UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERIA

TRABAJO DE GRADUACION PARA OPTAR AL GRADO DE
INGENIERO EN CIENCIAS DE LA COMPUTACION

DESARROLLO DE EDITOR DE LENGUAJE DE MODELAMIENTO UNIFICADO (UML) PARA DIAGRAMA DE CASOS DE USO

PRESENTADO POR:
GABRIELA EUGENIA HERRERA CARRILLO
FRANCISCO JAVIER ARÉVALO LÓPEZ

CIUDADELA DON BOSCO OCTUBRE, 2006.

UNIVERSIDAD DON BOSCO

FACULTAD DE INGENIERIA

“DESARROLLO DE UN EDITOR DE LENGUAJE DE MODELAMIENTO UNIFICADO (UML) PARA DIAGRAMAS DE CASOS DE USO”

ING. ROLANDO ALAS

ASESOR

JURADO EVALUADOR

ING. CRUZ ANTONIO GALDAMEZ

JURADO

ING. MARCO VINICIO LUNA

JURADO

ING. ANA MERCEDES CÁCERES

JURADO

AUTORIDADES DE LA UNIVERSIDAD DON BOSCO

ING. FEDERICO MIGUEL HUGUET RIVERA

RECTOR

PADRE VICTOR BERMUDEZ

VICERECTOR

LIC. MARIO RAFAEL OLMOS ARGUETA

SECRETARIO GENERAL

ING. ERNESTO GODOFREDO GIRÓN

DECANO FACULTAD DE INGENIERIA

LIC. MAURICIO COTO

DIRECTOR DE ESCUELA DE COMPUTACION

ÍNDICE GENERAL

Capitulo I: Marco Referencial

Capitulo II: Análisis de Resultado y Diagnostico

Capitulo III: Análisis del Sistema

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Introducción.

En el análisis orientado a objetos se desarrolla una serie de modelos que describen el software de una computadora, la manera como éste trabaja y cumple necesidades especificadas por los usuarios.

En el presente trabajo se realiza un estudio del Lenguaje de Modelamiento Unificado (UML), el cual es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de una aplicación. UML entrega una forma de modelar elementos conceptuales como procesos de negocio y funciones de sistema, además de objetos concretos, por ejemplo, escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

Inicialmente se presenta una investigación de todos los aspectos relacionados con la filosofía UML, específicamente con el diagrama de casos de uso.

Desde que UML fue adoptado por el OMG[1] como el lenguaje estándar para el modelado, se ha definido un buen número de modelos de proceso para el desarrollo de aplicaciones orientadas a objetos (OO), que utilizan este lenguaje como medio de expresión de diferentes modelos que se crean durante el desarrollo. Estas propuestas suelen estar dirigidas por los casos de uso, de manera que éstos se emplean para definir los requisitos funcionales del sistema, y todas las etapas del proceso (planificación de las iteraciones, análisis, diseño y pruebas), se articulan en torno a los casos de uso identificados.

Actualmente, en muchas discusiones sobre casos de uso se coincide en señalar que con frecuencia son mal interpretados, y que no hay guías precisas para resolver los aspectos que tienen que ver con su organización. En este sentido, se han publicado diferentes propuestas en las que se discuten cuestiones tales como la granularidad

[1]Object Management Group (OMG, www.omg.org)

de los casos de uso, el nivel de detalle en que deben describirse, o la conveniencia de crear una jerarquía de casos de uso.

Un caso de uso[2] puede ser definido como *una secuencia de acciones, incluyendo variaciones, que el sistema puede ejecutar y que produce un resultado observable de valor para un actor que interactúa con el sistema*. Aunque el éxito de los casos de uso se suele justificar con el hecho de que constituyen una técnica simple e intuitiva, algunos autores señalan las dificultades que presentan la obtención y la especificación de casos de uso verdaderamente útiles, y la falta de consenso sobre cómo organizarlos y manejarlos.

Estas son las razones que llevan a pensar que es necesario establecer un conjunto de guías para la identificación, descripción y organización de los casos de uso.

El diagrama de casos de uso es la técnica más efectiva y a la vez la más simple para modelar los requisitos del sistema desde la perspectiva del usuario. Los casos de uso se utilizan para modelar cómo un sistema o negocio funciona actualmente, o cómo los usuarios desean que funcione. No es realmente una aproximación a la orientación a objetos; es realmente una forma de modelar procesos.

Es, sin embargo, una manera muy buena de dirigirse hacia el análisis de sistemas orientados a objetos. Los casos de uso son generalmente el punto de partida del análisis orientado a objetos con UML.

El diagrama de casos de uso consiste en actores y casos de uso. Los actores representan usuarios y otros sistemas que interaccionan con el sistema. Se dibujan como "muñecos". Los casos de uso representan el comportamiento del sistema, los escenarios que el sistema atraviesa en respuesta a un estímulo desde un actor. Se dibujan como elipses.

[2] Desarrollo de Software Orientado a Objeto usando UML, Patricio Letelier Torres, Departamento Sistemas Informáticos y Computación (DSIC) Universidad Politécnica de Valencia (UPV) - España

Cada caso de uso se documenta por una descripción del escenario. La descripción puede ser escrita en modo de texto o en un formato paso a paso. Cada caso de uso puede ser también definido por otras propiedades, como las condiciones pre y post del escenario, condiciones que existen antes que el escenario comience y condiciones que existen después que el escenario se completa.

En los diagramas de casos de uso y en todos los diagramas en general un elemento muy importante son los estereotipos, estos vienen a representar cada elemento de los diagramas, siendo otro elemento importante las relaciones que se establecen como las comunicaciones que se pueden dar entre los elementos de los diagramas.

Este documento se estructura de la siguiente manera: en el primer capítulo se menciona las causas que dieron comienzo a la investigación, se presenta un planteamiento del problema a combatir, la evolución que ha tenido UML y los paquetes de software existentes para desarrollar estos diagramas; las técnicas utilizadas para la creación de un editor de diagramas de casos de uso.

Se describe un diagnóstico del tema a tratar, la metodología e instrumentos para realizar el proyecto de construcción de un editor de los diagramas antes mencionados, las ventajas, desventajas, alcances y limitantes la metodología que se sigue en el desarrollo del proyecto.

Se presenta un análisis de la investigación realizada para verificar la importancia del desarrollo de la aplicación y cómo la elaboración de la misma beneficia a la población a la que va dirigida; se describen las técnicas y herramientas utilizadas en la investigación y los resultados de dichos análisis.

También en el documento se incluye la etapa de análisis que contiene una perspectiva del producto, las funciones que éste contiene, las características de los usuarios a los que va dirigido, así como las restricciones del sistema; los

requerimientos específicos que surgen a partir de la investigación, atributos del sistema y herramientas que se utilizan para llevar a cabo el desarrollo.

Contiene además la fase de diseño del editor UML; diseño de programas, diagramas, pantallas y reportes, un mapa del sistema, interfaces para diagramar casos de uso, las herramientas a utilizar, la descripción del prototipo.

Se plantea una breve descripción de la filosofía de Lenguaje de Modelamiento Unificado, detallando de manera general sus componentes, enfocando nuestra investigación en los diagramas de casos de uso.

Para finalizar con el proceso de implementación que incluye el código fuente utilizado para la construcción del editor y un manual de usuario para operara la aplicación.

CAPITULO I
MARCO REFERENCIAL

Introducción

En el capitulo presentado a continuación se detallan elementos importantes para el correcto desarrollo del editor de diagramas de casos de uso UML, entre estos se encuentran los antecedentes, se plantea el problema a solucionar con la aplicación, los objetivos planteados, los alcances y las limitaciones que se presentaron en el proceso del sistema además de plantear también la delimitación y proyección social que este presenta.

Para finalizar se plantea el marco teórico que fundamenta la aplicación, la metodología a utilizar y el presupuesto de desarrollo.

1.1 Antecedentes

La técnica de UML surge con el objetivo de unificar un método de diseño y análisis orientado a objetos; permitiendo a un desarrollador de software contar con un modelo de análisis usando una notación con reglas prácticas a través de símbolos, haciendo comprensible para otras personas los diagramas.

En UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo; no puede ser el mismo el proceso para crear una aplicación en tiempo real, que el proceso de desarrollo de una aplicación orientada a gestión.

Lo que se intenta es lograr que los lenguajes que se aplican siguiendo los métodos más utilizados sigan evolucionando en conjunto y no por separado. Y además, unificar las perspectivas entre diferentes tipos de sistemas (no sólo software, sino también en el ámbito de los negocios), al aclarar las fases de desarrollo, los requerimientos de análisis, el diseño, la implementación de la ingeniería.

Con la creación del UML se persigue obtener un lenguaje que sea capaz de abstraer cualquier tipo de sistema, sea informático o no, mediante los diagramas, es decir,

mediante representaciones gráficas que contienen toda la información relevante del sistema.

Un diagrama es una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las relaciones entre los objetos y los vértices se corresponden con los elementos del modelo.

Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibuja de forma que se resaltan los detalles necesarios para entender el sistema.

El modelo de casos de uso fue descrito en 1992 por Ivar Jacobson dentro de la corporación Rational Rose. Actualmente, forma parte de la colección de diagramas que presenta la notación UML. Su especificación define la arquitectura de aplicaciones informáticas y precisa la funcionalidad de los procesos de negocio.

Los casos de uso son el eje central donde convergen todos los demás diagramas UML. Representan el núcleo a partir del cual puede establecerse una relación real entre un modelo de referencia y el código implementado.

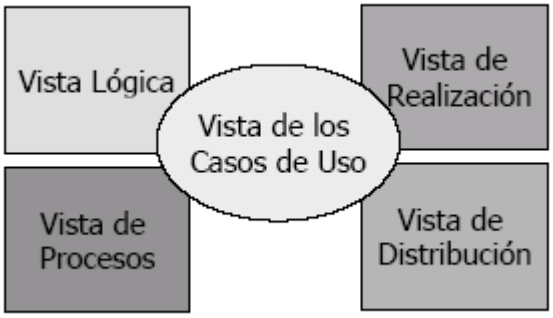


Figura 1.
Organización de modelos. Vista 4+1

Características de un proceso dirigido por casos de uso:

- Se capturan requisitos de usuario a través de casos de uso

- Son fundamentales para:
    - Identificar y especificar clases, subsistemas e interfaces
    - Identificar y especificar casos de prueba
- Planificar las iteraciones e integración del sistema
- Nos guían a través de los flujos de trabajo
- En cada iteración se identifican e implementan unos cuantos casos de uso
- Los casos de uso sirven para idear la arquitectura
- Se seleccionan los casos de uso más representativos
- Se utiliza como partida para escribir el manual de usuario
- Requisitos funcionales a través de casos de uso
- Requisitos no funcionales:
    - Se “adjuntan” a los casos de uso
    - Se guardan en una lista

El modelo de análisis a partir de casos de uso crece incrementalmente y se especifican a través de diagramas de clases y de colaboración, al principio se examinan unos pocos casos de uso y se crean sus realizaciones cada clasificador puede participar en varias realizaciones distintas con distintos roles como clases estereotipadas de análisis.

A nivel nacional no se tiene ningún antecedente de que exista una aplicación del tipo que se pretende llevar a cabo, debido que no se han desarrollado en el país editores de UML, a nivel internacional si existen editores, siendo su máximo exponente el desarrollado por la compañía Rational Rose.

1.2 Importancia de la Investigación

En el sistema que se lleva a cabo es importante realizar una investigación profunda y rica en contenidos, debido que UML es una filosofía muy diferente a la que se utilizaba al momento de hacer el análisis y diseño de un sistema; es una forma de pensar distinta a la que se ha venido manejando, además de implementar nuevos conceptos, dicho motivo lleva a prestar un interés especial en el área de la investigación para poder así realizar y edificar una base sólida sobre la cual descansara el sistema en desarrollo.

Este método es en la actualidad el más completo a la hora de llevar a cabo el desarrollo del sistema, donde va dirigido el futuro de la elaboración y desarrollo de software, por eso es importante desarrollar una aplicación que cumpla con los avances técnicos relacionados con el área del diseño de sistemas.

1.2.1 Planteamiento del Problema

En la actualidad para desarrollar software los profesionales del área de computación utilizan tecnología orientada a objetos, cuya herramienta es más completa a la hora de realizar el análisis y la programación, ahorrando así tiempo y dinero a las empresas encargadas de desarrollar software, con la implementación de esta técnica se necesitó actualizar los diseños del software, por lo que nació así el lenguaje UML, en este contexto se presenta la necesidad de un editor que pueda ayudar a los diseñadores a llevar a cabo dichos diagramas, dado que son los más idóneos a la hora de trabajar con programación orientada a objetos y no se pede acceder de manera fácil y económica a una herramienta que venga a facilitar la elaboración de los diagramas UML, específicamente los diagramas de casos de uso.

Es por esto que el Lenguaje de Modelamiento Unificado permite al desarrollador crear aplicaciones que faciliten la etapa de construcción a partir de la etapa de análisis de requerimientos y diseño.

El UML es una metodología relativamente nueva que en el entorno de desarrollo de software aun no es muy común y su uso es muy escaso sobre todo en nuestro país, donde los analistas pocas veces se toman el tiempo de crear un diagrama correctamente diseñado del software que se va ha crear sin tomar en cuenta que esto facilitaría las siguientes etapas en el desarrollo del sistema.

Un mal diseño de sistema provoca retrasos y malentendido en la etapa de construcción, además que usar diversas herramientas de diseño dificulta el proceso de entendimiento, sobre todo si el desarrollo se realiza por equipos de trabajo multidisciplinario ya que llegar a una estandarización del diseño es casi imposible.

Los problemas a abordarse son:

- La falta de herramientas para desarrollar dicho lenguaje. (El UML es poco utilizado en la elaboración de software).
- La poca calidad que se presenta en los diagramas elaborados por la falta de una herramienta adecuada al momento de realizarlos.
- Ausencia de estandarización a la hora de llevar a cabo las etapas de análisis de un sistema.

1.2.2 Justificación

La utilización de UML específicamente el diagrama de casos de uso ayuda a los gestores y practicantes de la ingeniería de software en todas las actividades asociadas a los procesos de diseño y desarrollo.

Por medio de un editor de diagramas de casos de uso se estandarizan los diseños de sistemas, y es posible establecer la serie de requerimientos y estructuras necesarias para plasmar un sistema de software previo al proceso intensivo de escribir código.

Entre más complejo es el sistema que se desea desarrollar más beneficios presenta el uso de UML, las razones son: que mediante un plano o visión global resulta más fácil detectar las dependencias y dificultades implícitas del sistema, los cambios en una etapa inicial (Análisis) resultan más fáciles de realizar que en una etapa final de un sistema, la fase intensiva de codificación.

Se optó realizar la investigación debido a que el lenguaje UML es la herramienta que actualmente se está utilizando, siendo la alternativa que viene a colaborar a la hora de trabajar en el diseño y desarrollo de software por ser una técnica muy útil y completa.

Se ha determinado enfocar la investigación en los diagramas de casos de uso ya que es el primer componente que conforma al lenguaje UML cuando se realiza el análisis/diseño de un sistema.[3]

Un caso de uso es una pieza de funcionalidad bien delimitada y reutilizable que da valor a un actor o a varios actores que interactúan con un sistema en discusión. Son un vehículo muy eficaz para capturar, organizar y visualizar requerimientos; representan el fundamento para una definición no ambigua de los requerimientos funcionales**.**

[3]Ver anexo1: Clasificación de los diagramas UML en el desarrollo de un sistema.

A continuación se presentan algunas de las características por las cuales se opto a realizar el editor de diagramas de casos de uso:

- Son muy útiles para delimitar claramente el comportamiento de un sistema ya que representan un lenguaje de comunicación eficaz entre clientes y desarrolladores, son el punto de partida para diseñar las interfaces gráficas de usuario y las interfaces de comunicación con sistemas externos y su revisión permite una comprensión detallada de la funcionalidad global de una aplicación.

- Facilitan una estimación más exacta de los recursos necesarios para implementar cada pieza funcional del proyecto.

- Son la base a partir de la cual es posible averiguar los objetos que configurarán el sistema y cómo interactuarán en los distintos escenarios posibles y escenarios probables. Permiten la definición del comportamiento de los objetos y componentes a través de sus interfaces y facilitan la descomposición del sistema en una arquitectura de n capas[4].

- Suministran una vista dinámica del sistema a la manera de una red semántica desde donde es posible invocar cualquier funcionalidad y comprobar sus vinculaciones.

- No ser simples usuarios de productos de software, sino ser creadores de esas herramientas.

[4]Ver anexo 2: Diagramas de casos de uso en diferentes etapas del proyecto.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un editor UML para diagramas de casos de uso que permita que los analistas y desarrolladores de proyectos de software puedan llevar a cabo de manera eficiente y funcional el proceso de diseño.

1.3.2 Objetivos específicos

- Desarrollar una herramienta con la cual se pueda trabajar con las características intrínsecas de cada estereotipo en los diagramas de casos de uso.
- Diseñar a través de un editor los diagramas de casos de uso, mostrando sus características y utilidad en la etapa de diseño.
- Construir un modelo de casos de uso que defina bien la funcionalidad y sus límites presentando como resultado un diagrama completo de la etapa de requerimientos.
- Elaborar diagramas de casos de uso con la especificación adecuada para descomponer el proyecto en piezas funcionales.
- Generar las vistas de diagramas de casos de uso en forma de diagrama y reporte.

1.4 Alcances

El sistema de información presenta un software, el cual permite al usuario trabajar de forma fácil, eficiente y ordenada al elaborar los diagramas de casos de uso UML, así a través de la aplicación se puede estandarizar cuando se esté diseñando un software determinado.

Con este sistema se logra optimizar la calidad de implementación de los diagramas de casos de uso UML.

El sistema tiene la capacidad de satisfacer las necesidades que puedan presentarse en el desarrollo de la elaboración de los diagramas de casos de uso ya que permite implementar las 3 fases de los diagramas los cuales son: inclusión, extensión y herencias; además la aplicación almacena en una estructura de datos los elementos del diagrama así como sus componentes y características.

El editor permite al usuario trabajar de forma amigable y fácil, además de trabajar con 5 estereotipos, (actor, caso de uso, paquete, asociación y dependencia) y proporciona la opción de cambiar las propiedades del elemento.

La aplicación presenta el manejo de archivos donde se almacenan los elementos utilizados en el diagrama y las modificaciones que se le puedan realizar en las propiedades, llevando una relación entre los elementos que componen dicho diagrama.

1.5 Limitaciones

Debido a la extensión del tema se ha decidido realizar la investigación enfocada para un tipo diagrama UML, dicho diagrama es el casos de uso.

Se desarrolla el editor de diagramas de casos de uso seleccionando cinco de los estereotipos necesarios para su desarrollo, debido a la complejidad que conlleva realizar un análisis completo de toda la filosofía UML.

1.6 Delimitación

En el sistema a desarrollar se ha determinado delimitar el tema global de la siguiente forma: “Desarrollo de Editor Lenguaje de Modelamiento Unificado (UML) para Diagramas de Casos de uso”.

De los diagramas UML que existen, se trabaja con uno, el diagrama de casos de uso debido a la extensión de los diagramas, para poder desarrollar el proyecto presentando una calidad aceptable.

Se pretende trabajar con 5 estereotipos[5]:

- Actor: metadato del elemento de tipo actor.
- Casos de uso: metadato del elemento de tipo caso de uso.
- Paquetes: metadato del elemento de tipo paquete, que engloba a otros elementos.
- Asociación: tipo de relación que asocia los elementos.
- Dependencia: tipo de relación en la que un elemento depende de otro.

Además el editor de diagramas de casos de uso UML trabaja con las relaciones de inclusión, extensión y herencias.

[5]Ver anexo3. Tipos de estereotipos .

El sistema satisface las necesidades de captura de requisitos cuando se elabore el software y es orientado en primer lugar a la población estudiantil como una herramienta didáctica tanto para enseñanza como para la elaboración de diversos proyectos, además está dirigida a los diseñadores de sistemas para que puedan elaborar de una forma clara y ordenada el análisis para la realización de un sistema; también está dirigida para todas aquellas personas interesadas en aprender el lenguaje UML específicamente los diagramas de casos de uso.

1.7 Proyección Social

El desarrollo de un editor de diagramas de casos de uso UML proporciona una herramienta de gran utilidad enfocada principalmente a beneficiar a la población estudiantil que desarrolla proyectos de software o de otras áreas que necesitan realizar un análisis de requerimientos de manera fácil y efectiva.

Se pretende crear una aplicación que esté al alcance de todas las personas interesadas en hacer uso del editor. Proporcionando una herramienta que sirva para el apoyo didáctico y que ayude a facilitar el entendimiento y la importancia del uso de diagramas de diseño en el desarrollo de cualquier tipo de proyecto.

La aplicación es un producto gratuito destinado a ser una herramienta de apoyo para el desarrollo de sistemas a pesar de que no pertenece a la categoría de aplicaciones de código abierto.

La importancia del desarrollo del editor de diagramas de casos de uso radica en su rol de tecnificar a la población estudiantil en el área de computación y crear una cultura de planificación en el desarrollo de aplicaciones, antes de profundizar en las etapas de programación.

1.8 Marco Teórico

1.8.1 Referencias Históricas

El UML fue creado por Grady Booch, James Rumbaugh e Ivar Jacobson, estos trabajaban en distintas empresas durante la década de los ochenta y principios de los noventa, en sus inicios, cada uno de ellos creó su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre las de sus competidores. A mediados de los noventa, iniciaron a intercambiar ideas entre sí y decidieron producir un producto común unificando la notación.[6]

En 1994 Rumbaugh ingresó a Rational Rose Software Corporation, donde ya trabajaba Booch. Luego Jacobson ingresó a Rational un año después. En ese momento se inició el proyecto de desarrollar un "Método Unificado", con la participación de Grady Booch y Jim Rumbaugh, el resultado fue presentado en el año de 1995. Una vez unido al dúo, Ivar Jacobson, estos se conformaron en socios de la compañía Rational Rose. Herramienta CASE Rational Rose.

Los anteproyectos del UML empezaron a circular en la industria del software y las reacciones resultantes trajeron consigo considerables modificaciones. Conforme diversos corporativos vieron que el UML era útil a sus propósitos, se conformó un consorcio del UML. Entre los miembros se encuentran DEC, Hewlett-Pakard, Intellicorp, Microsoft, Oracle, Texas Instrument y Rational. En 1997 el consorcio produjo la versión 1.0 del UML y lo puso a consideración del OMG (Grupo de Administración de Objetos) como respuesta a su propuesta para un lenguaje de modelos estándar.

[6]Ivar Jacobson, Grady Booch, James Rumbaugh, **"El Proceso Unificado de Desarrollo Software",** Addison Wesley, 1999.

1.8.2 Referencias Conceptual

UML (Lenguaje Unificado de Modelado), es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir, su objetivo es lograr que, además de describir con cierto grado de formalismo tales sistemas, puedan ser entendidos por los usuarios de aquello que se modela.

La investigación hace uso de las herramientas de UML para el modelado de datos, especialmente se basará en estereotipos y patrones, como una forma especial de representar atributos con imprecisión utilizando la teoría de conjuntos difusos (lógica difusa).

Estereotipo en UML: Es un nuevo elemento del lenguaje definido sobre la base de algún elemento PRE existente de UML. Extienden la semántica pero no la estructura de las clases del meta-modelo. Permite representar una variación de un elemento existente que posee otra intención, o distinción de uso. La definición de un estereotipo se hace en forma explícita en la vista estática, mediante una relación de generalización con el elemento de UML que es base para su definición. El nombre del estereotipo debe ser distinto de los elementos de UML, y se denota entre comillas francesas («nombre estereotipo»). También puede considerar una notación gráfica distintiva.

Patrones: Solución ya probada y eficaz para algún problema de diseño que puede expresarse como un conjunto de principios y heurísticas.

Notación: Conocimiento estándar para identificar los diferentes elementos de un diagrama. Esta puede tener variantes (símbolos) de una herramienta en otra, pero se mantienen el concepto y su acción.

Herramientas: El conocimiento de la herramienta de desarrollo del análisis y diseño.

Procesos: Es la identificación de los procesos que se involucran en un sistema, es decir conocer la problemática del sistema.

Diagrama de Casos de Uso: define la estructura fundamental de una aplicación. El diagrama de caso de uso permite diagramar en un nivel macro la interacción entre los actores externos (usuarios) y las actividades que estos realizan en la aplicación.

Actor es un rol que un usuario juega con respecto al sistema.

Asociación: Es el tipo de relación más básica que indica la invocación desde un actor o caso de uso a otra operación (caso de uso). Dicha relación se denota con una flecha simple.

Dependencia o Instanciación: Es una forma muy particular de relación entre clases, en la cual una clase depende de otra, es decir, se instancia (se crea). Dicha relación se denota con una flecha punteada.

1.8.3 Referencia Experiencial

A continuación se presenta una recopilación de algunos de los productos diseñados para la elaboración de diagramas UML.[7]

Compañía	Producto	Versión	Fecha	Plataforma	Localización	Costos
AndroMD	AndroMDA:	3	05/2005	Java VM	Inglaterra	gratis
Tigris	ArgoUML	0.14.1	12/2003	Java VM	EEUU	gratis
BOUML	BOUML	2.5.5	09/2005	Windows, Unix	EEUU	gratis
AlTOVA	UMODEL	2005	05/2005	WINDOWS	EEUU	$129
Sybase	PowerDesigner	9.5	10/2002	Windows	EEUU	$5990
Gentleware	Poseidon for UML - Community	3.2	10/2005	Java VM	EEUU	$875

Tabla No. 1: Editores de UML

En América Latina según la investigación realizada no se han desarrollado aplicaciones que permitan a los diseñadores trabajar en el diseño de UML.

La desventaja de estos productos mencionados en la tabla presentada es la complejidad al trabajar y sobretodo para las personas que no conocen el lenguaje UML, es importante mencionar que UML puede presentarse de una manera básica como lo es la elaboración únicamente grafica del diagrama o de una forma compleja como lo es el desarrollo y propiedades intrínsecas que de cada uno de los elementos de dicho diagrama presentan, además la mayoría de ellos tienen precios altos por ejemplo Rational Rose con un costo de $2495.

[7] http://www.objectsbydesign.com/tools/umltools_byPrice.html sitio dedicado a proporcionar información de la programación orientada a objetos 1999-2005 Objects by Design, Inc

1.9 Metodología del Proyecto

Disponer de técnicas eficaces de investigación de hechos es vital en el desarrollo de proyectos de sistemas; la investigación se lleva cabo durante todas las fases del ciclo de vida del desarrollo de sistemas.

Para dar apoyo a los nuevos sistemas, el analista debe recabar información sobre personas, datos, actividades, redes y tecnología.

Es por estas razones que para realizar el proyecto de un editor para diagramas de casos de uso UML se utilizan dos metodologías de investigación: las técnicas de tratamientos de datos y las técnicas de sistema.

Las técnicas de tratamiento de datos que se implementan son:

Encuestas: Se realiza encuestas en Universidades del área metropolitana para determinar el nivel de conocimiento de UML específicamente de los casos de estudio en los alumnos de educación superior.

Entrevistas: Se realiza entrevistas a profesionales en el área de computación con el objetivo de recolectar información del uso de UML y los diagramas de casos de uso para la realización de proyectos.

Las técnicas de sistema que se utilizaron son:

Diagramas de UML: Para realizar un análisis de diseño de sistema desde la etapa de requerimiento hasta su etapa de aplicación.

Visual Basic. Net: Para llevar a cabo el proyecto debido a que es la herramienta más idónea para desarrollar este tipo de aplicación, tomando en cuenta que es una tecnología de punta.

Base de Datos: Se utiliza los archivos binarios en el sistema para respaldar y almacenar cada uno de los elementos de los diagramas, además de utilizarlas también para determinar las propiedades de cada elemento del diagrama y así poder respaldar la representación grafica del diagrama con una base de datos de los elementos que la componen como de sus propiedades. Además se utiliza para detallar las relaciones existentes entre los elementos que pueden componer los diagramas de casos de uso elaborados.

La metodología utilizada para el desarrollo de aplicaciones está basada en el paradigma de "Desarrollo de software como equipo de solución de problemas", el cual consiste de una aproximación colaborativa de resolución de problemas que asegure un desarrollo eficiente y productos cada vez mejores. Esta metodología hace uso de un ciclo de desarrollo basado en iteraciones controladas e incrementales.

Como técnica para el modelado de las aplicaciones se presenta el UML (Unified Modeling Language), estándar actual para el modelado de aplicaciones orientadas a objetos basadas en componentes, categoría dentro de la cual se incluyen las aplicaciones Web.

El paradigma de Desarrollo de software como equipo de solución de problemas se caracteriza por los siguientes aspectos:

- Enfoque en calidad por diseño y no calidad por pruebas
- Diseño iterativo y desarrollo por fases
- Prevención de defectos en lugar de corrección de defectos
- Concurrencia del diseño
- Enfoque en procesos y métricas
- Establecimiento de Equipos con funciones entrecruzadas
- Responsabilidad compartida entre los miembros del equipo para asegurar la entrega de un producto exitoso.

FASES

La metodología se implementa por medio de cuatro fases:

1. Incepción: Alcanzar y acordar un entendimiento inicial de la definición del producto qué será entregado (Business Modeling). El producto de esta fase es el Documento de Visión del Proyecto.
2. Elaboración: Alcanzar y acordar un entendimiento inicial del diseño detallado del producto; cómo será construido. Los productos de esta fase son Documento de Arquitectura y Diagramas de Clases, Paquetes y Secuencia.

   Durante esta fase, se divide las aplicaciones en Casos de Uso, o fragmentos funcionales de cada aplicación. Cada uno de esos casos de uso será presentado al usuario para su validación y se establecerá su relevancia, lo cual determinará su prioridad al momento del desarrollo.
3. Construcción: Crear la versión inicial completamente funcional del producto.

El producto se construirá a partir de la implementación de los Casos de Uso respectivos de cada aplicación.

Estas fases se desarrollan de manera iterativa e incremental, lo cual consiste en presentar al usuario avances de cada fase para su validación, de forma que cambios en los requerimientos iniciales puedan ser implementados antes de tener el producto final.

Se realiza un máximo de tres iteraciones por avance (entregable) para cada una de estas fases. Donde cada entregable estará constituido por un grupo de Casos de Uso, agrupados según la relevancia asignada.

ROLES

Se utiliza los siguientes roles dentro del equipo del proyecto:

1. Arquitecto de Software: responsable de la arquitectura de las aplicaciones a desarrollar, lo que implica las principales decisiones técnicas que conforman el diseño e implementación del proyecto.
2. Desarrolladores: responsables del desarrollo de los componentes de las aplicaciones (Casos de Uso) y pruebas de estos. Se tendrá tres desarrolladores, uno de nivel avanzado y dos de nivel intermedio.
3. Usuario: el usuario directo del sistema, en la etapa de desarrollo realizará las pruebas al sistema.

