CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE CLASE, ORIENTADA BAJO EL MODELO DE SERVICIOS DE INTERNET CLOUD COMPUTING  
(Proyecto de Grado Dirigido)

GABRIEL LEONARDO DÍAZ CÁRDENAS

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER  
FACULTAD DE INGENIERIAS  
PLAN DE ESTUDIOS DE INGENIERIA DE SISTEMAS  
SAN JOSÉ DE CÚCUTA  
2013

CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE CLASE, ORIENTADA BAJO EL MODELO DE SERVICIOS DE INTERNET CLOUD COMPUTING.

GABRIEL LEONARDO DÍAZ CÁRDENAS  
Código 0152685

Trabajo de grado para ostentar el título de Ingeniero de Sistemas de la Universidad Francisco de Paula Santander

DIRECTOR  
MARCO ANTONIO ADARME JAIMES  
INGENIERO DE SISTEMAS  
MAGISTER EN CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER  
FACULTAD DE INGENIERIAS  
PLAN DE ESTUDIOS DE INGENIERIA DE SISTEMAS  
SAN JOSÉ DE CÚCUTA  
2013

**TABLA DE CONTENIDO**

[LISTADO DE ILUSTRACIONES 6](#_Toc361939748)

[LISTADO DE TABLAS 7](#_Toc361939749)

[INTRODUCCIÓN 8](#_Toc361939750)

[1. PRESENTACIÓN DEL PROYECTO 10](#_Toc361939751)

[1.1 TITULO 10](#_Toc361939752)

[1.2 PLANTEAMIENTO DEL PROBLEMA 10](#_Toc361939753)

[1.3 JUSTIFICACIÓN 11](#_Toc361939754)

[1.4 OBJETIVOS 13](#_Toc361939755)

[1.4.1 Objetivo General 13](#_Toc361939756)

[1.4.2 Objetivos Específicos 13](#_Toc361939757)

[1.5 ALCANCES Y DELIMITACIONES 14](#_Toc361939758)

[1.5.1 Alcance 14](#_Toc361939759)

[1.5.2 Delimitaciones 14](#_Toc361939760)

[2. MARCO REFERENCIAL 16](#_Toc361939761)

[2.1 MARCO DE ANTECEDENTES 16](#_Toc361939762)

[2.1.1 Draw.io 16](#_Toc361939763)

[2.1.2 Creately 16](#_Toc361939764)

[2.1.3 GWTUML 17](#_Toc361939765)

[2.1.4 JoinJS 17](#_Toc361939766)

[2.2 MARCO CONCEPTUAL 18](#_Toc361939767)

[2.2.1 Herramientas CASE 18](#_Toc361939768)

[2.2.2 Cloud Computing 18](#_Toc361939769)

[2.2.3 UML 18](#_Toc361939770)

[2.2.4 Diagramas de Clase 18](#_Toc361939771)

[2.2.5 Procesos de Desarrollo de Software 18](#_Toc361939772)

[2.2.6 Meta-Programación 18](#_Toc361939773)

[2.3 MARCO TEORICO 18](#_Toc361939774)

[2.3.1 Java 19](#_Toc361939775)

[2.3.2 Frameworks 19](#_Toc361939776)

[2.3.3 Servidor Web 19](#_Toc361939777)

[2.3.4 Servidor de Aplicaciones 19](#_Toc361939778)

[2.3.5 Extreme Programming (XP) 19](#_Toc361939779)

[2.3.6 PrimeFaces 19](#_Toc361939780)

[2.3.7 SVG 19](#_Toc361939781)

[2.3.8 HTML, JavaScript y CSS 19](#_Toc361939782)

[2.4 MARCO DE LEGAL 19](#_Toc361939783)

[2.4.1 Acuerdos de uso 19](#_Toc361939784)

[2.4.2 Contrato de licencia de código binario, SUN MICROSYSTEMS 20](#_Toc361939785)

[2.4.3 GNU General PublicLicence 20](#_Toc361939786)

[2.4.4 Propiedad Intelectual 21](#_Toc361939787)

[3. DISEÑO METODOLÓGICO 22](#_Toc361939788)

[3.1 TIPO DE INVESTIGACIÓN 22](#_Toc361939789)

[3.2 FUENTES DE INFORMACIÓN 22](#_Toc361939790)

[3.2.1 Fuentes de información primaria 22](#_Toc361939791)

[3.2.2 Fuentes de información secundaria 23](#_Toc361939792)

[3.3 RECOLECCIÓN Y ANALISIS DE INFORMACIÓN 23](#_Toc361939793)

[4. ANALISIS DE LAS HERRAMIENTAS CASE 25](#_Toc361939794)

[4.1 ENTERPRISE ARCHITECT 25](#_Toc361939795)

[4.1.1 Historia 26](#_Toc361939796)

[4.1.2 Características 27](#_Toc361939797)

[4.2 STARUML 28](#_Toc361939798)

[4.3 DRAW.IO 28](#_Toc361939799)

[5. CLASIFICACIÓN DE LA INFORMACIÓN 29](#_Toc361939800)

[5.1 MATRIZ DE CARACTERISTICAS 29](#_Toc361939801)

[5.2 CARACTERISTICAS PROPIAS 29](#_Toc361939802)

[5.3 LISTA DE FUNCIONALIDADES 29](#_Toc361939803)

[6. METODOLOGÍA DE DESARROLLO 30](#_Toc361939804)

[6.1 EXPLORACIÓN 30](#_Toc361939805)

[6.1.1 Identificación de Roles del Proyecto 30](#_Toc361939806)

[6.1.2 Identificación de Actores del Sistema 31](#_Toc361939807)

[6.1.3 Historias de Usuario 34](#_Toc361939808)

[6.2 PLANIFICACIÓN 38](#_Toc361939809)

[6.2.1 Priorización de Historias de Usuario 38](#_Toc361939810)

[6.2.2 Estimación de Historias de Usuario 38](#_Toc361939811)

[6.2.3 Definición de Iteraciones 38](#_Toc361939812)

[6.2.4 Metafora del Sistema 38](#_Toc361939813)

[6.3 DESCRIPCIÓN DE LAS ITERACIONES 38](#_Toc361939814)

[6.3.1 Iteración 1 39](#_Toc361939815)

[6.3.2 Iteración 2 39](#_Toc361939816)

[6.4 PLANIFICACIÓN DE LAS PRUEBAS 39](#_Toc361939817)

[6.4.1 Casos de Prueba 39](#_Toc361939818)

[6.4.2 Pruebas Unitarias 39](#_Toc361939819)

[7. IMPLEMENTACIÓN 40](#_Toc361939820)

[7.1 DEFINICIÓN DE LA ARQUITECTURA 40](#_Toc361939821)

[7.2 DEFINICIÓN DE LA TECNOLOGIA 40](#_Toc361939822)

[7.3 PATRONES DE DESARROLLO 40](#_Toc361939823)

[7.4 BASES DE DATOS 40](#_Toc361939824)

[7.5 CONFIGURACIÓN 40](#_Toc361939825)

[7.6 ESPECIFICACIÓN DE FUNCIONALIDADES 40](#_Toc361939826)

[7.7 DESPLIEGUE 40](#_Toc361939827)

[8. ASEGURAMIENTO DE LA CALIDAD 41](#_Toc361939828)

[8.1 CASOS DE PRUEBA 41](#_Toc361939829)

[8.2 PRUEBAS DE UNIDAD 41](#_Toc361939830)

[8.3 ANALISIS ESTATICO DE CODIGO 41](#_Toc361939831)

[CONCLUSIONES 42](#_Toc361939832)

[REFERENCIAS BIBLIOGRÁFICAS 43](#_Toc361939833)

# LISTADO DE ILUSTRACIONES

[Ilustración 1. Diagrama de Actores 29](#_Toc361683451)

# LISTADO DE TABLAS

[Tabla 1. Formato características generales – Enterprise Architect 28](#_Toc361939718)

[Tabla 2. Especificación del actor Visitante 31](#_Toc361939719)

[Tabla 3. Especificación del actor Usuario 32](#_Toc361939720)

[Tabla 4. Especificación del actor Invitado 32](#_Toc361939721)

[Tabla 5. Especificación del actor Diagramador 33](#_Toc361939722)

# INTRODUCCIÓN

En el desarrollo de software es común usar herramientas que faciliten las tareas complejas y agilicen el trabajo manual; la mayoría o casi todas las empresas denominadas *Fábricas de Software*, incorporan aplicaciones a su quehacer diario, permitiendo a sus trabajadores llevar a cabo su labor de manera más eficiente, conjunta y automatizada.

Este grupo de aplicaciones sonllamadas *Herramientas CASE*, debido a las siglas tomadas del nombre en inglés ***“****Computer****A****ided****S****oftware* ***E****ngineering”*,en nuestro idioma *Ingeniería de Software* asistida por computadora. Estasson la base para la creación de todas las aplicaciones que vemos hoy en día y cada una se especializa en diferentes aspectos, dependiendo de la o las fases del proceso de desarrollo que apoyan. Otras más complejas, permiten controlar prácticamente todo el ciclo de vida de una aplicación desde el análisis de requerimientos hasta el despliegue y mantenimiento.

