-de-investigacion-cientifica.htm

[34]. Probabilidades y Estadísticas, DDC Matemática Aplicada, Universidad de las Ciencias Informáticas. [cited 5/6/2010]; Available from: http://eva.uci.cu/file.php/69/Tema_4/Act.%2022.pdf

BIBLIOGRAFÍA

1. **Jacobson, Ivar; Booch, Grady; Rumbaugh, James.** El Proceso Unificado de Desarrollo de Software. Editorial Félix Varela, 2004.

2. **Booch, Grady. UML in Action.** Communications of the ACM. October 1999. Vol. 42, No. 10.

3. **Pérez Lovelle, Sonia.** Automatización de la Arquitectura de Componentes Genéricos usando UML.CUJAE, CUBA.

4. **Welcome to ArgoUML. 2008** [cited 12/12/2009]; Available from: http://argouml.tigris.org/

5. **Colectivo de Autores.** Modelo de decisión para soportar la selección de herramientas Case. 2001

6. **Kemerer, C.** (1992); "Learning curve models for integrated CASE tool management"; USA; Mit Center for InformationSystemResearch; IEEE Software.

7. Validación del sistema de acciones propuesto a través del Criterio de Especialistas. Universidad de Málaga.

8. **BOUML. 2010**. [cited 15/12/2009]; Available from: http://bouml.free.fr/

9. **Visual Paradigm For UML. 2010** [cited15/12/2009]; Available from: http://www.visual-paradigm.com/

10. **Eclipse.2010** [cited 11/1/2010]; Available from: http://www.eclipse.org/modeling/mdt/?project=uml2

11. **Umbrello UML Modeler. 2008** [cited 13/1/2010]; Available from: http://uml.sourceforge.net/

12. **Rational Software Architect. 2010.** [cited 14/1/2010]; Available from: http://www-01.ibm.com/software/awdtools/architect/swarchitect/

13. **¿Qué es NetBeans?** [cited 2/4/2010]; Available from: http://ayuda-java.blogspot.com/2007/07/qu-es-NetBeans.html

14. **La Arquitectura de NetBeans V2** [cited 7/4/2010]; Available from: http://www.slideshare.net/ralphkui/la-arquitectura-de-NetBeans-v2

15. **NetBeans** [cited 16/4/2010]; Available from: http://amap.cantabria.es/confluence/display/DEV/NetBeans?labelsString=

16. **NetBeans: your first plugin** [cited 28/4/2010]; Available from: http://silveiraneto.net/2008/02/24/NetBeans-your-first-plugin

17. **El método Delphi. 2005-2006** [cited 2/6/2010]; Available from: http://www.monografias.com/trabajos-pdf/pronostico-delphi/pronostico-delphi.pdf

18. **(Valdés, 1999) (Moráguez, 2001).** [cited 2/6/2010]; Available from: http://www.gestiopolis.com/canales6/eco/metodo-delphi-estadistica-de-investigacion-cientifica.htm

19. **OBJECT MANAGEMENT GROUP**, **2005 OMG Model Driven Architecture.** [cited 5/6/2010]; Available from: http://www.omg.org/mda/

20. **OBJECT MANAGEMENT GROUP, 2005 UML 2.0, The current Oficial Version.** [cited 5/6/2010]; Available from: http://www.uml.org/#UML2.0

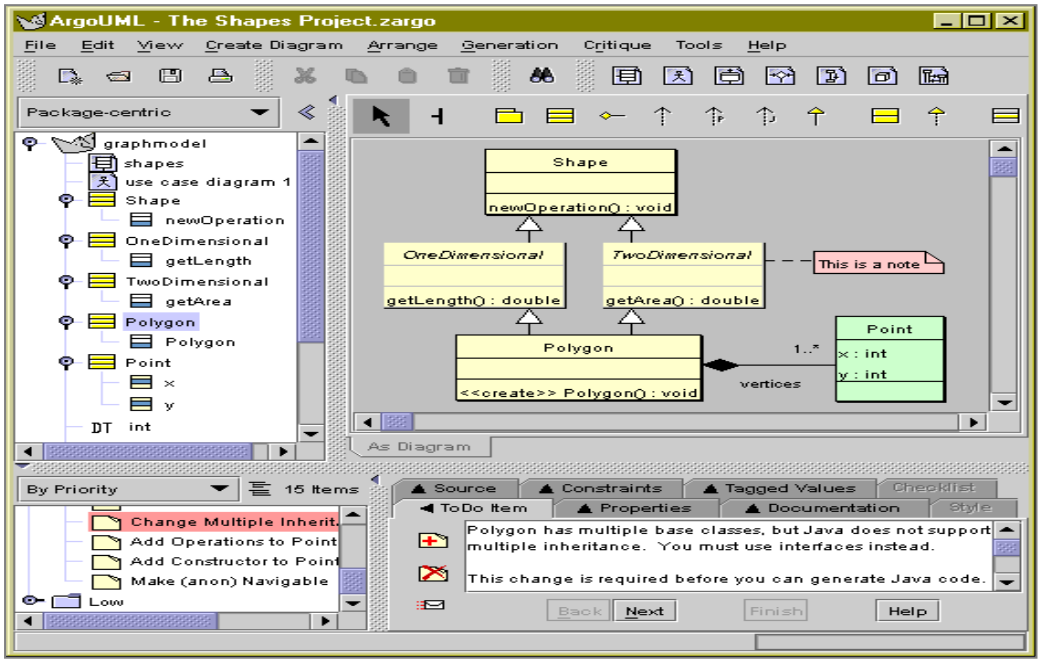
21. **OMG UML 2.0. Marcando un hito en el desarrollo del software, Avatar SRL** [cited 5/6/2010]; Available from: http://www.avatarsrl.com/website/articulos/AVATAR%20-%20Articulo%20OMG%20UML.pdf

22. **Santos, Armando.** *La toma de decisiones consensuales.* Ciudad de La Habana, 2003.