ACTIVIDADES

Las siguientes actividades se realizan por cada rol dentro de las fases que les corresponden:

1. Modelado del Negocio y Establecimiento de Requerimientos: Es responsabilidad del Administrador del Proyecto y el Arquitecto de Software establecer la modelación de los requerimientos del negocio y su documentación. Se realiza dentro de la fase de Incepción.
2. Análisis y Diseño: responsabilidad del Administrador del Proyecto y el Arquitecto de Software la especificación de Casos de Uso. Se realiza dentro de la fase de Elaboración.
3. Implementación: Responsabilidad de los desarrolladores durante la fase de Construcción. Implica el desarrollo de de cada Caso de Uso y sus prueba individual, esta quedara a la libertad de la universidad.
4. Aseguramiento de la Calidad: responsabilidad del Arquitecto de Software verificar la calidad de los productos, por medio de pruebas de integración de los distintos Casos de Uso que componen una aplicación. Realizada durante la fase de Construcción.
5. Desarrollo: Poner a disponibilidad de los usuarios las aplicaciones desarrolladas, será responsabilidad del Arquitecto de Software, durante la fase de Transición.

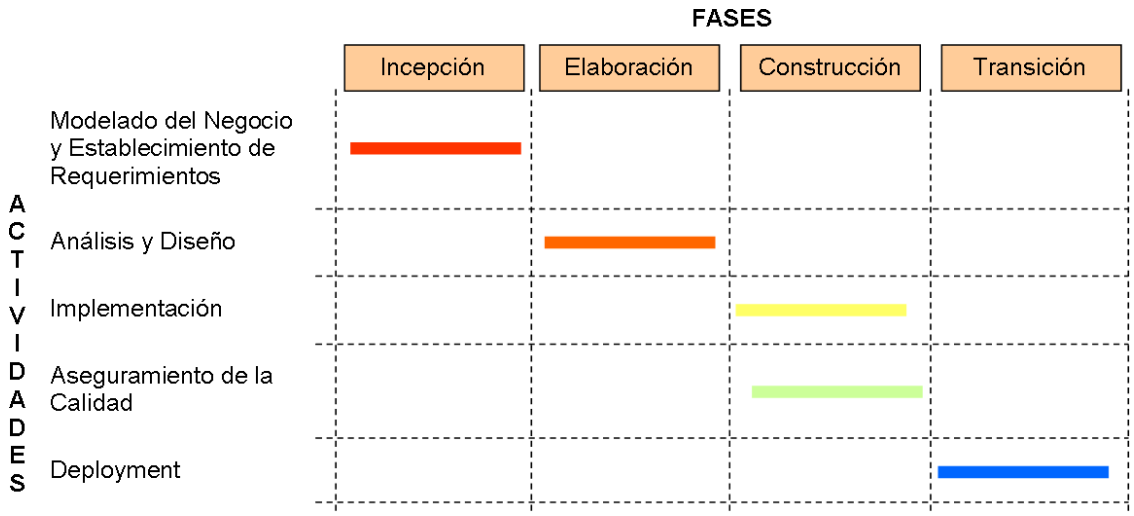


Figura 2. Componentes de la Metodología: Fases, Actividades, Iteraciones

1.10 Plan Capitular

A continuación se presenta las etapas que contendrá el proyecto Editor UML para diagramas de casos de uso:

Capitulo1: Marco Referencial

1.1 Antecedentes
1.2 Importancia de la Investigación
    1.2.1 Planteamiento del Problema
    1.2.2 Justificación
1.3 Objetivos
1.4 Alcances
1.5 Limitaciones
1.6 Delimitaciones
1.7 Proyección Social
1.8 Marco Teórico
    1.8.1 Referencias Históricas
    1.8.2 Referencias Conceptuales
    1.8.3 Referencia Experiencial
1.9 Metodología
1.10 Plan Capitular
1.11 Presupuesto de Implementación

Capitulo2: Análisis de Resultado y Diagnostico

2.1 Tipo de Investigación
2.2 Población y Muestra
2.3 Técnicas y Herramientas de Investigación
2.4 Presentación y Análisis de Resultados

Capitulo3: Análisis del Sistema

1.11 Presupuesto de Desarrollo

Para llevar a cabo el proyecto que consiste en el desarrollo de un editor de diagramas de casos de uso UML, se pretende implementar el siguiente presupuesto:

COSTO DE EQUIPO			
Cantidad	**Insumo**	**precio estimado /unitario**	**precio total**
2	Computadora.	$ 400.00	$ 800.00
1	computadora portátil	$ 1,000.00	$ 1,000.00
2	UPS	$ 25.00	$ 50.00
1	Impresora	$ 40.00	$ 40.00
1	Resma de Papel	$ 5.00	$ 5.00
1	Licencia Visual Estudio. Net	$ 80.00	$ 80.00
**	Presupuesto en caso de imprevistos.	$ 500	$ 500
	Gastos Profesionales	$ 500	$ 1000
TOTAL			**$3475.00**

Tabla No. 2: Presupuesto de desarrollo

CAPITULO II

ANÁLISIS DE RESULTADO Y DIAGNÓSTICO

Introducción

Tener claro a que población estará enfocado nuestro producto-servicio, constituye un esquema de trabajo definido, de esta manera también se podrá conocer las limitantes que como empresa se puede tener, al existir bien definidos los límites de acción.

Al referirse al sujeto de estudio, es claro que éste lo constituye la población o posibles clientes que puedan y quieran requerir de la aplicación a desarrollar. Además, ellos se vuelven el centro de estudio, para ello, se debe calcular el tamaño de la muestra; es decir, tomar una parte del universo para poder generar conclusiones a partir de los resultados obtenidos de las pruebas cubiertas en la muestra. Si bien es cierto, cada elemento de la muestra puede diferir en aspectos específicos, podemos generalizarlos a fin de crear esquemas de oferta y trabajo.

2.1 Tipo de Investigación

A continuación se describe el tipo de investigación realizada y los objetivos que se han planteado para esta investigación.

Los objetivos de la investigación son:

- Definir el contexto ambiental en el que se implementaría el editor.
- Precisar el objeto de estudio.
- Definir y delimitar el problema de investigación y los aspectos que intervienen.
- Seleccionar el método y las técnicas adecuadas al objeto de estudio.
- Organizar y sistematizar las acciones por desarrollar.
- Describir los recursos necesarios.
- Verificar la factibilidad del estudio.

La técnica que se utiliza es la técnica de campo la cual es el instrumento de observación diseñado según el objeto de estudio.

Para lograr obtener un resultado de la investigación que favorezca a la interpretación y solución del problema se realizan las siguientes actividades:

- Explorar. Precisar aspectos previos a la observación, como verificar la utilización de los diagramas de casos de uso en los estudiantes universitarios.
- Reunir información para interpretar hallazgos.
- Describir hechos.

Requisitos al observar

- Delimitar los objetivos de la observación.
- Especificar el procedimiento o instrumentos de observación.
- Comprobación continúa.

2.2 Población y Muestra

Como universo, se pretende abarcar usuarios que hayan cursado la materia de análisis y diseño de sistemas a nivel universitario.

Para conocer exactamente cual debe ser el espacio muestral para desarrollar o implementar nuestro estudio, se utilizó la siguiente fórmula:

$$n = \frac{Z^2 * N * p * q}{I^2(N-1) + Z^2 * p * q}$$

en donde:

- N: Tamaño de la población.
- Z: valor correspondiente a la distribución de Gauss 1.96 para probabilidad de 0.5.
- p: probabilidad de éxito esperada. En caso de desconocerse, aplicar la opción mas desfavorable (p=0.5), que hace mayor el tamaño muestral.
- q: probabilidad de fracaso 1-p.
- I: Error que se prevé cometer. Se tiene un error de 10%

$$n = \frac{1.96^2 * 27 * 0.5 * 0.5}{0.1^2 (27-1) + 1.96^2 * 0.5 * 0.5}$$

cuyos valores serían $Z = 1.96$, $p = q = 0.5$, $N = 27$; con lo que obtendríamos un valor de $n = 22$.

Por lo que nuestra muestra es de 22 estudiantes, a los cuales se les pasó la encuesta a fin de recabar información, y así poder obtener información que nos brinde pautas para poder determinar la importancia de la investigación.

2.3 Técnicas y herramientas de Investigación

La técnica es indispensable en el proceso de la investigación científica, ya que integra la estructura por medio de la cual se organiza la investigación. La técnica pretende los siguientes objetivos:

- Ordenar las etapas de la investigación.
- Aportar instrumentos para manejar la información.
- Llevar un control de los datos.
- Orientar la obtención de conocimientos.

En cuanto a las técnicas de investigación que se utilizarán para el desarrollo del proyecto son dos formas generales: técnica documental y técnica de campo.

La *técnica documental* permite la recopilación de información para enunciar las teorías que sustentan el estudio de los fenómenos de utilización de UML específicamente los diagramas de casos de uso y procesos que se deben tomar en cuanta para la creación del editor de dichos diagramas.

La *técnica de campo* permite la observación en contacto directo con el objeto de estudio, es decir, ex estudiantes de análisis y diseño de sistema de la universidad Don Bosco, y el acopio de testimonios que permitan confrontar la teoría con la práctica en la búsqueda de la verdad objetiva.

Técnica documental

El objetivo de la investigación documental es elaborar un marco teórico conceptual para formar un cuerpo de ideas sobre el objeto de estudio, se consolidará toda la información que sea relevante para el desarrollo del editor.

Fuentes primarias de información

Estas fuentes son los documentos que registran o corroboran el conocimiento inmediato de la investigación incluyen libros, revistas, informes técnicos y tesis.

Técnica de campo

El instrumento de observación se diseña según el objeto de estudio.

Objetivos de la observación

- Explorar. Precisar aspectos previos a la observación estructurada y sistemática.
- Reunir información para interpretar hallazgos.
- Describir hechos.

Al llevar a cabo la investigación se contemplará algunos requisitos necesarios para que el resultado sea favorable, entre estos están:

- Delimitar los objetivos de la observación.
- Especificar el procedimiento o instrumentos de observación.
- Comprobación continua.

La observación sistemática se realiza de acuerdo con un plan de observación preciso, en el que se han establecido variables y sus relación, objetivos y procedimientos de observación.

Para la *observación sistemática,* el instrumento que se uso es la Entrevista.

Entrevista. La encuesta es una pesquisa o averiguación en la que se emplean cuestionarios para conocer la opinión pública. Consiste en el acopio de testimonios

orales y escritos de personas vivas. En la investigación de campo, para la recopilación de información pueden utilizarse las entrevistas, los cuestionarios y el muestreo, entre otros. La entrevista es una de las técnicas más usuales en ciencias sociales. Puede definirse como la relación que se establece entre el investigador y los sujetos de estudio. Puede ser individual o grupal, libre o dirigida.

Objetivos de la entrevista:

1. Obtener información sobre el objeto de estudio.
2. Describir con objetividad situaciones o fenómenos.
3. Interpretar hallazgos.
4. Plantear soluciones.

2.4 Presentación y Análisis de Resultados

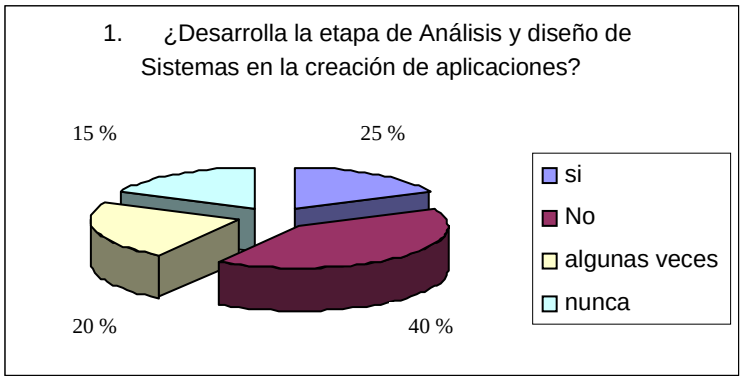
Objetivo general de la encuesta:

Identificar el nivel de conocimiento y aplicación de la filosofía UML orientada a los casos de uso que realizan los estudiantes durante el proceso de análisis y diseño de sistemas.

Resultados de Encuesta:

Pregunta 1:

Objetivo de la pregunta: Conocer el nivel de aplicación de la etapa de análisis y diseño de sistema que hacen los estudiantes al momento del desarrollo del programa.



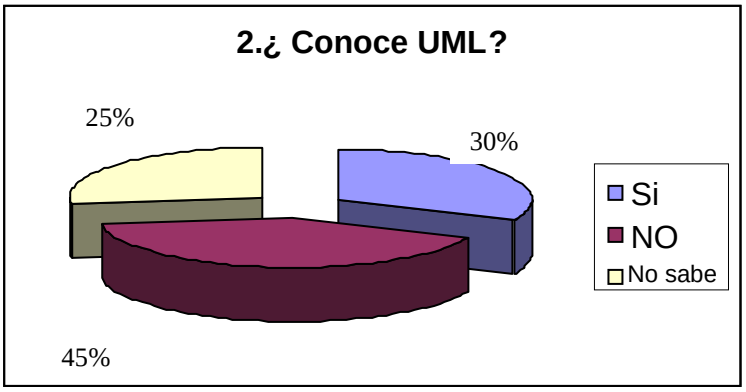
Conclusión:

Se puede inferir que:

- Los estudiantes no poseen buenos hábitos en el desarrollo de aplicaciones.
- No existen herramientas versátiles que estén al alcance de los estudiantes encuestados para realizar un adecuado desarrollo de etapa de análisis y diseño.
- Los docentes no incentivan el uso de herramientas para el análisis y diseño de sistemas.

Pregunta 2:

Objetivo de la pregunta: Verificar el conocimiento de los estudiantes en el tema en estudio.



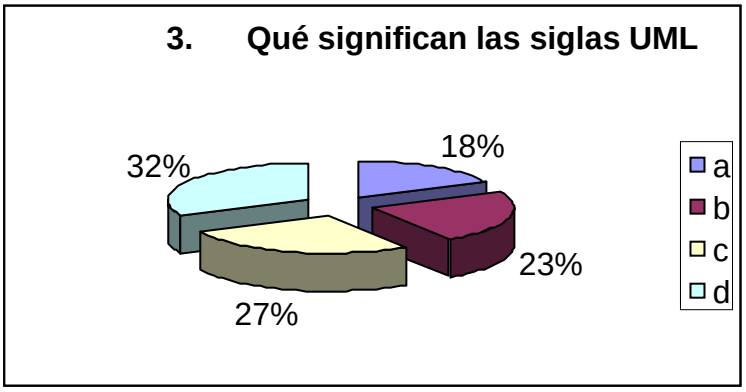
Conclusión:

La población estudiantil encuestada desconoce la filosofía UML por diferentes razones entre estas están, la poca importancia que los estudiantes aplican a la etapa de diseño y análisis dentro del desarrollo de un sistema.

Los planes de estudio contemplan pocas materias que enfatizan la importancia de la etapa de análisis en un sistema de software, o no están actualizados.

Pregunta 3.

Objetivo de la pregunta: Verificar el grado de conocimiento en la filosofía UML.

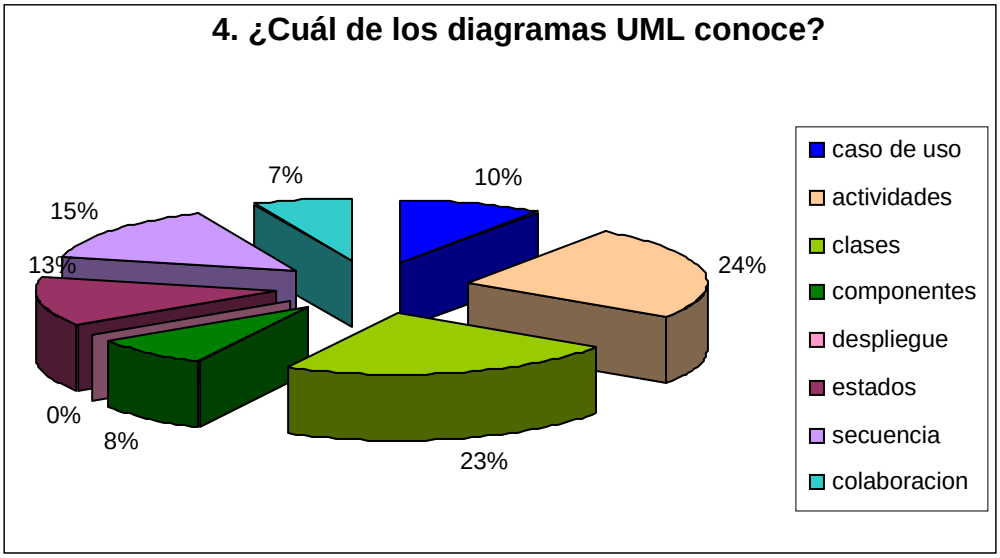


Conclusión:

La mayor parte de estudiantes no posee una base conceptual a cerca de UML y por tanto desconoce el significado preciso de sus siglas y de los demás conceptos que son base teórica para el desarrollo de un análisis.

Pregunta 4.

Objetivo de la pregunta: Corroborar el grado de conocimiento en los diferentes tipos de diagramas UML.



Conclusión:

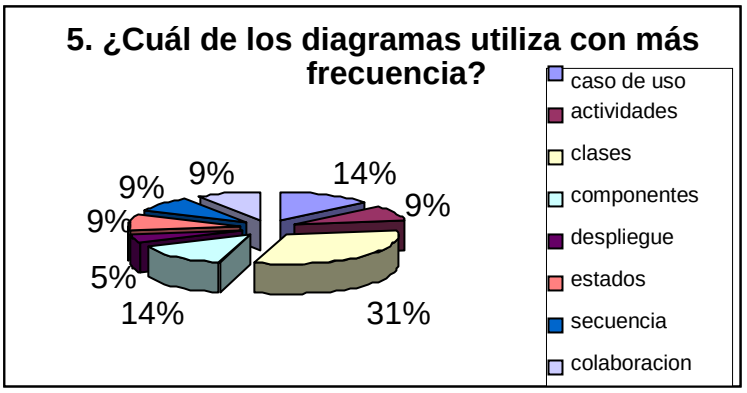
Los tipos de diagramas UML, con los que están más familiarizados según el porcentaje de conocimiento se presentan en el siguiente orden:

1. Diagramas de Actividades.
2. Diagramas de Clases.
3. Diagramas de Secuencia.
4. Diagramas de Estado.
5. Diagramas de Casos de Uso.
6. Diagramas de Componentes.
7. Diagramas de Colaboración.
8. Diagramas de Despliegue.

Estos resultados se deben a que los estudiantes hacen uso de los diagramas generalmente sin saber que estos pertenecen a la filosofía UML.

Pregunta 5

Objetivo de la pregunta: Dar a conocer el grado de utilización de los diagramas UML



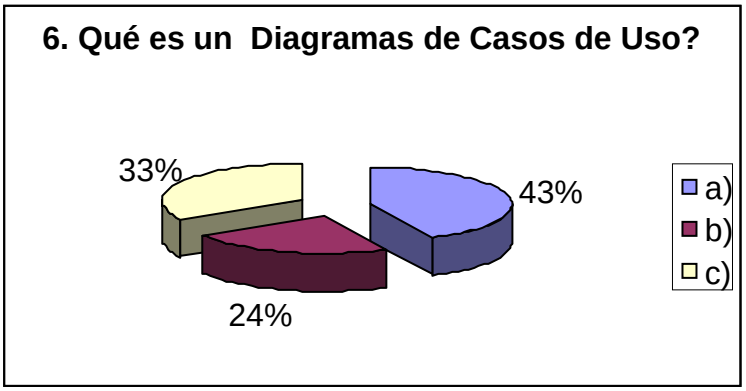
Conclusión:

A partir del resultado de la encuesta se presenta la utilización de los diagramas, siendo el más usado el diagrama de clases, seguido del diagrama de casos de uso y componentes, luego en igual porcentaje los demás diagramas a excepción del de despliegue que es el de menor grado de uso.

Se puede concluir que los resultados de la pregunta 4 con la pregunta 5 se contradicen ya que ambos resultados no concuerdan el uno con el otro, esto se puede deber a la falta de conocimiento de los encuestados en aspectos teóricos que se relacionan con UML; o también al poco interés con que los estudiantes abordan temas como la filosofía UML.

Pregunta 6:

Objetivo de la pregunta: Introducir al encuestado en el tema en estudio.

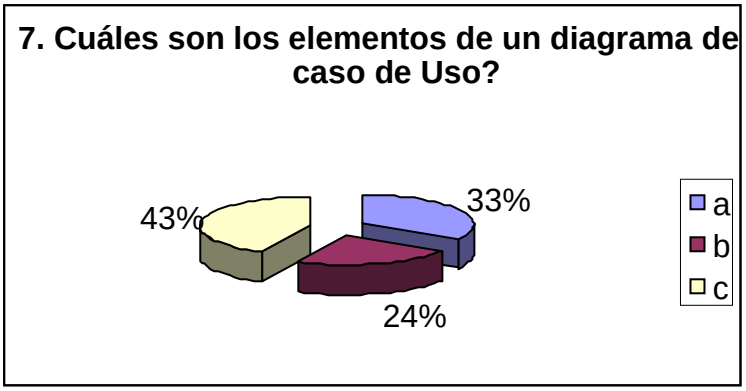


Conclusión:

Un grupo limitado de estudiantes tiene el concepto verdadero de lo que es un diagrama de casos de uso, esto se debe a el escaso uso de dichos diagramas en la etapa de análisis de requerimientos abonado con los resultados de las preguntas anteriores que enfatizan que los casos de uso son poco conocidos como diagramas para la etapa de diseño.

Pregunta 7:

Objetivo de la pregunta: Profundizar en la verificación de conceptos acerca de los casos de uso.

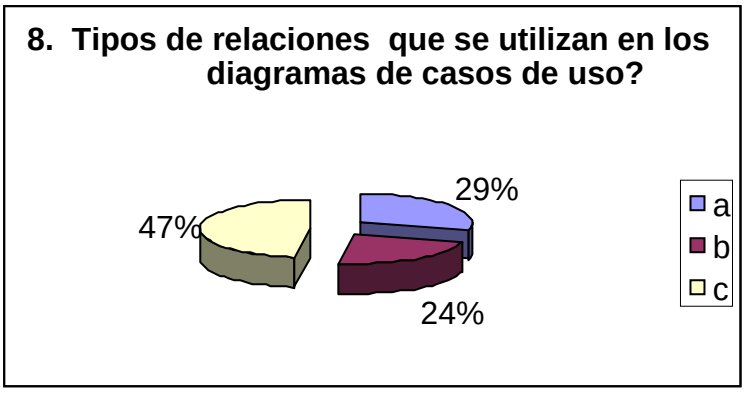


Conclusión:

El porcentaje mayor de estudiantes no tiene claro los elementos esenciales de los casos de uso, se puede inferir que dicho desconocimiento en estos elementos es por la poca familiaridad que los estudiantes tienen en el uso de los diagramas, o por que utilicen los diagramas pero no poseen los conceptos de terminología de los casos de uso.

Pregunta 8:

Objetivo de la pregunta: Dar a conocer el grado de utilización de los diagramas de casos de uso.

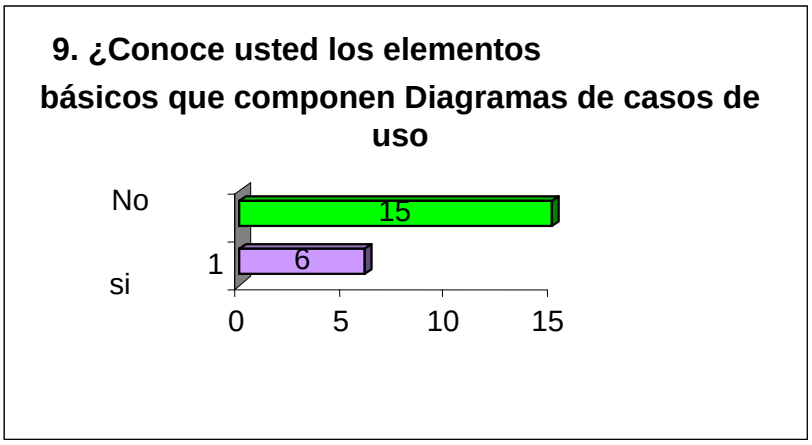


Conclusión:

Los encuestados no diferencian bien los tipos de relaciones concretos que se utilizan en los diagramas de casos de uso, se verifico que hacen asociación con las relaciones a las que se encuentran familiarizados a utilizar y no específicamente a las que se utilizan en los diagramas de casos de uso.

Pregunta 9:

Objetivo de la pregunta: Verificar los resultados obtenidos en el literal 7 a cerca del grado de conocimiento en los diagramas de casos de uso.

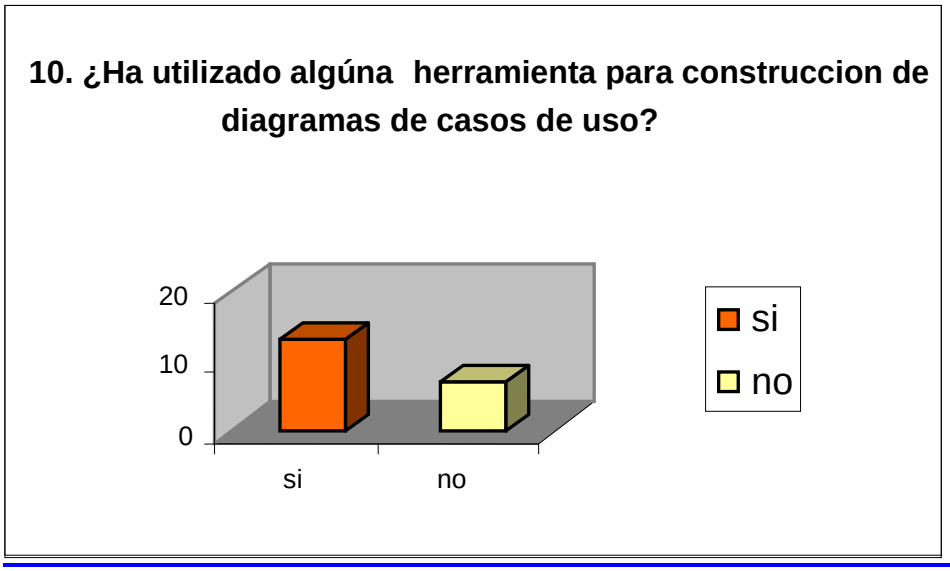


Conclusión:

El resultado corrobora que así como la pregunta 7 demostraba el poco conocimiento de los diagramas de casos de uso, los encuestados aseveran la falta de información a cerca de los aspectos teóricos de los diagramas, se puede inferir que esto es la causa del poco interés de los estudiantes por aprender metodologías de desarrollo nuevas o hacer uso de herramientas case que faciliten la construcción de proyectos de software.

Pregunta 10.

Objetivo de la pregunta: Aseverar la poca familiaridad que los estudiantes tienen en el uso de aplicaciones del tipo que se desarrolla en está investigación.



Conclusión:

La mayor parte de los estudiantes no han utilizado herramientas del tipo del editor que se pretende construir. Esto se puede interpretar como una falta de conocimiento de los estudiantes en herramientas de este tipo, ya que en muchas ocasiones se constato que si se han usado pero el estudiante no identifica el tipo de aplicación.

CAPITULO III
ANÁLISIS DEL SISTEMA

Introducción

A continuación de describe el análisis realizado para la elaboración del editor de diagramas de casos de uso UML, en este capitulo se describen los elementos y factores importantes para poder llevar a cabo la aplicación.

Se plantean las restricciones y se define a que usuarios será dirigido este sistema, los atributos que presenta y los requerimientos de software solicitados para el buen funcionamiento.

3.1 Descripción General

Se desarrolla una aplicación que permita crear un esquema simplificado donde se describa un sistema o realidad desde un determinado punto de vista para facilitar su estudio y compresión.

Los requerimientos son la base de estudio para los casos de uso desde su propia naturaleza, la herramienta permite la creación de modelos de un sistema expresándolo de manera visual mediante la filosofía UML; ayudando a hacer de un análisis complejo un modelo simplificado.

Los casos de uso representan gráficamente una vista del sistema, estudiando los requisitos desde los puntos de vista de los actores y permitiendo llevar el control de la información necesaria de cada elemento.

El proyecto que se construye consiste en un editor de diagramas de casos de uso el cual permite al usuario crear el dibujo del diagrama y además incluye la captura y el almacenamiento de la información relacionada con los diagramas facilitando así el proceso de documentación y enriqueciendo el análisis de requerimientos en el desarrollo de cualquier proyecto.

El editor consta de una interfase gráfica dividida en un área de trabajo, donde se elabora el diagrama, además de barras de herramientas para visualizar los elementos básicos que permitan desarrollar el caso de uso; entre estos están: actor, caso de uso, las relaciones, paquetes; además de los diferentes menús que son de utilidad para desarrollar la aplicación.

Se contemplan diferentes opciones de menú en las que se especifica funciones como crear nuevo, guardar, abrir, imprimir, entre otras.

La metodología del desarrollo del software que se utiliza para el Editor de diagramas de casos de uso UML es el RUP (Racional Unified Process), del cual se especifica y se han adaptado una serie de disciplinas parra llevar a cabo el desarrollo.

Se plantea un método iterativo desarrollando varias fases del análisis a partir de disciplinas como se muestran a continuación:

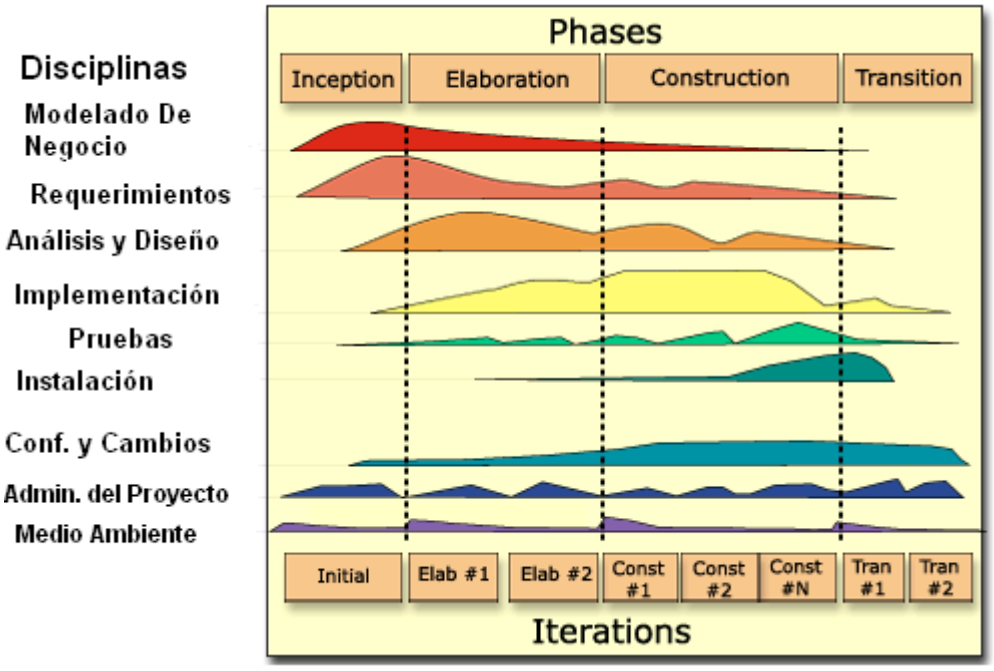


Figura 3*: Componentes de la Metodología*

3.1.1 Perspectiva del Producto

El producto es la creación de un editor UML para generar diagramas de casos de uso que pueda ejecutar las funciones básicas para poder ser una herramienta completa y que solucione las necesidades principales a la hora de llevar a cabo diagramas.