Tras un largo camino en la evolución del software y metodologías de desarrollo, se han incorporado nuevos modelos y tendencias para desarrollo de aplicaciones. Debido al alto impacto que tuvo el internet, ha surgido un modelo de negocio en donde se rompe el concepto tradicional de las aplicacionesqueresidenestáticas en la máquina del usuario y este a su vez compra los derechos para usar en todo momento determinada aplicación; hoy por hoy hablamos de un esquema en el que las aplicaciones se alojan en lugares remotos y los usuarios hacen uso de ellas bajo demandala red global Internet y pagando únicamente el consumo que se hace de estas (en el caso del Software Propietario). Desde hace varios años llamado *Cloud Computing o Computación en la Nube*, este modelo ha cambiado la concepción que se tenía anteriormente del uso de aplicaciones software.

Con el desarrollo de este proyecto se pretende incorporar ambos conceptos en una herramienta software, aprovechando las bondades que brindan las aplicaciones en la nube y resolviendo una necesidad particular de nuestro ambiente universitario. Deestemodo los estudiantes podrán disponer de una herramienta básica para su formación y que posea las características necesarias para ser considerada una aplicación en la nube.

El objetivo principal es el desarrollo de una herramienta para el modelado de Diagramas de Clase UML, la cual permita crear y manipular de manera online los componentes básicos de este tipo de diagrama; inclusivepermitaaúnmás que una mera representación visual, llegando a obtener archivos de código fuente generados a partir de los componentes o elementos del diagrama.

A lo largo de este documento se *CLASS Modeler* pretende ser una herramienta debido a su naturaleza, esta herramienta incorpora los conceptos básico de los diagramas de clase

# PRESENTACIÓN DEL PROYECTO

## TITULO

CONSTRUCCIÓN DE UNA APLICACIÓN WEB PARA LA ELABORACIÓN DE DIAGRAMAS DE CLASE, ORIENTADA BAJO EL MODELO DE SERVICIOS DE INTERNET CLOUD COMPUTING.

## PLANTEAMIENTO DEL PROBLEMA

En estudios realizados por el grupo de investigación y desarrollo de software (GIDIS) de la Universidad Francisco de Paula Santander, después convertido en un proyecto de grado[[1]](#footnote-1), se ha determinado que unos de los principales factores involucrados en los altos índices de mortalidad y deserción académica en la carrera de Ingeniería de Sistemas de las universidades de Cúcuta, son la falta de dedicación al estudio por parte de los estudiantes, poca comprensión de temas como abstracción y modelamiento de sistemas, y falta de ambientes prácticos donde los estudiantes tuvieran la posibilidad de llevar a cabo actividades para afianzar los conceptos aprendidos en clase. Teniendo esto en cuenta se hace evidente que se necesitan mecanismos para que los estudiantes ejerciten sus habilidades y de esta manera se disminuya esta problemática que ha afectado considerablemente al programa académico.

Dentro del programa de Ingeniería de Sistemas de la UFPS se tiene un clara tendencia e inclinación hacia el desarrollo de software, es claro para muchos estudiantes que este es uno de los puntos fuertes de la carrera y es por ello que muchos continúan su vida profesional por ese camino; por tal motivo desde los primeros semestres se trata de cultivar estas habilidades en ellos; considerando que el modelamiento y abstracción de sistemas son temas de vital importancia, se hace necesario disponer de herramientas automáticas (software) que permitan adquirir competencias en estas áreas y reforzar conceptos aprendidos en clase, pero que solo se consiguen mediante la práctica; además que permitan al estudiante forjar unos cimientos importantes para su vida profesional en aspectos tales como UML, Diagramas de clase, CASE Tools[[2]](#footnote-2) y razonamiento abstracto.

Por otro lado, se es consciente que actualmente ya existen herramientas muy completas que permiten realizar estas labores y que el programa académico ha venido utilizando desde hace algún tiempo (por ejemplo Enterprise Architect[[3]](#footnote-3), Netbeans UML, StarUML, ArgoUML), además que la universidad dispone de licencias de uso para algunas de ellas (por ejemplo, Enterprise Architect que es software privado), y otras son software libre de modo que pueden ser utilizados de cualquier forma; sin embargo, la mayoría de ellas, por no decir todas, son software que se instala de manera nativa, de modo que carecen de portabilidad y facilidad de acceso, aspectos que son muy importantes y atractivos hoy en día en los software de aplicación. Otras requieren ciertas características mínimas de hardware que aunque no es muy común, no siempre están disponibles y son suficientes para soportar que estas aplicaciones se ejecuten de manera adecuada.

Adicional pero no menos importante, se ha notado mediante observaciones indirectas y manifestaciones explicitas de algunos estudiantes, sobre todo en primeros semestres, que muchos de ellos aún tienen dificultades en la configuración de estas aplicaciones, otros expresan no tener computadoras propias o simplemente que los recursos hardware son insuficientes para el funcionamiento adecuado de las aplicaciones, aspectos que evidencian la necesidad planteada anteriormente e impulsan a diseñar modelos de servicios accesibles a través internet, los cuales podrían brindar ciertas flexibilidades y beneficios extras para los usuarios que no tienen las aplicaciones nativas, beneficios tales como facilidad de acceso, portabilidad, autoconfiguración, capacidad de compartir recursos e información.

## JUSTIFICACIÓN

El programa académico de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, tiene como objeto de estudio la “Construcción y Gestión de Sistemas de Información”[[4]](#footnote-4), por esta razón dentro del pensum existen muchos cursos que se enfocan en programación e ingeniería de software; esto significa que hay una clara tendencia a tratar temas como el modelado de aplicaciones, procesos de desarrollo y otros conceptos que hacen parte de esta área de la ingeniería de sistemas; de esta manera consideramos que se necesita que los estudiantes pongan en práctica técnicas y métodos para desarrollar y mantener software de calidad, teniendo siempre en mente que la tecnología existe para ser usada como herramienta facilitadora del trabajo.

Lo anterior sugiere la necesidad de disponer de una herramienta que permita realizar prácticas de laboratorio, más específicamente prácticas de abstracción de diagramas de clase, de modo que se puedan aplicar todos los conceptos aprendidos, bajo un ambiente real. Es por ello que nos hemos enfocado específicamente en este aspecto, de tal manera que se pueda obtener una herramienta adicional que sirva de apoyo al proceso de aprendizaje y permita adquirir con seguridad conceptos vitales y que serán de gran ayuda en la vida profesional de los estudiantes.

De igual manera sería ideal que los docentes puedan disponer de un mecanismo de control y supervisión de las actividades realizadas por sus estudiantes, permitiendo de manera fácil y cómoda la revisión de los trabajos de clase o evaluaciones parciales; por tal motivo se pretende construir una aplicación orientada bajo el modelo de servicios “Cloud Computing” [[5]](#footnote-5)(*en español, Computación en la nube*), e incorporar comportamientos como los desarrollados por “Google Docs”[[6]](#footnote-6), en donde los usuarios pueden compartir información con otros de manera fácil y cómoda a través de un click. Aplicando este concepto a la aplicación en construcción, se espera que los estudiantes puedan compartir con su docente los diagramas realizados de modo que el docente pueda consultarlos y corregirlos a través de la aplicación. Además es significativo incentivar el autoaprendizaje, permitiendo a los estudiantes el intercambio de conocimientos e información, de manera que puedan reforzar los conceptos, obviamente mediante algunos mecanismos de control que garanticen un proceso limpio y transparente.

Adicionalmente una motivación importante para el desarrollo de este proyecto es mejorar la infraestructura de servicios y de tecnologías de información disponibles para los estudiantes de la UFPS, mediante la construcción de herramientas innovadoras y que generen conocimientos sobre el desarrollo de aplicaciones web.

Por último, otro aspecto importante es la necesidad de impulsar e innovar en el desarrollo de tecnologías, el compromiso que tiene el plan de estudios de Ingeniería de Sistemas con la universidad y con la comunidad en general, para desarrollar alternativas de solución a problemáticas mediante el uso de la tecnología y generar conocimiento en base a investigación e innovación nos ha llevado a esta iniciativa, esperando propulsar el desarrollo de nuevos productos y herramientas que presten servicios dentro y fuera de la universidad, y que sean realizadas por los mismos estudiantes.

## OBJETIVOS

### Objetivo General

Implementar una aplicación web basada en el paradigma orientado a servicios de Cloud Computing, que permita el diseño y construcción de diagramas de clase UML y que sirva como medio de apoyo para proyectos de software realizados por los estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander.

### Objetivos Específicos

* Realizar un estudio de las principales características que poseen las herramientas CASE utilizadas por los estudiantes del programa de Ingeniería de Sistemas, para determinar los servicios que prestan en cuanto al diseño de diagramas de clase.
* Definir las funcionalidades que el software debe poseer, determinando las características específicas, los servicios que debe prestar a los usuarios y los procedimientos que debe llevar a cabo.
* Realizar el modelado del sistema, determinando una solución óptima a los procedimientos definidos, de modo que no se vea afectado el rendimiento y la estabilidad del software a construir.
* Realizar pruebas de funcionamiento que garanticen el comportamiento adecuado del software y que permitan verificar que se cumplan los requerimientos definidos.
* Documentar las pruebas realizadas junto con los procedimientos llevados a cabo y los resultados obtenidos.
* Desplegar la aplicación en un servidor de modo que pueda ser accedido por estudiantes y docentes.