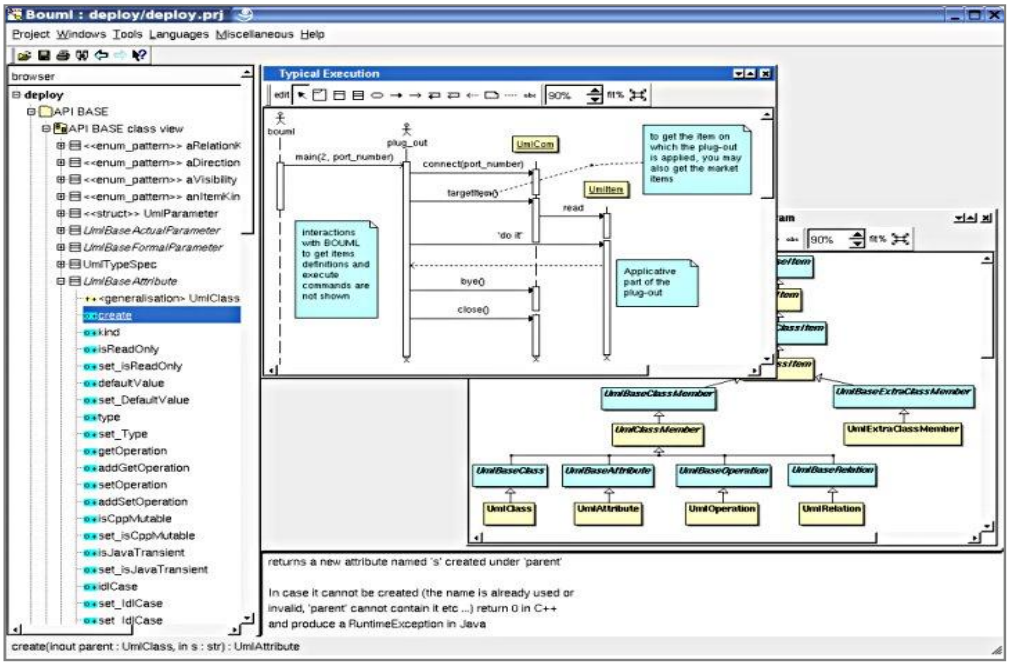
23. **El método DELPHI. (2 / 2005)** María de Lourdes Bravo Estévez, Universidad de Cienfuegos. Cuba. José Joaquín Arrieta Gallastegui, Universidad de Oviedo. España. [cited 5/6/2010]; Available from: http://www.rieoei.org/deloslectores/804Bravo.PDF

ANEXOS

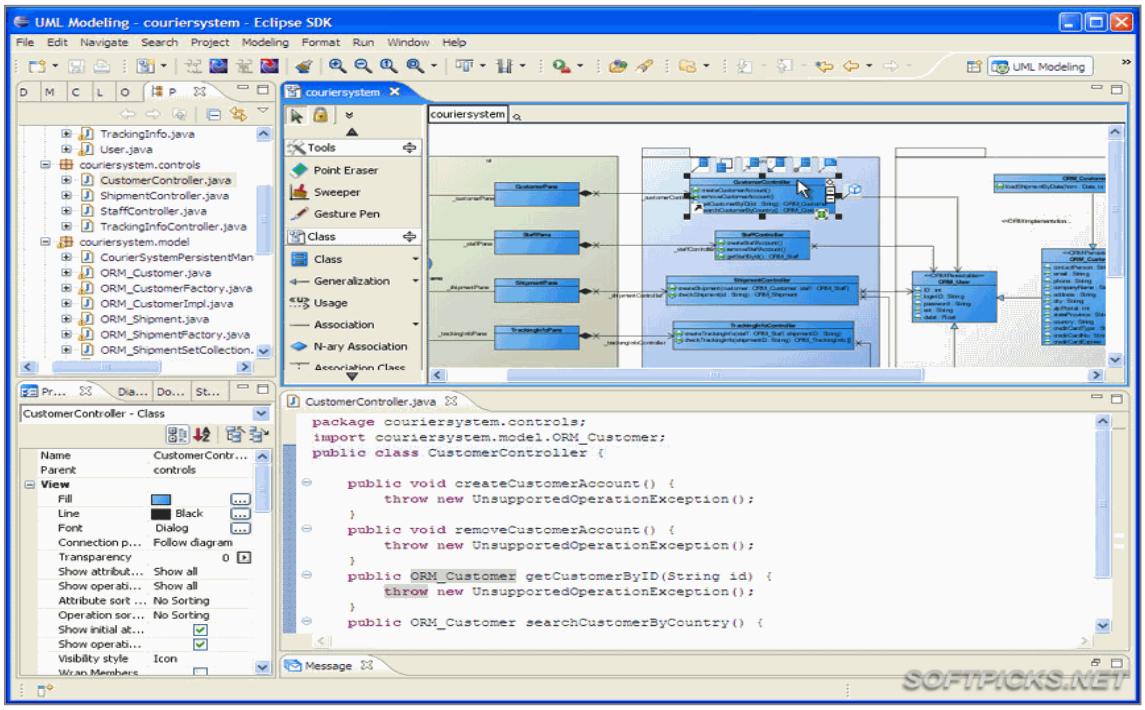
Anexo 1: Pantalla Principal de ArgoUml.