El producto global tiene las siguientes características:

- Diseñar un área especifica para los elementos de los diagramas de casos de uso.
- Elaborar un sector en la aplicación disponible para elaborar los diagramas.
- Permitir al usuario arrastrar los elementos desde el área que contenga los mismos hasta el área de trabajo ya definida.
- Tener la capacidad de cortar, copiar y pegar los elementos de los diagramas
- Determinar puntos en los componentes para poder unirlos, por ejemplo: actor con relación.
- Hacer validaciones de las relaciones para que cumpla con lo que indica la filosofía UML.
- Permitir guardar las características de los componentes en una base de datos.
- Generar reportes haciendo uso de archivos binarios.
- Guardar y abrir archivos.
- Accesible y fácil de operar.
- Generar el diagrama de caso de uso en prosa.

3.1.2 Características del Usuario

El usuario es un elemento fundamental en el desarrollo de cualquier investigación por tanto es de gran importancia analizar las características que debe tener.

A partir de la investigación que se llevó a cabo para el desarrollo del editor de diagramas de casos de uso se determinó como usuarios directos a los estudiantes de análisis y diseño de sistemas, pero cabe aclarar que el editor está desarrollado para satisfacer las necesidades de usuarios con las siguientes características:

- Los usuarios deberán tener conocimientos básicos en computación y de preferencia en el manejo de aplicaciones gráficas o aplicaciones similares al editor de diagramas de casos de uso.
- Los usuarios no deben tener ningún conocimiento avanzado, tan solo deberán conocer los distintos elementos que se utilizan para elaborar los diagramas de casos de uso.
- También deberá conocer el significado de las características de cada elemento para poder explotar al máximo el funcionamiento del editor.
- Deberán estar familiarizados con el manejo de la información que proporcionan o que es posible manejar con los diagramas de casos de uso; para interpretar los resultados que el editor proporcione.

3.1.3 Restricciones

Las restricciones de los casos de uso generalmente son tan pocas que la mayoría de los sistemas no se ven afectados por estas limitantes. Sin embargo, existe un tipo especial de sistema en el cual los casos de uso aun necesitan evolucionar para conseguir cubrir y especificar plenamente sus requerimientos: Estos son los sistemas de tiempo real.

Los sistemas de tiempo real tienen requerimientos particulares que no se encuentran en otros sistemas, un ejemplo de estos requerimientos es:
El Tiempo fuera de servicio (Ejemplo: "No mas de 6 minutos por año por máquina")
La Independencia ("Las fallas de software y hardware serán restablecidas sin intervención humana")

Si queremos modelar estos requerimientos en casos de uso nos encontraremos con varios obstáculos:
¿Quién es el actor?
Si el requerimiento se enfoca a fallas de software o hardware, ¿Quién Genera la falla?. Una falla de un sistema es prácticamente impredecible, por lo que señalar a una entidad como la provocadora de una falla es imposible. Más aun el especificar las acciones que esta entidad realiza con el caso de uso para provocar la falla son ilimitadas.
¿Cómo realizamos el caso de uso?
Si un caso de uso se realiza con la colaboración de clases, ¿qué clase género la falla?. Para realizar correctamente este caso de uso debería modelarse el sistema entero y especificar para cada componente las acciones que podrían llevar a una falla. Esto definitivamente no es un trabajo muy práctico que digamos.

El editor de diagramas de casos de uso contempla las siguientes restricciones:

- Se manejan cinco estereotipos para la clasificación de elementos.

- El editor UML trabaja exclusivamente con diagramas de casos de uso.
- En las funciones del editor no se contempla que sea un generador de código como lo son algunas otras aplicaciones de su tipo.
- El sistema está diseñado para trabajar únicamente en la plataforma Windows.
- Maneja únicamente diagramas, no se realizará estructuras de proyectos.
- El área de trabajo es un área finita a partir del tamaño de las pantallas auxiliándose con el manejo de barras de desplazamiento.
- Todos los elementos se manejarán en formato de imágenes.

3.2 Requerimientos Específicos

3.2.1 Interfases Externas

El editor de diagramas de casos de uso consta básicamente de dos áreas de trabajo; la primera es un espacio en el que usuario observa los iconos que corresponden al desarrollo de cualquier diagrama de casos de uso, la segunda es el área de trabajo donde se visualiza el diagrama terminado y permite desde ahí abrir las otras pantallas para captura de los atributos de los elementos que se han dibujado en el diagrama.
Las características que se almacenan para cada elemento son las siguientes:

Elemento: Actor. 

Características: nombre, estereotipo, documentación.

Elemento: Caso de Uso. 

Características: nombre, documentación, estereotipo, antecesor, predecesor.

Elemento: Paquete. 

Características: almacena el diagrama con los elementos que van dentro del paquete, además de colocar el nombre del mismo.

Elemento: Relación de asociación. 

Características: nombre, estereotipo, Antecesor, predecesor, documentación.

Elemento: Relación de dependencia. 

Características: nombre, estereotipo, documentación.

Elemento: Relación de generalización. 

Características: nombre, estereotipo, documentación, padre, hijo.

3.2.2 Característica del Sistema

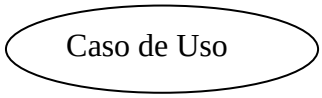
El sistema está desarrollado bajo la plataforma Windows, cuya implementación es compatible con Windows XP, Milenium y 98. El sistema utiliza estructura de datos binaria para almacenar las características de cada uno de los elementos.

Dicho sistema tiene la capacidad de presentar un reporte con los requerimientos obtenidos en base al diagrama o diagramas de casos de uso elaborados, dicho reporte se elabora haciendo uso de archivos binarios, además de tener la capacidad de imprimir el reporte deseado.

En base al diagrama realizado el usuario obtiene un reporte que va a satisfacer las necesidades básicas de dicha tarea, además de ofrecer a la persona un ambiente agradable para trabajo y sobre todo amigable y fácil de usar.

El sistema tiene la virtud de ofrecer una plantilla fácil de usar, detallada para poder ingresar de forma amigable los elementos de los diagramas así como cada una de sus propiedades. Esto proporciona un editor fácil de usar y que con un conocimiento básico de UML pueda desarrollar su tarea de la mejor forma posible.

El editor de diagramas de casos de uso permite desarrollar los diagramas tomando en cuenta las características y reglas de validación de cada elemento; a continuación se especifican algunas de ellas:



Casos de Uso:

Un caso de Uso es una secuencia de Acciones que un sistema posee y permiten observar resultados evaluados por un actor particular.

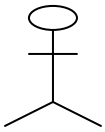
Este diagrama presenta una vista de alto nivel de cómo el sistema se usará desde la perspectiva del actor.

Un caso de Uso contiene:

- Actores.
- Casos de Uso.
- Interacción o relaciones entre actores y casos de uso en el sistema incluyen: la asociación, dependencia y generalización.

Un diagrama de casos de uso puede ser usado durante análisis y captura de requerimientos y es muy útil para entender cómo trabaja el sistema.

Durante la fase de diseño se puede utilizar el diagrama de casos de uso para especificar cómo será implementado el sistema.



Actor:

El actor representa al usuario del sistema, ayudan a delimitar el sistema y aclarar la visión de qué hará el proceso.

Es importante aclarar que un actor interactúa pero no tiene control sobre los casos de uso.

Un actor realiza las siguientes acciones:

- Interactúa o hace uso del sistema.
- Provee las entradas o recibe información del sistema.
- Es un agente externo y no tiene control sobre los casos de uso.

Un actor es:

- Representa un agente que interactúa con el sistema.
- No son parte del sistema que se desarrolla.
- Entran información al sistema.
- Quien usa directamente el sistema.
- Quien es el responsable del mantenimiento.
- Un hardware externo usado por el sistema.

- Otro sistema que se necesita para interactuar con el que se va crear.

¿Cómo identificar un actor?

1. ¿En qué dominios de la organización se usará el sistema?
2. ¿Quién será beneficiario de la nueva funcionabilidad?
3. ¿Quién proveerá/usará información?
4. ¿Quién dará soporte y administrará?
5. ¿Un usuario actuará con diferentes roles?
6. ¿Diferentes usuarios actuarán con el mismo rol?
7. ¿Interaccionará el nuevo sistema con un antiguo?

Características del actor:

- Un actor es un estereotipo de una clase.
- El nombre del actor debe ser desplegado bajo el icono.
- Se puede manejar las relaciones en el diagrama para Mostar la interacción entre actores y casos de uso.
- La relación de asociación puede dibujarse de un actor a un caso de uso.
- La relación de generalización se puede dibujar entre actores.

Casos de Uso:

Es el que identifica lo que el sistema hace.

En esta forma tan simple describe y especifica cómo será usado el sistema por un actor desde su perspectiva. Una descripción más detallada de las características del caso de uso es la siguiente:

- Una secuencia de las transacciones realizadas por un actor en el sistema.
- Envía algunos valores que evalúa el actor.

¿Cómo identificar un caso de uso?

1. ¿Cuáles son las tareas y responsabilidades de cada actor?
2. ¿Algún actor creará, almacenará, cambiará, borrará o leerá esta información?
3. ¿Qué casos de uso crearán, almacenarán o cambiarán la información?
4. ¿Es necesario que un actor informe al sistema sobre cambios externos?

5. ¿Qué casos de uso darán soporte y mantendrán el sistema?
6. ¿Pueden ser realizados por los casos de uso todos los requerimientos funcionales documentados?

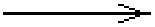
El caso de uso parte de:

- Captura de los requerimientos del sistema.
- Comunicación entre el usuario final y los expertos.
- Prueba del sistema.

Un caso de uso es la mejor manera de descubrir y examinar las definiciones que el actor quiere que el sistema habilite; aunque todas las necesidades del sistema no pueden ser cubiertas en un caso de uso, para remediarlo es usual que exista una colección de casos de uso para especificar las necesidades del sistema.
El caso de uso tiene un nombre, es recomendable escribir una descripción informal del actor y la secuencia de los eventos entre los objetos.
El nombre del caso de uso debe empezar con un verbo por ejemplo nombres de posible casos son: dispensar dinero, transferir fondos.
El nombre se coloca dentro del icono.
En cuanto a las relaciones se puede dibujar una relación de asociación para unir casos de uso y actores y usar la generalización entre dos casos de uso.

Estereotipos:

Un estereotipo representa la sub-clasificación del modelado de elementos, esto representa que forma parte del meta modelo de UML, algunos estereotipos son predefinidos pero también se pueden definir según el tipo de modelo.

Relación de Asociación: 

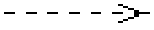
Una asociación proviene de la comunicación, la comunicación puede existir entre casos de uso y actores, clases o interfases. La asociación es la más general de las asociaciones; si dos objetos se consideran independientes se relacionan por asociación.

Por defecto la asociación es unidireccional y se dibuja en un diagrama por una flecha entre el inicio y el fin, el fin indica quién o qué recibe la comunicación.
Para cambiar y hacer la comunicación bidireccional se utiliza una flecha sin dirección.

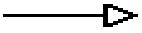
El nombre de la asociación y el estereotipo es un verbo o frase verbal usada para identificar el tipo de la relación.

Para entender la asociación entre el cliente servidor de una acción se sigue el orden de procedencia:

- Si una asociación es de agregación la punta de la flecha apunta al cliente.
- Si no es de agregación entonces se puede navegar determinando el cliente y servidor al final.

Relación de dependencia: 

Una dependencia es una relación entre dos elementos del modelo en el cual el cambio en uno de los elementos afecta también al otro. Se usa una relación de dependencia para conectar elementos con similar nivel generalmente, en diagramas de clases una dependencia indica que la operación en el cliente involucra las operaciones en el servidor.
Se pueden conectar elementos del modelo con dependencia en cualquier diagrama excepto de estado y objetos. Por ejemplo, se puede conectar casos de uso, paquetes, clases con paquetes.

Relación de generalización: 

Una relación de generalización es la relación entre lo más general entre clases o casos de uso y lo más específico. Una generalización muestra una línea sólida del elemento más específico al más general.
Existen tres estereotipos en la generalización:

Extensión: una relación de extensión entre el caso de uso A y el caso de uso B indica que una instancia del caso de uso B puede ser aumentada (Dependiendo de ciertas condiciones especificadas en la extensión) por el comportamiento especificado en el Caso de uso B. El comportamiento es insertado en el punto definido como punto de extensión del Caso de uso B, el cual es referenciado por la relación externa.

Inclusión: Una generalización de un caso de uso A hacia el caso de uso B indica que A es una especialización de B.

Generalización: una relación de inclusión de el caso de uso A hacia el Caso de uso B indica que una instancia de el caso de uso A también contendrá el comportamiento especificado por B. El comportamiento es incluido en el punto definido en A.

3.2.3 Requerimientos de rendimiento

Se necesita que el editor de diagramas de casos de uso permita realizar y cumplir con los siguientes requerimientos:

- Expresar formalmente los procesos de la organización como resultados del análisis de requerimientos.

- Presentar el modelo que represente en forma clara, simple, coherente y condensada las diferentes actividades de la organización.

- Identificar los objetos de la organización y las relaciones existentes entre ellos, de forma tal que se pueda representar el entorno de los procesos que se desarrollan.

- Proveer del esquema general y básico de las relaciones existentes entre los objetos, de forma que sirva de marco estructural para el diseño lógico de la BD, y que oriente su construcción modular y progresiva.

Para poder implementar el editor para diagramas de casos de uso UML se necesita que sea instalado en una maquina que cumpla las características mínimas para obtener un rendimiento aceptable:

Procesador:	433 mhz,
Disco duro:	40 GB o más
memoria Ram:	256 MB
Tarjeta de red	
Windows 98 instalado.	

Tabla No. 3: Requerimientos mínimos de una computadora

3.2.5 Atributos del sistema Software

Las herramientas que se utilizan en el desarrollo del editor son:

- La herramienta RUP para poder llevar a cabo la planificación del proyecto y como metodología de trabajo.
- Visual Estudio 2005 para llevar a cabo la implementación del sistema.
- Microsoft Project para la elaboración de la programación del proyecto.
- Diversos paquetes como son: Rational Rose, Power Builder, etc. Para poder obtener ideas básicas de herramientas similares.
- Se utilizan los paquetes de office para desarrollar el reporte escrito y otras aplicaciones necesarias.

3.2.6 Herramientas Utilizadas en el desarrollo

RUP

El Rational Unifed Process (RUP) muestra la estructura de los procesos de desarrollo desde dos perspectivas:

- El eje vertical representa los aspectos estáticos del RUP, desde este punto representa actividades, artefactos, personas y flujos de trabajo.
- El eje horizontal representa el tiempo y los aspectos dinámicos del RUP, desde esta perspectiva representa ciclos, fases, iteraciones.

Utilizando el RUP, el ciclo de desarrollo del editor se divide en ciclos de desarrollo individuales. Estos ciclos están a su vez divididos en componentes, llamadas fases.
En el RUP, estas fases se denominan: fase de concepción, de elaboración, de construcción, y de transición.

Es durante la fase de concepción que la mayoría del trabajo de modelado de los procesos se lleva a cabo, dicho modelado incluye actividades tales como definir las

necesidades que se desean solventar, incluyendo las razones del proyecto, los beneficios esperados, las opciones consideradas, los costos y los riesgos potenciales.

El objetivo del modelado de procesos es establecer un mejor entendimiento y mejorar el canal entre las partes interesadas con el proyecto.

Aplicación de la Metodología RUP.

Antes de proponer la metodología para efectuar el desarrollo es necesario enfocarse en diagnosticar las razones de porque los proyectos de desarrollo de software fallan. Al llevar a cabo esta actividad se reconoce que existen aspectos comunes que hacen que los proyectos fallen y que se reconocen como los 'síntomas' que se padecen, entre los que se incluyen:

- Poca administración de requerimientos.
- Comunicación ambigua e imprecisa.
- Una frágil arquitectura.
- Una complejidad muy grande
- Inconsistencias no detectadas en requerimientos, diseño e implementaciones.
- Pruebas insuficientes.
- Una evaluación subjetiva del estado del proyecto.
- Baja automatización.
- Baja compenetración de los usuarios y los administradores.

Por todas las razones antes mencionadas, es necesario utilizar un proceso de desarrollo de software que provea una aproximación disciplinada para la asignación de las tareas y de las responsabilidades, cuyo objetivo es asegurarse que el software desarrollado es de la más alta calidad y que satisface las necesidades de los usuarios finales con un tiempo predecible. Dicho proceso de ingeniería del software es el Racional Unified Process o RUP.

El RUP utiliza las mejores prácticas de desarrollo del software, las cuales se encargan de mitigar cada uno de los síntomas presentados con anterioridad,

asegurándose que el desarrollo del editor se lleve a cabo con la más alta calidad, a tiempo. Dichas prácticas se enumeran a continuación:

- *Desarrollar Iterativamente.* Un proyecto de desarrollo iterativo tiene un ciclo de vida que se compone de varias iteraciones que incorporan un conjunto de actividades secuenciales que incluyen modelado de procesos, requerimientos, análisis y diseño, implementación. Pruebas e instalación de software.
- *Administrar Requerimientos.* La administración de requerimientos es un proceso sistemático para encontrar, documentar, organizar y mantener a los requerimientos.
- *Utilizar una arquitectura de Componentes.* Una arquitectura de componentes permite dividir el proyecto de desarrollo en diferentes módulos interrelacionados que facilitan la reutilización de componentes comunes a cada uno de ellos.
- *Modelar Visualmente.* UML es un lenguaje de modelado visual y especificaciones comúnmente utilizado para modelar procesos de negocio y modelar estructuras de la organización. Modelado visual es el uso de notaciones textuales y gráficas para capturar el diseño del software. Una notación, como UML, incrementa la comunicación en el equipo de trabajo y permite una base no ambigua para la implementación.
- *Verificar la calidad continuamente.* Es importante que la calidad de los artefactos sea evaluada en varios momentos a lo largo del proyecto. Esto se logra demostrando y probando diferentes escenarios en todas las iteraciones que contrasta con otras metodologías en donde las pruebas se dejan al final del ciclo de vida del proyecto.
- *Administrar los cambios.* Un reto clave cuando se desarrolla un sistema de software es que debe coordinar desarrollo trabajando en conjunto en múltiples iteraciones y módulos. En ausencia de un control disciplinado y una excelente administración de los cambios al software y a la documentación del desarrollo, este proceso puede degenerar en un caos total.

Estándar ERS

Plantilla para la documentación de los requerimientos (ERS).
El análisis y desarrollo de requerimientos tiene como producto final: un acuerdo documentado entre el cliente y el grupo de desarrollo acerca del producto a ser construido.

El documento es conocido como: Especificación de Requerimientos del Software, Especificación Funcional o Especificación del Sistema.

Características de un documento realizado a través de ERS:

- El ERS debe comprender la totalidad de los requerimientos.
- Los desarrolladores y clientes no deben realizar presunción alguna.
- Si cualquier requerimiento funcional o no funcional no es identificado en el ERS, no es parte del acuerdo, y por lo tanto no aparece en el producto final.

IEEE Std 830 - Prácticas recomendadas para la Especificación de Requerimientos del Software.
Alcance: Son prácticas recomendadas para escribir especificaciones de requerimientos del software.

No identifica método, nomenclatura, o herramienta para preparar el documento de especificación de requerimientos (ERS).

Prácticas recomendadas para la Especificación de Requerimientos del Software.
Características de un buen documento de especificación de requerimientos:

- Correcto
- Claro, sin ambigüedades
- Completo
- Consistente
- Verificable
- Modificable

- Trazable

Preparación conjunta: El proceso de desarrollo de software debe comenzar con un acuerdo sobre lo que debe hacer el software.

Este acuerdo en la forma del documento de Especificación de Requerimientos debe ser preparado conjuntamente

Evolución del documento ERS
El ERS posiblemente necesite evolucionar a medida que el desarrollo de SW progresa.

Consideraciones:
Los requerimientos deben ser especificados con el mayor grado posible de completitud de acuerdo a lo que se sabe en ese momento. La especificación incompleta de algunos requerimientos debe ser mencionada.
Un proceso formal de cambios que identifique, controle y reporte debe ser iniciado.

Modelado de Negocios

Para conseguir sus objetivos, una empresa organiza su actividad por medio de un conjunto de *procesos de negocio*. Cada uno de ellos se caracteriza por una colección de *datos* que son producidos y manipulados mediante un conjunto de *tareas*, en las que ciertos *agentes* (por ejemplo, trabajadores o departamentos) participan de acuerdo a un *flujo de trabajo* determinado. Además, estos procesos se hallan sujetos a un conjunto de *reglas de negocio*, que determinan las políticas y la estructura de la información de la empresa. Por tanto, la finalidad del modelado del negocio es describir cada proceso del negocio, especificando sus datos, actividades (o tareas), roles (o agentes) y reglas de negocio.

El primer paso del modelado del negocio consiste en capturar los procesos de negocio de la organización bajo estudio. La definición del conjunto de procesos del

negocio es una tarea crucial, ya que define los límites del proceso de modelado posterior.

Capturamos los procesos del negocio a partir de los objetivos principales de la empresa. En primer lugar, consideramos los objetivos estratégicos de la organización. Teniendo en cuenta que estos objetivos van a ser muy complejos y de un nivel de abstracción muy alto, serán descompuestos en un conjunto de sub-objetivos más concretos, que deberán cumplirse para conseguir el objetivo estratégico. Estos sub-objetivos pueden a su vez ser descompuestos en otros, de manera que se defina una jerarquía de objetivos. En nuestro estudio, hemos experimentado que dos o tres niveles de descomposición son suficientes. Para cada uno de estos sub-objetivos de segundo nivel definimos un proceso de negocio que deberá dar soporte a dicho sub-objetivo. Representamos cada proceso del negocio como un *caso de uso del negocio*, que inicialmente será descrito de forma textual.

Identificación de Roles del Entorno del Negocio

Una vez se han identificado los procesos de negocio, es preciso encontrar los agentes involucrados en su realización. Cada uno de estos agentes o actores del negocio desempeña cierto papel (*juega un rol*) cuando colabora con otros para llevar a cabo las actividades que conforman dicho caso de uso del negocio. De hecho, identificaremos los roles que son jugados por agentes de la propia investigación o agentes externos (como clientes u otros sistemas).

Para tener una visión general de los diferentes procesos de negocio de la organización, puede construirse un *diagrama de casos de uso del negocio*, en el cual aparece cada proceso del negocio como un caso de uso. Este diagrama permite mostrar los límites y el entorno de la organización bajo estudio. Sólo se mostrarán en este diagrama los actores del negocio correspondientes a los roles externos al sistema, de forma que los procesos de negocio en los que sólo tomen parte roles internos a la organización no estarán conectados a ningún actor.

Descripción de los Casos de Uso del Negocio

El siguiente paso dentro del modelado del negocio es introducirse en cada uno de los casos de uso del negocio identificados, para describirlo en detalle.
A continuación, hemos de determinar los agentes internos que juegan un rol en cada caso de uso del negocio. Hasta el momento hemos identificado los roles que pertenecen al entorno de la organización. Ahora es necesario estudiar la descripción de cada caso de uso del negocio, y observar el conjunto completo de roles involucrados, tanto externos como internos a la organización.

UML

Se utiliza UML como lenguaje de modelado debido a las siguientes razones:

1. El sistema de software profesional es diseñado y documentado antes de que sea codificado. Sabremos exactamente los que conseguiremos, por adelantado.
2. Ya que el sistema del sistema está antes de crear la primera parte del código, es fácil descubrir el código reutilizable y tratarlo para una mejor eficacia.
3. Los vacíos en el diseño del sistema podrán ser descubiertos antes sobre los diagramas del mismo.
4. El software se comportará de la forma esperada y surgirán menos sorpresas.
5. El diseño total del sistema dicta el modo en que se desarrollará el software. Las decisiones finales se harán antes de que nos encontremos con código mal escrito. Con esto ahorraremos tiempo en nuestro desarrollo.
6. Cuando necesitemos hacer modificaciones en el sistema, es mucho más fácil hacerlo sobre la documentación UML. Hay que recurrir mucho menos a rehacer un nuevo estudio. Volvemos a ahorrar tiempo en nuestro desarrollo.
7. Si se incorporan nuevos desarrolladores al proyecto, los diagramas UML les permitirá hacerse rápidamente una idea del sistema.
8. La comunicación con nuestros desarrolladores, y con desarrolladores externos, es mucho más eficiente.

Se utilizan los diferentes diagramas:

Herramienta: Diagrama de Casos de Uso

Entender el Uso del Sistema: Para realizar un análisis de requerimientos En la reunión con los usuarios potenciales, los desarrolladores se reunieron con los usuarios para descubrir a los actores que inician cada Caso de Uso y a los actores que se beneficiaran de ellos (Un actor puede ser tanto un sistema como una persona).

Herramienta: Diagrama de Estados

Analizar los cambios de Estado de los Objetos: También el modelador de objetos refina el modelo mostrando los cambios en el estado de un objeto.

Herramienta: Diagramas de Secuencia y Diagramas de Colaboración

Definir las interacciones entre los objetos: En este momento el equipo de desarrollo cuenta con juego de casos de uso y un diagrama de clases refinado, es hora de definir como los objetos actúan entre sí. El modelador de objetos desarrolla un juego de diagramas que incluye los cambios de estado.

Herramienta: Los diagramas de Clases

Descubrir los procesos de negocio: Aquí los analistas deben entender perfectamente los procesos cómo se trabajará en el editor, en este diagrama se describe la estructura de datos con los que se lleva un control de la información del sistema.

Visual Estudio .NET

Características de una aplicación desarrollada en VB. NET:

- Funcionales.
- Simplicidad.
- Escalabilidad.
- Permite diseños con alto grado de vistosidad.
- Permite manejo de objeto.

Características de Visual Estudio NET:

Asistente para la actualización El asistente para la para migrar las aplicaciones de Visual Basic 6.0 a Visual Basic .NET ha sido mejorado y admite ahora Controles de Usuario y las Web Clases.

Depurador Mejorado Un depurador mejorado muestra más vistas de datos legibles e intuitivos para ayudar a detectar y corregir errores en el código antes de su implementación.

Nuevos proveedores administrados ADO.NET Los nuevos proveedores administrados ofrecen acceso fácil a Oracle 7i, y 8i y fuentes de datos ODBC.

Sitios de Comunidad en línea Obtenga conocimiento y consejo del sistema mejorado de Ayuda Dinámica así como hospedaje de sitios de comunidad en línea

Entorno de Desarrollo Integrado en Studio Visual .NET Le proporciona mejoras de tiempo, rendimiento y mayor confianza.

Para poder llevar a cabo el desarrollo del sistema se utilizara visual Basic. Net 2005, basado en el modo grafico los cuales tendrán la opción de ser arrastrados, copiados, cortados y pegados, para poder desplazar los elementos que componen el diagrama de casos de uso se utilizan referencias en los planos X e Y para poder así referenciar lo puntos para poder moverlos, además se toma un punto central para referenciar el objeto.

Por ejemplo en el caso del elipse se tomara un punto en el centro de dicho elemento también almacenando los extremos tanto en el eje X como en el eje Y siendo estos los puntos de referencia.

Se ha agregado un contador para poder visualizar el arrastrado de los diversos elementos que componen los diagramas de casos de uso, además de poder referenciar la posición anterior en la que se dejo.

CAPITULO IV
DISEÑO DEL SISTEMA

Introducción

El diseño de sistemas es el proceso por el cual las necesidades finales se transforman en un paquete de software. Este capítulo desarrolla la fase de diseño del sistema y contiene los diseños de programas, pantallas, diagramas y reportes el mapa del sistema un prototipo de diseño de la interfaz y la documentación de especificaciones técnicas.

Para el sistema se plantea el siguiente diseño:
La pantalla será dividida en dos partes globales, la primera es la general en donde se encuentra todas las barras y botones necesarios obtenidos según el análisis previamente elaborado.

La segunda sección es el área de trabajo en donde se lleva a cabo los diversos diagramas que el usuario desee realizar estas se generan de manera dinámica al activar un botón.

4.1 Diseño del Programa

El sistema se ha planeado de la forma como se presenta a continuación:

- El área de trabajo: Se pretende realizar un sistema que conste de dos partes en lo que se refiere a la aplicación en general, dichas partes son:
  1. Un sector fijo o predeterminado en donde se colocan todos los menús y otros componentes de la consola.
  2. Un área de trabajo destinada solamente para que el usuario pueda desarrollar los proyecto.

En la primera opción se opta por realizar un modelo estándar fijo; éste contiene las pestañas en las cuales aparecen los diferentes menús, también contienen los

botones para las opciones básicas, en pocas palabras se pretende colocar una barra de menús y una barra de herramientas de forma estática.

Otro elemento que contiene el sistema es el sector donde se encuentren almacenados los botones de los elementos que componen a los diagramas de casos de uso; además un área para elementos secundarios, y un sector para visualizar los elementos que han sido insertados en el diagrama; el objetivo de este sector facilitar la navegación y agilizar la búsqueda.

En la segunda fase del sistema se ha contemplado un sector de la aplicación para que el usuario pueda construir los diagramas, éste es el segundo elemento que compone la aplicación.

Para esta se pretende tener una pantalla finita que es el área visible, en donde las personas que usan la aplicación puedan elaborar los diagramas, dicha pantalla se adapta a la resolución de monitor en la que se ejecute.

- Barra Estándar: Se ha realizado un menú en donde se encuentren las opciones básicas:

1. Archivo
2. Edición
3. Ver
4. Opciones
5. Ventana
6. Ayuda.

Dichos elementos contienen en su interior opciones propias de cada sub menú, para facilitar al usuario las tareas a realizar.

Se utiliza en los componentes de cada uno de los menús antes mencionados características propias las cuales se describen a continuación:

1. Menú Archivo:

Sus opciones son:

1. Nuevo
2. Abrir
3. Guardar
4. Guardar Como
5. Imprimir
6. Vista Preliminar
7. Salir

Nuevo: En esta opción se genera una nueva ventana de trabajo, para esto se crea de forma dinámica áreas de trabajo para el usuario, al aparecer el nuevo sector de trabajo este se autodenomina Diagrama1, el usuario al almacenarlo podrá cambiarle el nombre.

Abrir: se abre una ventana auxiliar en forma de explorador para poder realizar la búsqueda del archivo por carpetas, y el usuario elige el directorio.

En esta opción el editor identifica todos los archivos con extensión ecu, es decir, todos los archivos con la forma *.ecu se ejecutan en el editor.