## ALCANCES Y DELIMITACIONES

### Alcance

La ejecución de este proyecto pretende alcanzar lo siguiente:

* Obtener un producto software que permita a los usuarios diseñar de manera visual diagramas de clase UML.
* Disponer de una plataforma online que pueda ser utilizada desde cualquier lugar y a cualquier hora a través de internet.
* Disponer de una herramienta que permita a los docentes asesorar de manera continua y cómoda a sus estudiantes en los proyectos desarrollo de software.
* Un software para realizar modelado de objetos que sirva como soporte de los proyectos de software y permita generar código fuente de manera automática, agilizando el proceso de desarrollo.
* Un documento mediante el cual se pueda conocer el funcionamiento básico del software y los servicios que este presta, que sirva como guía a los usuarios sobre la manera en la cual deben utilizar la herramienta (Manual de usuario).

### Delimitaciones

Con el propósito de determinar la viabilidad y posibilidad de desarrollo del proyecto se han establecido las delimitaciones mencionadas a continuación:

* El software será totalmente independiente del Sistema de Información Académico de la Universidad (SIA), de modo que no se validara que los usuarios registrados estén en la base de datos del mismo. Sin embargo se tendrá en cuenta la información básica de los usuarios como: el código dentro de la Universidad, la facultad y el programa académico al cual pertenecen, entre otros.
* Solo se podrá realizar diagramas de clase UML, la aplicación no abarca el diseño de otro tipo de diagrama.
* El software no dispondrá de un administrador, de modo que no será necesario disponer de usuarios que realicen configuraciones específicas y únicas en el software.
* El código fuente generado a partir de los diagramas de clase será únicamente en lenguaje Java.
* Considerando que existen varios cursos de programación e Ingeniería de Software dentro del pensum académico actual del programa de Ingeniería de Sistemas, el proyecto no estará dirigido específicamente a un curso en particular, sino que estará disponible para cualquier estudiante en cualquier curso; sin embargo, se tomara una población base de uno o dos cursos para realizar el levantamiento de información. En el diseño metodológico se especificará con más detalle este tema.
* El software no permitirá editar el código fuente generado directamente, para esta labor se hace necesario que el usuario disponga de un editor.

# [MARCO REFERENCIAL](#TablaContenido)

## MARCO DE ANTECEDENTES

Después de realizar una investigación de las soluciones existentes a la problemática planteada, se encontraron algunas aplicaciones en el mercado que ofrecen parte de las características que esperamos obtener con el software en construcción. A continuación se mencionan algunas de estas aplicaciones:

### Draw.io

Draw.io es una aplicación de diagramado online desarrollada usando la librería javascript mxGraph creada por la compañía JGraph Ltd. Esta empresa fue fundada en el año 2000 y dedicada únicamente a la comercialización y soporte de componentes gráficos de visualización desde el 2004. Actualmente se ha impuesto en el mercado con su producto líder *mxGraph*, el cual comprende una librería javascript que permite dibujar componentes para diagramas en cualquier navegador web usando HTML, CSS, Javascript y SVG, además que permite realizar el backend[[7]](#footnote-7) de las aplicaciones en diferentes lenguajes como Java, PHP y .NET.

Draw.io es el sitio de demostración para la tecnología desarrollada por la empresa JGraph y su producto mxGraph, el cual permite dibujar diagramas creativos en el navegador, compartir y editar con otros usuarios en tiempo real. Esta aplicación puede ser accedida directamente desde la URL: <https://www.draw.io/>.

### Creately

Creately es una herramienta online UML para crear diagramas de Casos de uso, Diagramas de clase, Actividad, Secuencia, Modelos Entidad Relación, entre muchos otros. Esta es una herramienta comercial la cual es propiedad de la empresa CinergixPty Ltd., es el primer producto de la empresa y se encuentra actualmente en una etapa madura; es posible acceder a él en modo prueba aunque no se pueden guardar los proyectos y las funciones son un poco limitadas; para poder tener a disposición toda la herramienta es necesario comprar una licencia de uso. Esta herramienta permite la generación de código fuente, documentación, ingeniería inversa, entre otras funciones; características que lo hacen muy llamativo aunque cuenta con la limitante de ser software propietario.

### GWTUML

GWTUML es una herramienta open source desarrollada por la empresa francesa Object Direct, dirigido por el ODLabsdepartament. Esta es una compañía dedicada a la consultoría y el desarrollo de tecnologías de información utilizando metodologías de desarrollo ágil, orientación a objetos y Cloud computing, brindando soporte a proyectos open source.

GWTUML es una herramienta CASE en proceso de desarrollo, la cual permite el modelado visual de diagramas de clase, diagramas de objetos y diagramas de secuencia. El desarrollo de esta herramienta se inició en enero de 2009 y la versión actual aún se encuentra en modo experimental, sin embargo, los desarrolladores han liberado algunas APIs[[8]](#footnote-8) y versiones beta de la aplicación, las cuales pueden ser descargadas y utilizadas en otros proyectos que requieran sus servicios. Adicionalmente existe alguna documentación del código fuente (Javadocs) de modo que los usuarios pueden consultarlos en caso de utilizar las APIs disponibles.

Este proyecto está basado en GWT[[9]](#footnote-9) (Google Web Toolkit) el cual es un framework para el desarrollo de aplicaciones web en lenguaje java, liberando al desarrollador de la complejidad implícita de varios aspectos de la tecnología AJAX, que a su vez es la herramienta más utilizada hoy en día para la construcción de aplicaciones para la internet

### JoinJS

JoinJS es una librería JavaScript para crear diagramas totalmente interactivos en la web. Esta librería puede ser usada tanto para implementar herramientas de diagramado o para simplemente publicar diagramas online.

Las principales características de esta librería son:

* Permite conectar objetos vectoriales con varios tipos de líneas y flechas.
* Interacción con objetos y conectores.
* Personalizar eventos y manejadores para mouse, teclado, entre otros.
* Elementos pre construidos para diagramas comúnmente usados (Entidad relación, UML, Org Chart, entre otros).
* Serialización (formato JSON, exportación a SVG en navegadores que lo permiten).
* Soporte para la mayoría de navegadores en versiones antiguas.

Página principal de la herramienta: [http://www.jointjs.com](http://www.jointjs.com/).

## MARCO CONCEPTUAL

### Herramientas CASE

Se puede definir a las Herramientas CASE (Computer Aided Software Engineering) como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Como es sabido, los estados en el Ciclo de Vida de desarrollo de un Software son: Investigación Preliminar, Análisis, Diseño, Implementación e Instalación.

La realización de un nuevo software requiere que las tareas sean organizadas y completadas en forma correcta y eficiente. Las Herramientas CASE fueron desarrolladas para automatizar esos procesos y facilitar las tareas de coordinación de los eventos que necesitan ser mejorados en el ciclo de desarrollo de software.

La mejor razón para la creación de estas herramientas fue el incremento en la velocidad de desarrollo de los sistemas. Por esto, las compañías pudieron desarrollar sistemas sin encarar el problema de tener cambios en las necesidades del negocio, antes de finalizar el proceso de desarrollo.

Las herramientas CASE también permiten a los analistas tener más tiempo para el análisis y diseño y minimizar el tiempo para codificar y probar.

Con un CASE integrado, las organizaciones pueden desarrollar rápidamente sistemas de mejor calidad para soportar procesos críticos del negocio y asistir en el desarrollo y promoción intensiva de la información de productos y servicios.

Estas herramientas pueden proveer muchos beneficios en todas las etapas del proceso de desarrollo de software, algunas de ellas son:

♦Verificar el uso de todos los elementos en el sistema diseñado.

♦Automatizar el dibujo de diagramas.

♦Ayudar en la documentación del sistema.

♦Ayudar en la creación de relaciones en la Base de Datos.

♦Generar estructuras de código.

La introducción de las herramientas CASE para ayudar en este proceso ha permitido que los diagramas puedan ser fácilmente creados y modificados, mejorando la calidad de los diseños de software. Los diccionarios de datos, un documento muy usado que mantiene los detalles de cada tipo de dato y los procesos dentro de un sistema, son el resultado directo de la llegada del diseño de flujo de datos y análisis estructural, hecho posible a través de las mejoras en las Herramientas CASE.

Pronto se reemplazaron los paquetes gráficos por paquetes especializados que habilitan la edición, actualización e impresión en múltiples versiones de diseño. Eventualmente, las herramientas gráficas integradas con diccionarios de base de datos para producir poderosos diseños y desarrollar herramientas, podrían sostener ciclos completos de diseño de documentos. Como un paso final, la verificación de errores y generadores de casos de pruebas fueron incluidos para validar el diseño del software. Todos estos procesos pueden saberse integrados en una simple herramienta CASE que soporta todo el ciclo de desarrollo.

La primera herramienta comercial se remonta a 1982, aunque algunos especialistas indican que algunos ejemplos de herramientas para diagramación ya existían.

No fue sino hasta 1985 en que las herramientas CASE se volvieron realmente importantes en el proceso de desarrollo de software. Los proveedores prometieron a la Industria que muchas actividades serían beneficiadas por la ayuda de las CASE. Estos beneficios consistían, por ejemplo, en el aumento en la productividad. El objetivo en 1985 para muchos vendedores era producir software más rápidamente. Las herramientas del CASE serían una familia de métodos favorablemente estructurados para planeamiento, análisis y diseño. Esto llevaría a la generación automática de código para desarrollo de software vía una especificación formalmente diseñada.

Las herramientas CASE, en función de las fases del ciclo de vida abarcadas, se pueden agrupar de la forma siguiente:

1. Herramientas integradas, I-CASE (Integrated CASE, CASE integrado): abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.

2. Herramientas de alto nivel, U-CASE (Upper CASE - CASE superior) o front-end, orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.

