# RECOLECCIÓN DE INFORMACIÓN

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 decido realizar 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. *Enterprise Architect* que es un software comercial y *StarUML* que a su vez es open source pero ampliamente utilizado y reconocido. A continuación se describirá cada uno de ellos, resaltando en detalle cada una de sus características:

## ENTERPRISE ARCHITECT

Enterprise Architect de Sparx Systems[[1]](#footnote-1) es 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, cubriendo 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 2.1 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

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[[2]](#footnote-2) 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 algunos procesos o metodologías de desarrollo de software.

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.

### CARACTERISTICAS GENERALES

Dentro de las características más importantes de Enterprise Architect podemos destacar las siguientes:

|  |  |
| --- | --- |
| **Característica** | **Descripción** |
| Soporte de UML | Diagramas de Casos de Uso, Actividades, Estado, Interacción (secuencia y comunicación), Paquetes, Clases, Objetos, Componentes, Despliegue. |
| Documentación | Las plantillas soportan todas las características de los elementos del modelo de EA y los datos extendidos tales como Pruebas, Riesgos, Recursos, Cambios, etc. Generador RTF y HTML. |
| Ingeniera de código directa e Inversa | Incorporado para código fuente C++, Java, C#, VB.Net, Visual Basic, Delphi, PHP, Python y ActionScript. |
| Plugins para vincular EA a Visual Studio .NET o Eclipse | Usar EA desde el propio IDE. |
| Modelado de datos | Ingeniería Inversa para muchos de los sistemas populares DBMS, incluyendo Oracle, SQL Server, MySQL, Access, PostgreSQL y otros. Modela tablas de base de datos, columnas, claves, claves foráneas, y relaciones complejas usando UML y perfiles de modelado de datos incorporados. |
| Control de Versiones | Soporta CVS nativo más para herramientas SCC. |
| Archivos binarios de ingeniería inversa para Java y .NET | Soporta la importación de archivos .JAR en java y ensamblados .NET. |
| Importación y exportación de modelos en XML |  |
| Pruebas | Pruebas de Unidad, de Integración, de sistema, de aceptación, escenarios. |
| Ingeniería WSDL | Perfil incorporado para WSDL para simplificar el desarrollo de Servicios web usando UML. |

Tabla 1. Principales características de Enterprise Architect

### DISEÑO DE DIAGRAMAS DE CLASE

Para un análisis adecuado de las herramientas, servicios y características que posee Enterprise Architect para la construcción de diagramas de clase, se realizó una clasificación de estas mismas dependiendo del tipo de funcionalidad que representa, de este modo las características fueron agrupadas 3 tipos: de Interfaz Gráfica, de Servicios y de Construcción de Diagramas. A continuación se mencionará cada una de las características encontradas en cada uno de los grupos:

### INTERFAZ GRÁFICA

Cuando se crea un proyecto para el desarrollo de software en Enterprise Architect, por defecto se crea una carpeta raíz para el proyecto y a partir de ella se colocan todos los demás componentes o elementos del mismo. El software permite realizar ingeniería de software de inicio a fin, incluyendo soporte para los modelos básicos del ciclo de vida del software: Requerimientos, Análisis, Diseño, Implementación y Pruebas; los diagramas de clase van incluidos dentro del modelo de diseño y hacen parte del modelo lógico del software en construcción.

Una vez creado un Proyecto en Enterprise Architect, la pantalla principal está distribuida de forma intuitiva en Paneles, que agrupan funcionalidades mostrando los componentes actuales del proyecto y las herramientas que ofrece al usuario para ejercer control sobre estos y del proyecto en general. A continuación se muestra una imagen tomada de la interfaz principal:

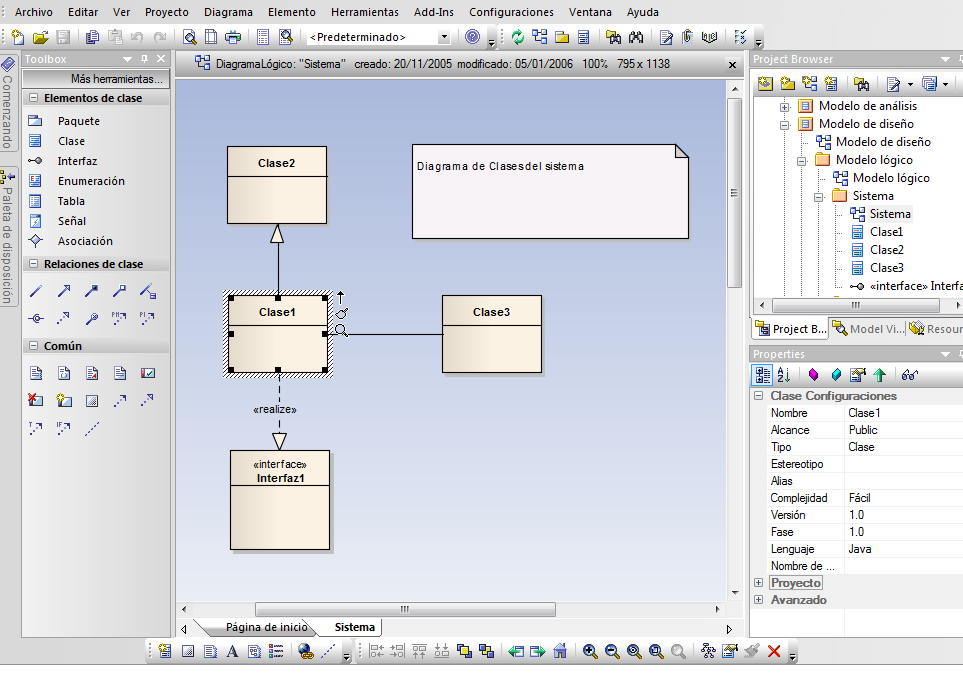


Ilustración 1 Pantalla principal - Enterprise Architect

La interfaz principal está distribuida de la siguiente forma: En la parte superior la *barra de menú* donde se encuentran todas las opciones el software brinda a los usuarios. Los menús en la barra se encuentran distribuidos y agrupados por unidades funcionales, en donde cada menú se encuentra enfocado en una funcionalidad o componente específico del proyecto, de este modo existen opciones que se aplican específicamente a los Proyectos, otros a los Diagramas y otros a los Elementos dentro de estos diagramas. Adicionalmente existen otros menús para configurar la herramienta, gestionar plantillas de código, tipos de atributos para las clases y otras más enfocadas en personalizar la interfaz de usuario y configuración adicional del entorno de trabajo.

Justo debajo de la barra de menú se encuentra *la barra de herramientas* donde se encuentran accesos directos a funcionalidades comúnmente usadas.

El contenido principal de la pantalla se divide en 3 zonas, la zona izquierda representa una *Caja o Paleta de Herramientas* donde se encuentran la mayoría de los componentes y elementos que se pueden añadir a los diagramas, además de los diferentes conectores que se pueden usar para relacionarlos; estos elementos son mostrados dependiendo el tipo de diagrama que se esté diseñando (en la imagen se encuentran los componentes para el diseño de diagramas de clase).

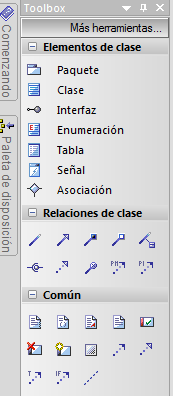


Ilustración 2 Paleta de Elementos de Diagramas de Clase - Enterprise Architect

La zona central es el *Área de Trabajo*, donde se encuentra el lienzo para dibujar los diagramas y demás artefactos que permite realizar la aplicación, esta área permite gran cantidad de comportamientos y funcionalidades que permiten diseñar de manera fácil y cómoda cualquier tipo de diagrama UML, por ejemplo, tiene disponibilidad de Drag & Drop de elementos, Zoom, Relaciones entre elementos mediante Drag & Drop, menús emergentes para seleccionar los tipos de relaciones, posibilidad de Copiar y Pegar elementos. Adicional a esto el Área de Trabajo permite editar o diseñar varios diagramas al tiempo, usando un sistema de pestañas para cada diagrama que se encuentra en edición, por lo tanto el usuario puede fácilmente hacer cambios en uno u otro dependiendo sus requerimientos; por otro lado, los elementos pueden ser incluidos en varios diagramas de modo que los cambios realizados en algún diagrama se ven reflejados en todos los demás donde el mismo elemento está incluido.