Guardar: Al igual que la función anterior ésta al ser activada la primera vez desplegará una ventana auxiliar para poder almacenar el trabajo, cabe mencionar que cuando el diagrama ya ha sido guardado solamente se almacenará de nuevo con la ruta ya establecida.

Como se menciona anteriormente la extensión con la que se guardaran los archivos que se generen con el editor será ecu.

El formato para guardar los archivos generados es el binario, se ha elegido este por su funcionalidad y su simplicidad.

Guardar Como: Su comportamiento es similar a la opción anterior solamente que en este caso la ventana emergente siempre aparece, dicha ventana se llamada al momento de activar dicha función.

Imprimir: Llama a la propiedad de impresión, en este submenú se selecciona detalles específicos relacionados con dicha opción como son selección de impresora, número de páginas, número de copias y además una opción de vista previa de modo que el usuario pueda seleccionar las diversas facilidades.

Vista Preliminar: Para la vista preliminar existe una opción para visualizar cómo se ve la pantalla en el diseño de impresión, así la persona que se encuentre utilizando el producto puede verificar la calidad del diagrama y así poder realizar las modificaciones deseadas antes de imprimir, esta opción se activa al darle clic para que aparezca en una nueva ventana la visualización de cómo queda el diagrama realizado.

Salir: Al igual que las funcionalidades mencionadas anteriormente ésta se encuentra en el menú archivo y su objetivo es cerrar el sistema, al activar esta función automáticamente el usuario cierra el sistema en su totalidad de no haber guardado se presenta una opción que pregunta si desea guardar antes de desactivar la aplicación llamando así a la función guardar en caso de darse una respuesta afirmativa, caso contrario solamente terminara de ejecutar la aplicación sin realizar cambio alguno.

2. Menú Edición:

Está compuesto por los siguientes elementos:

1. Deshacer
2. Rehacer
3. Cortar
4. Copiar
5. Pegar
6. Seleccionar Todo

Deshacer: Dicho componente elimina la acción realizada, es decir que regresa el diagrama a su estado anterior de realizada la acción.

Rehacer: Esta función es lo contrario a la anterior, si deshacer borraba la ultima acción realizada, ésta rehace la acción que se ha eliminado.

Cortar: Esta función desaparece el elemento seleccionado, dicho elemento se almacena en memoria de manera temporal. la función de este es el de poder seleccionar componentes y trasladarlos a otros sectores o borrarlos de el archivo.

Copiar: Se implementa para que la persona que haga uso de la aplicación pueda crear una replica del elemento o elementos seleccionados, al igual que el componente Cortar almacena en memoria temporalmente él o los elementos copiados y al seleccionar otro éste ocupa su lugar, es importante mencionar que al cerrar la aplicación se limpia automáticamente el contenedor que almacena los elementos.

Pegar: Esta opción cierra el ciclo de las 2 anteriores ya que inserta de nuevo los elementos que se encuentren en memoria, es importante definir que el componente que ingresa es el último al cual se le acciono las 2 funciones mencionadas.

Seleccionar Todo: Como su nombre lo dice seleccionara todos los componentes que posea el archivo para poder aplicarles un evento.

3. Menú Ver:

Este almacena los siguientes elementos:

1. Barra de Herramientas
2. Barra de Estado
3. Alejar
4. Acercar
5. Vista Reporte

Barra de Herramientas: Le permite a la persona que utiliza el sistema aparecer y desaparecer la barra de herramientas, cuando aparezca seleccionada la pestaña el usuario sabe que la barra de herramientas está en modo visible; cuando esté desactivado permite dejar más espacio de visualización del diagrama.

Barra de Estado: La funcionalidad es similar a la barra de herramientas esta opción desaparece la barra de estado al igual que su predecesor al estar activada la pestaña indica que la barra se encuentra visible.

Alejar: Se elabora para que el usuario al hacer clic en esta función pueda aumentar el área de visualización y así poder contemplar todo el diagrama, al activar esta opción el porcentaje de visualización ira disminuyendo esto quiere decir que el área aumenta en tamaño y la visualización de los elementos será en menor tamaño.
Acercar: Es la acción contraria al componente Alejar, permite que el sector de trabajo disminuya y que los elementos puedan aumentar en tamaño de visualización.

Vista de Reporte: Se busca generar una sub pantalla en la cual pueda aparecer el reporte con un formato ya preestablecido, el objetivo es que el usuario pueda

visualizar como se va generando el reporte creado en base a los diagramas realizados.

Esta es una función muy importante, que permite generar la opción de crear en prosa el diagrama de casos de uso.

El formato del reporte esta compuesto por:

1. Nombre del Elemento
2. Estereotipo del Elemento
3. Relación de Procedencia
4. Elemento de Procedencia
5. Relación Destino
6. Elemento Destino
7. Descripción

El reporte se ve de la siguiente forma:

Nombre del Elemento	
Estereotipo del Elemento	
Relación de Procedencia	**Elemento de Procedencia**
Relación Destino	**Elemento Destino**
Descripción	

Figura 4: Diseño del Reporte

Este formato se aplica para cada uno de los componentes que formen parte del diagrama, si se encuentran por ejemplo 20 elementos el reporte consta de 20 de este esquema de datos y así sucesivamente todo depende del número de elementos que contenga el diagrama.

Éste se genera al momento de almacenar los datos en el sector detallado para dicho fin.

4. Menú Opciones:

Se encuentra compuesto por los siguientes componentes:

1. Exportar a Word
2. Colores
3. Formato

Exportar a Word: Esta opción permite exportar el reporte a un documento de Microsoft Word para poder así satisfacer cualquier necesidad que el usuario crea conveniente.

Colores: Permite cambiarle el color de fondo al elemento seleccionado, el número de colores permitido para el cambio es de 5 colores, estos son:

- Rojo
- Azul
- Verde
- Negro
- Blanco

Los colores mencionados anteriormente son con los que se va a cambiar el fondo de los elementos, es importante mencionar que las relaciones y otros elementos por sus dimensiones el fondo es demasiado reducido y no se permite llevar a cabo el llenado de color.
También esta opción se usa para almacenar el color deseado del borde, dicho color se aplica al área de desarrollo del elemento además del la línea que cubre el entorno.

Formato: Al activar este control el usuario modifica el formato de letra que posee el sistema, aparece una ventana auxiliar donde se encuentra opciones cómo**:**

- Tamaño
- Tipo de Letra
- Negrita
- Cursiva
- Subrayado

Tamaño: Incrementa o decrementa el tamaño de la letra.

Tipo de Letra: El usuario puede elegir entre 8 tipos de letra, inicialmente estos tipos de letra son:

Arial
Verdana
Times New Roman
Arial Black
Tahoma
Comic Sans
Arial Narrow
Arial Unicote

Negrita: Permite poner el texto seleccionado en negrita resaltándolo así de los otros textos.

Cursiva: El texto seleccionado toma el efecto cursivo al activar esta opción.
Subrayado: Dibuja una línea debajo del texto que se ha marcado resaltando este.

5. Menú Ventana:

Incluye los siguientes ítems:

1. Nueva Ventana
2. Cerrar Todo

3. Área para las ventanas abiertas

Nueva Ventana: La función es similar a la de la opción nuevo del menú archivo, creando nuevas ventanas con un correlativo y con la opción de almacenar y cambiar el nombre.

Cerrar Todo: Esta funcionalidad cierra todas las ventanas o áreas de trabajo que se encuentren abiertas en el proyecto con la validación de guardar los cambios antes mencionada en el fichero salir del menú Archivo.

Área para Ventanas Abierta: Es un área destinada para presentar todas las áreas abiertas. Que se encuentren abiertas al momento de estar trabajando en la aplicación.

- **Barra de Herramientas:** Es otra facilidad que ofrece el sistema, permite acceder de forma mas fácil a opciones siguientes:

1. Nuevo: La funcionalidad será prácticamente igual a la descrita en la opción Nuevo del menú archivo.

2. Abrir: La opción trabaja de forma idéntica a la descrita en el apartado de la funcionalidad abrir del menú archivo.

3. Guardar: El comportamiento de este botón será el mismo al de la opción guardar del menú archivo.

4. Imprimir: La funcionalidad en este caso será un poco diferente debido que imprime todo el documento sin necesidad de presentar una pantalla auxiliar.

5. Vista Previa: Su funcionamiento será similar a la opción de vista preliminar en el menú archivo.

6. Cortar: Trabaja de la misma forma como se ha planteado en la opción cortar del menú edición.

7. Copiar: Su funcionalidad es la misma de la opción copiar en la carpeta edición.

8. Pegar: Se comporta de la misma manera como lo hace la opción con el mismo nombre en la opción edición.

- **Barra de Herramientas de Casos de Uso:** Esta es una de las funcionalidades nuevas que se implementan en el sistema, propias de la herramienta.

Esta barra esta constituida por seis elementos representados por botones a modo que al darles clic aparezca en el área de trabajo el componente de los diagramas de casos de uso seleccionado, estos son:

1. Actor
2. Caso de Uso
3. Paquete
4. Relación de Asociación
5. Relación de Generalización
6. Relación de Dependencia

La característica de estos botones es que cuando se presione uno de ellos aparece el respectivo componente en la pantalla, este componente es una caja de dibujo la cual se puede modificar en tamaño y en su posición original según sea el criterio de la persona que esta utilizando la aplicación, es decir que cuando el usuario le de un clic al botón aparezca automáticamente en el área de trabajo el elemento correspondiente a la opción seleccionada y que esta imagen pueda tener propiedades de arrastre por la pantalla y de variación de tamaño.

Hay otro punto importante, que la propiedad de arrastre solo es valida dentro del área de trabajo, es decir que si se desea mover un objeto fuera del área de trabajo este sistema no lo permite.

Al hacer doble clic en el objeto del área de trabajo despliega un formulario auxiliar para que el usuario pueda llenar los datos propios de dicho componente.

A continuación se detallan datos propios de cada componente:

Actor: Para poder activar este componente es necesario que al momento de darle clic al botón el sistema cree un cuadro de imagen con el archivo jpg que le corresponde y con un contador para ponerle un correlativo de nombres provisionales, el nombre estaría compuesto por la palabra actor y el correlativo obtenido del contador, el contador se inicializa cada vez que se entra a la aplicación.

Es importante mencionar que el nombre asignado por defecto se puede cambiar.

Otra opción es la de variar las dimensiones físicas del actor tanto su posición como su tamaño pero siempre con la validación de que se pueda realizar solamente en el área de trabajo.

Al momento de crear este elemento también presenta algunas restricciones propias de UML como es el que no se pueda unir a ningún otro elemento si no es a través de una relación.

El elemento contiene terminales específicas donde se pueda realizar la unión entre las relaciones y el actor, para generar así una mejor presentación, más estética y ordenada, en total el número de terminales son cuatro y están ubicadas en los 4 puntos cardinales del actor.

Al hacer doble clic encima del objeto se presentará un formulario auxiliar para agregar los datos del actor, es necesario que el usuario llene todos los datos que ahí se le indican para poder enriquecer así el reporte y completar el diagrama, debido a eso habrá una validación para que si no ha llenado todos los campos no le permita almacenar los otros datos.

Los datos que posee son almacenados en variables para ser utilizados a la hora de generar el reporte, los datos se capturan en cajas de texto y al darle en el botón aceptar automáticamente el contenido de esas cajas de texto pasan a las variables ya destinadas para ello y serán almacenadas en el formato para generar el reporte.

Los campos a llenar son los siguientes:

- Nombre
- Estereotipo
- Descripción
- Elemento Destino
- Relación Asociada

Nombre: En este apartado el usuario pondrá el nombre que el desee asociarle al actor, la longitud del nombre podrá ser lo máximo de 20 caracteres.

Estereotipo: Para este apartado el usuario no digitará debido que para actor solo se utilizará un estereotipo el que lleva el mismo nombre del elemento, por dicha razón este campo aparecerá bloqueado a modo que no le permita digitar en el.

Descripción: Este campo es el de mayor longitud de todos podrá almacenar hasta 1000 caracteres y en el se utilizará una caja de texto multilíneas con una barra de desplazamiento vertical para facilidad de digitación.

Elemento Destino: En este apartado se selecciona el elemento con el cual está enlazado a través de la relación, este elemento lo puede ser almacenado automáticamente o digitado.

Relación Asociada: En este apartado el sistema automáticamente toma la relación que asocia al actor con otro elemento.

Caso de Uso: El modo de operación de este componente es similar al del actor, se activa al momento de presionar el botón y aparece un elemento caso de uso listo para poder trabajar, cambiarle el nombre y poder arrastrar el elemento por toda el área destinada para ello.

Además al crear un componente de CU (Caso de Uso) este adquiere un nombre automáticamente que está compuesto por las iniciales CU y por un correlativo, es importante mencionar que también presenta la opción para poder cambiar el nombre del elemento.

Este además presenta la propiedad de variar el tamaño del elemento a modo de aumentarlo o disminuirlo según sea la necesidad del usuario.

El componente de los diagramas de casos de uso también presenta sus reglas, por eso es importante determinar cuales serán las restricciones, por ejemplo siempre se tiene que unir a un actor o a un paquete a través de una relación nunca se pueden unir sin relaciones.

Al activar el control a través de su botón respectivo el caso de uso aparecerá en el área de trabajo y el usuario lo podrá movilizar solamente en dicho sector.

Al darle doble clic en el objeto creado se despliega una ventana que se utiliza para que se ingresen los datos correspondientes a este.

Los campos que se deben de completar son los siguientes:

- Nombre
- Estereotipo
- Descripción
- Elemento Destino
- Relación Asociada

Nombre: En este apartado el usuario pondrá el nombre que el desee asociarle al caso de uso, la longitud del nombre al igual que en el actor podrá ser lo máximo de 20 caracteres.

Estereotipo: Para este apartado el usuario no digitará, debido que para caso de uso sólo se utiliza un estereotipo el que lleva el mismo nombre del elemento, por dicha razón este campo aparecerá inhabilitado, para poder así mantener la integridad del campo y reducir el margen de error que se pueda presentar a la hora de realizar el reporte o cualquier presentación del diagrama elaborado.

Descripción: El espacio destinado para este campo es el de mayor longitud, debido que irá almacenado cualquier detalle específico que a la hora de estar trabajando, siendo uno de los campos principales y por lo cual se le ha decidido dejar el suficiente espacio para trabajar, dicha longitud es de 1000 caracteres.

Elemento Destino: Se optó por este campo debido que es importante saber el origen que le esta dando vida al caso de uso, para esta opción el sistema será capaz de almacenar el nombre del elemento que viene a preceder al caso de uso.

Relación Asociada: En esta sección el sistema automáticamente toma la relación que asocia al caso de uso con otro elemento.

Estos datos recolectados serán almacenados en archivos binarios, y así asignarles la extensión que tendrán al final y el mismo caso se dará al recuperar el archivo

Verificación de Validaciones

1. Validaciones de Unión

- **Actor**

Inserta un elemento

Se realiza una unión

Si la unión es de un actor con una relación

Se puede realizar

Sino

No se puede realizar debe ser una relación

- **Caso de Uso**

Inserta un elemento

Se realiza una unión

Si la unión es de un Caso de Uso con una relación

Se puede realizar

Sino

No se puede realizar debe ser una relación

- **Paquete**

Inserta un elemento

Se realiza una unión

Si la unión es de un Paquete con una relación

Se puede realizar

Sino

No se puede realizar debe ser una relación

- **Relación**

Inserta un elemento

Se realiza una unión

Si la unión es de una Relación con otro elemento

Se puede realizar

Sino

No se puede realizar debe ser una relación

Validaciones de Relación según UML.

Si la relación une a un actor con un actor

No se puede realizar

Si la relación une un actor con un Caso de Uso

Si se puede realizar

Si la relación es de un actor con un paquete

Si se puede realizar

Si la relación es de un Caso de Uso con un actor

Si se puede realizar

Si la relación es de Caso de Uso con Caso de Uso

Si se puede realizar

Si la relación es de Caso de Uso con un paquete

Si se puede realizar

Si la relación es de un paquete con un actor

Si se puede realizar

Si la relación es de un paquete con un caso de uso

Si se puede realizar

Si la relación es de un paquete con un paquete

Si se puede realizar

Algoritmos

- **Entrada.**

Inicializa los componentes

Refresca el área de trabajo

Limpia la memoria

El editor esta listo para trabajar

- **Opción Abrir**

Se active la opción abrir

Manda a llamar al subformulario de abrir

Abre el directorio que se le ha determinado de estándar

Carga los archivos de dicho directorio en pantalla

Si el archivo se encuentra en dicho directorio entonces

Activa el botón Abrir

Sino

Realiza búsqueda en un nuevo directorio

Si se presiona el botón Abrir

Abre el documento Sino Se espera otra orden por parte del usuario

Si se presiona el botón Cancelar

Cierra la pantalla

Si se abre el archivo

Este se carga en pantalla

- **Opción Guardar**

Se activa la opción Guardar

Si es la primera vez que se guarda

Se manda a llamar el subformulario de guardar

Sino

Guarda lo que contiene el área de trabajo

Si se llama el subformulario de guardar

Manda a llamar al directorio PRE establecido

Carga dicho directorio en el área ya definida para ello

Presenta en pantalla el directorio

Si es el directorio deseado

Se le pone el nombre

Se presiona el botón Guardar

Se almacena el archivo en el directorio

Sino

Se busca otro subdirectorio

- **Opción Guardar Como**

Se le da clic a la funcionalidad

Manda a llamar a la subventana de Guardar Como

Manda a llamar el directorio PRE definido

Carga el directorio en el área destinada para ello

Si el directorio es el indicado

Da clic en el botón guardar

Almacena el diagrama

Sino

Realiza Búsqueda del directorio

Si da clic en el botón cancelar

    Cierra la ventana de Guardar como sin almacenar nada

- **Opción Imprimir**

Se activa dándole clic al botón

Manda a llamar a la ventana auxiliar de impresión

Manda a llamar a todas las impresoras disponibles o conectadas al sistema

Carga las impresoras

Si se le da clic al botón imprimir

    Manda a impresión el archive

Sino

    Si le da clic al botón Salir

        Cierra la ventana de impresión sin provocar cambio alguno

Algoritmo de Validación de Arrastre.

Posición_objeto

Posición_ini_pantalla_amp

Posición_fin_pantalla_amp

Posición_ini_pantalla_altura

Posición_fin_pantalla_altura

Sí la Posición_objeto < Posición_ini_pantalla_amp

    El objeto no se puede colocar en dicha área

    Sino

Si la Posición_objeto > Posición_fin_pantalla_amp

El objeto no se puede colocar en dicha área

Sino

Si la Posición_objeto > Posición_ini_pantalla_altura

El objeto no se puede colocar en dicha área

Sino

Si la Posición_objeto > Posición_fin_pantalla_altura

El objeto no se puede colocar en dicha área

Sino

El objeto puede ser colocado exitosamente en dicha área

Algoritmo para los paquetes

Dar doble clic en el elemento

Si el elemento <> paquete

Abrir propiedades del elemento

Sino

Si el elemento ya existe

Abrir área de trabajo con el nombre ya establecido

Almacena elementos colocados en área existente

Se elabora los diagramas dentro del paquete o se modifica

Se almacena el diagrama dentro del paquete

Sino

Abrir un área de trabajo nueva

Coloca elemento en área nueva

Se elabora los diagramas

Se almacena el diagrama dentro del paquete

4.2 Diseño de Diagramas.

Diagramas de Negocios.

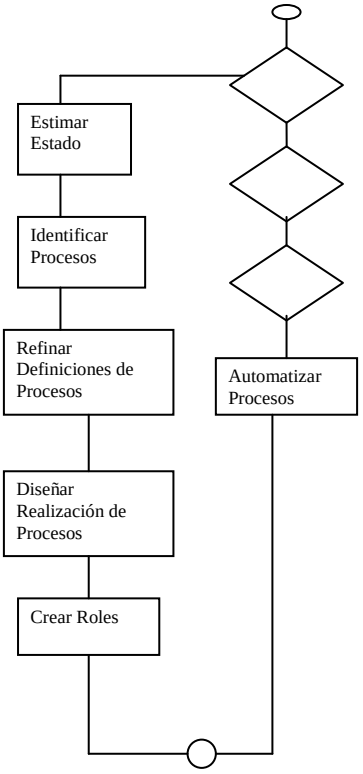


Figura 5: Diagrama de Negocio

Descripción de Diagrama de negocios.

Estimar Estado de Negocio.

Metas: Crear un editor de Diagramas de Casos de Uso UML.

Situación Actual: Se parte de que no existe un editor, es por esta razón que se decide trabajar los casos de uso, por ser los diagramas diseñados para la búsqueda de requerimientos.

Razones: Se crea el editor para tener una herramienta fácil para el desarrollo de casos de uso.

Problemas:

UML: Validación de elementos que constituyen los casos de uso.

Programación: Aumentar, disminuir, cortar, pegar, unir, mover, eliminar elementos.

Mejoras: Crear un editor de fácil uso que permita manejar los casos de uso de forma eficiente fácil y confiable, además de esto que sea gratuito.

Reglas de negocio: como reglas de negocio se ubican las relaciones validas entre los elementos; a continuación se presenta una tabla donde se describen las uniones permitidas a nivel de objetos de programa; anteriormente se describió en el algoritmo la validación según UML:

	Actor	Caso de uso	paquete	Relación Asociativa	Relación Dependencia	Relación Generalización
Actor	NO	NO	NO	Si	Si	Si
Caso de Uso	NO	NO	NO	Si	Si	Si
Paquete	NO	NO	NO	Si	Si	Si
Relación Asociativa	Si	Si	Si	NO	NO	NO
Relación Dependencia	Si	Si	Si	NO	NO	NO
Relación Generalización	Si	Si	Si	NO	NO	NO

Tabla4: Validaciones UML

Definir, Identificar, Diseñar realización de Procesos

Procesos y descripciones:

Iniciar.

Objetivo: Abrir aplicación y preparar escenario.

Entradas: El usuario inicia aplicación desde el sistema operativo.

Salidas: La aplicación en sus condiciones iniciales con el área de trabajo y barras de menú cargados.

Recursos: cargar aplicación.

Secuencia de actividades: El usuario abre la aplicación, carga el área de trabajo con un nuevo documento y la barra de herramientas.

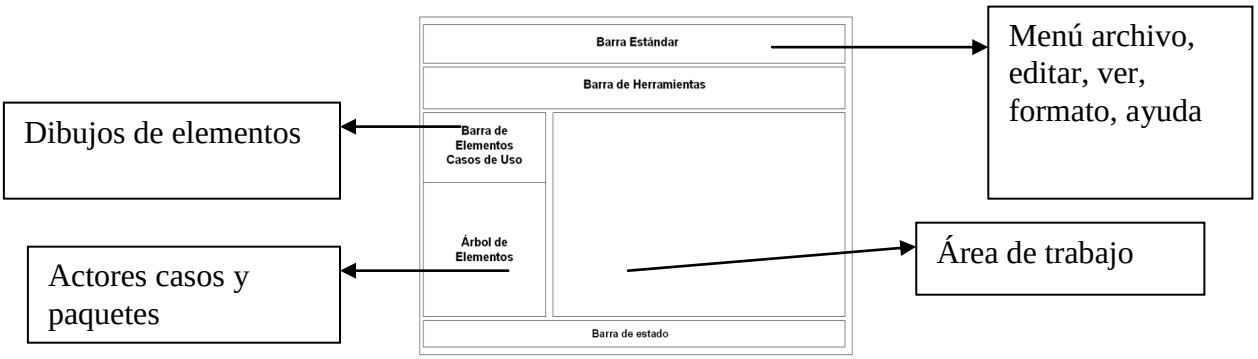


Figura 6: Estructura del Editor

Eventos: Abrir aplicación, Abrir archivo creado.

Arrastrar.

Objetivo: Llevar elemento al área de trabajo.

Entradas: Dar click y arrastrar elemento.

Salidas: Elemento colocado en área de trabajo.

Recursos: Barra de elementos.

Eventos: Seleccionar elemento, dar clic, arrastrar, soltar.

Llenar Información.

Objetivo: Almacenar información.

Entradas: Dar clic en elemento y guardar información.

Salidas: Datos del elemento almacenados.

Recursos: Formulario para llenar información.

Eventos: Clic en elemento, llenar form y guardar.

Copiar / Cortar.

Objetivo: Copiar y cortar elemento.

Entradas: Seleccionar elemento.

Salidas: Elemento Duplicado.

Recursos: Elemento.

Eventos: Clic en elemento, menú cortar/ copiar.

Unir Elementos.

Objetivo: Unir elementos.

Entradas: Elementos de diagrama.

Salidas: Símbolos unidos.

Recursos: Elementos.

Eventos: Seleccionar antecesor y seleccionar destino y unir.

Imprimir.

Objetivo: Imprimir diagrama.

Entradas: Diagrama.

Salidas: Diagrama impreso.

Recursos: Impresos y diagrama.

Eventos: Menú imprimir, vista previa imprimir.

Guardar.

Objetivo: Almacenar información.

Entradas: Opción guardar del menú.

Salidas: Diagrama almacenado.

Recursos: Diagrama y barra de herramienta.

Eventos: opción guardar, almacenar cambios.

Reporte

Objetivo: generar reporte.

Entradas: Diagrama y opción reporte.

Salidas: Reporte.

Recursos: Diagrama.

Eventos: opción imprimir.

Salir

Objetivo: Cerrar aplicación.

Entradas: Opción menú.

Salidas: Aplicación cerrada.

Recursos: aplicación abierta.

Eventos: salir.

Refinar Roles.

Roles	**Eventos**	**Resultados**
Usuario	Abrir	Aplicación cargada
	Nuevo	Iniciar diagrama
	Copiar	Copiar elemento
	Cortar	Cortar elemento
	mover	Mover elemento
	Arrastrar	Elemento en área
	Guardar	Almacenar información
	Reporte	Generar vista caso de uso
	imprimir	Generar diagrama

Tabla 5: Roles

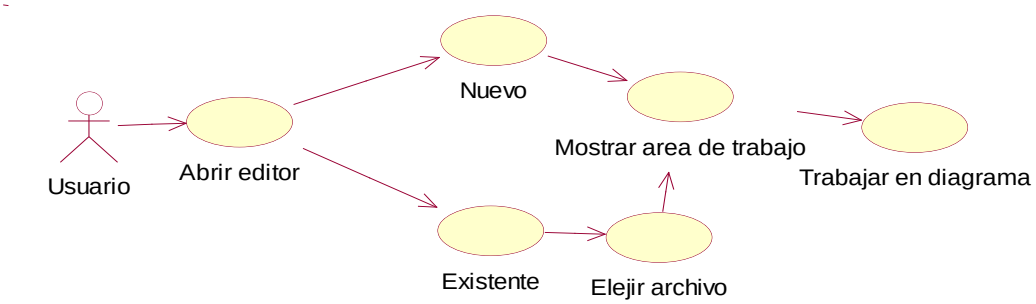
Automatizar Procesos.

Proceso	**Automatizado**
Iniciar	100 %
Arrastrar	100 %
Llenar información	100 %
Copiar/ cortar	100 %
Imprimir	100 %
Reporte	100 %
Guardar	100 %
salir	100 %

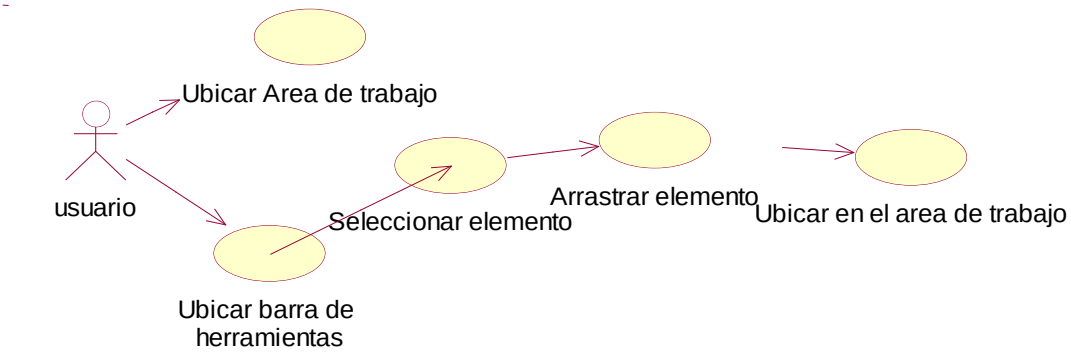
Tabla 6: Listado de Procesos

Diagramas de Casos de Uso.

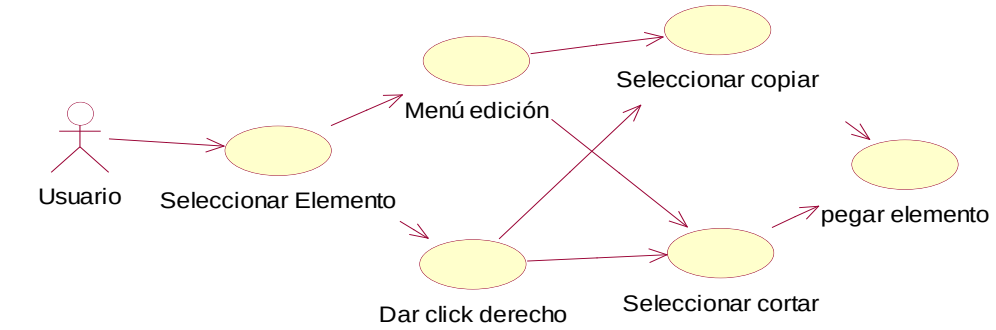
Caso de Uso 1. Iniciar Aplicación



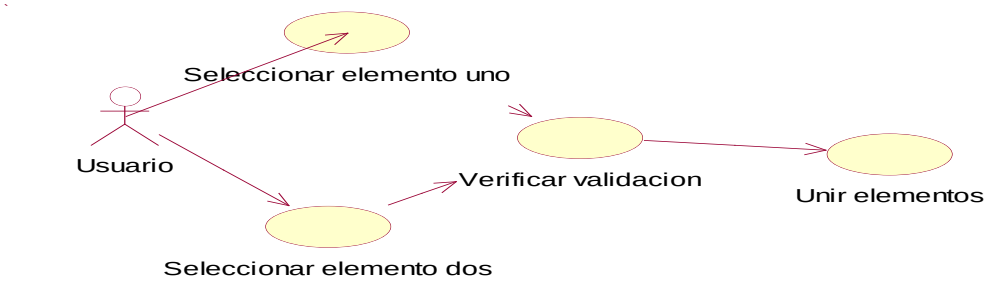
Caso de Uso 2. Manejo de Elementos



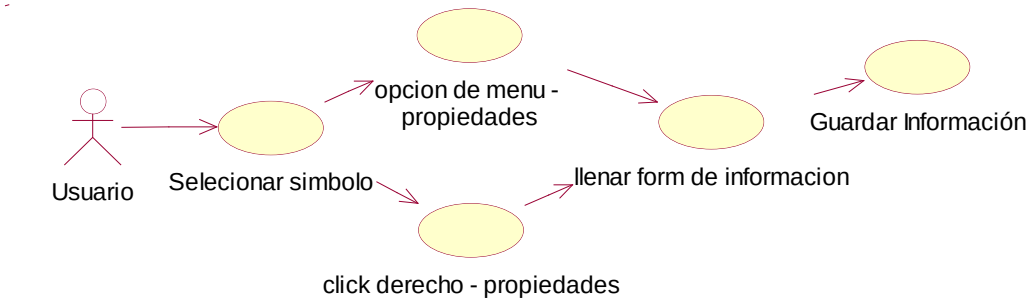
Caso de Uso 3. Copiar / Cortar elementos



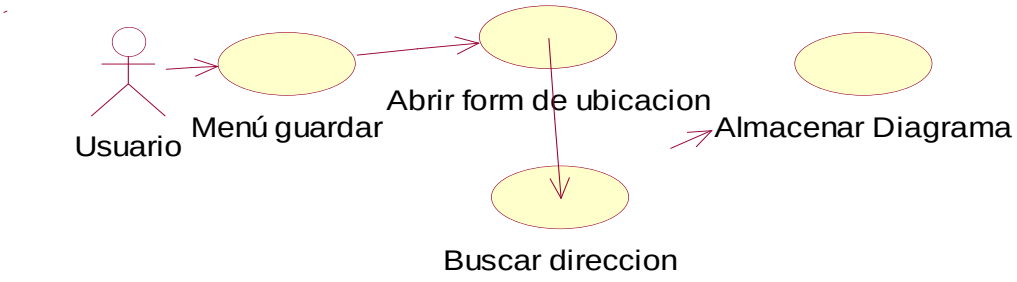
Caso de Uso 4. Unir Elementos



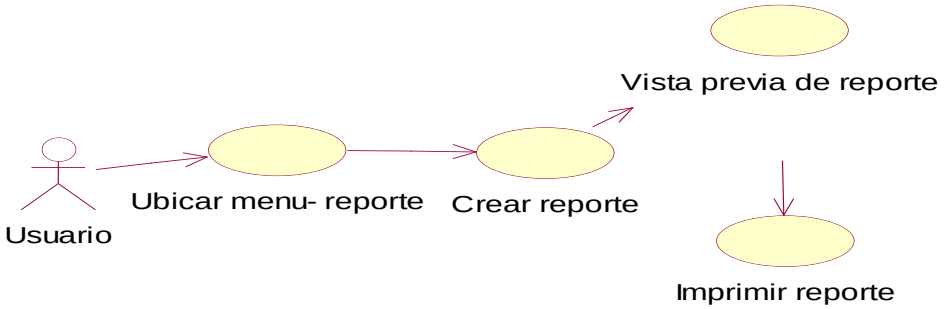
Caso de Uso 5. Llenar Información



Caso de Uso 6. Guardar



Caso de Uso 7. Generar Reporte



Diagramas de Colaboración.