3. Herramientas de bajo nivel, L-CASE (Lower CASE - CASE inferior) o back-end, dirigidas a las últimas fases del desarrollo: construcción e implantación.

4. Juegos de herramientas o Tools-Case, son el tipo más simple de herramientas CASE. Automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

### Cloud Computing

La nube (Cloud Computing) es el procesamiento masivo de datos y almacenamiento de información en servidores, ubicados en cualquier parte del mundo,  conectados a una conexión de internet y a los que se puede acceder desde cualquier dispositivo: computador, Smartphone, tabletas, etc.

En los años 70’s John McCarthy, famoso científico e informático ganador del Premio Turing, expresó: “algún día la computación podrá ser organizada como un servicio público”. El primer uso académico de la expresión “computación en nube”, fue durante una conferencia de Ramnath Chellappa (actual profesor de la Emory University) en 1997. Desde ese momento el famoso concepto de la nube ha venido trascendiendo y hoy en día estamos entiendo con más claridad a qué se referían estos visionarios de la tecnología.

EL NIST (National Institute of Standards and Technology) de EE.UU lo define como: “Cloud Computing es un modelo para permitir el acceso adecuado y bajo demanda a un conjunto de recursos de cómputo configurables (ejemplos: redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provistos y puestos a disposición del cliente con un mínimo esfuerzo de gestión y de interacción con el proveedor del servicio”.

Sin embargo, para algunos el concepto de nube sigue siendo confuso. Lo cierto es que muchas personas  ya realizan actividades diarias en la nube y lo desconocen. Cada vez que alguien consulta  su correo electrónico,  hace una transacción bancaria por Internet,  ve un video de su artista favorito en YouTube, sube fotos de una fiesta en Flickr, escucha música en  Grooveshark o  envía documentos por Dropbox, está  viviendo en la nube.

### UML

El UML (Lenguaje Unificado de Modelado) es una de las herramientas más emocionantes en el mundo actual de desarrollo de sistemas. Estos se debe a que permite a los creadores de sistemas generar diseños que capturen sus ideas en una forma convencional y fácil de comprender para comunicarlas a otras personas.

La comunicación de la idea es de suma importancia. Antes del advenimiento del UML, el desarrollo de sistemas era, con frecuencia, una propuesta al azar. Los analistas de sistemas intentaban evaluar los requerimientos de sus clientes, generar un análisis de requerimientos en algún tipo de notación que ellos mismos comprendieran (aunque el cliente no lo comprendiera), dar tal análisis a uno o más programadores y esperar que el producto final cumpliese con lo que el cliente deseaba.

Dado que el desarrollo de sistemas es una actividad humana, hay muchas posibilidades de cometer errores en cualquier etapa del proceso, por ejemplo, el analista pudo haber malentendido al cliente, es decir, probablemente produjo un documento que el cliente no pudo comprender. Tal vez ese documento tampoco fue comprendido por los programadores quienes, por ende, pudieron generar un programa difícil de utilizar y no generar una solución al problema original del cliente.

Hoy en día, es necesario contar con un plan bien organizado. Un cliente tiene que comprender que es lo que hará un equipo de desarrolladores; además tiene que ser capaz de señalar cambios si no se han captado claramente sus necesidades (o si cambia de opinión durante el proceso). A su vez, el desarrollo es un esfuerzo orientado a equipos, por lo que cada uno de sus miembros tiene que saber qué lugar toma su trabajo en la solución final (así como saber cuál es la solución general).

La clave está en organizar el proceso de diseño de tal forma que los analistas, clientes, desarrolladores y otras personas involucradas en el desarrollo del sistema lo comprendan y convengan con él. El UML proporciona tal organización.

El UML es la creación de Grady Booch, James Rumbaugh e Ivar Jacobson. Estos caballeros, apodados recientemente “Los tres amigos”, trabajaban en empresas distintas durante la década de los años ochenta y principios de los noventa y cada uno diseño su propia metodología para el análisis y diseño orientado a objetos. Sus metodologías predominaron sobre la de sus competidores. A mediados de los años noventa empezaron a intercambiar ideas entre si y decidieron desarrollar su trabajo en conjunto.

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 de UML. Entre los miembros se encuentran DEC, Hewlett-Packard, Intellicorp, Microsoft, Oracle, Texas Instruments y Rational. En 1997 el consorcio produjo la versión 1.0 del UML.

El UML está compuesto por diversos elementos gráficos que se combinan para conformar diagramas. Debido a que el UML es un lenguaje, cuenta con reglas para combinar tales elementos entre los cuales encontramos: diagrama de clases, diagrama de objetos, diagrama de casos de uso, diagrama de estados, diagrama de secuencias, diagrama de actividades, diagrama de colaboraciones, diagrama de componentes y diagrama de distribución.

### Diagramas de Clase

La técnica del diagrama de clase se ha vuelto medular en los métodos orientados a objetos. Virtualmente, todos los métodos han incluido alguna variación de esta técnica.

El diagrama de clase, además de ser de uso extendido, también está sujeto a la más amplia gama de conceptos de modelado, aunque los elementos básicos son necesarios para todos, los conceptos avanzados se usan con mucha menor frecuencia,

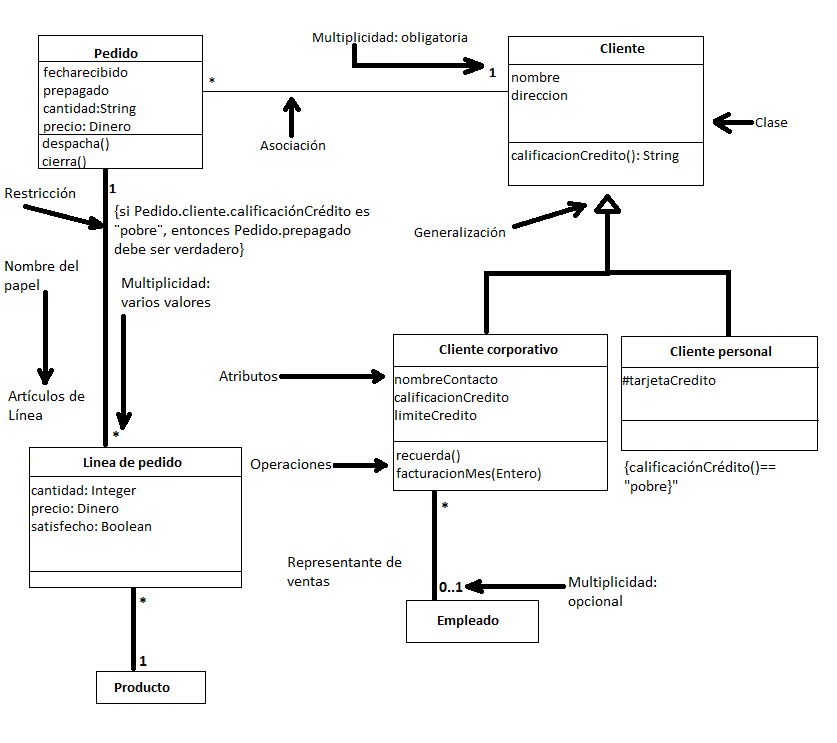
El diagrama de clase describe los tipos de objetos que hay en el sistema y las diversas clases de relaciones estáticas que existen entre ellos. Hay dos tipos principales de relaciones estáticas:

Asociaciones: (Por ejemplo, un cliente puede rentar diversas videocintas).

Subtipos: (Una enfermera es un tipo de persona).

Los diagramas de clase también muestran los atributos y operaciones de una clase y las restricciones a que se ven sujetos, según la forma en que se conecten los objetos.

La siguiente ilustración es un ejemplo de un Diagrama de clases en el cual se indicaran los principales elementos y características de un diagrama de clases.



Algunos conceptos importantes sobre diagramas de clase:

**Clase:** Es la definición de las características concretas de un determinado tipo de objeto. Es decir, de cuáles son los datos y los métodos de los que van a disponer todos los objetos de ese tipo. Por esta razón, se suele decir que el tipo de dato de un objeto es la clase que define las características del mismo.

**Asociación:** Representan relaciones entre instancias de clases (una persona trabaja para una empresa; una empresa tiene cierta cantidad de oficinas).

**Multiplicidad:** Es una indicación de la cantidad de objetos que participaran en la relación dada. En la figura anterior, la \* entre Cliente y Pedido indica que el primero pude tener muchas Órdenes asociadas a él; el 1 indica que un Pedido viene de un solo Cliente.

En general, la multiplicidad indica los límites inferior y superior de los objetos participantes. El \* representa de hecho el intervalo 0…infinito: el Cliente no necesita haber colocado un Pedido, y no hay un tope superior (por lo menos en teoría) para la cantidad de Pedidos que puede colocar. El equivale a 1..1: cada Pedido debe haber sido solicitado por un solo Cliente.

**Atributos:** El atributo de nombre de un Cliente indica que los Clientes tienen nombres. Desde el nivel de especificación, este atributo indica que un objeto Cliente puede decir su nombre y tiene algún modo de establecer un nombre. En el nivel de implementación, un Cliente tiene un campo (también llamado variable de instancia o miembro de datos) para su nombre.

Dependiendo del detalle del diagrama, la notación de un atributo puede mostrar el nombre, el tipo y el valor predeterminado de un atributo (la sintaxis del UML es visibilidad nombre: tipo = valor por omisión).