En la parte izquierda existen dos paneles importantes que ayudan a visualizar fácilmente la estructura del proyecto y del elemento seleccionado actualmente en el diagrama. Estos son *El Árbol del Proyecto* y el *Panel de Propiedades.* El árbol permite una funcionalidad muy importante y básica para el usuario, como es poder desplazarse por todo el proyecto y tener la posibilidad de editar cualquier elemento de manera rápida, a su vez permite mover o desplazar elementos de un paquete a otro en caso de necesitarlo o eliminar un elemento del proyecto en caso de no necesitarlo más. El panel de propiedades permite visualizar las propiedades básicas del elemento seleccionado actualmente, de modo que se puede editar fácilmente campos como el nombre, el tipo, entre otros. A continuación se muestra una imagen de ambos paneles:

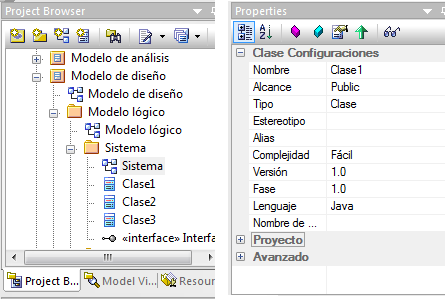


Ilustración 4 Árbol del Proyecto y Panel Propiedades - Enterprise Architect

### SERVICIOS

Enterprise Architect posee 4 servicios básicos relacionados con la creación de Diagramas de Clase, estas son: *Ingeniería de Código*, *Gestión de Tipos de datos* y *Gestión de Plantillas de código* y *Exportación e Importación de Proyectos*.

*La Ingeniería de Código* permite al usuario importar o generar código fuente desde o hacia el proyecto, esta es una utilidad primordial en el software, debido a que permite agilizar el proceso de desarrollo y evita tener que transcribir los diagramas a código de manera manual. Actualmente la herramienta tiene soporte por defecto de varios lenguajes de programación como Java, C++, C#, Python, Delphi, PHP, Visual Basic y VB .NET, para los cuales se puede importar o generar código fuente. La importación de archivos se puede realizar de manera individual o de carpetas completas de código fuente, sin embargo también es posible la importación de archivos binarios, por ejemplo archivos EXE, DLL o hasta JAR en el caso del lenguaje Java.

Otra de las opciones que incluye esta funcionalidad es la capacidad de *sincronizar el modelo con el código generado*, de esta manera los cambios que se realicen tanto en uno como en el otro se pueden ver reflejados inmediatamente y sin la necesidad de volver a generar el código. A continuación se muestra una imagen tomada del asistente de Enterprise Architect para la importación de código:

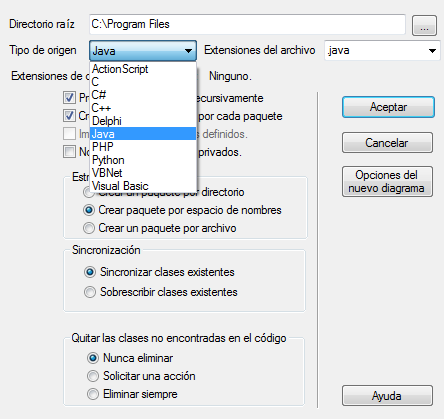


Ilustración 5 Asistente para la importación de código - Enterprise Architect

*La Gestión de Tipos de Datos* permite al usuario configurar o personalizar los tipos de datos para los atributos y métodos de las clases o interfaces que se crean para los diagramas. En la configuración por defecto del software se incluyen varios tipos de datos básicos, como por ejemplo: integer, long, string, float, byte, entre otros, sin embargo se pueden incluir muchos más dependiendo de las necesidades del usuario.

También se puede añadir soporte para tipos de datos de otros lenguajes que no vienen incluidos por defecto. A continuación se muestra una imagen del asistente para incluir nuevos tipos de datos, como se puede observar en la parte superior existe una lista desplegable con todos los “productos” (lenguajes) soportados.