En el siguiente diagrama se representa el inicio del sistema como se llama a cada uno de los elementos que componen la aplicación.

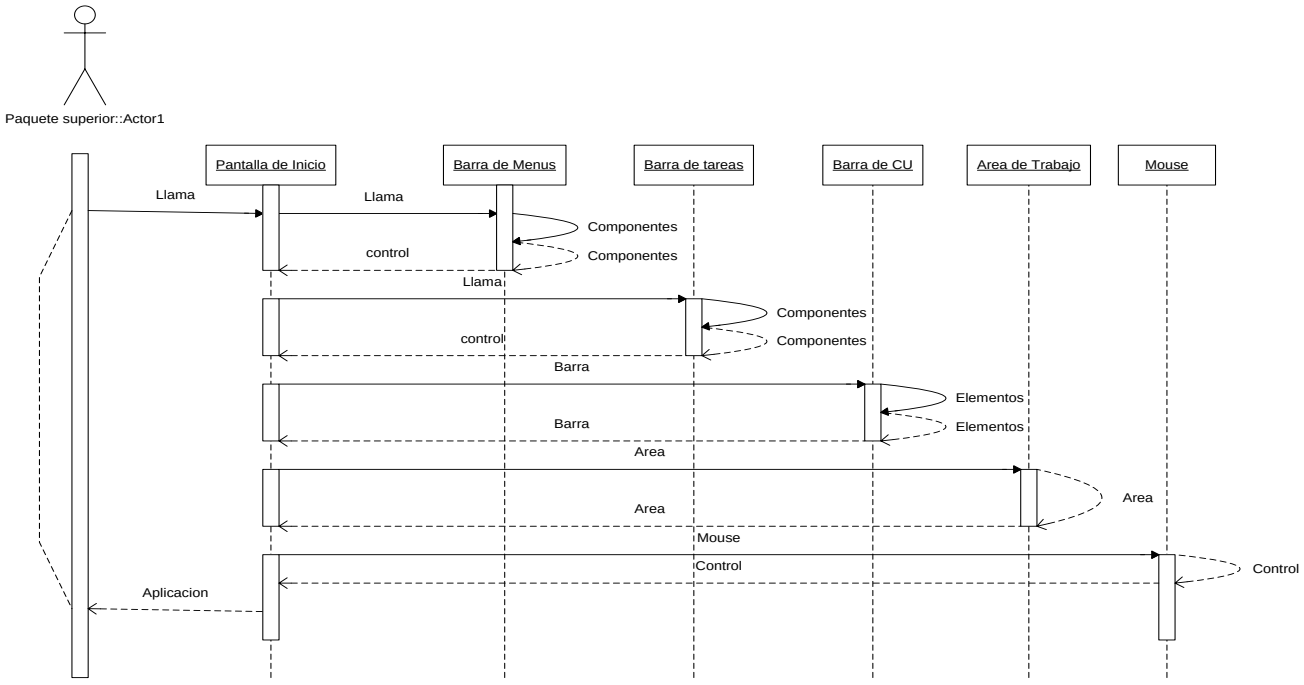


Figura 14: Diagrama de Colaboración

Diagramas de Clases

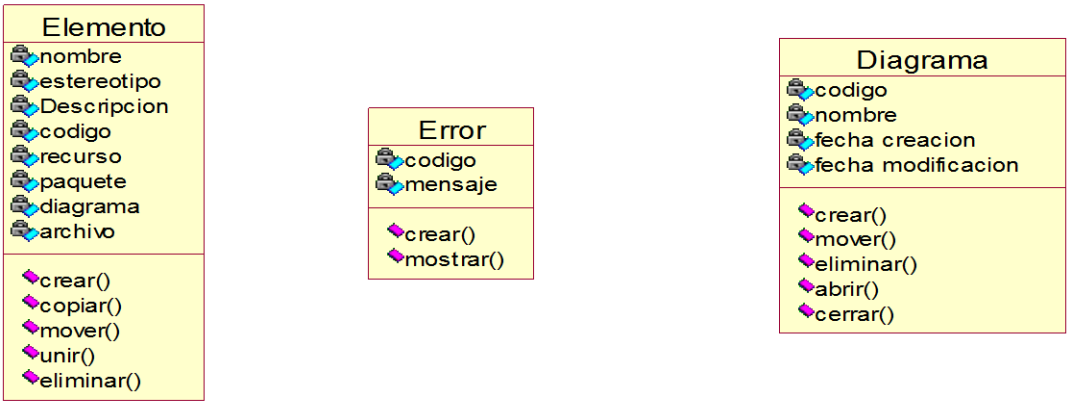


Figura 15: Diagrama de Clases

4.3 Diseño de Pantallas del Sistema.

El siguiente bosquejo muestra en forma general como se ha planeado la visualización del proyecto:

Figura 16: Diseño de pantalla 1

A continuación se detalla como se distribuye de los componentes en la pantalla.
Para empezar comenzaremos a detallar la parte fija de la pantalla que es siempre la misma sin importar cuantas ventanas podamos tener abiertas.

A continuación se presentara el bosquejo de la distribución de los componentes en dicha barra:

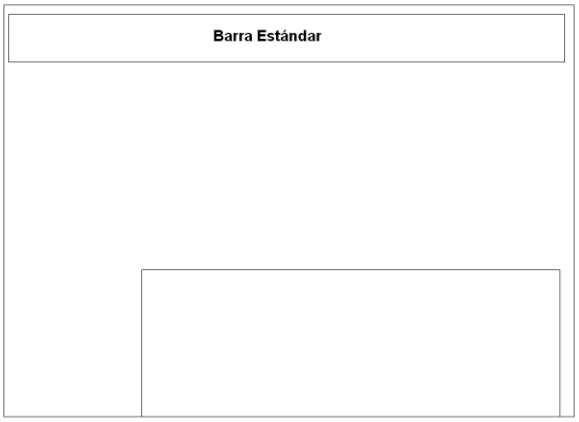


Figura 17: Diseño de pantalla 2

Para iniciar se le ubica en la parte superior una barra estándar que convierte el editor en una herramienta mas completa, los componentes que contiene la barra estándar son:

1. Archivo
2. Edición
3. Ver
4. Opciones
5. Ventana
6. Ayuda.

El otro elemento que contendrá nuestro sistema es la barra de herramientas, a continuación se visualiza el lugar donde se encontrara dicha barra:

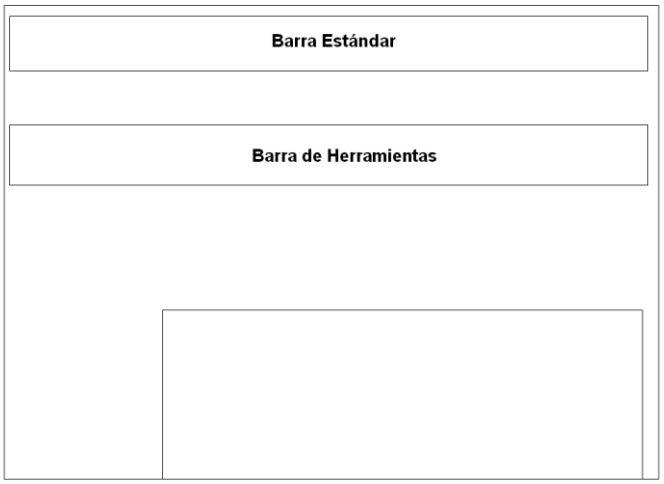


Figura 18: Diseño de pantalla 3

La barra de herramientas como podemos ver se encuentra a continuación de la barra estándar como podemos observar ambas se extienden a lo ancho de la aplicación.

Los componentes de la barra de herramientas son:

1. Nuevo
2. Abrir
3. Guardar
4. Imprimir
5. Vista Previa
6. Cortar
7. Copiar
8. Pegar

Otro elemento importante en el sistema es la barra de elementos de los diagramas de casos de uso, esta es una parte fundamental porque en ella se contendrán los elementos fundamentales en el editor.

La posición en la que se encuentra dicha barra es la siguiente:

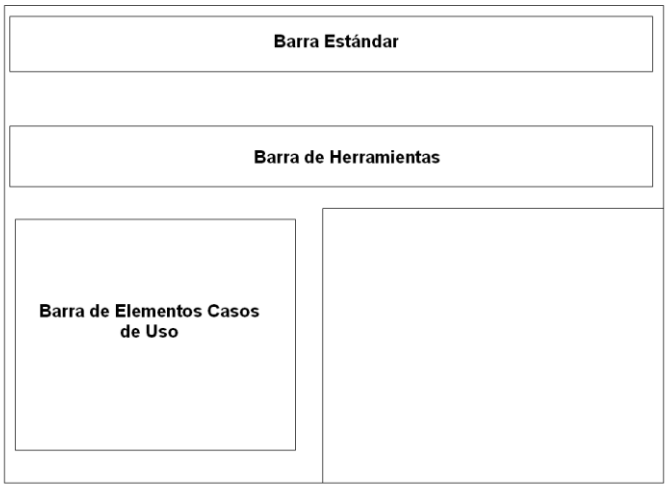


Figura 19: Diseño de pantalla 4

Como se aprecia en la imagen la ubicación de esta es por debajo de las barras mencionadas anteriormente ubicándose en la parte izquierda de la pantalla, esta contiene 6 botones con las características que se mencionan a continuación:

1. Actor
2. Caso de Uso
3. Paquete
4. Relación de Asociación
5. Relación de Generalización
6. Relación de Dependencia

Otro de los componentes que se presentan es un árbol que contiene los elementos que se vayan utilizando en el diagrama facilitando así la búsqueda de los elementos dentro de esta.

La ubicación de este se vera a continuación:

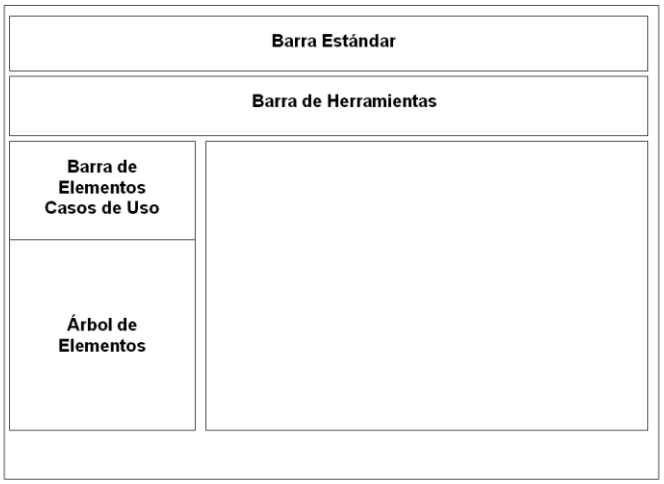


Figura 20: Diseño de pantalla 5

Como podemos apreciar la ubicación es en la parte inferior izquierda de la pantalla, en este componente aparecen los nombres de los elementos que forman parte del diagrama.

Estos son ordenados según la categoría a la cual pertenecen, por ejemplo todos los miembros del diagrama que sean actores estarán juntos, los que sean casos de uso estarán juntos y así sucesivamente para que se venga a facilitar a la hora de buscar un elemento, esta facilidad vera mayor resultado a la hora de trabajar con diagramas demasiado extensos.

Para finalizar la descripción general de cómo es el diseño del sistema se utiliza también una barra de estado la cual estará ubicada en la parte inferior de la aplicación.
Como se puede apreciar en el siguiente diagrama:

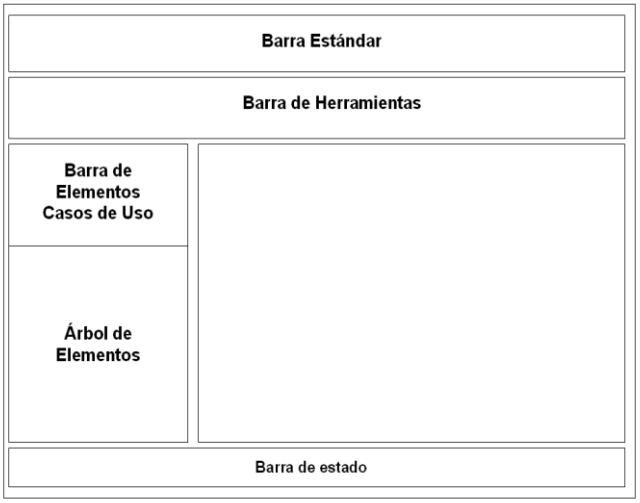


Figura 21: Diseño de pantalla 6

Con esto se finaliza como es en forma general el proyecto, se puede apreciar que se ha dividido en dos partes globales el sistema y la parte fija se ha dividido en 5 sub elementos, es importante mencionar que cada uno de ellos también contiene elementos hijos.

A continuación también se hará el diseño general de las pantallas secundarias que son llamadas a través de los botones de la barra de elementos de casos de uso.

Para poder llevar a cabo las pantallas secundarias se hace mediante un estándar, para iniciar estas pantallas contienen un espacio para el nombre del elemento como se puede apreciar a continuación:

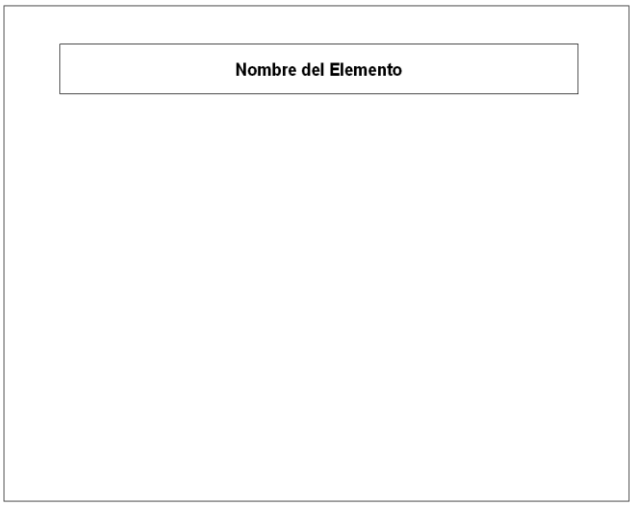


Figura 22: Diseño de reporte 1

Como se aprecia el nombre aparece en la parte superior del estándar de las pantallas secundarias.

A continuación se presentara el segundo elemento de los sub formularios y la posición de este en la pantalla.

El segundo elemento es el estereotipo, que en cada uno de los componentes de los diagramas de casos de uso podemos encontrar diferentes estereotipos, en nuestro caso cada elemento tiene uno diferente.

A continuación podemos apreciar la ubicación del estereotipo en el estándar:

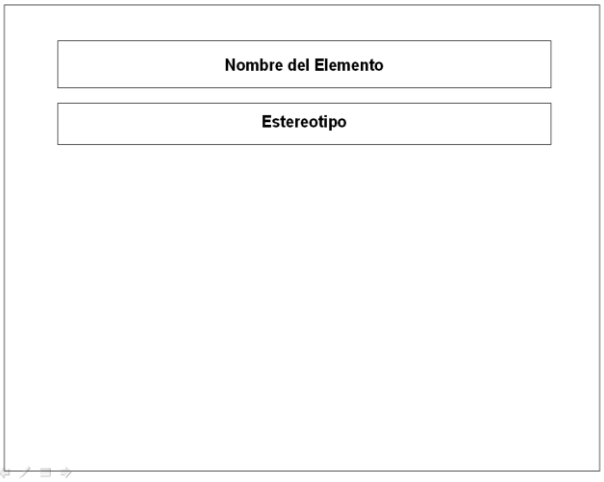


Figura 23: Diseño de reporte 2

El campo del estereotipo como se aprecia en la figura esta colocado justamente debajo del nombre.

Es importante recalcar que debido a que cada componente solo tiene asociado un estereotipo este campo estará bloqueado a modo que automáticamente al crear el objeto este lo pueda adquirir.

El otro factor que viene a integrarse a la pantalla que estamos diseñando es el de la descripción, que se vuelve un factor importante, debido que en ella se colocan aspectos importantes que son retomados a la hora de generar el reporte en prosa del diagrama que se esta realizando.

La ubicación de esta en la pantalla es:

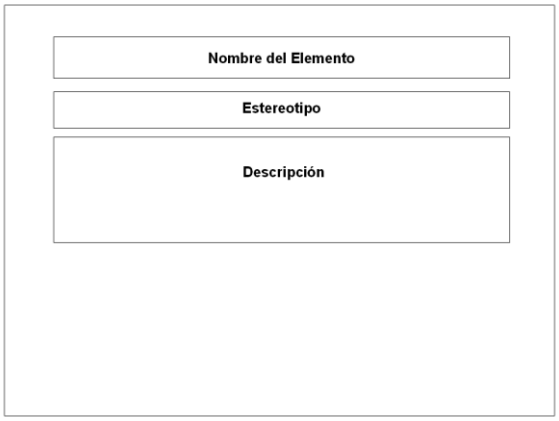


Figura 24: Diseño de reporte 3

Según el orden que lleva la pantalla después del estereotipo se ha ubicado la descripción, un campo debido a la cantidad de información se ha determinado de mayor tamaño que los demás.

El siguiente elemento es el componente antecesor al que se esta trabajando, esta opción puede ser digitada por el usuario o el mismo sistema puede capturarlo automáticamente y así ahorrar el tiempo de digitación y ser mas exacto a la hora de procesar la información, aumentando así la eficiencia del sistema.

A continuación se visualiza la posición de este:

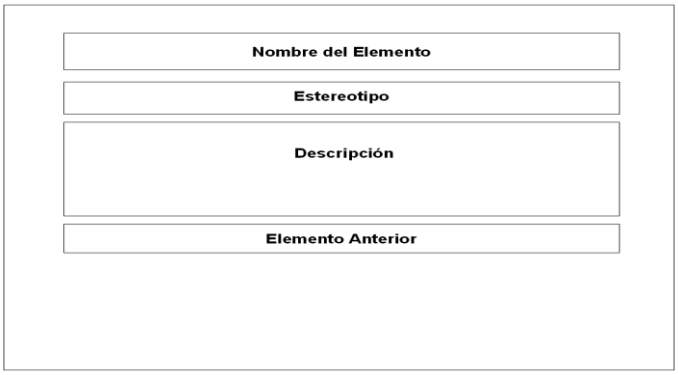


Figura 25: Diseño de reporte 4

En este se almacenan los elementos insertados con anterioridad, es decir que se almacena el objeto del cual procede el que estamos trabajando.

Para finalizar se le introduce el ultimo elemento el cual es la relación que une al elemento anterior con el que se esta modificando de modo que sea mas exacto y mayor eficiencia a la hora de realizar el diagrama.

Al final el sistema quedara así:

Figura 26: Diseño de reporte 5

Siendo este el producto final del diseño del sistema.

Diseño de Reportes

Nombre del Elemento	
Estereotipo del Elemento	
Relación de Procedencia	Elemento de Procedencia
Relación Destino	Elemento Destino
Descripción	

Figura 27: Diseño de reporte final

4.4 Documentación de especificación técnica

A continuación se presentan las diferentes características técnicas y estándares utilizados en la elaboración del editor de diagramas de casos de uso UML.

1. Estándar de Manejo.

El editor de diagramas de casos de uso se identificara con el nombre de ECUSV y los archivos creados en éste se identificarán con la extensión **.ecu,** así por ejemplo Diagrama.ecu.

Al iniciar la aplicación se abre un nuevo documento llamado por default **Diagrama1.ecu**.
Por default al dar guardar el archivo nuevo se direccionará al directorio raíz de la maquina y el usuario decide si lo cambia de posición.

2. Estándar de programación

Los estándares de programación tienen como finalidad poder unificar criterios de programación.
Para la realización de la aplicación se utilizara los siguientes elementos.

Clases Comunes

Las clases comunes son clases que representan la estructura de una tabla, dichas clases se pueden representar como una clase simple o una clase derivada
El nombre de dicha clase esta definido por un prefijo y por el nombre de la tabla que representa.

Por ejemplo:
Se tiene una tabla Elementos

Su clase común será DataElementos.

Las clases poseen como métodos básicos:

- Insertar: Permite crear el comando para la inserción de un nuevo registro y hace las validaciones necesarias antes de realizar un ingreso de datos a la base de datos
- Modificar: Permite crear el comando para la modificación de un registro y hace la validación necesaria antes de actualizar un registro en la base de datos
- Eliminar: Permite la creación el comando para la eliminación de un registro Validando que el registro a eliminar pueda ser eliminado.
- Buscar: Permite crear el comando para realizar la búsqueda de registros.

Las clases tienen como finalidad interactuar con el usuario final. Para nuestro caso estas clases de presentación serán los winForm

El nombre de las clases de presentación esta formado por un prefijo y un nombre

Por ejemplo:
FmMntElementos.vb: Esta clase será utilizada para ingresar datos.
Esta clase poseerá como elementos básicos los siguientes métodos.

- Guardar.
- Eliminar.
- Retornar: El botón estará siempre activo y permitirá retornar al diagrama

Prefijo para el nombre de objetos nativos de .NET

Prefijo	Objeto	Ejemplo
ds	DataSet	dsElemento
dg	DataGrid	dgElemento
dt	DataTable	dtElemento
dr	DataRow	drRegistro
da	DataAdapter	daBuscar
cmm	SQLCommand	cmmInsertar
txt	TextBox	txtCodigo
lbl	Label	lblCodigo
ddl	DropDownList	ddlEstado
lstb	ListBox	lstEstado
cb	CheckBox	cbActivo
rfv	RequiredFieldValidator	rfvCodigo
rbl	RadioButtonList	rblOpciones
rb	RadioButton	rbActivo
rev	RegularExpressionValidator	revNit
cv	CompareValidator	cvValidar
rv	RangeValidator	reCompararFechas
Lb	LinkButton	lbNuevo
ib	ImageButton	ibBuscar
cbl	CheckBoxList	cblForma
lsb	ListBox	LsbValores
dl	DataList	dlValores
hlk	HyperLink	hlkCatalogos
img	Image	imgLista
crv	CrystalReportView	crvResumen
cmb	ComboBox	cmbCombo1

Tabla 7: Prefijos de Programación

Estándares de variables.

A partir de la tabla anterior se presenta los prefijos utilizados a nivel de programación; de ahí que el manejo de variables se realice de la forma descrita a continuación:

- variables del programa

Prefijo<NombreIdentificador>

Ejemplo:

imgActor: para manejar el icono del actor dentro de un diagrama.

txtActorNombre: es el text box q almacena el nombre del actor.

cmbEstereotipo: representa el conjunto de estereotipos.

Estándares de funciones.

Las Funciones se manejan de la siguiente forma:

<NombreFuncion> : es la descripción de la acción que se realiza, se maneja con nombres representativos al objetivo de la función.

Ejemplo:

MoverElemento: realiza el movimiento de un elemento de un lugar a otro dentro del área de trabajo.

GuardarDiagrama: Almacena todos los elementos contenidos en el diagrama.

Estándares de Formularios.

Los diferentes formularios que se utilicen se identifican de la siguiente forma:

Fm<Nombre>. Donde <Nombre> es el identificador del formulario.

Ejemplo:

FmDiagrama: es el formulario que almacena el diagrama.

FmCasoUso: es el formulario que contiene la información de un caso de uso.

CAPITULO V

PROPUESTAS DE IMPLEMENTACIÓN

5.1 Pruebas.

En ésta etapa intervienen solo los desarrolladores, los cuales actúan sobre un código fuente congelado, por lo que están restringidos a agregar nuevas funcionalidades y la imposibilidad de realizar grandes modificaciones a las interfaces gráficas de usuario.

A pesar de que las pruebas no pueden asegurar la ausencia de defectos, ya que sólo se puede demostrar que existen defectos en el software, son parte fundamental antes de entregar el software final.

El esquema general de pruebas finales aplicadas se rige por pruebas de tipo caja negra que estudia la especificación del software, las funciones que debe realizar, las entradas y las salidas esperadas. Se detalla a continuación:

Las pruebas se hacen principalmente en función de los requerimientos determinados. También pruebas técnicas no siempre visibles para el usuario final, que una vez que son superadas pueden permitir la entrega del software finalizado.

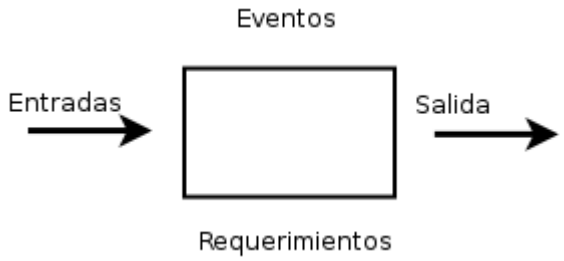


Figura28: Esquema representativo de las entradas al módulo a probar frente a los requerimientos del usuario, sumado a los eventos y análisis de la salida.

Las pruebas realizas en el editor UML han sido aplicadas por estudiantes que cursaron la materia de análisis y diseño de sistemas de distintas universidades de El Salvador que poseen diferentes niveles de conocimientos de los diagramas de casos de uso UML.

5.1.1 Técnicas

La técnica usada para realizar pruebas al editor UML fue la siguiente:

Pruebas por pantallas independientes.

Se realizo la prueba individual de cada pantalla para verificar su correcta funcionabilidad.

No.	Nombre	Entrada	Salida esperada	Salida Real
1	Realizar conexión	Elementos de origen y destino	Conexión de elementos establecida	Conexión establecida
2	Mover Ratón	dirección (v , ,)	Movimiento x, y	Correcto
3	Arrastre de elementos	Elemento y posición inicial	Arrastre a posición final	Correcto
4	Adicionar Elementos	Selección del elemento	Colocación en área de trabajo	Correcto
5	Llenar Propiedades	Hacer clic sobre elemento	Abrir formulario de propiedades	correcto
6	Almacenamiento de propiedades	Ingreso de propiedades	Adición de propiedades al objeto seleccionado	Correcto

Tabla 8: Lista de pruebas generales.

5.1.2 Validación

El software totalmente ensamblado se prueba como un todo para comprobar si cumple los requisitos funcionales y de rendimiento, facilidad de mantenimiento, recuperación de errores, etc.

El editor UML para diagramas de casos de uso cumple con las validaciones planteadas con anterioridad en el diseño[8].

	Actor	Caso de uso	paquete	Relación Asociativa	Relación Dependencia	Relación Generalización
Actor	NO	NO	NO	Si	Si	Si
Caso de Uso	NO	NO	NO	Si	Si	Si
Paquete	NO	NO	NO	Si	Si	Si
Relación Asociativa	Si	Si	Si	NO	NO	NO
Relación Dependencia	Si	Si	Si	NO	NO	NO
Relación Generalización	Si	Si	Si	NO	NO	NO

Tabla4: Validaciones UML

[8] Tabla 4 : Capitulo Diseño del sistema (Pagina 91)

Fuentes de Información

A. Bibliografía.

WHITTEN, Jeffrey. ANÁLISIS Y DISEÑO DE SISTEMAS DE INFORMACIÓN. Editorial Mc Graw Hill, España, 1996

PRESSMAN, Roger. INGENIERÍA DEL SOFTWARE. Editorial Mc Graw Hill.
BOOCH / JACOBSON / RUMBAUGH. THE UNIFIED MODELING LANGUAGE USER GUIDE. Editorial Addison – Wesley

[BOOC, 1999] BOOCH, G / RUMBAUGH, J / JACOBSON, I. "EL LENGUAJE UNIFICADO DE MODELADO". Ed. Addison Wesley Iberoamericana, Madrid, 1999

[LARM, 1999] LARMAN, C. "UML Y PATRONES, INTRODUCCIÓN AL ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS". Ed. Prentice Hall, México, 1999

CRAIG Larman, APPLYING UML AND PATTERNS, Editorial Prentice Hall, second edition, New Jersey2001

B. Sitios de Internet.

- http://www.vico.org/ Vico Virtual Campus, Institución de ELearning
- http://www.dsic.upv.es/~uml/ Universidad Politécnica de Valencia, Curso: "Desarrollo de Software Orientado a Objetos usando **UML**
- http://www.objectsbydesign.com/tools/umltools_byPrice.html Sitio orientado para programadores que trabajan con lenguaje orientado a objetos.
- http://www.planetacodigo.com Sitio agregador de weblogs sobre programación y desarrollo en castellano.

GLOSARIO

A

Abstracción:

La característica esencial de una entidad que la distingue de otra padre o entidad.

Activación:

Ejecución de una acción.

Actor:

Agente o sistema externo que interactúa con el sistema, Entra, manipula o recibe información, puede ser una persona o aplicación.

Artefacto:

Una pieza física en la ejecución, que es usada o producida por el software en un proceso.