Operaciones:

Generalización o Herencia:

Composición:

Agregación:

### Procesos de Desarrollo de Software

### Meta-Programación

## MARCO TEORICO

### Java

### Frameworks

### Servidor Web

### Servidor de Aplicaciones

### Extreme Programming (XP)

### PrimeFaces

### SVG

### HTML, JavaScript y CSS

## MARCO DE LEGAL

### Acuerdos de uso

El software, resultado de la ejecución del proyecto, se regirá bajo los marcos legales fijados por la licencia pública CreativeCommon (LPCC) que dicta lassiguientes condiciones de uso, encontradas más detalladamente en el documento Atribución No Comercial Compartir Igual 2.5 (Colombia)[[10]](#footnote-10):

* Se posee la libertad de compartir la obra, entendiendo compartir como la capacidad de copiar, distribuir, ejecutar y comunicar públicamente la obra.
* Además puede llevar a cabo obras derivadas de la original.
* Es obligatorio reconocer los créditos de la obra de manera especificada por el autor o el licenciante, pero no de manera que sugiera que tiene el apoyo de este último, o que apoya el uso que hacen a su obra.
* No puede ser utilizada para uso comercial.
* Si se altera o transforma, o genera un derivado a partir de esta obra, solo puede ser distribuida bajo una licencia idéntica a la presente.

### Contrato de licencia de código binario, SUN MICROSYSTEMS

Licencia de uso de software concedidos por SunMicrosystems, para la utilización de Java SE, siempre y cuando sean aceptados los términos de licencia, expuestos en el documento SUN MICROSYSTEMS, INC. CONTRATO DE LICENCIA DE CÓDIGO BINARIO[[11]](#footnote-11).

“SUN MICROSYSTEMS, INC. (EN ADELANTE DENOMINADO “SUN”) LE CONCEDE LA LICENCIA DEL SOFTWARE DEFINIDO A CONTINUACIÓN ÚNICAMENTE CON LA CONDICIÓN DE QUE USTED ACEPTE TODOS LOS TÉRMINOS ESTIPULADOS EN EL PRESENTE CONTRATO DE LICENCIA DE CÓDIGO BINARIO Y TÉRMINOS DE LICENCIA ADICIONALES (EN CONJUNTO DENOMINADOS “CONTRATO”). POR FAVOR, LEA EL CONTRATO DETENIDAMENTE. EL USO DEL SOFTWARE SIGNIFICA QUE HA LEÍDO LAS CONDICIONES Y QUE LAS ACEPTA. SI ACEPTA ESTAS CONDICIONES EN REPRESENTACIÓN DE UNA COMPAÑÍA U OTRA ENTIDAD LEGAL, SIGNIFICA QUE ESTÁ EN POSESIÓN DE LA AUTORIDAD QUE LE PERMITE VINCULAR LA ENTIDAD LEGAL A ESTAS CONDICIONES. SI NO ESTÁ EN POSESIÓN DE DICHA AUTORIDAD O SI NO DESEA QUEDAR VINCULADO A ESTAS CONDICIONES, NO DEBE UTILIZAR EL SOFTWARE EN ESTA INSTALACIÓN NI EN NINGÚN OTRO SOPORTE EN EL QUE SE UBIQUE EL SOFTWARE”.

### GNU General PublicLicence

La Licencia Pública General de GNU (GNU GPL, por sus siglas en inglés) es una licencia libre y gratuita con derecho de copia para software y otros tipos de obras.

Las licencias para la mayoría del software y otras obras de índole práctica están diseñadas para privarle de la libertad para distribuir y modificar las obras. Por el contrario, la Licencia Pública General de GNU garantiza la libre distribución y modificación de todas las versiones de un programa, a fin de asegurarle dicha libertad a todos los usuarios. En la Fundación para el Software Libre utilizamos la Licencia Pública General de GNU para la mayoría de nuestro software; también se aplica a cualquier otra obra publicada de esta manera por sus autores. Usted también puede aplicarla a sus programas.

Cuando hablamos de software libre, nos referimos a la libertad, no al precio. Nuestras Licencias Públicas Generales están diseñadas para garantizarle a usted la libertad de distribuir copias de software libre (y cobrar por ellas, si así lo desea), obtener el código fuente, o tener la posibilidad de obtenerlo, modificar el software o utilizar partes del mismo en nuevos programas libres, y saber que puede hacer estas cosas[[12]](#footnote-12).

### Propiedad Intelectual

De acuerdo con lo estimulado en el artículo 156 del acuerdo 065 del 26 de 1996, correspondiente al estatuto estudiante de la Universidad Francisco de Paula Santander, el cual dicta de la siguiente manera: “Los trabajos de grado son propiedad intelectual de la Universidad y su uso estará sujeto a las normas que para tal fin estén vigentes”[[13]](#footnote-13).

# [DISEÑO METODOLÓGICO](#TablaContenido)

El diseño metodológico del proyecto está enmarcado en la investigación aplicada y se fundamentará en la ingeniería de software, de ella se tomarán los conceptos, metodologías y técnicas propias para la construcción de la aplicación.

## TIPO DE INVESTIGACIÓN

Según la problemática planteada, el proyecto pretende dar solución a una necesidad vista dentro del programa de Ingeniería de Sistemas de la UFPS, de tal manera que se englobará el proyecto como una INVESTIGACIÓN APLICADA en términos del desarrollo de software que permitan solventar necesidades existentes dentro de la organización.

## FUENTES DE INFORMACIÓN

### Fuentes de información primaria

Teniendo en cuenta que la problemática afecta a la comunidad perteneciente al programa de Ingeniería de Sistemas, se consideran las siguientes fuentes de información primaria:

* Estudiantes que están cursando las asignaturas, “Programación Orientada a Objetos” y “Análisis y diseño de Sistemas”, las cuales se enfocan principalmente en el modelado de aplicaciones. Estos son cursos de 2º y 6º semestre respectivamente según el pensum 115[[14]](#footnote-14). No obstante, a manera de observación indirecta se tomaran otros cursos que puedan usar el producto en construcción.
* Docentes del programa Ingeniería de Sistemas, los cuales son fuente de información de vital importancia, ya que ellos son los que acompañan el proceso de aprendizaje de los estudiantes.

### Fuentes de información secundaria

Estas fuentes hacen referencia a estudios, técnicas, teorías, documentos y toda clase de información existente sobre el tema en particular de modo que se establecen las siguientes:

* Herramientas CASE existentes, comerciales o de código abierto, que permitan el diseño de diagramas de clase. Se tomará en cuenta las herramientas utilizadas actualmente por los estudiantes, de modo que se pueda determinar que funcionalidad o que servicios ofrecen.
* Manuales o documentación existente sobre las herramientas mencionadas anteriormente.
* Documentación de proyectos de software online que permitan diseñar diagramas de clase.

Libros enfocados en temas como: Cloud computing, Ingeniería de Software, Lenguaje Unificado de Modelado, Procesos de desarrollo de software, desarrollo de aplicaciones Web, desarrollo de aplicaciones con J2EE, AJAX.

## RECOLECCIÓN Y ANALISIS DE INFORMACIÓN

El desarrollo del proyecto requiere obtener información general sobre la construcción de diagramas de clase, esto implica conocer a fondo conceptos tales como: Clases, Interfaces, Relaciones, Navegabilidad, entre otros; además de todos los elementos de los cuales dispone UML para el diseño de este tipo de diagramas. Para esto se realizará una investigación profunda sobre el tema, consultado bibliografía existente y diferentes autores.

Adicionalmente se necesita determinar las funcionalidades básicas que poseen las herramientas CASE utilizadas actualmente por los estudiantes de la UFPS, de manera que se pueda conocer qué procedimientos, servicios y características poseen con respecto al diseño de diagramas de clase; esto se hace con el objetivo de determinar los requerimientos de funcionalidad mínimos que debe poseer el software en construir y los servicios que este debe ofrecer. De cierta manera, mediante observación indirecta y a manera de tradición se conocen las principales herramientas CASE utilizadas por estudiantes y docentes de Ingeniería de Sistemas de la UFPS (Enterprise Architect, ArgoUML, StarUML), y considerando que el programa académico dispone de algunas licencias de uso para este tipo de herramientas (como es el caso de Enterprise Architect), se procederá a realizar un análisis concreto de estas herramientas específicamente. De igual manera y como se mencionó anteriormente, este análisis servirá como punto de partida del proyecto y además ayudará a tener un idea clara de la aplicación que se desea llegar a construir.

El procedimiento utilizado para la recolección de información en este caso será de manera literaria, mediante consultas e investigaciones de material bibliográfico, logrando una apropiación de conceptos y técnicas importantes para la construcción del proyecto.

Por último es necesario aclarar que este proyecto no pretende dar respuestas a incógnitas o hipótesis que se tengan sobre algún fenómeno académico o social dentro de la Universidad, sino que se encargara de suplir una necesidad detectada dentro del establecimiento.

Para el análisis de la información se procederá a clasificar los datos según su naturaleza, determinando los aspectos importantes para cada fase del desarrollo y la información necesaria para construir los componentes en cada una de ellas.

El análisis de la información consistirá en evaluar los datos obtenidos de las fuentes mencionadas, documentación de las herramientas CASE existentes, manuales de usuario, análisis del funcionamiento de las aplicaciones utilizadas actualmente, literatura especifica del caso, y en general, fuentes que sean cercanas al tema. Se realizará un proceso de clasificación y agrupamiento de los datos, de manera que se pueda mediante ellos, apoyar la siguiente fase del desarrollo y que sean las directrices de todo el proyecto en general, de modo que mediante estos se puedan tomar decisiones importantes para el desarrollo.