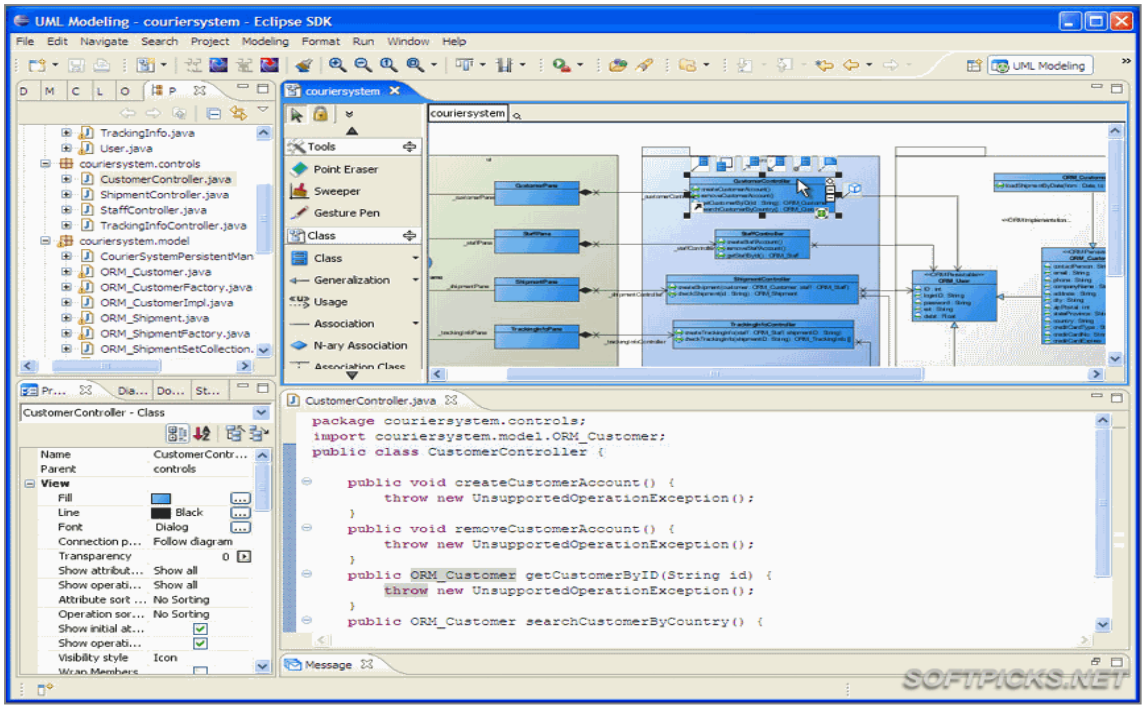
Anexo 2: Pantalla Principal de BOUML.



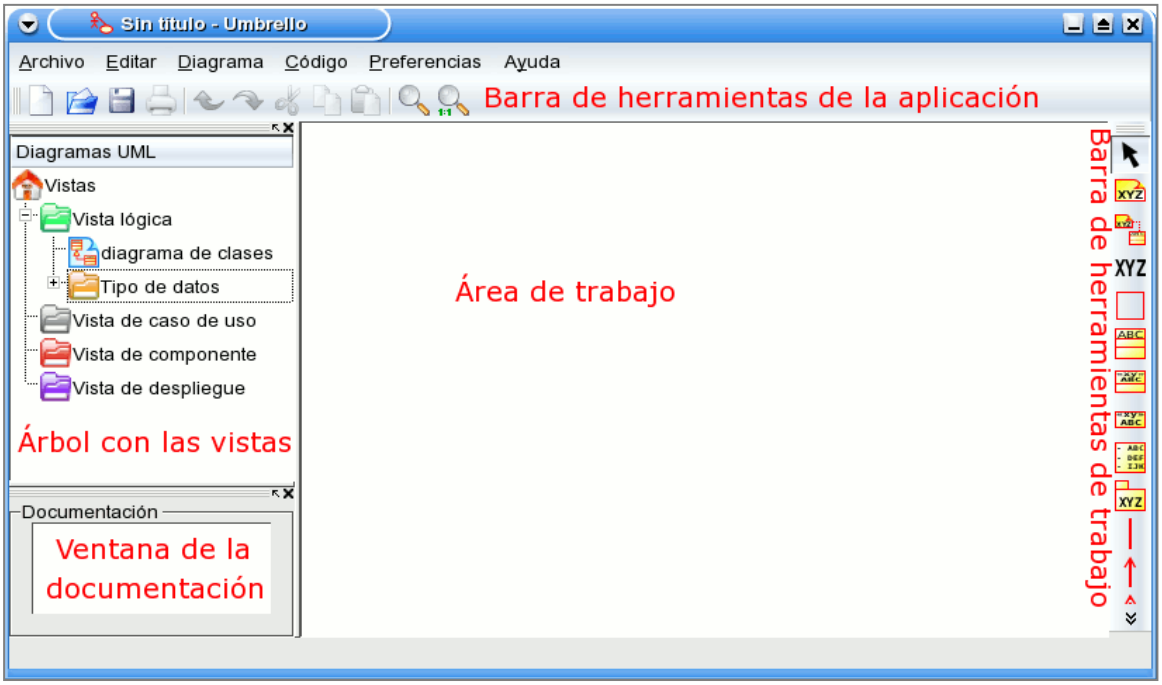
Anexo 3: Pantalla Principal de Visual Paradigm.