Asociación:

Relación semántica entre dos o más clasificadores que especifica un tipo de conexión entre ellos.

Análisis de requisitos:

Proceso de estudio de las necesidades del usuario para conseguir una definición de los requisitos del sistema o del software.

C

Caso de Uso:

Denota actividad dentro del diagrama.

Codificación:

Proceso de descripción de un programa de ordenador en un lenguaje de programación.

Componente:

Una de las partes que forman un sistema. Un componente puede ser hardware, software, y puede a su vez subdividirse en otros componentes.

D

Descripción del sistema:

Documento orientado al cliente que describe las características del sistema desde el punto de vista del usuario final. El documento se utiliza para coordinar conjuntamente los objetivos del sistema del usuario, cliente, desarrollador e intermediarios.

Diseño:

Proceso de definición de la arquitectura, componentes, interfaces y otras características de un sistema o de un componente**.**

Diseño arquitectura:

Proceso que define una colección de componentes de software y hardware junto con sus interfaces, para definir el marco de desarrollo de un sistema.

Diseño Detallado:

Proceso de definición y ampliación del diseño preliminar de un sistema o de un componente hasta un grado de detalle suficiente para llevar a cabo la implementación.

Disponibilidad:

El grado con el que se mide la accesibilidad de un sistema o de un componente cuando es necesario su uso. Suele expresarse en términos de probabilidad.

E

Escalabilidad:

Facilidad con la que un sistema o un componente pueden modificarse para aumentar su capacidad funcional o de almacenamiento.

Especificación de interfaz:

Documento que especifica las características de interfaz de un sistema o de un componente.

Especificación de requisitos de software:

Documentación de requisitos fundamentales (necesarios, esenciales e indispensables) de funcionalidades, rendimiento, restricciones y atributos del software, y sus interfaces externas. Su acrónimo inglés es SRS.

Elemento de configuración:

Parte de un desarrollo de software (planes, software, documentación de especificación y diseño, manuales, etc.) tratada como una unidad independiente en el proceso de gestión de configuración**.**

I

Implementación:

Proceso de transformación de un diseño en componentes de hardware, software o de ambos.

Ingeniería del software:

Aplicación de procesos sistemáticos y disciplinados para el desarrollo, operación y mantenimiento de software.

Interfaz:

Componente de hardware o software que conecta dos o más componentes con el propósito de transmitir información entre ellos.

Interfaz de usuario:

Interfaz que permite la comunicación entre un usuario y un sistema, o los componentes de un sistema**.**

M

Manual Usuario:

Documento que contiene la información necesaria para obtener de un sistema o de un componente los resultados deseados. Nota: se establece diferencia entre un manual de operador y un manual de usuario, cuando en el sistema hay funciones propias de operación (cambio de discos o cintas, mantenimiento de base de datos, etc.) diferenciadas de las de uso normal del sistema para realizar las funciones que le son propias.

Metodologías Ágiles:

Estrategias de desarrollo de software que promueven prácticas que son adaptativas en vez de predictivas; centradas en las personas o los equipos, iterativas, orientadas hacia la funcionalidad y la entrega, de comunicación intensiva y que requieren implicación directa de cliente.

O

OO (Orientado por objetos)

Enfoque para el desarrollo de sistemas de software que representa el dominio de aplicación de forma natural y directa basándose en los objetos que se implican en dicho dominio.

OOA

Análisis orientado por objetos. Método de análisis que examina los requisitos desde la perspectiva de clases y objetos encontrados en el vocabulario del dominio del problema.

OOP

Programación orientada por objetos. Método de implementación de los programas que los organiza como grupos cooperativos de objetos, cada uno de los cuales representa instancias de una clase, que a su vez forman parte de una jerarquía a través de relaciones de herencia.

P

Prototipo:

Versión preliminar de un sistema que sirve de modelo para fases posteriores.

R

Requisito:

Condición o facultad que necesita un usuario para resolver un problema.

RUP

Proceso de Ingeniería del Software que proporciona un enfoque disciplinado para asignar tareas y responsabilidades en las organizaciones de desarrollo de software. Se trata de un proceso integrado en un producto, desarrollado y mantenido por Racional Software, e integrado en su conjunto de herramientas de desarrollo. Se encuentra disponible a través de IBM. //Inglés: Rational Unified Process//

S

Sistema:

Conjunto de procesos, hardware, software, instalaciones y personas necesarios para realizar un trabajo o cumplir un objetivo.

Software:

Los programas de ordenador, procedimientos, y opcionalmente la documentación y los datos asociados que forman parte de un sistema.

U

UML:

UML (Lenguaje Unificado de Modelado) es un lenguaje gráfico para visualizar, especificar, construir y documentar los componentes de un sistema software. UML permite tanto la especificación conceptual de un sistema como la especificación de elementos concretos, como pueden ser las clases o un diseño de base de datos.

V

Validación:

Confirmación mediante examen y aportación de pruebas objetivas de que se cumplen los requisitos concretos para un uso determinado.

X

XP:

Extreme Programming o programación extrema**.**

ANEXOS

Anexo 1.

A continuación se define el instrumento que se utilizó y se distribuyó entre la población seleccionada:

Universidad Don Bosco
Escuela de Computación

Diagramas de Casos de Uso con el Lenguaje UML

Objetivo. Determinar la utilización de los Diagramas de Casos de Uso con el Lenguaje UML

1. ¿Desarrolla la etapa de Análisis y diseño de sistemas en la creación de aplicaciones?
   Si No algunas veces nunca
2. ¿Conoce UML?
   Si No no sabe

3. Qué significan las siglas UML

a) Lenguaje de Modelamiento Unificado
b) Unión lenguajes modelados
c) Unificación de Modelos Lingüísticos
d) Ninguno de las anteriores

4. ¿Cuál de los diagramas UML conoce?

Casos de Uso	Distribución
Clases	Actividad
Objetos	Estados
Componentes	Colaboración
Secuencia	

5. ¿Cual de los diagramas utiliza con más frecuencia?

Diagrama de:

Casos de Uso	Distribución
Clases	Actividad
Objetos	Estados
Componentes	Colaboración
Secuencia	

6. ¿Qué es un Diagrama de Casos de Uso?
    a) representación de actividades de un proyecto
    b) Representación de los requerimientos para un sistema
    c) Recolección de datos

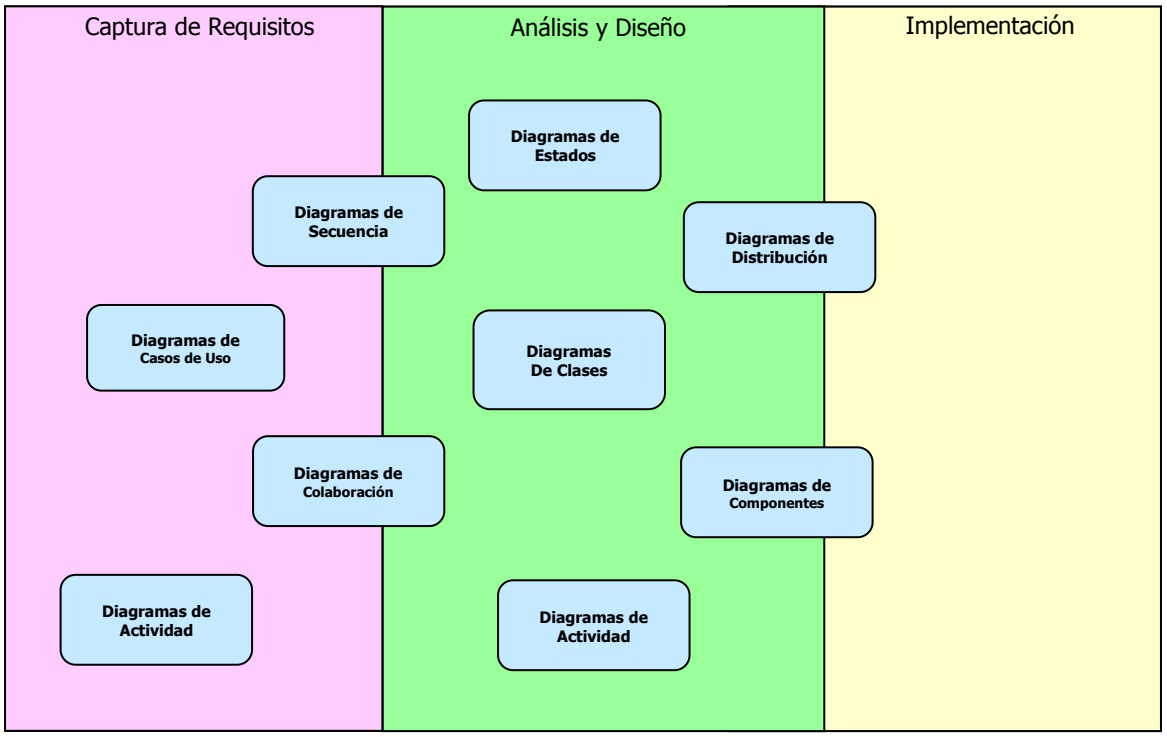
7. ¿Cuáles son los elementos de un diagrama de caso de Uso?
    a) caso de uso, actividad, actor
    b) actor , caso de uso, relaciones, paquetes
    c) caso de uso, relaciones, entidades

8. ¿Qué tipos de relaciones se utilizan en los diagramas de casos de uso?
    a) asociación, generalización, recursividad
    b) asociación, generalización, dependencia y herencia
    c) herencia, distribución, colaboración

9. ¿Conoce los elementos básicos de los diagramas de casos de uso?
    Si NO
10. ¿Ha utilizado alguna herramienta para construcción de diagramas de Casos de Uso?
    Si NO

Anexo 2:

Clasificación de los diagramas UML en el desarrollo de un sistema.



Anexo 3.

Requerimientos y Casos de Uso

Fig. A

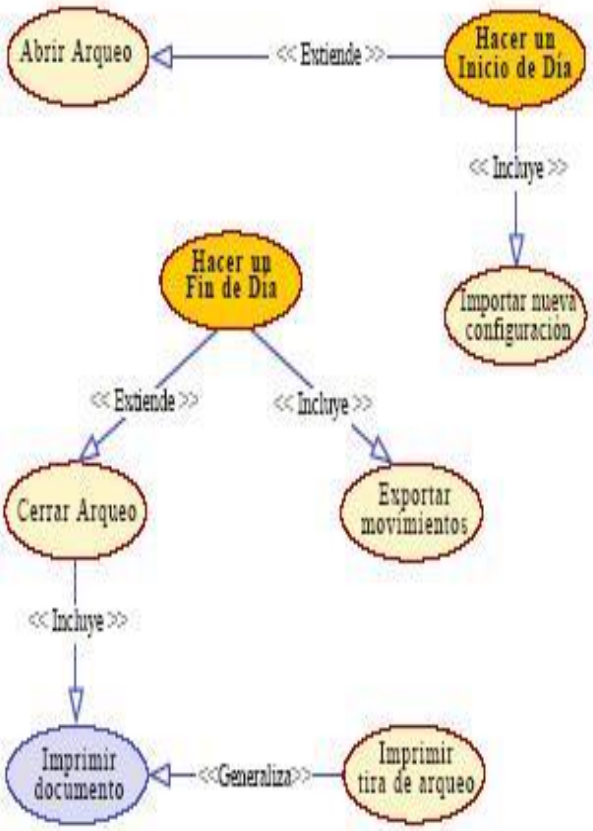
En la Figura A se muestra la utilidad de los Diagramas de Casos de Uso en las diferentes etapas de la creación de un proyecto describiendo los distintos procesos que se presentan a la hora de elaborarlo.

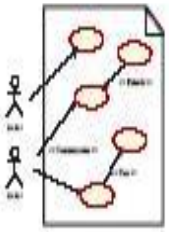
Anexo 4.

Listado de estereotipos.

Aplicado a	Tipo
Actor	Elemento
Paquete	Paquete
Caso de Uso	Elemento
Colaboración	Elemento
Clase	Elemento
Tabla	Elemento
Componente	Elemento
Nodo	Elemento
Objeto	Elemento
Secuencia	Elemento
Entidad	Elemento
Elemento guia	Elemento
Requerimiento	Elemento
Estado	Elemento
Actividad	Elemento
Interface	Elemento
Evento	Elemento
Cambio	Elemento
Hiperlink	Elemento
Atributo	Elemento
Operación	Operación
Asociación	Conector
Fin de asociación	Fin asociación
Generalización	Conector
Dependencia	Conector
Objeto de flujo	Conector
Estado de flujo	Conector
Nodo inicio	Elemento
Nodo de paro	Elemento
Nota	Elemento
Decisión	Elemento
agregación	Conector

Ventajas del modelo Use Case





Funcionalidad
Diagramas de Casos de Uso

Piezas de funcionalidad reutilizables en diferentes procesos de negocio

1. Lenguaje de comunicación entre usuarios y desarrolladores.

2. Comprensión detallada de la funcionalidad del sistema.

3. Acotación precisa de las habilitaciones de los usuarios.

4. Gestión de riesgo más eficiente para gobernar la complejidad.

5. Estimación más exacta para determinar tiempo, recursos y prioridades en la dosificación de esfuerzo de desarrollo.

6. Fiel trazabilidad para verificar la traducción de requerimientos en código ejecutable.

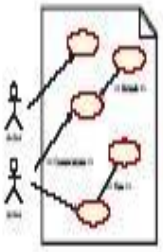
7. Mayor control para mantener las sucesivas revisiones de los programas.

8. Certificación contractual Cliente-Desarrollador.

9. Documentación orientada al usuario: Helps - Manual de Procedimientos - Reglas de Negocio.

10. Documentación orientada al administrador del sistema: Soporte de Mantenimiento.

Man ejo



Funcionalidad
Diagramas de Casos de Uso

Especificación de un Caso de Uso

L Límites: Cuando empieza y cómo termina el Caso de Uso.

I Interacciones: Comportamiento de Actores y Sistema. Acción-Reacción dentro del Caso de Uso.

M Masa: Conjunto de Objetos e Interfaces que requiere el Caso de Uso.

I Índice de escenarios: Flujo principal de eventos y secuencia de variaciones posibles dentro de un Caso de Uso.

T Tribulaciones: Contingencias probables que pueden afectar al flujo de los eventos y son excepciones del Caso de Uso.

Propósito - Regla de Negocio -	Precondiciones	Activación	Flujo Principal	Variaciones	Excepciones
Cerrar un periodo de movimientos de caja para obtener una situación real de dinero en efectivo y en documentos.	• Arqueo abierto • Actor habilitado • TPV abierto • TPV habilitado	• A discreción de un Actor habilitado	1. Sistema *requiere confirmación de cierre* de arqueo. 2. Sistema *comprueba la configuración* de TPV para identificar tipo de cierre. 3. Sistema *calcula los totales teóricos* para cada forma de cobro. 4. Actor *introduce totales reales* para cada forma de cobro. 5. Sistema *registra el cierre*, importes reales para cada forma de cobro y descuadres. 6. Sistema *imprime el documento* "Tira de Arqueo".	a. Si Actor decide *aparcar* el cierre de arqueo, sistema activa UC Aparca_arqueo y termina el UC. a. Cierre de *arqueo de tipo abierto*: • Actor decide el momento de imprimir el documento "Tira de Arqueo". b. Cierre de *arqueo de tipo ciego*: • Sistema no muestra los totales teóricos para cada forma de cobro.	a. Si el servidor está off-line, *sistema informa al usuario*, termina el UC manteniendo el arqueo abierto. a. Si impresora está off-line, *sistema informa al usuario* y le *pide confirmación* para continuar.

Manual de usuario

EDITOR DE CASOS DE USO – ECUSV

Herramienta para la elaboración de Diagramas de Casos de Uso.

ÍNDICE GENERAL

Introducción al Editor de Diagramas de Casos de Uso

ECUSV

Casos de Uso

El ECUSV es una útil herramienta que le ayuda a conocer y realizar diagramas UML para casos de uso su construcción y elementos que lo forman.

Simulando Casos de Uso

El ECUSV es una herramienta para la modelación de Diagramas de casos de uso.

Con el uso del ECUSV se pueden implementar y desarrollar diagramas de casos de uso según la filosofía UML.

Este documento presenta los pasos necesarios para utilizar el ECUSV e inducir al usuario en los procesos a seguir para obtener un mayor provecho del esta útil herramienta.

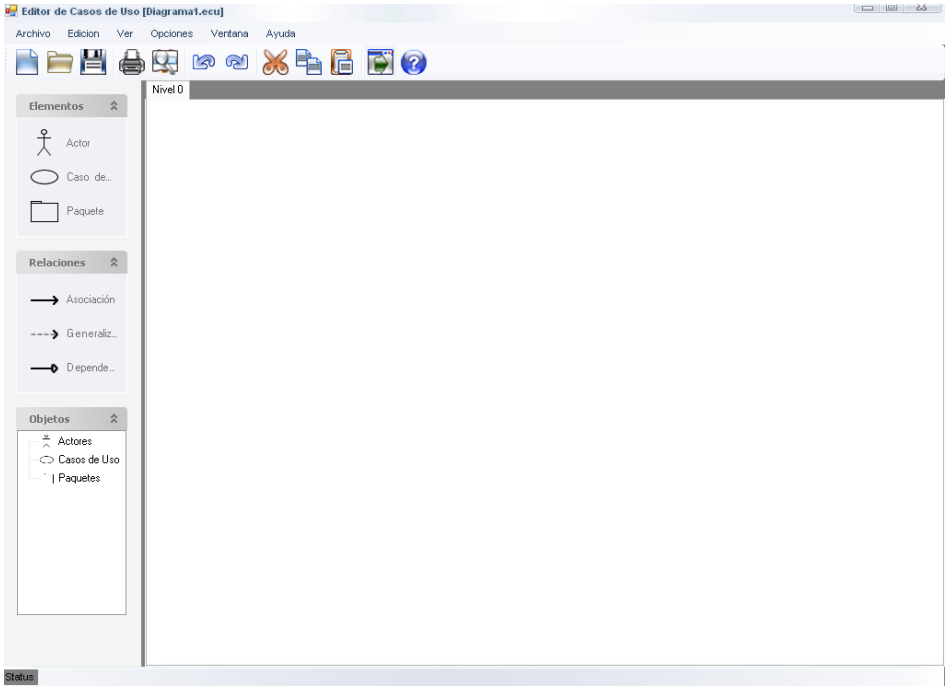


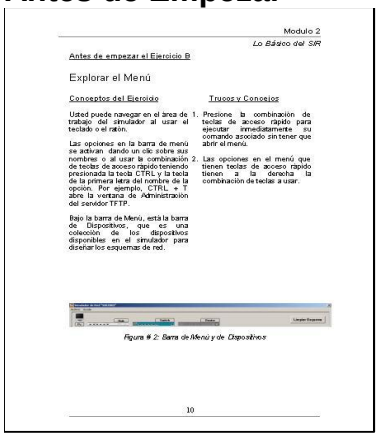
Figura # 1: Interfaz del ECUSV

Como usar este manual

Página Titular

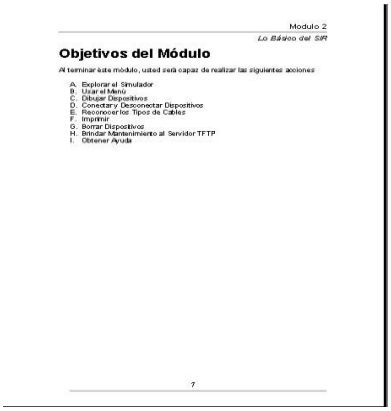
Se ha escrito el manual en módulos de capacitación que presentan al usuario conceptos y ejercicios. Cada módulo empieza con su propio título que identifica el inicio de este, y resume los conceptos tocados en el módulo.

Antes de Empezar


Modulo 2
Lo Básico del SIR
Antes de empezar el Ejercicio B
Explorar el Menú
Conceptos del Ejercicio
Trucos y Consejos
Usted puede navegar en el área de trabajo del simulador al usar el teclado o el ratón.
Las opciones en la barra de menú se activan dando un clic sobre sus nombres o al usar la combinación de teclas de acceso rápido teniendo presionada la tecla CTRL y la tecla de la primera letra del nombre de la opción. Por ejemplo, CTRL + T abre la ventana de Administración del servidor TFTP.
Bajo la barra de Menú, está la barra de Dispositivos, que es una colección de los dispositivos disponibles en el simulador para diseñar los esquemas de red.
1. Presione la combinación de teclas de acceso rápido para ejecutar inmediatamente su comando asociado sin tener que abrir el menú.
2. Las opciones en el menú que tienen teclas de acceso rápido tienen a la derecha la combinación de teclas a usar.
Figura # 2: Barra de Menú y de Dispositivos
10

Cada ejercicio es precedido por una explicación de los conceptos. Hay una lista de recomendaciones y atajos que posiblemente le sean útiles, incluidos para su referencia.

Los Objetivos


Modulo 2
Lo Básico del SIR

Objetivos del Módulo

Al terminar éste módulo, usted será capaz de realizar las siguientes acciones:

A. Explorar el Simulador
B. Usar el Menú
C. Dibujar Dispositivos
D. Conectar y Desconectar Dispositivos
E. Reconocer los Tipos de Cables
F. Imprimir
G. Borrar Dispositivos
H. Brindar Mantenimiento al Servidor TFTP
I. Obtener Ayuda

7

Los objetivos del módulo son mostrados claramente en una hoja independiente. Generalmente cada objetivo corresponde a un ejercicio del módulo y son identificados por letras para su referencia.

Ejercicios. Paso a Paso

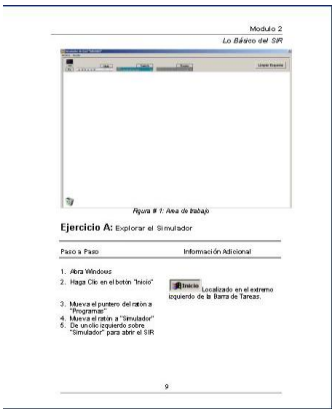

Modulo 2
Lo Básico del SIR

figura # 1: Area de trabajo

Ejercicio A: Explorar el Simulador

Paso a Paso	Información Adicional
1. Abra Windows	
2. Haga Clic en el botón "Inicio"	Localizado en el extremo izquierdo de la Barra de Tareas.
3. Mueva el puntero del ratón a "Programas"	
4. Mueva el ratón a "Simulador"	
5. De un clic izquierdo sobre "Simulador" para abrir el SIR	

9

Los ejercicios modulares son presentados en un formato diseñado para todo tipo de usuarios. La columna izquierda provee instrucciones paso a paso. La columna derecha complementa los pasos, y provee información adicional, que ofrece un entendimiento más profundo del ejercicio entero.

Editor Casos de Uso

Modulo

1

La Instalación

La instalación del Editor UML para diagramas de casos de uso con lleva un serie de pasos que incluyen recursos mínimos de Hardware y Software.

Objetivos del Módulo

Al terminar éste módulo, usted será capaz de realizar las siguientes acciones

A. Identificar los requisitos necesarios para la instalación del ECUSV
B. Instalar el ECUSV

A. Requisitos necesarios para la instalación del ECUSV

a. Hardware

En la tabla # 1 se detallan los recursos recomendados y mínimos para instalar el ECUSV.

Recomendado		**Mínimo**	
Procesador	Pentium 300	Procesador	Pentium 200
RAM	32Mb	RAM	16Mb
Video	8Mb	Video	4Mb
CD-ROM	16x	CD-ROM	16x
Color	Verdadero 24bits	Color	Verdadero 16bits
Disco Duro	4Gb	Disco Duro	2GB

Tabla # 1: Requisitos de Hardware para instalación del ECUSV

b. Software

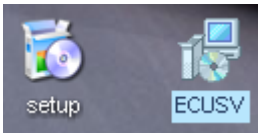
- Se requiere un Sistema Operativo Windows 95, 98 y Windows 2000 Profesional o Server.
- La resolución de la pantalla debe establecerse en 1024 x 768 píxeles, para una visualización optima.

B. Instalación del ECUSV

Paso a Paso

Información Adicional

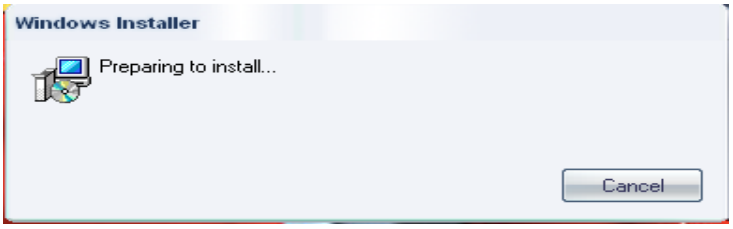
1. Con un doble clic ejecute el instalador del ECUSV



Paso a Paso

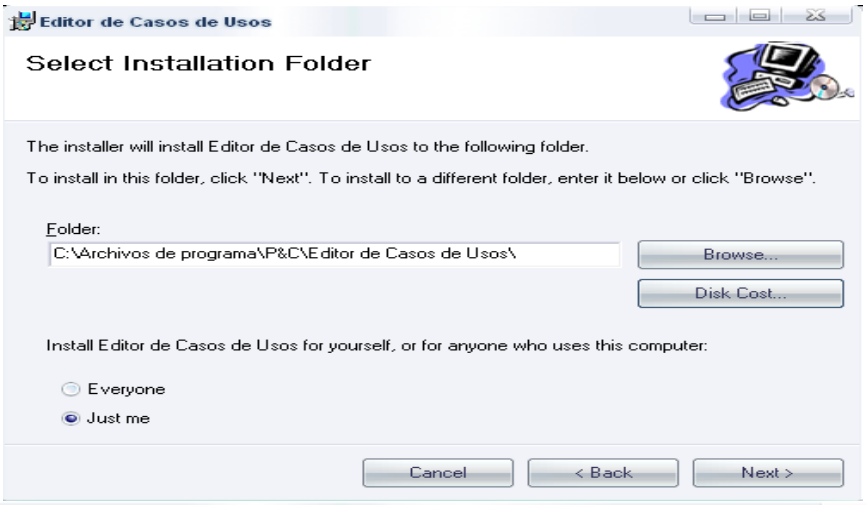
Información Adicional

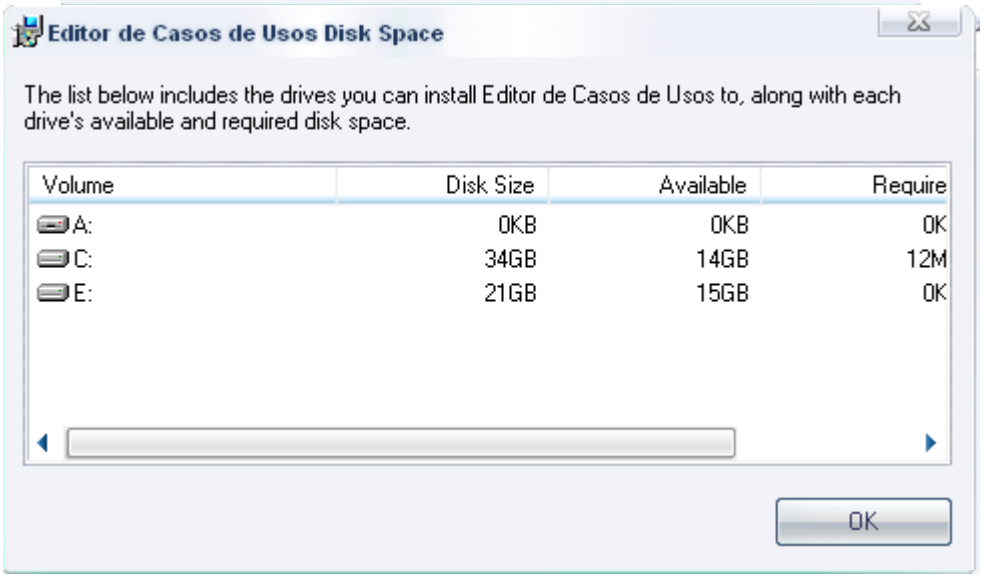
2. Cuando aparezca el ayudante del instalador, presione el botón “Next” para iniciar la instalación



3. Seleccione la ruta en donde se instalará el simulador, por defecto el

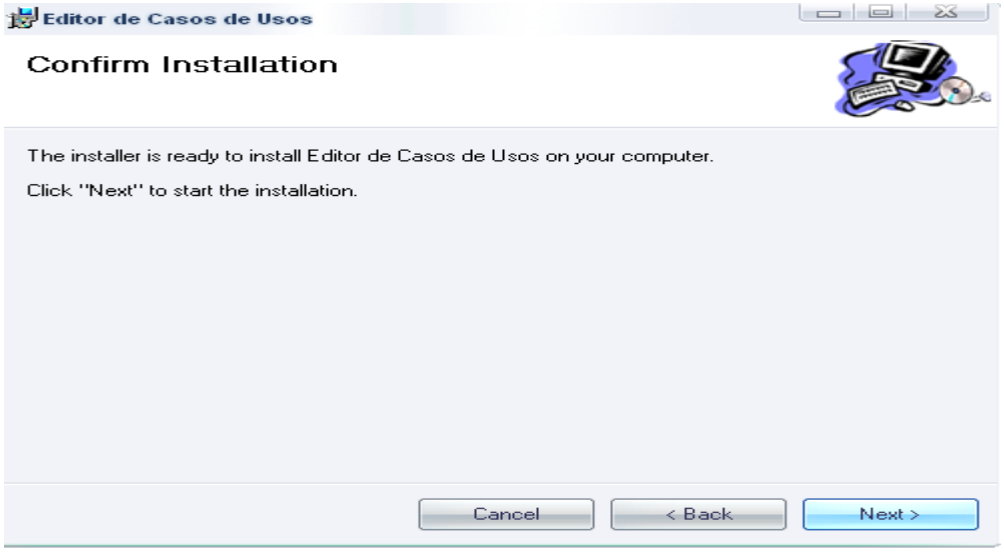
ECUSV se instala en la carpeta ECUSV dentro del directorio Archivos de Programas. Presione “Next” para continuar



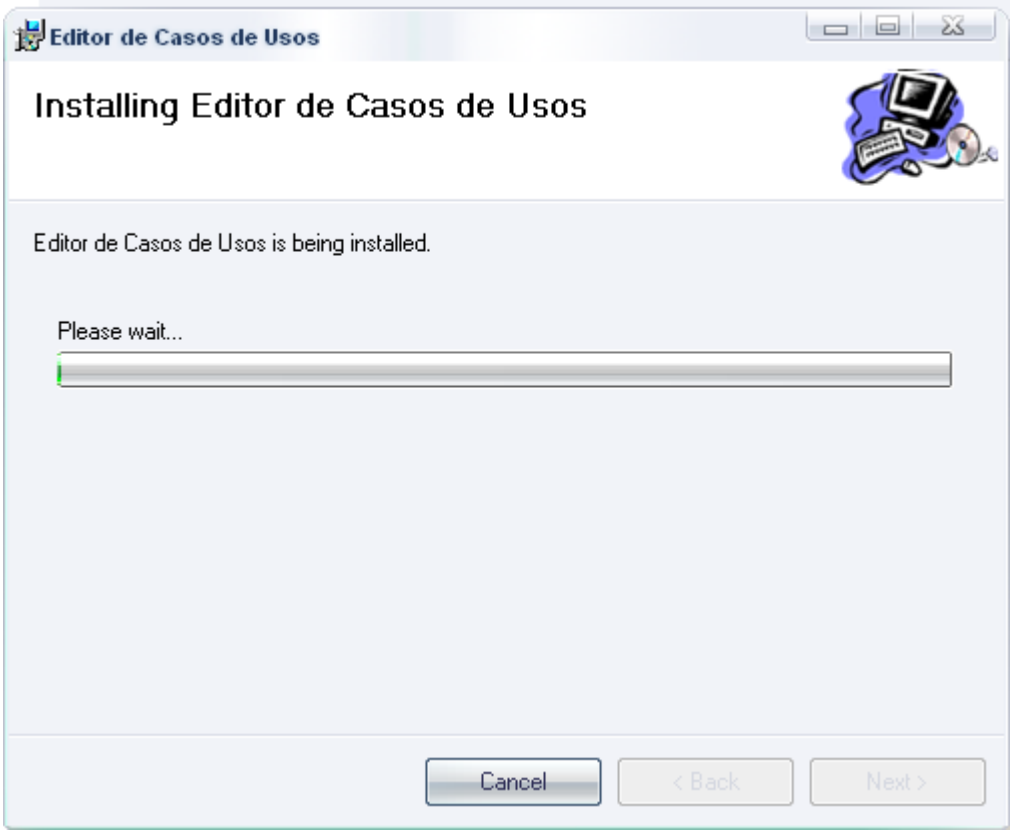


Paso a Paso

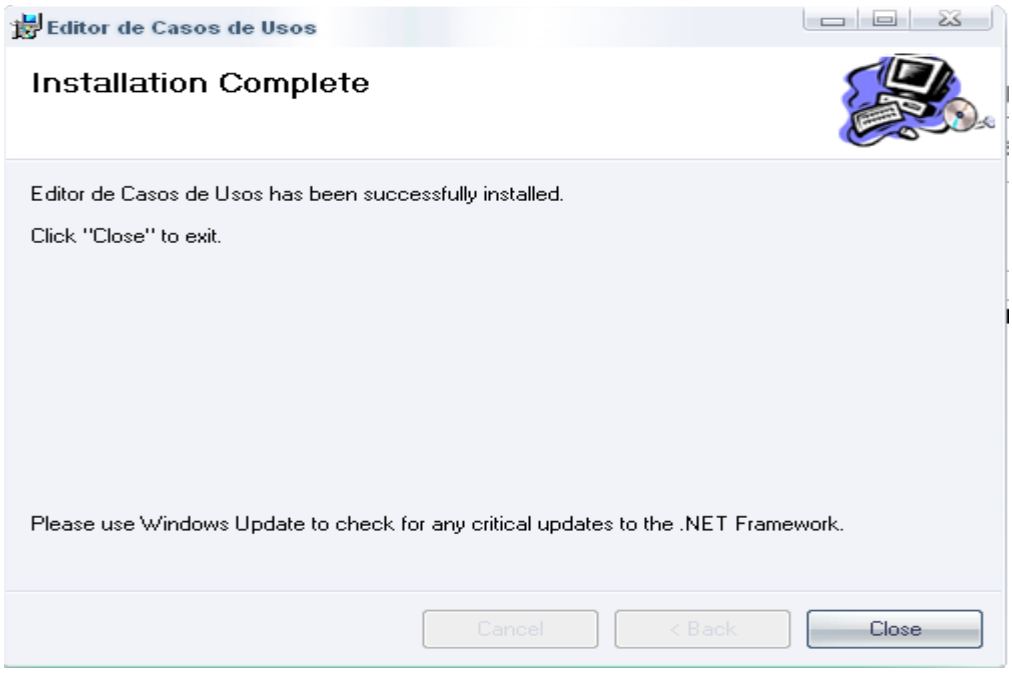
4. El simulador esta listo para se instalado, presione “Next”



5. El ECUSV iniciará el proceso de instalación.



6. Presione “Close” para finalizar con la instalación



7. Seleccione con un clic el botón “Inicio” de la barra de su computador

8. Seleccione con un clic izquierdo el icono del ECUSV que esta en la barra de programas de su computador

Editor Casos de Uso

Modulo

2

Lo Básico del ECUSV

El Editor está compuesto por una serie de ventanas fácilmente accedidas, las cuales permiten al usuario la navegación y uso de sus opciones.