# ANALISIS DE LAS HERRAMIENTAS CASE

Con el objetivo de obtener información importante para la realización del proyecto y con el ánimo de iniciar el proceso de levantamiento de requerimientos se ha realizado un pequeño análisis de algunas de las herramientas utilizadas dentro del plan de estudios de Ingeniería de Sistemas para realizar modelado UML, de modo que se pueda establecer cuáles son las principales características y servicios que estas ofrecen a los usuarios en cuanto al diseño de *Diagramas de Clase* y que puedan ser incluidos en las funcionalidades del software en construcción.

De modo aleatorio se han seleccionado dos de las herramientas más conocidas y que a manera de observación se ha determinado que son ampliamente utilizadas por estudiantes y docentes del plan de estudios, adicionalmente se ha seleccionado una herramienta similar a la que está en construcción, de modo que podamos conocer qué características ofrece a los usuarios y cuales son imprescindibles en una herramienta de modelamiento UML. Las herramientas seleccionadas son: *Enterprise Architect* que es un software comercial conocido a nivel mundial por su gran calidad, *StarUML* que a su vez es open source pero ampliamente utilizado, y por ultimo una herramienta de modelado online llamada *Draw.IO*la cual se asemeja mucho a lo que estamos tratando de lograr.

Al momento de realizar este análisis se tomó la última versión de las aplicaciones que se instalan de manera nativa, Enterprise Architect en su versión 9.0.9 con un periodo de prueba de 30 dias yStarUMLen su versión 5.0. En el caso de Draw.IO la librería mxGraph usada para la construcción de esta aplicación se encuentra en la versión 1.10.4.0.

A continuación se describirá cada uno de ellos, resaltando en detalle cada una de sus características:

## ENTERPRISE ARCHITECT

Enterprise Architect de Sparx Systemses una herramienta CASE (Computer Aided Software Engineering) para el diseño y construcción de sistemas de software, para el modelado de procesos de negocios, y para objetivos de modelado más generalizados.

EA es una herramienta de análisis y diseño UML comprensivo, que cubre el desarrollo de software desde la obtención de los requisitos, diseño del modelo, pruebas, cambio de control y mantenimiento para la implementación, con completa trazabilidad. EA combina el poder de la última especificación UML 2.4.1 con alta performance, una interfaz intuitiva, para brindar un modelado avanzado a todo el equipo de desarrollo. Es una herramienta visual con multi-usuario con un gran establecimiento de características ayudando a los analistas, testers, administradores de proyectos, personal del control de calidad y desarrolladores alrededor del mundo a construir y documentar, software sostenible.

### Historia

Sparx Systems es una empresa australiana dedicada a la creación de herramientas para el modelado UML, fundada en 1996 por Geoffrey Sparx y situada en Creswick una pequeña población de Victoria, Australia. La primera versión de Enterprise Architect fue liberada en el año 2000 y fue originalmente diseñada como una herramienta de modelado UML para la versión 1.1 del estándar, de allí en adelante el producto ha evolucionado para soportar otras especificaciones OMG[[15]](#footnote-15) UML como 1.3, 2.0, 2.1, 2.3 y 2.4.1.

Durante el tiempo de vida de la aplicación se han liberado alrededor de 20 versiones, desde la versión 1.0 hasta la más reciente 9.3 liberada en Agosto de 2012, cada una con mejoras y características importantes según los cambios que ha sufrido el estándar UML y la adopción de algunas tecnologías

Actualmente la cantidad de estándares para el diseño y modelado de software y sistemas de negocios que son soportados por Enterprise Architect es considerable, teniendo en cuenta la adopción de UML 2.4.1 como principal herramienta para modelar aplicaciones e incluyendo algunos otros BPMN, WSDL, BPEL y XSD, ha permitido que la herramienta se imponga como uno de los líderes del mercado y que su aceptación por parte de la industria del software haya sido grandiosa.

Enterprise Architect se encuentra disponible en diferentes ediciones, según las necesidades del usuario o de las organizaciones que usan la aplicación, de modo que se pueden adquirir diferentes tipos de licencias para diferentes usos. Actualmente existen dos ediciones y dentro de las cuales se pueden comprar varias versiones del producto:

* Ediciones base
* Corporativa
* Profesional
* Escritorio
* Ediciones Suite
* Ultimate
* Ingeniería de Sistemas
* Ingeniería de Software y negocios

Los precios del producto dependen de la versión adquirida y de la cantidad de licencias que se obtengan de cada uno, siendo la versión de Escritorio la más simple hasta la versión Ultimate que a su vez posee la capacidad de trabajo en múltiples dominios de manera remota y un soporte completo de ingeniería de software y de negocios.

### Características

Enterprise Architect posee un gran abanico de características, todas encaminadas al desarrollo y mantenimiento de proyectos de software, aunque también brinda soporte para otro tipo de proyectos.

La información aquí consignada fue tomada de la documentación explicita de Enterprise Architect, disponible en la página web, además se usó parte de la información del “Contenido de Ayuda” que viene incluido en la versión del software.

A continuación mencionaremos varias de las características generales de EA:

| **Característica** | **Descripción** |
| --- | --- |
| Plataforma | Diseñado para ser usado bajo sistemas operativos Windows, sin embargo puede ser instalado y utilizado mediante emuladores como CrossOver en sistemas Linux o MAC. CrossOver es una herramienta propietaria basada en una tecnología open source llamada Wine, el cual es una API Win32 re-desarrollada para sistemas operativos basados en Unix como Mac OS X o Linux. |
| Categoría de Software | Software propietario distribuido bajo licencias comerciales, el costo de estas depende de la versión del software y la cantidad de licencias adquiridas. |
| Versión de UML | UML 2.4.1 Liberada en agosto de 2011. |
| Diagramas UML | *Estructurales*: Clases, Objetos, Paquetes, Estructura Compuesta, Componentes, de Perfiles y Despliegue.  *Comportamentales*: Actividad, Casos de Uso, de Estados, de Tiempo, Secuencia, Comunicación e Interacción.  *Adicionales no UML*: Análisis, Requerimientos, Bases de Datos, Mantenimiento, Interfaces de Usuario, Interacción y modelos del negocio. |
| Tecnologías MDG[[16]](#footnote-16) | Soporta tecnologías comoICONIX, ArcGIS, BPMN, SMOF, SoaML, SysML. |
| Fases del desarrollo | Modelamiento del Negocio, Ingeniería de Requerimientos, Análisis, Diseño, Implementación, Pruebas, Mantenimiento, Gestión de la Calidad. |
| Estándar XMI | Soporte completo del estándar en su versión 2.1, permite exportar e importar proyectos en archivos .XMI de modo que puedan ser llevados a otras herramientas de modelado. |
| Proyectos y equipos de trabajo | Permite generar repositorios de servidor DBMS[[17]](#footnote-17), de modo que se pueda trabajar de manera concurrente en proyectos de gran envergadura y con gran cantidad de usuarios.  Adicionalmente permite integración con repositorios de control de versiones como Subversion, CVS y SCC. |
| Importación / Exportación | *Exportación*: Existen varias formas de exportar proyectos, diagramas o cualquier otro tipo de artefacto creado dentro del software, archivos XML, XMI, Código fuente, imágenes planas de los diagramas en formato PNG, JPG, GIF, entre otros.  *Importación*: Desde XML, XMI, archivos CSV[[18]](#footnote-18), Desde código fuente en varios lenguajes, desde archivos binarios como JARs o ensamblados .NET. |
| Ingeniería Directa |  |
| Ingeniería Inversa |  |
| Documentación generada | Soporte completo de WYSIWYG[[19]](#footnote-19) manejador de plantillas para generar archivos RTF. |
| Soporte de Versiones |  |
| Seguridad |  |

Tabla 1. Listado de características de Enterprise Architect

## STARUML

## DRAW.IO

# CLASIFICACIÓN DE LA INFORMACIÓN

## MATRIZ DE CARACTERISTICAS

## CARACTERISTICAS PROPIAS

## LISTA DE FUNCIONALIDADES

# [METODOLOGÍA DE DESARROLLO](#TablaContenido)

La metodología de desarrollo seleccionada para el proyecto fue*eXtreme Programming* (XP), debido a su naturaleza y al tamaño del equipo de trabajo se puede adaptar fácilmente al desarrollo del proyecto, considerando que brinda flexibilidad y la posibilidad de adaptación a los cambios que pueden surgir en el transcurso del proyecto.

Uno de los principales principios del

## EXPLORACIÓN

### Identificación de Roles del Proyecto

El desarrollo del proyecto será llevado a cabo por un grupo pequeño de personas, sin embargo cada una tiene una labor específica e importante para la correcta ejecución del mismo. A continuación se describen los roles del proyecto y su participación en el proyecto:

* **Desarrollador**: Gabriel Leonardo DíazCárdenas, estudiante de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander.
* **Cliente**:En un proceso de desarrollo ágil es de vital importancia para el desarrollo del proyecto, sin embargo en nuestro caso no tenemos un ente o persona específica a la cual se le entregará la aplicación, en su lugar tenemos un grupo general de personas a las cuales irá dirigido el proyecto:

“Estudiantes del Plan de Estudios de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, en especial, estudiantes que cursan materias como Programación Orientada a Objetos y Análisis y Diseño de sistemas. Adicionalmente los Docentes que hacen parte de estas cursos del pensum academico”.