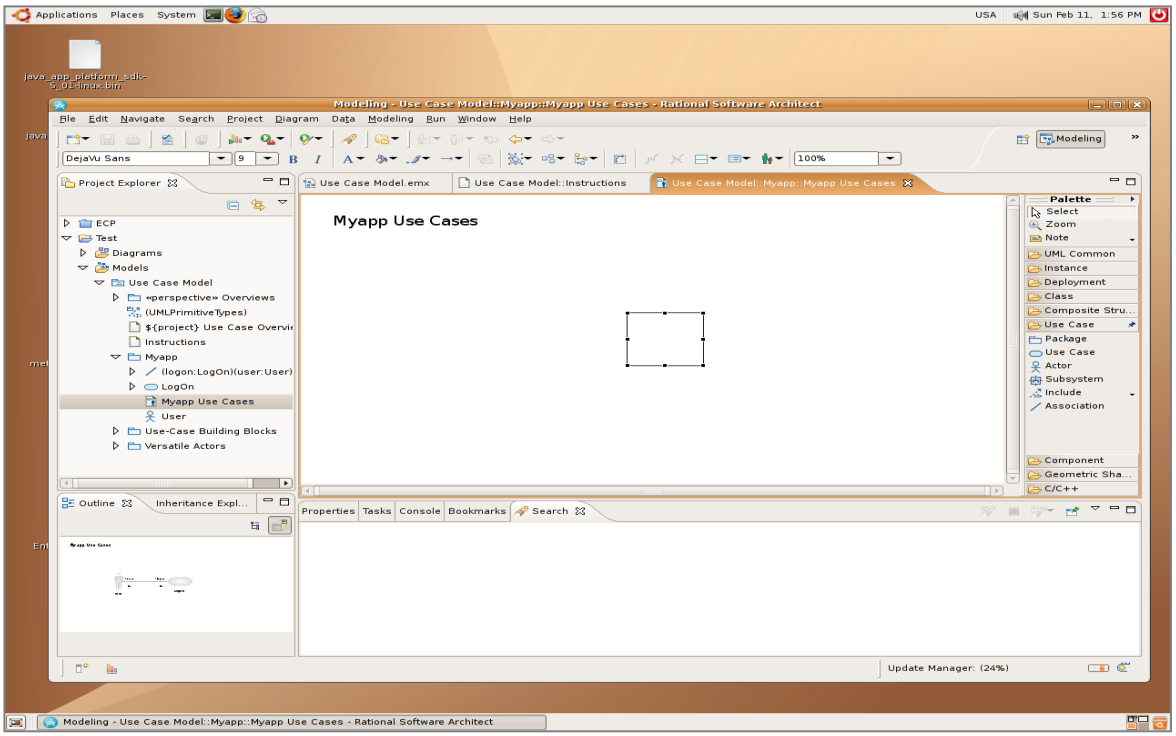
Anexo 4: Pantalla de Modelado UML del Eclipse.



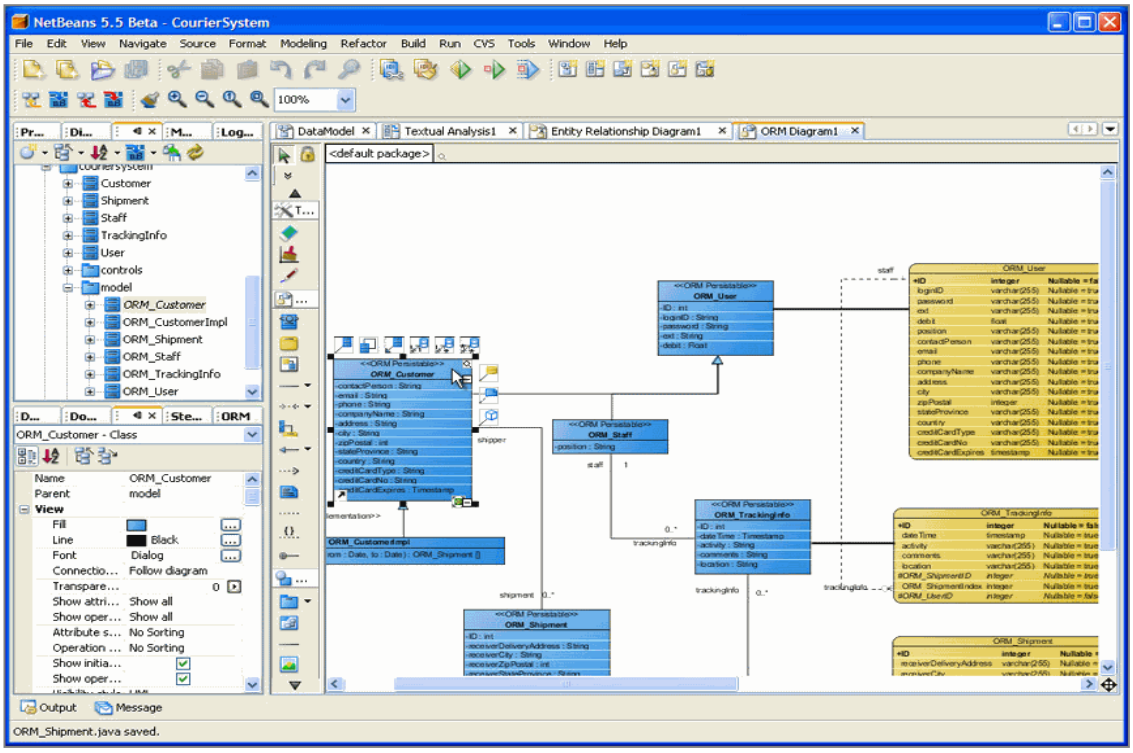
Anexo 5: Pantalla Principal de Umbrello.