Objetivos del Módulo

Al terminar éste módulo, usted será capaz de realizar las siguientes acciones

A. Explorar el Editor
B. Conocer la Utilidad del editor
C. Iniciar la Aplicación
D. Utilizar el Menú
E. Obtener Ayuda

Antes de empezar el Ejercicio A

Explorar el Editor

Conceptos del Ejercicio

La ventana principal del Editor está formada por dos partes:

- Barra de Menú y Barra de Elementos
- Área de trabajo

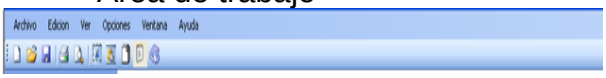


Figura 2. Barra de Menú

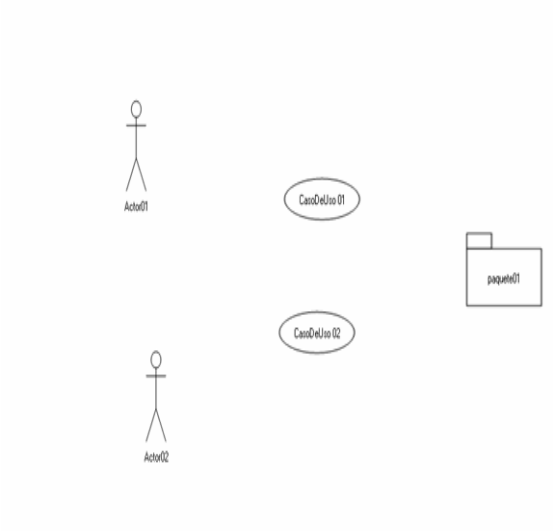


Figura 3. Área de trabajo

Trucos y

1. Utilice el ratón para moverse dentro del Editor, dibujar y seleccionar dispositivos.

Ejercicio A: Explorar el Editor

Paso a Paso

1. Abra Windows
2. Haga Clic en el botón ”Inicio”

Localizado en el extremo izquierdo de la Barra de Tareas.

3. Mueva el puntero del ratón a “Programas”
4. Mueva el ratón a “Editor”
5. De un clic izquierdo sobre “ECUSV” para abrir el Editor

B. Conocer Utilidad del Editor

Descripción General

a. El ECUSV es una aplicación completa, sencilla y funcional para la creación de los casos de uso

2. El ECUSV es una aplicación enfocada en el diseño de sistemas desde el punto de vista del usuario del sistema a modelar

3. El ECUSV permite que tanto un desarrollador experto como un novato pueda realizar sus casos de uso de una manera fácil, ágil y correcta.

C. Iniciar Editor

Conceptos del Ejercicio

Para iniciar la aplicación es necesario localizar el icono de la aplicación, dar doble clic.

Al comenzar la aplicación tendremos por defecto una diagrama inicial donde se podrá iniciar a trabajar.

Trucos y

2. Utilice el ratón para moverse dentro del Editor, dibujar y seleccionar dispositivos.

Antes de empezar el Ejercicio D

Explorar el Menú

Conceptos del Ejercicio

Usted puede navegar en el área de trabajo del editor al usar el ratón.

Las opciones en la barra de menú se activan dando un clic sobre sus nombres o al usar la combinación de teclas de acceso rápido teniendo presionada la tecla CTRL y la tecla de la primera letra del nombre de la opción. Por ejemplo, CTRL. + A abre la ventana.

Bajo la barra de Menú, está la barra de Elementos, que es una colección de los elementos disponibles en el simulador para diseñar los esquemas de red.

Trucos y

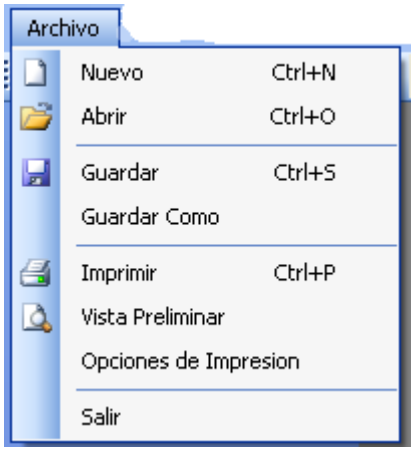
1. Presione la combinación de teclas de acceso rápido para ejecutar inmediatamente su comando asociado sin tener que abrir el menú.
2. Las opciones en el menú que tienen teclas de acceso rápido tienen a la derecha la combinación de teclas a usar.

Ejercicio D: Explorar el Menú

Paso a Paso

1. Abra el Editor.
2. Haga clic en “Archivo” en la barra de Menú para ver las opciones que lo conforman.

Se abre el menú Archivo



3. Haga clic en “Salir” o en el botón de cerrado de ventana

El botón de cerrado de ventana esta localizado en la esquina superior derecha de cada ventana.

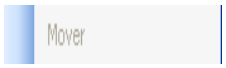
4. Presione CTRL.+ A para abrir la ayuda del Editor sin tener que abrir el menú

A continuación se detalla la pantalla y se describe cada uno de sus componentes, su funcionamiento y el papel que cumple dentro del sistema.
A continuación se presentaran los contenidos de los diferentes menús que el sistema va a contener.

Menú Básico.

En el menú básico el usuario tendrá la opción de llevar a cabo tareas fundamentales, dichas opciones son utilizadas para la hoja donde se llevaran a cabo los diagramas, estas funcionalidades se detallaran a continuación:

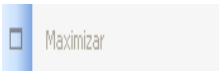
La opción restaurar le sirve al usuario para regresar al tamaño original el área de trabajo de la aplicación, es decir que si la hoja donde se realiza el diagrama esta maximizada la retorna a su tamaño original (normal).



Esta opción es para poder movilizar el área de trabajo en todo el espacio designado para esta cabe mencionar que esta opción se activa solamente cuando el botón ha sido restaurado a su tamaño original.

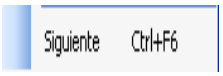
La pestaña tamaño se utiliza para variar como su nombre lo dice funciona para cambiar las dimensiones del sector destinado para desarrollar los diagramas, al igual que la opción anterior solo se activa cuando dicha área ha sido restaurada.

Minimiza el área de trabajo situándolo en la parte inferior del sector destinado para esta.



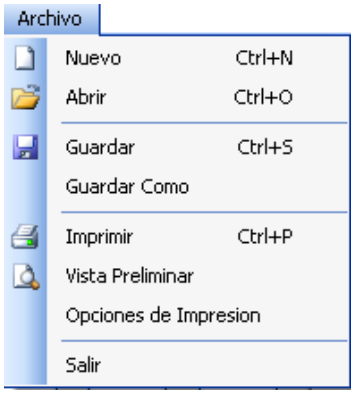
Permite que el área de trabajo alcance todo el espacio definido para esta, le asigna todo el sector determinado para dicho componente.

Cierra el área de trabajo, esta opción se puede llevar a cabo con las teclas que se detallan a continuación ctrl. + F4.

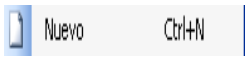


Si el usuario tiene 2 o mas áreas de trabajo (ventanas) abiertas esta opción le permite navegar a través de ellas, es decir, permite ver el contenido de cada una de las páginas donde se han desarrollado los diagramas de casos de uso.

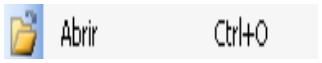
Menú Archivo.



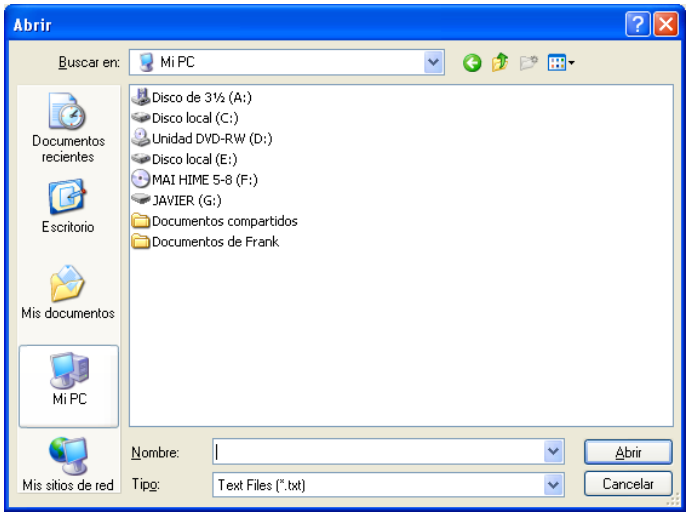
En el menú archivo se despliegan opciones de control como se pueden apreciar en el diagrama, que le permiten al usuario una forma diversa y amigable a la hora de trabajar, cada uno de sus componentes se verán a continuación:



Permite al usuario agregar un área nueva donde este pueda llevar a cabo sus diagramas, le permite al usuario agregar nuevos sectores de trabajo sin la necesidad de cerrar el sector anterior, dicha opción se puede activar además con la combinación de teclas ctrl.+N siendo esta muy practica a la hora de estar diagramando.

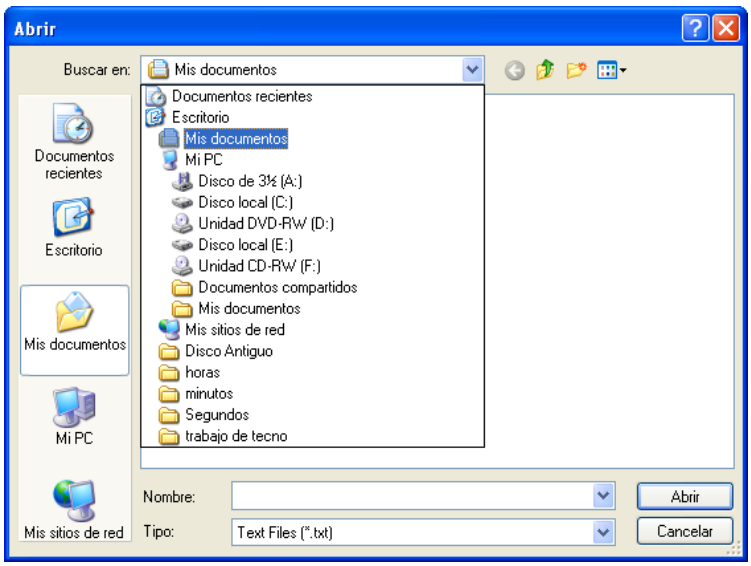


Le permite a la persona que este trabajando con la herramienta abrir archivos previamente almacenados, además de presentarles una pantalla que viene a facilitar la búsqueda de dichos archivos, la pantalla es la siguiente:



La cual consta de:

En esta barra se puede buscar las diferentes carpetas donde se da la posibilidad de encontrar el archivo que estamos buscando, al presionar el botón  , el cual nos sirve para desplegar los otros directorios y así poder hacer búsquedas como lo podemos apreciar en la siguiente imagen:



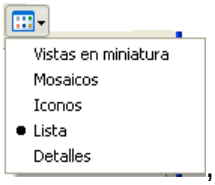
Entonces ahí se selecciona la carpeta deseada y con un clic se abre el contenido de esta.

Otros componentes son los siguientes botones:

El primer botón  sirve para retornar al área donde se encontraba anteriormente, es decir que si un usuario se encontraba en Mis Documentos y se trasladaba a Mi PC por ejemplo con este botón se puede regresar al presionarlo al área anterior en este caso Mis Documentos.

Otro de los botones que aparecen en dicha barra es  el cual permite navegar a un nivel superior en las carpetas, es una herramienta funcional porque le da la facilidad al usuario de acceder a carpetas de nivel superior de forma más fácil.

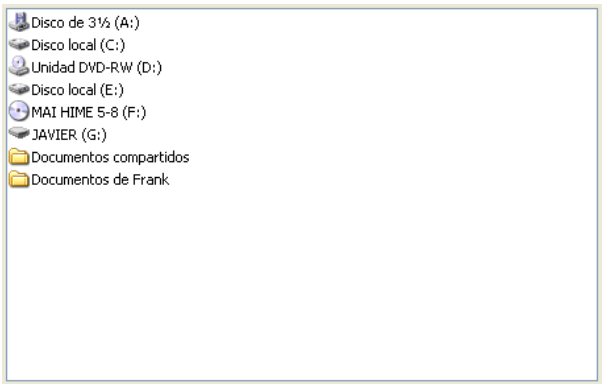
El elemento siguiente es  que permite de forma sencilla agregar una nueva carpeta en el directorio que se esta trabajando, es importante destacar que dicho elemento se desactiva al momento de estar en el directorio de Mi Pc.

Para finalizar se presenta  , dicho componente que al ser desplegado se puede apreciar de la siguiente forma 

, podemos destacar que este control nos permite visualizar de diferentes formas las carpetas y archivos de un directorio, estas son:

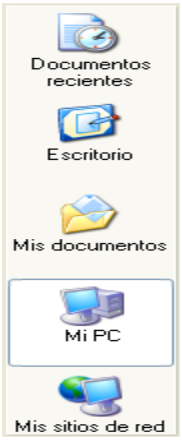
- Vistas en Miniatura
- Vistas en Mosaicos
- Iconos
- Lista
- Detalles

Hay otras formas que vienen a complementar la pantalla de abrir, una de ellas es el área donde se puede apreciar el contenido de un directorio como se puede visualizar en la imagen:



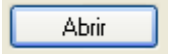
Esta área es la destinada para contener los elementos que componen un directorio, en este sector es donde se aplican los cambios de los botones antes mencionados.

El editor en el componente mencionado presentara la facilidad de activar algunos ficheros principales de manera inmediata para ello se ha elaborado la siguiente barra:



En esta parte 

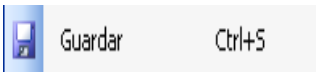
el usuario podrá digitar el nombre del archivo y así agilizar la búsqueda además de presentar la opción para la selección de los diagramas en base a su extensión.

Para finalizar se encuentran los botones de abrir 

y el cual como su nombre lo dice permite presentar el documento seleccionado y el de cancelar

que cierra este componente y retorna la pantalla principal sin ninguna alteración.

La opción guardar:

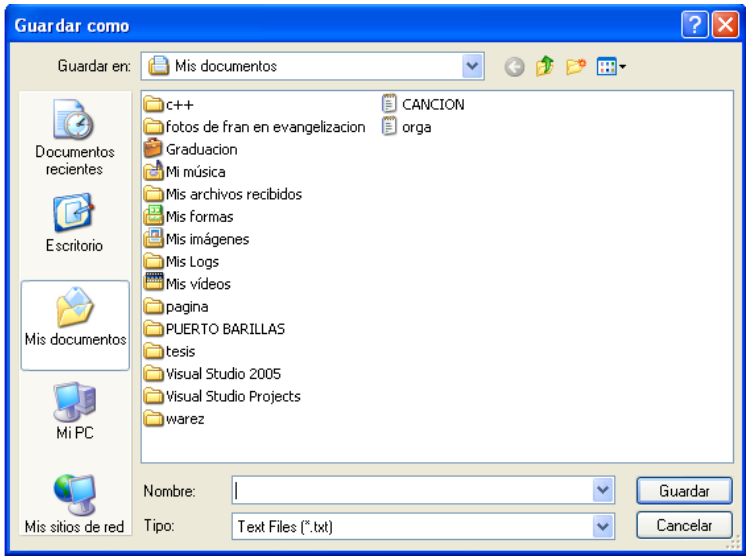


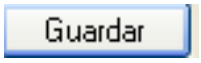
Permite como su nombre o dice almacenar el diagrama que se esta elaborando cuando es la primera vez que se utiliza dicho componente aparece una ventana similar a la de abrir que se detallara mas adelante cuando se comente de la opción guardar como, depuse al utilizarlo nuevamente solo lo almacenara de manera automática, dicha opción esta disponible además en la barra de herramientas a través del botón  que realiza las mismas funciones que el componente mencionado.

Otra de las alternativas que nos ofrece el sistema es la facilidad Guardar Como 

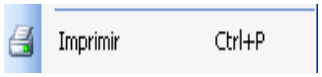
.

Al seleccionarla despliega una pantalla parecida a la que despliega el botón abrir en la cual se puede seleccionar la ubicación en donde se desee guardar el diagrama, la pantalla es la siguiente:



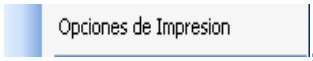
En esta pantalla las funcionalidades de los botones es similar a la de la de la opción abrir con la única diferencia que esta pantalla tiene un botón Guardar 

que almacena el registro en el directorio que el usuario ha seleccionado.

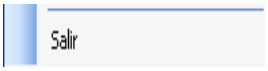
La siguiente opción del menú Archivos es la de imprimir 

con la cual se activa la función de imprimir el documento.

Otra forma de acceder a dicha funcionalidad es a través del botón  con la diferencia que en esta se imprime todo el documento automáticamente sin presentar los detalles de forma mas especifica como en la opción anterior.

La penúltima opción es la de opciones de impresión 

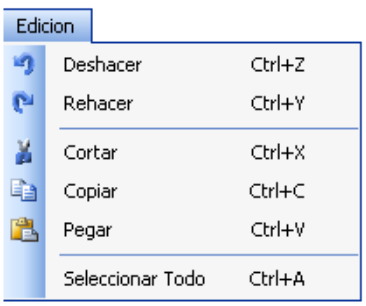
, la cual ha sido destinada para seleccionar si se va a imprimir el reporte o el diagrama del proyecto en el cual se esta trabajando.

Y para finalizar la opción Salir 

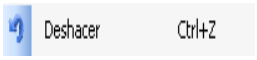
la cual se utiliza para salir de la aplicación.

Menú Edición.

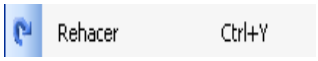
El menú Edición esta compuesto por los siguientes ítems:



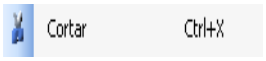
En dicha pestaña se encuentran las opciones de:

• Deshacer: 

regresa al paso anterior de la ultima acción realizada y así sucesivamente a modo de volver al estado pre-acción realizada.

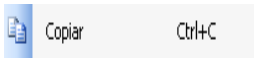
• Rehacer: 

es lo contrario a la función anterior este regresa la acción hasta el paso actual.

• Cortar: 

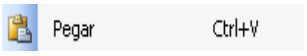
su función es eliminar de manera temporal, permite almacenar el objeto cortado para poderlo invocar con la opción pegar, cabe mencionar que al cortar otro elemento este se almacena borrando el que se encontraba en memoria.

Otra forma de llevar a cabo dicha actividad es a través del botón  que es una facilidad alternativa que da el sistema.

• Copiar: 

realiza una copia del componente seleccionado al igual que en el ítem anterior se lleva a cabo el clon con la función pegar.

La otra manera de poder acceder a esta es a través del botón en la barra de herramientas.

• Pegar: 

la función de esta es como su nombre lo dice pega o ingresa el elemento que con anterioridad fue cortado o copiado, es decir, que es la opción que complementa a las mencionadas anteriormente.

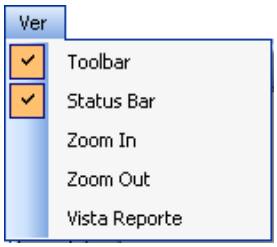
Al igual que las facilidades mencionadas anteriormente esta se puede acceder a través de un botón en la barra de herramientas dicho botón es:  .

• Seleccionar Todo: 

permite elegir todos los componentes que forman parte del diagrama.

Menú Ver.

Este está compuesto por los siguientes elementos:



A continuación se detalla cada una de sus funciones:

- Toolbar:  al activarlo permite visualizar la barra de herramientas.

- Status Bar:  le da la facilidad al usuario de presentar u ocultar la barra de estado.

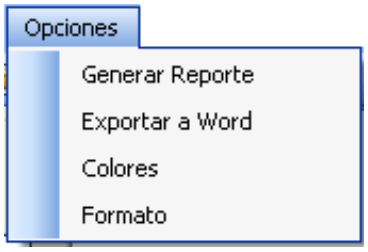
- Zoom In:  la finalidad de esta es aumentar las proporciones visuales del diagrama, es decir, incrementar en aspectos de visualización el tamaño.

- Zoom Out:  al contrario de la opción anterior este disminuye las proporciones visuales, es decir, aleja el diagrama.

- Vista Reporte:  presenta una visualización preliminar de cómo se vera el reporte.

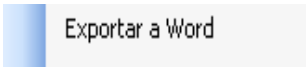
Menú Opciones.

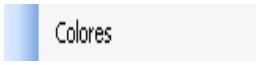
Está constituido por:



Estas son:

• Generar Reporte:  esta opción se utiliza para generar el Caso de Uso en prosa, es decir un reporte básico de lo que se plantea en el diagrama.

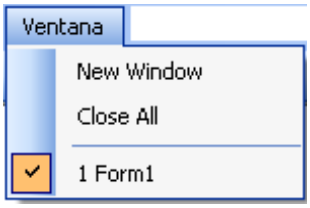
• Exportar a Word:  como su nombre lo dice se ha destinado esta opción para poder facilitar el uso de los diagramas al usuario exportándolos a Microsoft Word.

• Colores:  permite cambiar el color de las líneas y del fondo del elemento elaborado, por lo tanto el usuario podrá cambiarle el relleno y el color de líneas a los componentes del diagrama.

• Formato:  esta opción permitirá desplegar una pantalla donde se le pueda dar el formato a la letra.

Menú Ventana.

El menú ventana consta de los siguientes elementos:

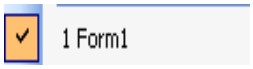


• New Windows: 

tiene una función similar a la opción de nuevo, genera una nueva área donde se puede realizar el diagramado.

• Close All: 

esta tiene la tarea de cerrar todas las ventanas o áreas de trabajo que podamos tener abiertas o habilitadas.

• Presentación de Pantallas que componen el proyecto: 

en esta sección aparecerán todas las áreas de trabajo que contenga el proyecto o que se puedan mantener abiertas, la sección donde se esta diagramando permanecerá chequeada.

Antes de empezar el Ejercicio E

Obtener Ayuda

Conceptos del Ejercicio

El ECUSV cuenta con un menú de ayuda que permite obtener todos los temas relacionados con el uso de la aplicación.

La ayuda del ECUSV cuenta con una introducción teórica de UML enfocada en los diagramas de casos de uso, proporcionando así una herramienta más completa.

Trucos y

3. Presione la combinación de teclas de acceso rápido para ejecutar inmediatamente su comando asociado sin tener que abrir el menú.

Editor Casos de Uso

Modulo

3

Manejo de Editor

Manejo de Funciones Básicas para la Elaboración de Un Diagrama

Objetivos del Módulo

Al terminar éste módulo, usted será capaz de realizar las siguientes acciones

A. Utilizar barra de Herramientas
B. Agregar elementos al área de trabajo.
C. Unir elementos.
D. Almacenar datos del elemento y las propiedades.

Antes de empezar el Ejercicio A

Usar la Barra de Herramientas

Conceptos del Ejercicio

Trucos y

La barra de herramientas es el medio que permite realizar distintas acciones comunes de trabajo.

Las opciones que contempla la barra de herramientas son:

- Copiar
- Cortar
- Pegar
- Nuevo
- Imprimir
- Vista preliminar
- Ayuda

Barra de Herramientas.

La barra de herramientas se encuentra compuesta por lo siguientes elementos:

- Nuevo: Elije un nuevo formulario o área de trabajo.
- Abrir: Permite seleccionar un área de trabajo ya existente.
- Guardar: Almacena el formulario.
- Imprimir: Permite imprimir el documento.
- Vista Preliminar: Presenta una vista preliminar o de impresión del área.
- Cortar: Permite cortar un elemento seleccionado.
- Pegar: Permite pegar el elemento que ha sido cortado o copiado en el área de trabajo.
- Copiar: Copia el elemento que ha sido seleccionado.
- Generar Reporte: Genera el reporte en base al diagrama de Caso de Uso elaborado.
- Ayuda: Genera una ventana auxiliar donde hay temas que pueden ayudarle al usuario a trabajar en el sistema.

Antes de empezar el Ejercicio B

Agregar Elementos al Área de Trabajo

Conceptos del Ejercicio

Trucos y

El ECUSV un manejo fácil de elementos para realizar un caso de uso.

Los elementos que se manejan con el ECUSV son:

Actor: es el ente que realiza la acción.

Caso de Uso: es la acción que se realiza en el sistema.

Paquete: es el elemento que engloba un grupo de actores y casos de uso.

Botones de Casos de Uso.

Esta área esta compuesta por los siguientes botones:

- 

Botón de Actor: Al activar este botón se le da un clic en el área de trabajo y diseña el elemento actor de los diagramas de casos de uso ya instalado se selecciona y se puede arrastrar.

- 

Botón de Caso de Uso: Sirve para ubicar el componente caso de uso sobre el sector determinado para la elaboración del los diagramas, al igual que el actor ya en el área se puede arrastrar.

- 

Botón de Paquete: Ubica el elemento paquete en el área de trabajo presenta las mismas propiedades que los elementos anteriores con la única diferencia que dentro de este se pueden tener más diagramas agrupados.

- 

Botón de Relación de Asociación: Selecciona la relación de asociación para ubicarla en el área.

- 

Botón de Relación de Generalización: Se utiliza para llevar el elemento de Generalización al sector de trabajo.

- 

Botón de Relación Dependencia: Al igual que en los componentes mencionados este ubica la relación Dependencia en el lugar destinado para elaborar el diagrama.

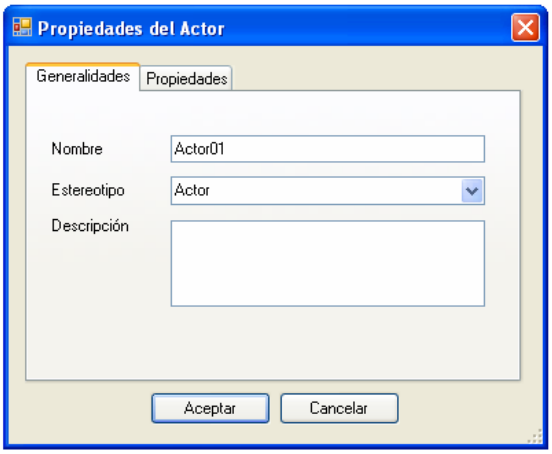
Cabe mencionar que al darle doble clic en los elementos antes mencionados se desplegaran las propiedades de cada uno de ellos.

Árbol.

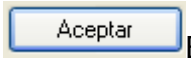
En esta parte del sistema se va generando en forma de árbol los elementos que se ingresan en el diagrama, ahí se puede examinar de una manera más fácil los componentes del documento.

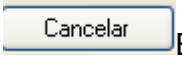
Pantallas Secundarias.

Actor y Caso de Uso.