* **Director de Proyecto**: Marco Antonio Adarme Jaimes, Msc. en Computación. Director general del proyecto quien estará controlando su ejecución y orientando el proceso de implementación, además quien estará verificando que se cumplan los objetivos planteados al inicio del proyecto.
* **Asegurador de la Calidad**: Las labores de verificación y pruebas del funcionamiento de la aplicación estarán a cargo del Msc. Margo Antonio Adarme Jaimes, a quien se le entregaran las liberaciones de cada iteración (mencionadas más adelante) y quien dará el visto de aprobación.
* **Consultor**: Este rol es asumido por personas externas, expertos en algunos de los temas que vamos a tratar durante la ejecución del proyecto.

### Identificación de Actores del Sistema

En UML se define un *actor* como aquella persona, proceso o cosa que interactúa con un sistema, subsistema o clase[[20]](#footnote-20). Para nuestro proceso de desarrollo es de vital importancia determinar cuáles son los actores de nuestro sistema, de modo que se pueda tener una idea clara de las necesidades conjuntas y particulares de cada uno de ellos. A continuación se mencionan cada uno de los actores y su descripción:

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **VISITANTE** | |
| Representa cualquier persona que llega a la aplicación en búsqueda de información, ya sea de manera accidental o voluntaria. Este actor no tiene una representación real dentro de la aplicación y solamente es manejado de manera conceptual, en efecto cualquier persona inicia siendo un visitante de la aplicación. | |
| **Clase** | Ninguna |
| **Super Clase** |  |
| **Interacción con la aplicación** | | |
| * Busqueda de Información o curiosidad. * Registrarse en el sistema. | | |

Tabla 2. Especificación del actor Visitante

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **Usuario** | |
| Representa una persona que hace login dentro de la aplicación y que le es permitido usar el diseñador de diagramas. Este es un actor abstracto y necesita ser redefinido en una representación concreta. Dentro de la aplicación Usuario es una interfaz implementada por dos clases. | |
| **Clase** | *IUser* |
| **Super Clase** |  |
| **Interacción con la aplicación** | | |
| * Inicio de Sesión * Cerrar Sesión * Ver demostración: Esta opción puede ser usada por cualquier visitante, sin embargo en el momento de ingresar a la aplicación se convierte en un usuario invitado. * Diseñar diagrama * Exportar XMI * Generar Codigo * Generar Imagen | | |

Tabla 3. Especificación del actor Usuario

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **Invitado** | |
| Un invitado en la aplicación es la representación de un usuario que utiliza la opción de *Ver Demostración* e inicia sesión dentro de la aplicación sin registro previo. Este actor posee una representación real dentro de la aplicación, sin embargo sus privilegios son limitados y no tiene acceso a algunas funcionalidades. | |
| **Clase** | *Guest* |
| **Super Clase** | *IUser* |
| **Interacción con la aplicación** | | |
| * Todas las interacciones heredadas | | |

Tabla 4. Especificación del actor Invitado

|  |  |  |
| --- | --- | --- |
| Visitante.bmp | **Diagramador** | |
| Representa una persona registrada en la aplicación mediante el formulario designado para tal labor. A esta persona se le asigna una cuenta usuario mediante su dirección de correo electrónico, y por medio de la cual ingresa a la aplicación de manera controlada para hacer uso de todos los servicios que están a su disposición. | |
| **Clase** | *User* |
| **Super Clase** | *IUser* |
| **Interacción con la aplicación** | | |
| * Todas las interacciones heredadas * Ver panel de control (Dashboard) * Gestionar diagramas (Crear, editar, borrar, copiar) * Compartir diagramas * Importar XMI * Gestionar Perfil * Cambiar Password * Desactivar cuenta * Reactivar cuenta * Recuperar contraseña | | |

Tabla 5. Especificación del actor Diagramador

Dentro de la aplicación nose dispone de un usuario administrador, ya que no existen tareas o configuraciones especiales que se tengan que hacer de manera controlada. Cada usuario es capaz de interactuar con la aplicación de la misma manera y todos ellos disponen de los mismos privilegios. Esto evita la necesidad de tener a una persona realizando configuraciones en la aplicación y la libera de este tipo de dependencia.

Ambos *Invitado* y *Diagramador* son *Usuarios* de la aplicación debido a que pueden iniciar sesión dentro de ella e ingresar al diseñador. Los actores pueden ser representados mediante el siguiente diagrama:



Ilustración 1. Diagrama de Actores

### Historias de Usuario

**H1. Crear cuenta de usuario**

Mediante esta funcionalidad un visitante puede crear una cuenta de usuario para iniciar sesión en el sistema. Para ello, dispone de un formulario de registro donde se captura información básica del visitante, como los nombres, apellidos, correo electrónico y contraseña. El correo electrónico es usado como identificador de la cuenta de usuario. Una vez creada la cuenta de usuario, esta permanece inactiva hasta que el usuario realice la activación respectiva mediante el correo de confirmación. Este correo se envía a la dirección ingresada al momento de crear la cuenta.

**H2. Activar cuenta de usuario**

Este es un requerimiento para poder usar la aplicación una vez se ha registrado como usuario. La activación se hace mediante un link que se envía a la dirección de correo que el usuario ingresó al momento del registro, este link contiene la información necesaria para validar los datos y un códigoúnico encriptado para brindar seguridad.

Básicamente las cuentas pueden tener 3 diferentes estados: INACTIVA, DESACTIVADA y ACTIVADA; en el momento de crear la cuenta esta adquiere el estado INACTIVA y permanece allí hasta que el usuario realice la activación. En este estado no es permitido iniciar sesión en la aplicación. Una vez hecho el proceso de activación, la cuenta pasa al estado ACTIVADA y permanece allí hasta que el usuario decida por voluntad propia desactivar la cuenta o darse de baja, en cuyo caso adquiere el estado DESACTIVADA. Si el usuario lo desea puede volver a activar la cuenta y volver a usar la aplicación.

**H3. Desactivar cuenta de usuario**

Por medio de esta opción el usuario podrá desactivar su cuenta temporalmente en caso de asi quererlo. Cuando se desactiva una cuenta de usuario, se mantiene toda la información que el usuario yá tenia, pero no le es permitido iniciar sesión en la aplicación, tampoco le podrán ser compartidos diagramas de otros usuarios.

**H4. Reactivar cuenta de usuario**

Esta es la operación inversa a la descrita en H3, el usuario reactiva su cuenta mediante un correo de confirmación en donde se le envia un codigo encriptado y la información necesaria autocontenida en un link, el usuario lo unico que debe hacer es clickear el link y lo llevará automaticamente a la aplicación donde la cuenta debe estar ACTIVADA nuevamente.

**H5. Recuperar contraseña**

Esta funcionalidad permite a un usuario registrado recuperar su contraseña en caso de haberla olvidado. El proceso consiste en ingresar la dirección de correo y de manera controlada se le envia un email de confirmación con un link para hacer la recuperación. En realidad, la operación es cambiar la contraseña, ya que el link enviado apunta a un formulario donde el usuario ingresa una nueva contraseña para su cuenta.

**H6. Iniciar sesión**

Funcionalidad mediante la cual un usuario ingresa a la aplicación. Esta operación puede ser llevada acabo de dos maneras diferentes, un usuario registrado que ingresa a la aplicación con su direccion de correo o un usuario invitado que ingresa a la aplicación sin registro previo.

**H7. Ver demostración**

Un visitante de la aplicación podrá hacer uso de ella sin registro previo, con el objetivo de obtener información o verificar los beneficios que esta puede prestarle. En la pagina de bienvenida se encuentra un link de ingreso, este link redirige al usuario al *Diseñador* de diagramas y crea un diagrama vacio, a partir de allí el usuario podrá hacer modificaciones a su antojo. Sin embargo, algunas funcionalidades estan restringidas, como el compartir diagramas, guardarlos, general codigo fuente o imagenes. Unicamente se podran hacer exportaciones en formato XMI del diagrama construido. Esta opción esta estrechamente relacionada con H14.

**H8. Gestionar perfil de usuario**

Esta opción permite cambiar la información de registro del usuario, tal como nombres, apellidos, fecha de nacimiento, sexo y avatar (imagen de identificación). No se puede cambiar la dirección de correo electronico del usuario.

**H9. Cambiar Contraseña**

Permite cambiar la contraseña del usuario, es necesario siempre ingresar la contraseña anterior. El formulario solicita una nueva contraseña y una confirmación para garantizar que el usuario puede recordarla facilmente. Solo se podrá cambiar la contraseña si la información requerida es correcta, es decir, si la anterior contraseña coincide con la guardada en base de datos y la nueva contraseña y su confirmación son exactamente iguales.

**H10. Crear diagrama**

Esta opción le permite a un usuario crear un diagrama para sí mismo. Los datos basicos de entrada son solamente el nombre y una descripción opcional del diagrama. No se hace ninguna validación especial aparte de que exista un nombre. El usuario quien crea el diagrama es asignado como owner(propietario) del mismo y puede compartirlo con otros. Inicialmente la representación del diagrama en XMI es vacia, solamente contiene un package (paquete) principal que tiene el mismo nombre del diagrama y no tiene elementos hijos.

**H11. Editar diagrama**

Permite cambiar los datos de creación del diagrama (nombre y descripción).