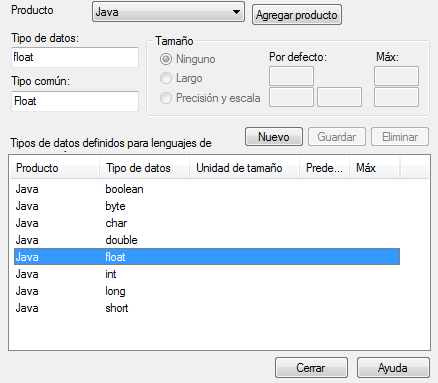


Ilustración 6 Asistente para la creación de tipos de datos - Enterprise Architect

*La Gestión de Plantillas de Código* permite a los usuarios personalizar las plantillas que el software usa para la generación de código fuente, modificar las configuraciones que el software incluye por defecto, así como la posibilidad de incluir soporte de nuevos lenguajes. Esta es una funcionalidad muy útil en el caso de que el lenguaje que estemos usando no esté incluido entre la lista inicial, aunque su uso es un poco complejo y se requiere un conocimiento previo para poder configurarlo de manera adecuada. Una buena alternativa es seguir la documentación del software de modo que se pueda realizar la incorporación de un nuevo lenguaje de manera satisfactoria. A continuación se muestra una imagen tomada del asistente para esta funcionalidad:

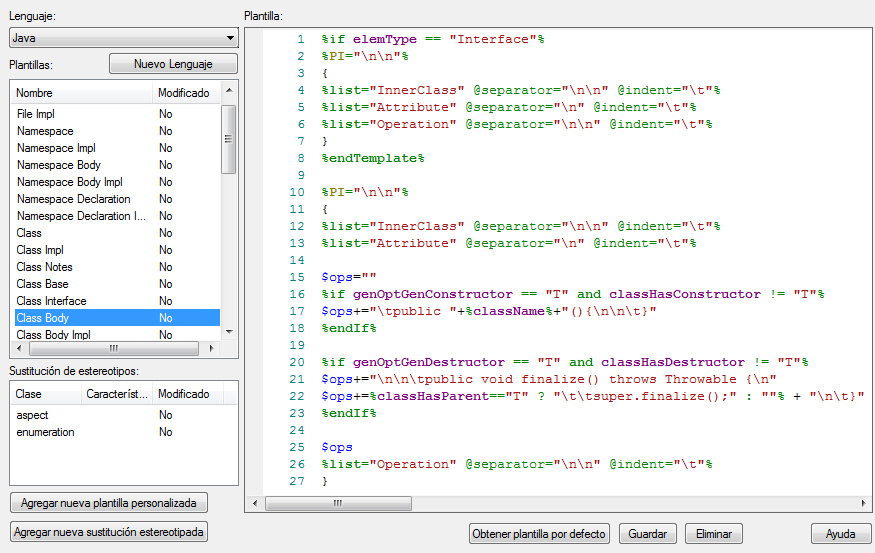


Ilustración 7 Asistente para Editor de Plantillas de Código - Enterprise Architect

Con respecto a la *Exportación e Importación de Proyectos*, anteriormente vimos uno de los mecanismos que Enterprise Architect posee para realizar esta labor, el cual se basa en la Ingeniería de Código; adicionalmente se puede realizar 2 tipos más de importación y exportación, estos son mediante el uso de archivos XMI[[3]](#footnote-3) o la generación de Imágenes planas a partir de un diagrama.

La generación de imágenes se refiere a la creación de un archivo de imagen plano en formato PNG, JPG, BMP o GIF, de modo que el usuario pueda luego ya sea imprimirlo o compartirlo con alguien más. La generación de archivos XML o XMI se puede realizar del proyecto completo o de diagramas específicos, esto facilita que los modelos o diagramas que realicemos se puedan abrir y modificar con otras herramientas de modelado aparte de Enterprise Architect. En el momento de generar el archivo XML se puede escoger la opción de generar en formato XMI o no y una vez generado el archivo Enterprise incluye también un archivo DTD el cual representa la estructura del XML generado.

### CONSTRUCCIÓN DE DIAGRAMAS DE CLASE

Para la construcción de diagramas Enterprise Architect posee varias funcionalidades muy importantes que ayudan a facilitar el trabajo, entre ellas podemos destacar las siguientes:

* Agregar Elementos: A los diagramas se pueden agregar varios tipos de elementos como Clases, Enumeraciones e Interfaces, a su vez a estos se les puede agregar atributos u operaciones (métodos). Existen otros tipos de elementos como Paquetes, Clase Asociación y Tablas, que también hacen parte de los diagramas de clase o que Enterprise Architect permite añadir para prestar un poco más de flexibilidad y lograr mejor trazabilidad con respecto a los demás modelos. A continuación se muestra una imagen del asistente de Atributos y otra del asistente de Operaciones para Clases e Interfaces.