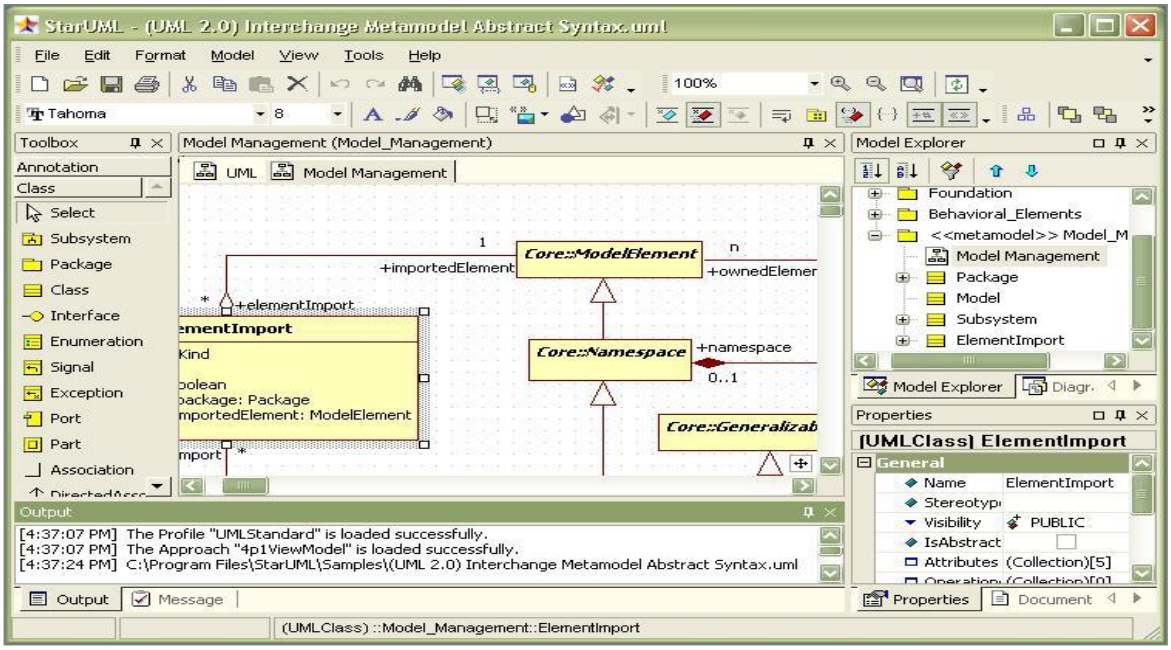
Anexo 6: Pantalla Principal de Rational Software Architect.



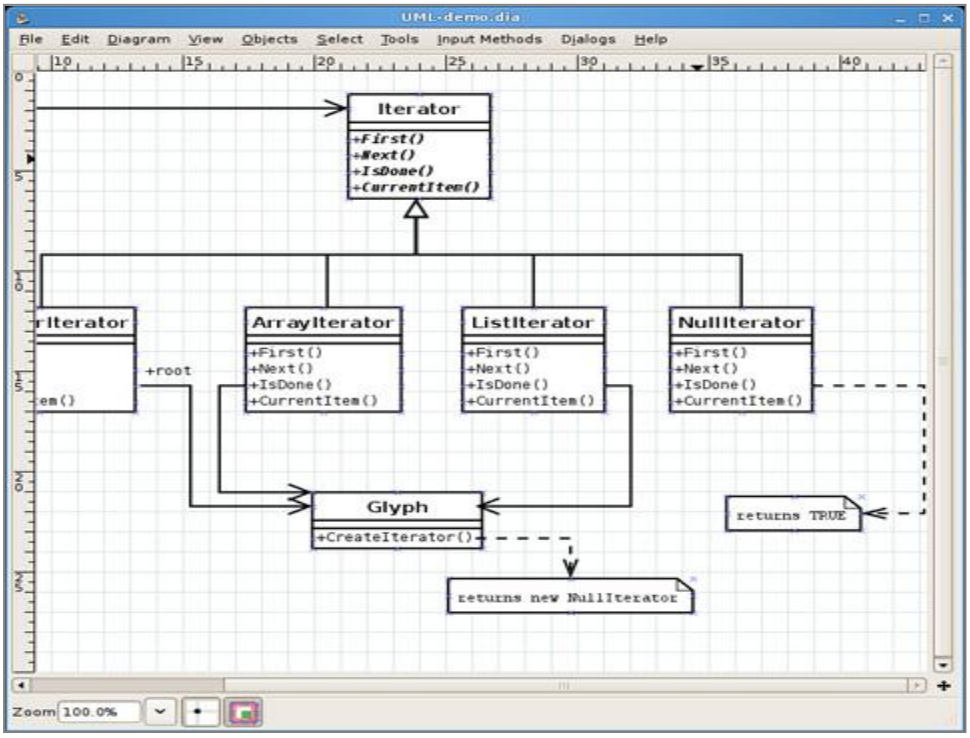
Anexo 7: Pantalla de Modelado UML del NetBeans.