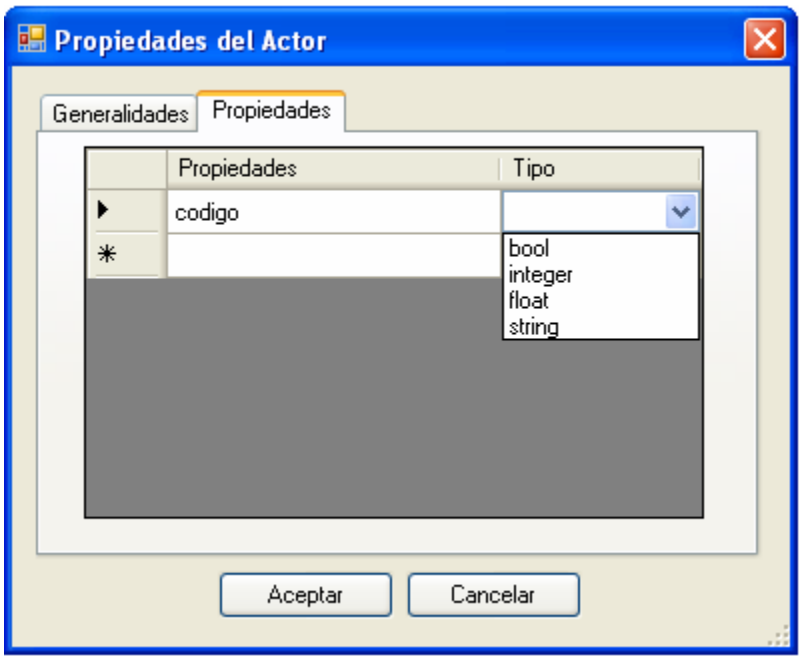


Las propiedades de los actores y de los casos de uso son similares a continuación se van a describir cada una de ellas.

- Nombre: aquí se colocara el nombre que el usuario desee darle al actor o caso de uso.
- Estereotipo: Se colocara el estereotipo al cual se le desea asignar el elemento seleccionado.
- Descripción: Como su nombre lo indica características del componente.
- Propiedades: Aquí se detallaran las propiedades que puedan presentar el actor o caso de uso.
- 

Botón Aceptar: Al llenar los campos con este botón se le agregan las propiedades antes mencionadas
- 

Botón Cancelar: Sale de la pantalla sin guardar sus atributos.



En la columna de propiedades se agregan dinámicamente los atributos que posee la clase actor.

En la columna tipo se determina el tipo de dato que corresponde al atributo, este puede ser: booleano (bool), entero (integer), flotante (float), cadena (string).

El botón aceptar almacena las propiedades insertadas.

El botón cancelar suspende toda la operación.

Antes de empezar el Ejercicio C

Unir Elementos

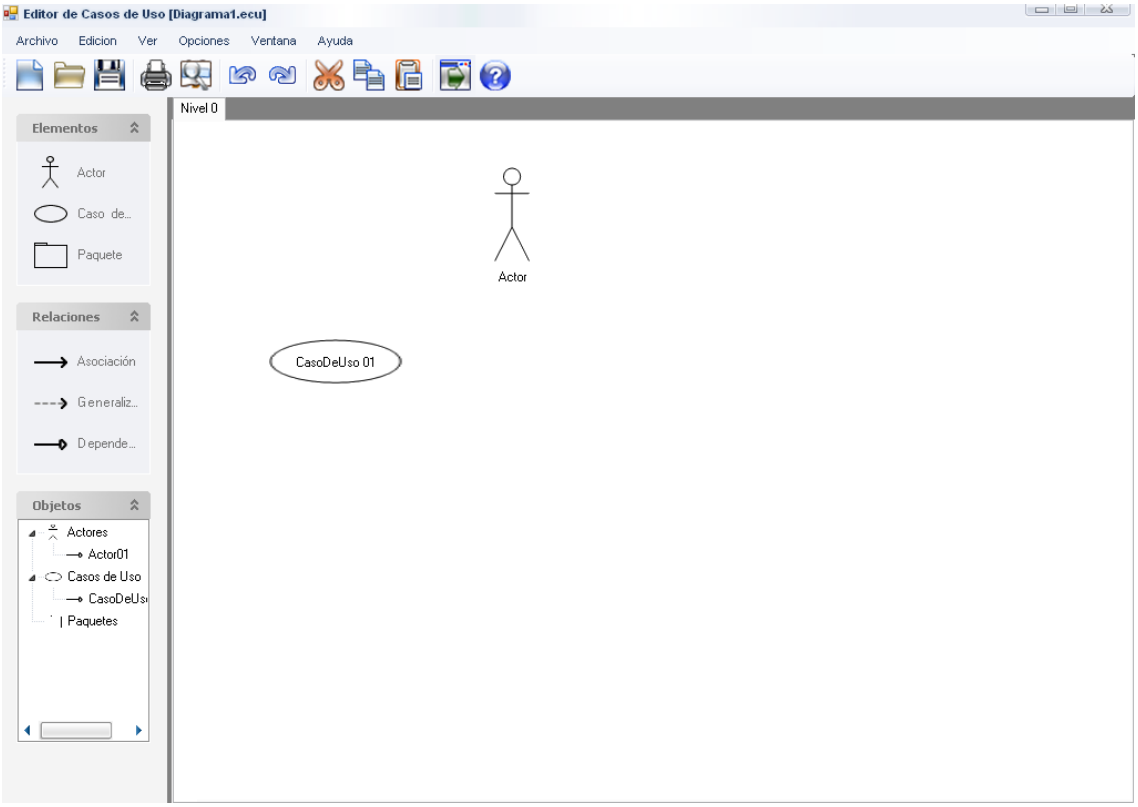
Conceptos del Ejercicio

El ECUSV cuenta permite realizar diagramas a partir de las validaciones que se utilizan enla filosofía UML.

Los diagramas realizados con el ECUSV permiten desarrollar casos de uso sencillos y mas complejos según el tipo de sistema que se este modelando.

Trucos y

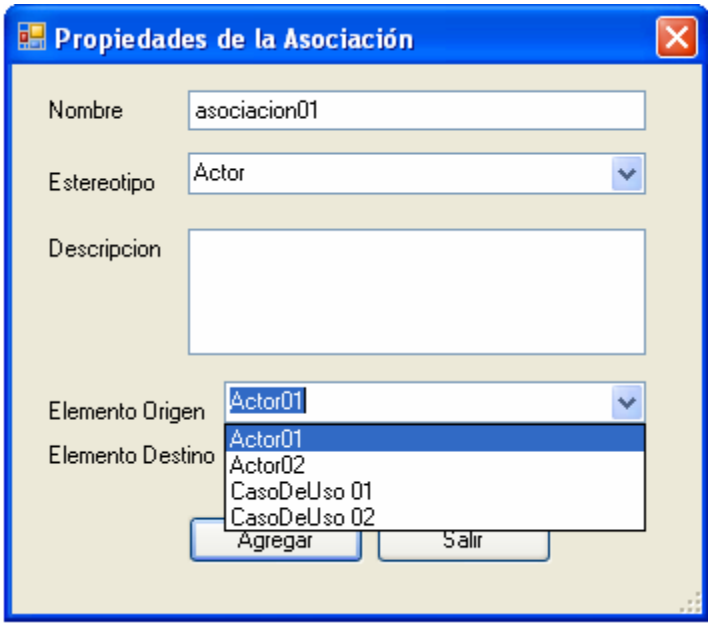
4. Presione la combinación de teclas de acceso rápido para ejecutar inmediatamente su comando asociado sin tener que abrir el menú.



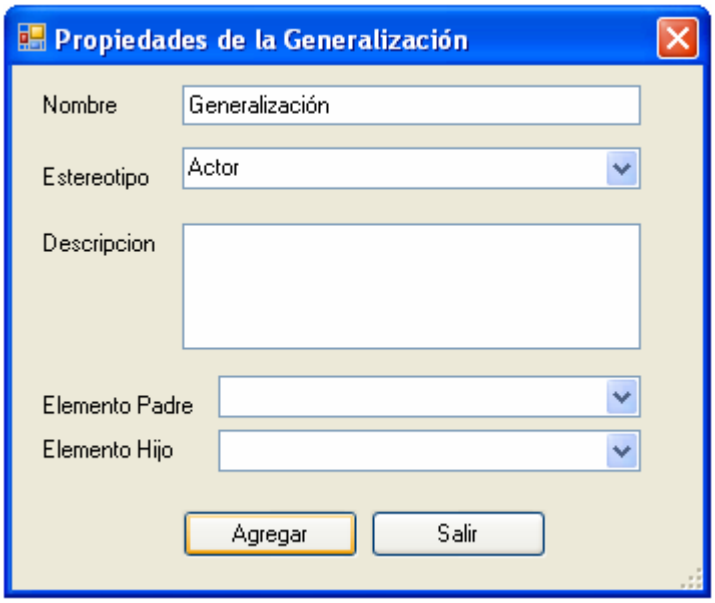
La unión de elementos se realiza seleccionando el elemento y llenando la información necesaria con los datos de la relación.

Relaciones.

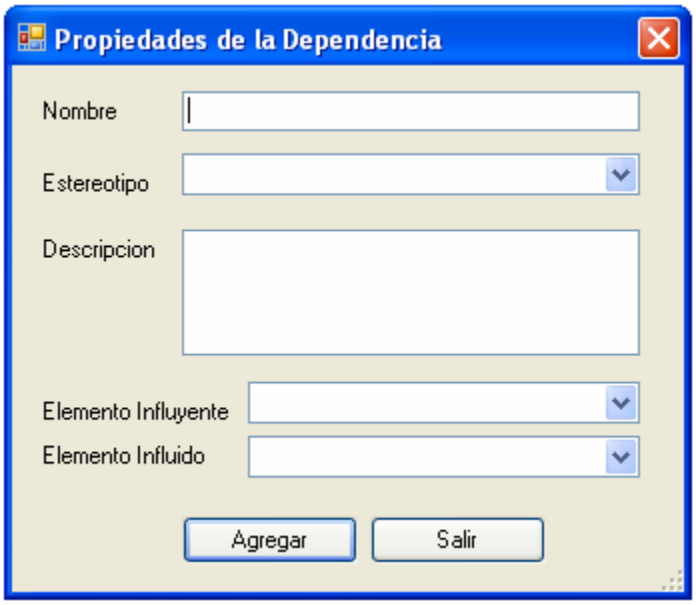
Relación de Asociación.



Relación de Generalización.



Relación de Dependencia.



MANUAL DEL PROGRAMADOR

ÍNDICE GENERAL

Introducción

Este documento tiene la finalidad de mostrar la estructura y modo de programación que posee el código fuente que constituye el software del ECUSV, el cual incluye las funciones y procedimientos que permiten ejecutar las tareas correspondientes para el correcto funcionamiento del editor, de tal forma que permita al usuario de este documento conocer las características, funciones, eventos y clases que se han utilizado.

El modelo de programación esta orientado a objetos, debido a que esta diseñado con clases. Dichas clases son las estándar brindadas por el lenguaje de Microsoft Visual Studio 2005 y están acompañadas por procesos y codificación de los eventos de los objetos utilizados. El sistema posee ciertas clases para el manejo de los elementos, siendo estas el núcleo del sistema.

El editor esta constituido por los objetos estándares del visual C# utilizando sentencias que son propias de este lenguaje de programación para mayor eficiencia, la sintaxis esta compuesta por clases y procedimientos necesarios para el buen funcionamiento de la aplicación.

Definición del programa

El ECUSV es un software para la elaboración de Diagramas de Casos de Uso que pertenece a la filosofía UML, siendo esta una de las más usadas para llevar a cabo el desarrollo de un sistema de una manera adecuada. El editor puede cumplir las funciones básicas de forma completa para poder llevar a cabo un diagrama de Casos de Uso supliendo las necesidades que a este se le pueda presentar a la hora de estar generando el proyecto, opciones para presentar las dos formas de visualización: diagrama y prosa.

Estructura del Programa

El programa está basado en el uso de las clases creadas en Visual C#, cada componente que forma parte del ECUSV es una clase y esta formada por sus propios formularios, módulos de clases, librerías y objetos.

Todas las clases son invocadas a partir de un formulario principal, donde se controlan todos los eventos que intervienen en el funcionamiento del editor.

A continuación se mencionan los elementos que constituyen el proyecto que forma parte del ECUSV.

Clases.

arrActor:

Esta clase maneja de forma dinámica la creación del elemento Actor

Funciones Miembro	Descripción
addActor ()	Inicializa el objeto Actor
arrActor()	Constructor
ctrActor	Invocación a la clase ctrActor
Remove	Elimina el Control

arrCasoDeUso:

Maneja de forma dinámica la creación del componente Caso de Uso

Funciones Miembros	Descripción
addActor ()	Inicializa el objeto Actor
arrActor()	Constructor
ctrActor	Invocación a la clase ctrActor
Remove	Elimina el Control

arrPaquete:

Maneja de forma dinámica la creación del componente Paquete

Funciones Miembro	Descripción
addActor ()	Inicializa el objeto Actor
arrActor()	Constructor
ctrActor	Invocación a la clase ctrActor
Remove	Elimina el Control

ctrActor, ctrCaso de Uso y ctrPaquete

Se encarga de todo el manejo de las propiedades del objeto

Funciones Miembro	Descripción
ctrActor()	Constructor
Imagen	Maneja las propiedades nombre, estereotipo, descripción y propiedades y lo asigna a un objeto común Actor
Imagen_mouse_down	Maneja el movimiento hacia abajo
Imagen_mouse_move	Maneja el movimiento
Imagen_mouse_move_up	Maneja el movimiento hacia arriba

Ubicación de los archivos

El proyecto se almacenara en una carpeta principal llamada Editor de Casos de Uso, dentro de esta carpeta se almacenarán los archivos de tipo clases y windows form que se utilizan en el editor. En otra carpeta llamada Resources se encuentran guardadas las imágenes y en la carpeta BIN se almacena los archivos de tipo .DLL y los ejecutables.

Nombre de los Archivos:

Se utilizan prefijos para clasificar los archivos.

- El prefijo arr se refiere a los archivos de creación de los elementos.
- El prefijo ctr se refiere al manejo de las propiedades de los elementos.
- El prefijo frm corresponde a los formularios para cada elemento.

Se maneja el archivo wfPrincipal que es el formulario principal del editor

Estructura XML

Código Fuente.

Clase Utilizada para la elaboración del Actor.

```
class arrActor :
    System.Collections.CollectionBase
    {
        private readonly System.Windows.Forms.Form HostForm;
        //private readonly wfPrincipal HostForm;

        public void AddActor()
        {
            ctrActor aActor= new ctrActor();
            this.List.Add(aActor);
            HostForm.Controls.Add(aActor);

            aActor.Top = Count * 25;
            aActor.Left = 200;
            aActor.Tag = this.Count;
            aActor.Text = "Actor " + this.Count.ToString();

            //aActor.Click += new
System.EventHandler(DobleClickHandler);
        }

        public arrActor(System.Windows.Forms.Form host)
        {
            HostForm = host;
            this.AddActor();
        }

        public ctrActor this[int Index]
        {
```

```
            get
            {
                return (ctrActor)this.List[Index];
            }
        }

        public void Remove()
        {
            if (this.Count > 0)
            {
                HostForm.Controls.Remove(this[this.Count - 1]);
                this.List.RemoveAt(this.Count - 1);
            }
        }


    }
```

Clase Utilizada para la elaboracion de los Casos de Uso.

```
class arrCasoDeUso :
    System.Collections.CollectionBase
    {
        private readonly System.Windows.Forms.Form HostForm;

        public void AddCasoDeUso()
        {
            ctrCasoDeUso aCasoDeUso= new ctrCasoDeUso();
            this.List.Add(aCasoDeUso);
            HostForm.Controls.Add(aCasoDeUso);

            aCasoDeUso.Top = Count * 25;
            aCasoDeUso.Left = 250;
            aCasoDeUso.Tag = this.Count;
            aCasoDeUso.Text = "Caso de Uso " + this.Count.ToString();

            //aActor.Click += new
System.EventHandler(DobleClickHandler);
        }

        public arrCasoDeUso(System.Windows.Forms.Form host)
        {
            HostForm = host;
            this.AddCasoDeUso();
        }

        public ctrCasoDeUso this[int Index]
        {
            get
            {
                return (ctrCasoDeUso)this.List[Index];
            }
        }
```

```
        public void Remove()
        {
            if (this.Count > 0)
            {
                HostForm.Controls.Remove(this[this.Count - 1]);
                this.List.RemoveAt(this.Count - 1);
            }
        }

}
```

Clase Utilizada para la elaboracion de los Paquetes.

```
class arrPaquete :
    System.Collections.CollectionBase
    {
        private readonly System.Windows.Forms.Form HostForm;

        public void AddPaquete()
        {
            ctrPaquete aPaquete = new ctrPaquete();
            this.List.Add(aPaquete);
            HostForm.Controls.Add(aPaquete);

            aPaquete.Top = Count * 25;
            aPaquete.Left = 300;
            aPaquete.Tag = this.Count;
            aPaquete.Text = "Paquete " + this.Count.ToString();

            //aActor.Click += new
System.EventHandler(DobleClickHandler);
        }

        public arrPaquete(System.Windows.Forms.Form host)
        {
            HostForm = host;
            this.AddPaquete();
        }

        public ctrPaquete this[int Index]
        {
            get
            {
                return (ctrPaquete)this.List[Index];
            }
        }

        public void Remove()
```

```
        {
            if (this.Count > 0)
            {
                HostForm.Controls.Remove(this[this.Count - 1]);
                this.List.RemoveAt(this.Count - 1);
            }
        }
}
```

Clase Utilizada para el movimiento del Mouse.

```
public class MouseMove
    {
        //the Current X and Y position
        public int curX;
        public int curY;

        //the last recorded X and Y position
        private int lastX;
        private int lastY;

        //the control being moved
        private PictureBox Control;

        //Whether the control is in the middle of being moved
        private bool Moving;
        private Timer timer;

        // Being the move
        public void Begin(PictureBox  c, int X, int Y)
        {
            Moving = true;
            Control = c;
            lastX = X;
            lastY = Y;
        }
        public void InProgress(int X, int Y)
        {
            if (!Moving) return;
            timer.Enabled = true;

            curX = X - lastX;
            curY = Y - lastY;
        }
        public void EndIt()
        {
            Moving = false;
            timer.Enabled = false;
        }

        public void ToNewPosition()
        {
            int X;
```

```
            int Y;

            //Set it to the new position
            X = Control.Left + curX;
            Y = Control.Top + curY;

            Control.Location = new System.Drawing.Point(X, Y);
        }

        public void Initialize(System.Windows.Forms.Timer tmr)
        {
            tmr.Enabled = false;
            tmr.Interval = 30;
            timer = tmr;
        }
}
```

Clase Utilizada para el movimiento del Actor.

```
public class movActor
    {
        //the Current X and Y position
        public int curX;
        public int curY;

        //the last recorded X and Y position
        private int lastX;
        private int lastY;

        //the control being moved
        private ctrActor Actor;

        //Whether the control is in the middle of being moved
        private bool Moving;
        private Timer timer;

        // Being the move
        public void Begin(ctrActor c, int X, int Y)
        {
            Moving = true;
            Actor = c;
            lastX = X;
            lastY = Y;
        }
        public void InProgress(int X, int Y)
        {
            if (!Moving) return;
            timer.Enabled = true;

            curX = X - lastX;
            curY = Y - lastY;
        }
        public void EndIt()
        {
```

```
            Moving = false;
            timer.Enabled = false;
        }

        public void ToNewPosition()
        {
            int X;
            int Y;

            //Set it to the new position
            X = Actor.Left + curX;
            Y = Actor.Top + curY;

            Actor.Location = new System.Drawing.Point(X, Y);
        }

        public void Initialize(System.Windows.Forms.Timer tmr)
        {
            tmr.Enabled = false;
            tmr.Interval = 30;
            timer = tmr;
        }
}
```

Clase Utilizada para el movimiento del Caso de Uso.

```
public class movCasoDeUso
    {
        //the Current X and Y position
        public int curX;
        public int curY;

        //the last recorded X and Y position
        private int lastX;
        private int lastY;

        //the control being moved
        private ctrCasoDeUso CasoDeUso;

        //Whether the control is in the middle of being moved
        private bool Moving;
        private Timer timer;

        // Being the move
        public void Begin(ctrCasoDeUso c, int X, int Y)
        {
            Moving = true;
            CasoDeUso = c;
            lastX = X;
            lastY = Y;
        }
        public void InProgress(int X, int Y)
        {
            if (!Moving) return;
            timer.Enabled = true;
```

```
            curX = X - lastX;
            curY = Y - lastY;
        }
        public void EndIt()
        {
            Moving = false;
            timer.Enabled = false;
        }

        public void ToNewPosition()
        {
            int X;
            int Y;

            //Set it to the new position
            X = CasoDeUso.Left + curX;
            Y = CasoDeUso.Top + curY;

            CasoDeUso.Location = new System.Drawing.Point(X, Y);
        }

        public void Initialize(System.Windows.Forms.Timer tmr)
        {
            tmr.Enabled = false;
            tmr.Interval = 30;
            timer = tmr;
        }
}
```

Clase Utilizada para el movimiento del Paquete.

```
public class movPaquete
    {
        //the Current X and Y position
        public int curX;
        public int curY;

        //the last recorded X and Y position
        private int lastX;
        private int lastY;

        //the control being moved
        private ctrPaquete Paquete;

        //Whether the control is in the middle of being moved
        private bool Moving;
        private Timer timer;

        // Being the move
        public void Begin(ctrPaquete c, int X, int Y)
        {
            Moving = true;
            Paquete = c;
            lastX = X;
            lastY = Y;
        }
```

```
        public void InProgress(int X, int Y)
        {
            if (!Moving) return;
            timer.Enabled = true;

            curX = X - lastX;
            curY = Y - lastY;
        }
        public void EndIt()
        {
            Moving = false;
            timer.Enabled = false;
        }

        public void ToNewPosition()
        {
            int X;
            int Y;

            //Set it to the new position
            X = Paquete.Left + curX;
            Y = Paquete.Top + curY;

            Paquete.Location = new System.Drawing.Point(X, Y);
        }

        public void Initialize(System.Windows.Forms.Timer tmr)
        {
            tmr.Enabled = false;
            tmr.Interval = 30;
            timer = tmr;
        }
}
```

Clase Utilizada para el movimiento del Paquete.

```
static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new wfPrincipal());
        }
}
```

Codigo de la Pantalla Principal.

```
public partial class wfPrincipal : Form
    {
        MouseMove mMove = new MouseMove();
        movActor mActor = new movActor();
```

```
        movCasoDeUso mCasoDeUso = new movCasoDeUso();
        movPaquete mPaquete = new movPaquete();

        arrActor actor;
        arrCasoDeUso casodeuso;
        arrPaquete paquete;

        TreeNode nodoActor = new TreeNode("Actores", 3, 3);
        TreeNode nodoCasoDeUso = new TreeNode("Casos de Uso", 8, 8);
        TreeNode nodoPaquete = new TreeNode("Paquetes", 13, 13);


        public wfPrincipal()
        {
            InitializeComponent();

            actor = new arrActor(this);
            actor[0].Visible = false;

            casodeuso = new arrCasoDeUso(this);
            casodeuso[0].Visible = false;

            paquete = new arrPaquete(this);
            paquete[0].Visible = false;

            mMove.Initialize(timer1);
        }

        private void wfPrincipal_Load(object sender, EventArgs e)
        {
            mMove.Initialize(timer1);

            mActor.Initialize(tmrActor);
            mCasoDeUso.Initialize(tmrCasoDeUso);
            mPaquete.Initialize(tmrPaquete);


            tvObjetos.Nodes.Add(nodoActor);
            tvObjetos.Nodes.Add(nodoCasoDeUso);
            tvObjetos.Nodes.Add(nodoPaquete);
        }

        private void ctrActor1_Load(object sender, EventArgs e)
        {

        }

        private void btnActor_Click(object sender, EventArgs e)
        {
            actor.AddActor();
            actor[actor.Count - 1].BringToFront();

            if ( (actor.Count - 1) < 10)
            {
                actor[actor.Count - 1].Nombre = "Actor0" +
(actor.Count - 1);
            }
            else
            {
                actor[actor.Count - 1].Nombre = "Actor" + (actor.Count
- 1);
```

```
            }

            this.actor[actor.Count - 1].MouseDown += new
System.Windows.Forms.MouseEventHandler(this.ctrActor1_MouseDown);
            this.actor[actor.Count - 1].MouseMove += new
System.Windows.Forms.MouseEventHandler(this.ctrActor1_MouseMove);
            this.actor[actor.Count - 1].MouseUp += new
System.Windows.Forms.MouseEventHandler(this.ctrActor1_MouseUp);

            TreeNode nuevoActor = new TreeNode(actor[actor.Count -
1].Nombre, 9, 9);
            nuevoActor.Tag = (actor.Count - 1) ;
            nodoActor.Nodes.Add(nuevoActor);
            nodoActor.Expand();


        }

        private void btnCasoDeUso_Click(object sender, EventArgs e)
        {
            casodeuso.AddCasoDeUso();
            casodeuso[casodeuso.Count - 1].BringToFront();


            if ((casodeuso.Count - 1) < 10)
            {
                casodeuso[casodeuso.Count - 1].Nombre = "CasoDeUso 0"
+ (casodeuso.Count - 1);
            }
            else
            {
                casodeuso[casodeuso.Count - 1].Nombre = "CasoDeUso" +
(casodeuso.Count - 1);
            }

            this.casodeuso[casodeuso.Count - 1].MouseDown += new
System.Windows.Forms.MouseEventHandler(this.ctrCasoDeUso1_MouseDown);
            this.casodeuso[casodeuso.Count - 1].MouseMove += new
System.Windows.Forms.MouseEventHandler(this.ctrCasoDeUso1_MouseMove);
            this.casodeuso[casodeuso.Count - 1].MouseUp += new
System.Windows.Forms.MouseEventHandler(this.ctrCasoDeUso1_MouseUp);

            TreeNode nuevoCasoDeUso = new
TreeNode(casodeuso[casodeuso.Count - 1].Nombre, 9, 9);
            nuevoCasoDeUso.Tag = casodeuso.Count - 1;
            nodoCasoDeUso.Nodes.Add(nuevoCasoDeUso);
            nodoCasoDeUso.Expand();

        }

        private void btnPaquete_Click(object sender, EventArgs e)
        {
            paquete.AddPaquete();
            paquete[paquete.Count - 1].BringToFront();

            if ((paquete.Count - 1) < 10)
            {
                paquete[paquete.Count - 1].Nombre = "paquete0" +
(paquete.Count - 1);
            }
            else
```

```
            {
                paquete[paquete.Count - 1].Nombre = "paquete" +
(paquete.Count - 1);
            }

            paquete[paquete.Count - 1].MouseDown += new
System.Windows.Forms.MouseEventHandler(this.ctrPaquete1_MouseDown);
            paquete[paquete.Count - 1].MouseMove += new
System.Windows.Forms.MouseEventHandler(this.ctrPaquete1_MouseMove);
            paquete[paquete.Count - 1].MouseUp += new
System.Windows.Forms.MouseEventHandler(this.ctrPaquete1_MouseUp);

            TreeNode nuevoPaquete = new TreeNode(paquete[paquete.Count
- 1].Nombre, 9, 9);
            nuevoPaquete.Tag = paquete.Count - 1;
            nodoPaquete.Nodes.Add(nuevoPaquete);
            nodoPaquete.Expand();

        }

        private void pnControles_Paint(object sender, PaintEventArgs
e)
        {

        }



        private void timer1_Tick(object sender, EventArgs e)
        {
            mMove.ToNewPosition();


        }

        private void imgActor1_MouseDown(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Left)
                mMove.Begin((PictureBox)sender , e.X, e.Y);
            //mMove.Begin(sender, int.Parse(e.X.ToString()),
int.Parse(e.Y.ToString()))
        }

        private void imgActor1_MouseMove(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Left)
                mMove.InProgress(e.X, e.Y);
        }

        private void imgActor1_MouseUp(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Left)
                mMove.EndIt();
        }
        //******************************
        private void ctrActor1_MouseDown(object sender, MouseEventArgs
e)
        {
```

```
            if (e.Button == MouseButtons.Left)
                this.mActor.Begin((ctrActor)sender, e.X, e.Y);
        }

        private void ctrActor1_MouseMove(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Left)
                this.mActor.InProgress(e.X, e.Y);
        }

        private void ctrActor1_MouseUp(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Left)
                this.mActor.EndIt();
        }

        private void btnAsociacion_Click(object sender, EventArgs e)
        {
            frmAsociacion Asociacion = new frmAsociacion();
            //Asociacion.Actores = actor;
            Asociacion.ShowDialog();

        }

        private void ctrActor1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("eventoexterior");
        }

        private void ctrActor1_DoubleClick(object sender, EventArgs e)
        {
            MessageBox.Show("doble eventoexterior");
        }

        private void btnGeneralizacion_Click(object sender, EventArgs
e)
        {
            //ctrActor1.Initialize(timer1);
        }

        private void tmrActor_Tick(object sender, EventArgs e)
        {
            mActor.ToNewPosition();
        }

        private void ctrCasoDeUso1_MouseDown(object sender,
MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
                this.mCasoDeUso.Begin((ctrCasoDeUso)sender, e.X, e.Y);
        }

        private void ctrCasoDeUso1_MouseMove(object sender,
MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
                this.mCasoDeUso.InProgress(e.X, e.Y);

        }
```

```
        private void ctrCasoDeUso1_MouseUp(object sender,
MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
                this.mCasoDeUso.EndIt();
        }

        private void ctrPaquete1_MouseDown(object sender,
MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
                this.mPaquete.Begin((ctrPaquete)sender, e.X, e.Y);
        }

        private void ctrPaquete1_MouseMove(object sender,
MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
                this.mPaquete.InProgress(e.X, e.Y);
        }

        private void ctrPaquete1_MouseUp(object sender, MouseEventArgs
e)
        {
            if (e.Button == MouseButtons.Left)
                this.mPaquete.EndIt();
        }

        private void ExitToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Documento.Visible = false;
            pnControles.Enabled = false;
        }

        private void NewToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Nuevo();
        }

        protected void Nuevo()
        {
            Documento.Visible = true;
            pnControles.Enabled = true;
            this.Text = "Editor de Casos de Uso [Diagrama1.ecu]";
            //tvObjetos.Nodes.Clear();

        }

        private void NewToolStripButton_Click(object sender, EventArgs
e)
        {
            Nuevo();

        }

        private void Documento_Paint(object sender, PaintEventArgs e)
        {
```

```
}

private void ctrActor2_DoubleClick(object sender, EventArgs e)
{
    MessageBox.Show("jhgsdkhjgfhsd");
}

private void tmrCasoDeUso_Tick(object sender, EventArgs e)
{
    this.mCasoDeUso.ToNewPosition();
}

private void tmrPaquete_Tick(object sender, EventArgs e)
{
    this.mPaquete.ToNewPosition();
}

private void ctrActor2_Load(object sender, EventArgs e)
{

}

                                }
```