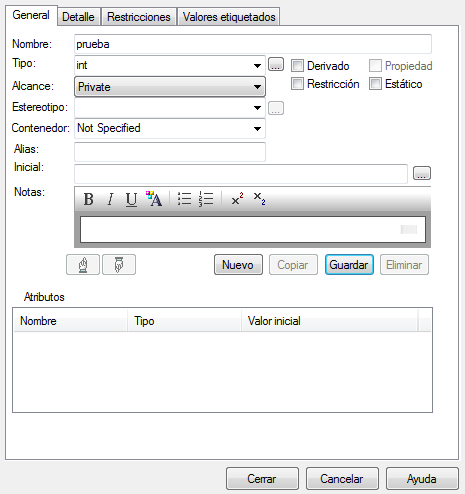


Ilustración 8 Asistente para la creación de atributos - Enterprise Architect

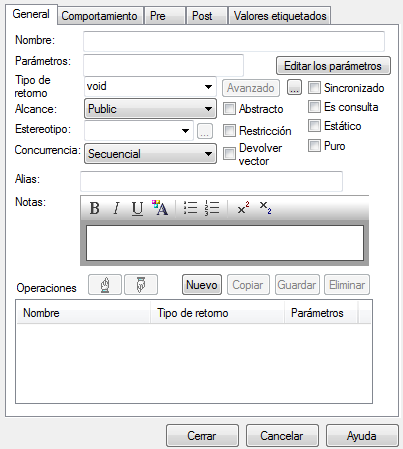


Ilustración 9 Asistente para la creación de Operaciones - Enterprise Architect

* Agregar Conectores: Los conectores son los diferentes tipos de relaciones que se pueden realizar entre los elementos de un diagrama, entre ellos podemos mencionar, Asociación, Herencia, Composición, Agregación, Realización, entre otros, todos estos son representados mediante líneas que pueden ser quebradas a voluntad por el usuario en el momento del diseño, de manera que se pueda personalizar la distribución de los elementos en el lienzo.
* Arrastrar y Soltar: Esta opción también conocida como Drag & Drop es vital para el diseño de diagramas ya que permite facilidad, comodidad y rapidez a la hora de agregar un elemento nuevo o existente al Área de Trabajo, también permite mover elementos sobre el árbol del proyecto de manera muy sencilla e intuitiva.
* Herramienta Zoom: Esta herramienta permite visualizar el diagrama completo o visualizar partes específicas, esto resulta muy útil ya que permite diseñar de manera enfocada en las partes que estamos necesitando o posibilita una vista general de todo el diagrama completo y realizar conexiones entre elementos que se encuentran distanciados con respecto a la distribución en el lienzo.
* Hacer y Deshacer (Undo-Redo): Esta es una de las funcionalidades más importantes para el usuario, ya que permite retroceder o avanzar en las modificaciones que se están realizando actualmente en el diagrama, de manera que si el usuario realizó un cambio involuntariamente o simplemente desea deshacer algún cambio realizado lo puede hacer de manera fácil. El buffer de operaciones que almacena Enterprise Architect es de más o menos 100 cambios, sin embargo, estos están limitados únicamente a los cambios realizados en el diagrama, por ejemplo cambiar la distribución, ordenamiento y posición de los elementos en el diagrama. Cambios a nivel del árbol del proyecto no están soportados, de modo que si se agrega un nuevo elemento, este es creado dentro del árbol y a su vez en el diagrama, pero únicamente podrá ser revertida la inclusión en el diagrama ya que en el árbol seguirá existiendo el elemento.
* Auto ordenado de elementos (Auto Layout): Esta funcionalidad permite realizar un ordenamiento automático de los elementos dentro del diagrama, es realmente útil cuando se realiza ingeniería inversa a partir de código fuente, ya que los elementos quedan desordenados y superpuestos unos sobre otros.

Estas son las características más importantes y que mayor utilidad tienen para el usuario, pero existe un sinnúmero de características pequeñas que ayudan mucho en la labor de diseño y creación de los diagramas.