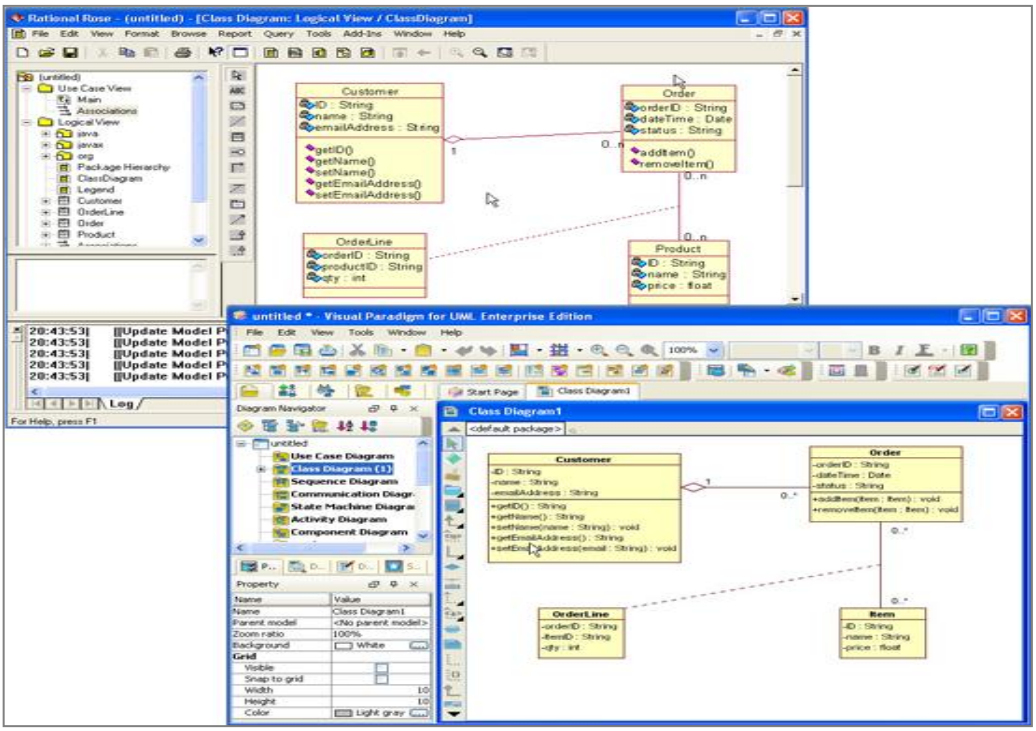
Anexo 8: Pantalla Principal de StarUml.



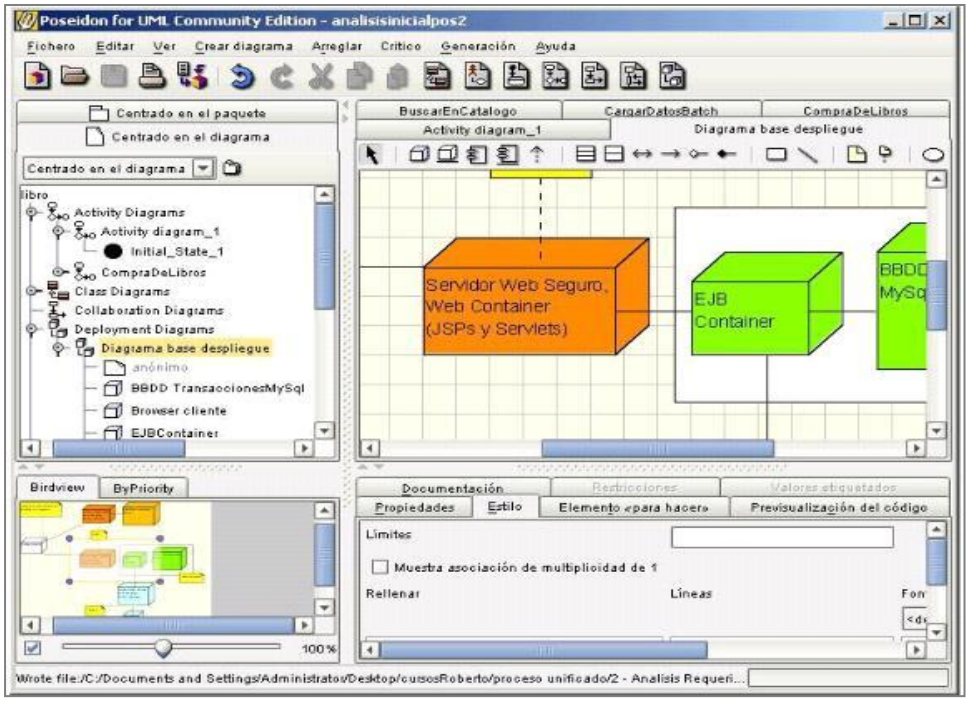
Anexo 9: Pantalla de modelado UML DIA.