**H12. Eliminar diagrama**

Permite eliminar un diagrama de manera controlada. Esta opción tiene varias restricciones:

* Un diagrama solo puede ser borrado por el propietario del mismo
* Un usuario quien tenga compartido un diagrama, no puede borrar dicho diagrama, solamente puede borrar su acceso a este.
* Cuando se borra un diagrama que esta compartido por parte de su propietario, se informa a los usuarios que este fué eliminado por medio de correo electronico.

**H13. Copiar diagrama**

Esta opción permite copiar un diagrama al cual se tiene acceso (compartido o propio). Los datos son copiados exactamente pero el propietario de la copia será quien realiza la opción. Los diagramas compartidos pueden ser copiados pero ninguno de los usuarios podrá ver el nuevo diagrama a menos que se comparta explicitamente.

**H14. Diseñar diagrama**

Permite abrir el Diseñador y editar los elementos UML del diagrama de clase. Dentro de este diseñador se encuentran todas las funciones basicas para manejar los diagramas, agregar elementos, editarlos, copiarlos, agregar comentarios.

**H15. Compartir diagrama**

Permite al usuario registrado compartir diagramas con otros usuarios del sistema. Esta opción es permitida unicamente para los diagramas de los cuales el usuario tenga privilegios. En general existen 3 diferentes tipos de privilegios ordenados de menor a mayor rango: READ\_ONLY (solo lectura), EDIT (edición) y RESHARE (volver a compartir). Basicamente un usuario solamente podrá compartir diagramas de su propia autoria o cuando se le ha sido asignado el privilegio de RESHARE, en cuyo caso el usuario es un segundo propietario del diagrama. Como se mencionó anteriormente, un usuario con su cuenta INACTIVADA o DESACTIVADA no podrá recibir ningun diagrama compartido.

**H16. Generar código fuente**

Esta opción le permite al usuario generar el codigo fuente del diagrama que se encuentra editanto mediante el Diagramador. El codigo fuente generado es unicamente en lenguaje Java. Los archivos fuentes pueden ser descargado uno a uno, o descargar un empaquetado que contenga todos los archivos generados.

**H17. Generar imagen del diagrama**

Permite obtener una representación plana del diagrama en una imagen con diferentes tipos de formatos (.gif, .png, .jpg). Esta opción es muy util para realizar documentaciones o impresiones del diagrama completo.

**H18. Importar/Exportar XMI**

Permite importar o exportar un diagrama en el formato del estandar XMI.

## PLANIFICACIÓN

### Priorización de Historias de Usuario

### Estimación de Historias de Usuario

### Definición de Iteraciones

### Metafora del Sistema

## DESCRIPCIÓN DE LAS ITERACIONES

### Iteración 1

### Iteración 2

Iteración n..

## PLANIFICACIÓN DE LAS PRUEBAS

### Casos de Prueba

### Pruebas Unitarias

# [IMPLEMENTACIÓN](#TablaContenido)

## DEFINICIÓN DE LA ARQUITECTURA

## DEFINICIÓN DE LA TECNOLOGIA

## PATRONES DE DESARROLLO

## BASES DE DATOS

## CONFIGURACIÓN

## ESPECIFICACIÓN DE FUNCIONALIDADES

## DESPLIEGUE

# [ASEGURAMIENTO DE LA CALIDAD](#TablaContenido)

## CASOS DE PRUEBA

## PRUEBAS DE UNIDAD

## ANALISIS ESTATICO DE CODIGO

# [CONCLUSIONES](#TablaContenido)

# [REFERENCIAS BIBLIOGRÁFICAS](#TablaContenido)

Instituto Nacional de Estadística e Informática. (1999) Herramientas Case. Impreso en los Talleres de la Oficina de Impresiones de la Oficina Técnica de Difusión Estadística y Tecnología Informática del Instituto Nacional de Estadística e Informática (INEI), pág. 9, 10, 11, 13, 17.

# Kumo [Web en línea]. <>http://www.kumo.com.co/blog/que-es-cloud-computing//>. [Consulta: 21-07-2013].

Schmuller, J. Pearson Educación Latinoamérica. México, pág. 5, 6, 7.

Martin, F. UML Gota a Gota. (1999) Addison Wesley Longman de México, S.A. de C.V., pág. 60, 61, 62, 63, 64.

1. AMAYA TORRADO, Yegny Karina y HERRERA ANGARITA, Lady Torcoroma. Identificación de las causas que generan problemas en el aprendizaje de fundamentos de programación de computadores en las facultades de ingeniería de las universidades de la ciudad de Cúcuta. Trabajo de Grado Ingeniería de Sistemas. 2003. Cúcuta N.de S. Universidad Francisco de Paula Santander. Facultad de Ingeniería Disponible en la Biblioteca Eduardo Cote Lamus bajo la signatura TIS 371.334A489i. [↑](#footnote-ref-1)
2. CASE (Computer Aided Software Engineering, Ingeniería de software asistida por computadora), son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo los costos del mismo en términos de tiempo, esfuerzo y dinero. [↑](#footnote-ref-2)
3. Página web oficial de Enterprise Architect, <http://www.sparxsystems.com/> [↑](#footnote-ref-3)
4. Tomado del portal web del programa Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, en la sección Presentación del Programa – Objeto de Estudio.

   Disponible en : <http://ingsistemas.ufps.edu.co/presentacion/objeto-estudio.html> [↑](#footnote-ref-4)
5. Cloud Computing, definición, características, tipos y ventajas, información disponible en la enciclopedia virtual Wikipedia, <http://en.wikipedia.org/wiki/Cloud_computing> [↑](#footnote-ref-5)
6. Google Docs es parte de la nube de servicios proporcionada por la empresa Google para sus usuarios, a través de esta nube los usuarios pueden disponer de un kit de herramientas entre los que se incluye un procesador de texto, hojas de cálculo, diseñador de presentaciones, entre otras. [↑](#footnote-ref-6)
7. Backend se le llama a la parte de la aplicación que se ejecuta en el servidor, la cual está encargada de procesar los datos enviados desde el cliente (Frontend) y generar una respuesta. [↑](#footnote-ref-7)
8. API (“Application Programming Interface”, en español “Interfaz de programación de aplicaciones”), se refiere a un conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizada por otro software. [↑](#footnote-ref-8)
9. GWT es un framework para la construcción de aplicaciones web en lenguaje Java, el cual permite trabajan con la tecnología AJAX (*Asynchronous Javascript and XML*) de forma trasparente para el desarrollador, evitándole tener que lidiar con los aspectos complejos y tediosos de la tecnología; además de ser compatible con la mayoría de navegadores web, lo cual es verdaderamente significativo ya que el código escrito funciona igual para todos. [↑](#footnote-ref-9)
10. Creative Commons Corporation. Código Legal. Atribución no comercial compartir igual 2.5 (Colombia). <http://creativecommons.org/licenses/by-nc-sa/2.5/co/legalcode> [↑](#footnote-ref-10)
11. CONTRATO DE LICENCIA DE CÓDIGO BINARIO. Sun Microsystems, Inc. Java 2 Plataform Standard edition, Runtime Environment 5.0.Disponible: <http://www.java.com/es/download/license_jre5.jsp> [↑](#footnote-ref-11)
12. GNU GENERAL PUBLIC LICENSE Versión 3, 29 june 2007. Disponible: <http://www.gnu.org/licenses/gpl.txt> [↑](#footnote-ref-12)
13. Universidad Francisco de Paula Santander. Estatuto estudiantil. Acuerdo 065, agosto 26, 1996.

    Disponible en : <http://www.ufps.edu.co/ufpsnuevo/archivos/reglamentacion/acuerdo065.pdf> [↑](#footnote-ref-13)
14. Universidad Francisco de Paula Santander. Ingeniería de Sistemas. Pensum académico vigente a partir del año 2011. Disponible en : <http://ingsistemas.ufps.edu.co/images/Descargas/pensum115.pdf> [↑](#footnote-ref-14)
15. El Object Management Group u OMG (o por su nombre en español, Grupo de Gestión de Objetos) es un consorcio dedicado al cuidado y el establecimiento de diversos estándares de tecnologías orientadas a objetos, tales como UML, XMI, CORBA. Es una organización sin ánimo de lucro que promueve el uso de tecnología orientada a objetos mediante guías y especificaciones para las mismas. El grupo está formado por diversas compañías y organizaciones con distintos privilegios dentro de la misma. [↑](#footnote-ref-15)
16. Model Driven Generation (MDG) [↑](#footnote-ref-16)
17. Data Base Management System (DBMS) o Sistema de gestión de bases de datos, son aplicaciones que permiten el almacenamiento, modificación y extracción de información de una base de datos, también provee herramientas para adicionar, modificar, borrar, acceder y analizar datos almacenados en una localización especifica. [↑](#footnote-ref-17)
18. CSV acrónimo de CommaSeparatedValues (en español: Valores separados por coma), son archivos de texto usados para representar datos en forma de tabla, en donde las columnas van separadas por comas y las filas por saltos de línea. [↑](#footnote-ref-18)
19. WYSIWYG es el acrónimo de WhatYouSeeIsWhatYouGet (en español: Lo que usted ve es lo que usted obtiene) Se aplica a los procesadores de texto y otros editores de texto con formato que permiten escribir un documento viendo directamente el resultado final. [↑](#footnote-ref-19)
20. *The Unified Modeling Language Reference Manual.*RUMBAUGH James, JACOBSON Ivar, BOOCH Grady. La guía completa del proceso unificado escrita por sus creadores. *Chapter 5*, *Use Case View.* Pág 63. [↑](#footnote-ref-20)