## STARUML

StarUML es un proyecto de código abierto para desarrollar de manera rápida, flexible, extensible y de libre acceso una plataforma UML/MDA[[4]](#footnote-4) bajo arquitectura Windows 32. El objetivo del proyecto StarUML es construir una herramienta para el modelado de software que sea un reemplazo irresistible para otras herramientas UML comerciales como Rational Rose, Together, entre otros.

StarUML está basado en una arquitectura de Plug-in de tal manera que el software puede soportar múltiples funcionalidades configurables dependiendo de las exigencias del usuario, esto hace que el software sea extensible y adaptable para diferentes tipos y categorías de usuarios. La usabilidad es uno de los temas mas importantes en el desarrollo de software, StarUML esta implementado para proveer muchas características amigables al usuario final, tal como asistentes, manipulación a través del teclado, diferentes vistas del diagrama, etc. StarUML es escrito en su mayoría en lenguaje Delphi. Sin embargo, cualquier lenguaje de programación puede ser usado para desarrollar StarUML, por ejemplo Java, Visual Basic, JScript, C#, VB .NET, entre otros).

### HISTORIA

### CARACTERISTICAS

### DISEÑO DE DIAGRAMAS DE CLASE

# INGENIERIA DE REQUERIMIENTOS

Una vez realizado el análisis de las herramientas case anteriores se determinaron las características básicas de las cuales debe disponer el software en construcción, algunos tomados de los resultados del análisis y otros añadidos a voluntad dependiendo de las características que se desea que el software tenga una vez terminado. Se establecieron los requerimientos funcionales y no funcionales para la aplicación, a continuación se mencionaran cada uno de ellos:

## 5.1 REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

**REQUERIMIENTOS FUNCIONALES**

*RF1.* Los usuarios de la aplicación son todos de un mismo tipo, no existen privilegios ni usuarios administradores.

*RF2.* Cualquier persona puede ingresar a la aplicación en modo demo, no se requiere un registro previo, sin embargo no se podrán guardar los diagramas generados ni ninguna otra información referente a estos.

*RF3.* Los usuarios se registraran en la aplicación mediante un formulario en el cual se pedirá algunos datos básicos como: Código, Facultad, Plan de estudios, Nombre, Contraseña.

*RF4.* Para poder guardar, compartir o valorar proyectos se debe estar logueado.

*RF5.* Los usuarios podrán compartir sus proyectos y estos a su vez podrán modificarlos, valorarlos y comentarlos.

*RF6.* Los usuarios no podrán compartir proyectos o diagramas que les fueron compartidos, esto es permitido únicamente para el creador del proyecto.

*RF7*. Los usuarios podrán crear tantos proyectos como quieran y dentro de cada proyecto pueden existir varios diagramas.

*RF8.* Los proyectos deben poseer una bitácora en donde se muestre todos los cambios que han recibido, además del usuario quien los efectuó y la fecha-hora exacta.

*RF9.* Cada proyecto debe almacenar información importante tal como: Nombre, Fecha de creación, propietario.

*RF10.* Los proyectos tendrán una estructura de árbol iniciando en un paquete principal y seguido por n paquetes secundarios y los elementos para los diagramas estarán distribuidos dentro de esos paquetes.

*RF11.* El código fuente generado tendrá la estructura de paquetes que posea el proyecto.

*RF12.* Se debe disponer de un mecanismo de valoración para los proyectos, de modo que los demás usuarios puedan calificar proyectos que les fueron compartidos según su criterio y juicio.

*RF13.* Las valoraciones realizadas por varios usuarios deben ser promediadas de modo que se obtenga un valor único, sin perder los valores previos de modos que se pueda saber qué usuarios y qué valoración le dieron al proyecto.

*RF14.* Los diagramas (o el proyecto completo según se requiera) deben tener diferentes mecanismos de exportación: Imagen plana, XML, Código Fuente.

*RF15.* Los mecanismos de importación podrán ser: A partir de código fuente o a partir de un archivo XML.

*RF16.* Los diagramas compartidos pueden ser modificados concurrentemente por varios usuarios, de modo que los cambios que hagan uno u otro deben verse reflejados en ambos lados.

*RF17.* NO existirá sincronización entre el código fuente generado y los diagramas.