Anexo10: Pantalla Principal de Rational Rose.



Anexo11: Pantalla Principal de Poseidon.



Anexo12: Entrevista realizada a los analistas y desarrolladores de CEDIN.

Entrevista

Estimado profesor nos dirigimos a usted debido a la necesidad planteada en nuestra investigación de conocer cuáles son las necesidades que debe solventar la herramienta CASE que se proponga en el CEDIN.

Es necesario que nos de los siguientes datos:

Nombre:

Categoría:

Rol que desempeña:

Muchas gracias anticipadas por su colaboración.

Las preguntas son las siguientes:

¿Cuántas herramientas utilizan actualmente?

¿Qué diagramas se necesita que realice la herramienta?

¿Necesita que genere documentación?

¿Necesita que genere ingeniería inversa?

¿Cuáles son las quejas más frecuentes de los usuarios respecto a los sistemas desarrollados?

Anexo13: Entrevista realizada a algunos líderes de los proyectos pertenecientes al CEDIN.

Entrevista

Estimado profesor nos dirigimos a usted debido a la necesidad planteada en nuestra investigación de conocer los diagramas UML que su proyecto necesita para su buen desarrollo.
Es necesario que nos de los siguientes datos:

Nombre del líder del proyecto:

Nombre del proyecto:

Muchas gracias anticipadas por su colaboración.

A continuación se presentan los diagramas pertenecientes a los estándares de UML existentes. Los pertenecientes al estándar UML2 son una evolución de los del UML1 diferenciándose el mismo en los diagramas que aparecen en negrita. ¿Diga de los estándares presentados cuáles son los diagramas que su proyecto necesita?

Estándar UML1

__Diagrama de casos de uso.
__Diagrama de colaboración.
__Diagrama de componentes.
__Diagrama de secuencia.
__Diagrama de despliegue.
__Diagrama de actividades.
__Diagrama de estados.
__Diagrama de objetos.
__Diagrama de clases.

Estándar UML2

1. **Diagramas de Modelado Estructurado**
   __ Diagrama de clases.
   __ Diagrama de estructura de paquetes.
   __**Diagrama de estructura compuesta.**
   __ Diagrama de objetos.
   __ Diagrama de despliegue.
   __ Diagrama de componentes.
2. **Diagramas de Modelado de Comportamiento**
   __ **Diagrama de comunicación.**
   __**Diagrama de tiempos.**
   __**Diagrama general de interacción.**
   __ Diagrama de casos de uso.
   __Diagrama de máquina de estados.
   __ Diagrama de actividad.
   __Diagrama de secuencia.

Anexo14: Encuesta para la selección de los expertos.

Estimado(a) experto(a):

Usted, por su experiencia y calificación profesional, ha sido seleccionado para pertenecer al panel de expertos que evaluará la propuesta del NetBeans con plugins para UML como herramienta de modelado para el CEDIN. Para llevar a cabo esta actividad se necesita conocer el nivel de conocimiento que posee sobre la temática planteada, por lo que es importante que responda el siguiente test de autoevaluación:

1. Evalúe su grado de conocimiento sobre las herramientas CASE para el modelado UML. Marque con una X sobre la siguiente escala (1: grado de conocimiento mínimo; 10: grado de conocimiento máximo).

0	1	2	3	4	5	6	7	8	9	10

2. Marque con una cruz (X) las fuentes que le han servido para argumentar el conocimiento que tiene Ud. de la temática que se investiga.

Fuentes de Argumentación	**Grado de influencia de cada fuente**		
	Alto	**Medio**	**Bajo**
Análisis teóricos realizados sobre el tema.			
Experiencia.			
Trabajos de autores nacionales.			
Trabajos de autores extranjeros.			
Conocimiento en el trabajo con la propuesta.			
Su propio conocimiento del tema.			

Anexo15: Encuesta para la evaluación de la propuesta.

Estimado(a) experto(a):

Usted, por su experiencia y calificación profesional, ha sido seleccionado para pertenecer al panel de expertos que evaluará la propuesta de una herramienta CASE para el CEDIN. Para evaluar esta propuesta, se necesita que usted de acuerdo a sus consideraciones le otorgue el valor a cada indicador de evaluación que se presenta en la tabla. Para ello marque con una X la categoría que corresponda a su respuesta.

Categoría: (1: Muy Baja; 2: Baja; 3: Media; 4: Alta; 5: Muy alta).

Indicadores de evaluación	**Calificación**				
	1	**2**	**3**	**4**	**5**
Calidad de la propuesta.					
Novedad científica de la propuesta.					
Necesidad de aplicación de la propuesta.					
Aporte social.					
Posibilidad de aplicación de la propuesta.					
Influencia de la propuesta en el mejoramiento de las soluciones del Software en el CEDIN.					
Satisfacción de las necesidades de modelado del CEDIN.					
Preparación del CEDIN para la adopción de la herramienta propuesta.					

Anexo 16: Tabla del percentil de la distribución Chi cuadrado.

Tabla 3: Valores Críticos de la Distribución Chi-Cuadrado.

		FUNCION DE DISTRIBUCION									
		0.005	0.010	0.025	0.050	0.100	0.900	0.950	0.975	0.990	0.995
	1	0.000039	0.000157	0.000982	0.003932	0.0158	2.71	3.84	5.02	6.63	7.88
	2	0.0100	0.0201	0.0506	0.10	0.21	4.61	5.99	7.38	9.21	10.60
	3	0.0717	0.11	0.22	0.35	0.58	6.25	7.81	9.35	11.34	12.84
	4	0.21	0.30	0.48	0.71	1.06	7.78	9.49	11.14	13.28	14.86
	5	0.41	0.55	0.83	1.15	1.61	9.24	11.07	12.83	15.09	16.75
	6	0.68	0.87	1.24	1.64	2.20	10.64	12.59	14.45	16.81	18.55
	7	0.99	1.24	1.69	2.17	2.83	12.02	14.07	16.01	18.48	20.28
G	8	1.34	1.65	2.18	2.73	3.49	13.36	15.51	17.53	20.09	21.95
R	9	1.73	2.09	2.70	3.33	4.17	14.68	16.92	19.02	21.67	23.59
A	10	2.16	2.56	3.25	3.94	4.87	15.99	18.31	20.48	23.21	25.19
D											
O	11	2.60	3.05	3.82	4.57	5.58	17.28	19.68	21.92	24.72	26.76
S	12	3.07	3.57	4.40	5.23	6.30	18.55	21.03	23.34	26.22	28.30
	13	3.57	4.11	5.01	5.89	7.04	19.81	22.36	24.74	27.69	29.82
D	14	4.07	4.66	5.63	6.57	7.79	21.06	23.68	26.12	29.14	31.32
E	15	4.60	5.23	6.26	7.26	8.55	22.31	25.00	27.49	30.58	32.80
	16	5.14	5.81	6.91	7.96	9.31	23.54	26.30	28.85	32.00	34.27
L	17	5.70	6.41	7.56	8.67	10.09	24.77	27.59	30.19	33.41	35.72
I	18	6.26	7.01	8.23	9.39	10.86	25.99	28.87	31.53	34.81	37.16
B	19	6.84	7.63	8.91	10.12	11.65	27.20	30.14	32.85	36.19	38.58
E	20	7.43	8.26	9.59	10.85	12.44	28.41	31.41	34.17	37.57	40.00
R											
T	21	8.03	8.90	10.28	11.59	13.24	29.62	32.67	35.48	38.93	41.40
A	22	8.64	9.54	10.98	12.34	14.04	30.81	33.92	36.78	40.29	42.80
D	23	9.26	10.20	11.69	13.09	14.85	32.01	35.17	38.08	41.64	44.18
	24	9.89	10.86	12.40	13.85	15.66	33.20	36.42	39.36	42.98	45.56
	25	10.52	11.52	13.12	14.61	16.47	34.38	37.65	40.65	44.31	46.93
	26	11.16	12.20	13.84	15.38	17.29	35.56	38.89	41.92	45.64	48.29
	27	11.81	12.88	14.57	16.15	18.11	36.74	40.11	43.19	46.96	49.64
	28	12.46	13.56	15.31	16.93	18.94	37.92	41.34	44.46	48.28	50.99
	29	13.12	14.26	16.05	17.71	19.77	39.09	42.56	45.72	49.59	52.34
	30	13.79	14.95	16.79	18.49	20.60	40.26	43.77	46.98	50.89	53.67

GLOSARIO DE TÉRMINOS

CEDIN: Centro de Desarrollo de Informática Industrial.

CASE: Ingeniería de Software Asistida por Ordenador.

TIC: Tecnologías de la Informática y las Comunicaciones.

UCI: Universidad de las Ciencias Informáticas.

UML: Lenguaje Unificado de Modelado.

OMG: Grupo de Gestión de Objeto.

UML: Lenguaje Unificado de Modelado.

IDE: Entorno de Desarrollo Integrado.

XML: Lenguaje de marcas extensible.

EJB: Enterprise JavaBeans. Proporcionan un modelo de componentes distribuido estándar del lado del servidor.

XMI: XML de Intercambio de Metadata. XMI es el nombre que recibe el estándar para el intercambio de metamodelos usando XML.

GUI: Interfaz Gráfica de Usuario.

AWT: Kit de Herramientas de Ventana Abstracta.

API: Interfaz de programación de aplicaciones.

Widget: Es una pequeña aplicación o programa, usualmente presentado en archivos o ficheros pequeños que son ejecutados por un motor de widgets o Widget Engine. Entre sus objetivos están los de dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.

Plugins: Es una aplicación informática que interactuando con otra aplicación le aporta una función específica a ésta última.

BSD: Distribución de Software Berkeley. Es la licencia de software otorgada principalmente para los sistemas BSD.

GPL: Licencia Pública General de GNU. Está orientada principalmente a proteger la libre distribución, modificación y uso de software.

JAR: (Java Archives): es un formato desarrollado por "Sun" que permite agrupar las clases diseñadas en el lenguaje Java, este formato es ampliamente utilizado en ambientes Java de todo tipo, esto se

debe a que otorga un nivel de compresión y reduce la carga administrativa al distribuir clases en el lenguaje.