*RF18.* Se debe disponer de mecanismos de UNDO y REDO para la edición de diagramas de modo que se puedan revertir o rehacer cambios en estos, no olvidar el tema del performance en este aspecto y considerar una capacidad de almacenamiento adecuada.

*RF19.* Los cambios en los diagramas deben ser guardados automáticamente en periodos de tiempo regulares y sin la interacción del usuario.

*RF20.* Se debe disponer de un mecanismo para el ZOOM.

*RF21.* De ser posible se debe disponer de plantillas de código, de modo que el código fuente generado a partir del proyecto pueda ser en diferentes lenguajes.

*RF22.* El código fuente se generará del proyecto completo y no de únicamente diagramas individuales.

*RF23.* La aplicación debe disponer de todos los componentes o elementos que hacen parte del estándar UML 2.0 para los diagramas de clase, tales como:

|  |  |
| --- | --- |
| **ELEMENTOS** | **CONECTORES** |
| Paquete  Clase  Interfaz  Enumeración  Nota  Instancia de Objeto | Asociación  Herencia  Composición  Agregación  Clase asociación  Anidamiento  Ensamble  Vinculo de nota |

Tabla 2. Elementos y conectores soportados por la aplicación.

*RF24.* Los diagramas solo podrán ser eliminados de la aplicación por su creador, y si un diagrama es borrado inmediatamente se remueve cualquier vínculo de este a los usuarios que lo tengan compartido.

*RF25.* Los usuarios pueden dejar de visualizar un diagrama o proyecto que les fue compartido.

*RF26.* Los diagramas compartidos tendrán los siguientes niveles de seguridad.

*RF27.* Una vez el usuario inicie sesión se les mostrará la lista de diagramas que tengan y de allí se podrá escoger la opción de crear nuevos, editar, eliminar o compartir según sea el caso.

*RF28.* Los diagramas pueden ser agrupados en paquetes o folders y un paquete puede contener muchos diagramas.

*RF29.* Los paquetes representan un contenedor de elementos (clases, interfaces y otros paquetes) y sobre estos se pueden crear diagramas, el código fuente tendrá la estructura de paquetes que tenga el proyecto.

*RF30.* Solo se podrá compartir proyectos completos, no se podrá realizar esta operación para diagramas específicos.

*RF31.* Los diagramas están contenidos en proyectos de modo que el usuario deberá primero crear un nuevo proyecto y luego dentro de este crear sus respectivos diagramas.

*RF32.* Los elementos de los diagramas existen en el proyecto dependiendo la jerarquía de paquetes dentro del mismo, un mismo elemento puede aparecer en varios diagramas del mismo proyecto.

*RF33.* Los elementos nuevos pueden incluidos en los diagramas mediante “Drag & Drop” o con ayuda de un asistente.

*RF34.* Elementos existentes en el árbol del proyecto también pueden ser incluidos de esta manera, sin embargo para estos se debe preguntar si se desea crear un elemento nuevo o si solamente se coloca una instancia del mismo elemento dentro del diagrama.

RF35. Se debe disponer de un mecanismo eficiente de Drag & Drop soportado por los navegadores más conocidos, en las versiones que permitan esta opción.

RF36. Los elementos como clases, enumerados, interfaces tienen modificadores, así como los atributos, métodos y constantes, todos estos aspectos deben ser soportados.

RF37. En cuanto a los tipos de datos para los atributos y métodos, se debe disponer de un set básico de tipos de datos (String, int, short, Long, byte… etc.), pero además es posible crear nuevos tipos de datos como por ejemplo clases, interfaces o enumeraciones creadas dentro del proyecto de modo que se puedan crear relaciones entre los elementos del diagrama.

RF38. Tipos de datos nuevos que no son necesariamente algún otro elemento o no se encuentran dentro del set básico, también podrán ser creados.

RF39. Tener en cuenta los parámetros de los métodos y las variables de retorno.

RF40. Los atributos o métodos pueden ser ordenados bajo el antojo del usuario al igual que los diagramas y elementos dentro del árbol del proyecto.

RF41. Se debe ofrecer la capacidad de sobre escribir métodos para las clases o interfaces que heredan de otras.

RF42. En cuanto a usabilidad de la aplicación, se debe disponer de un mecanismo de copy-paste.

RF43. Se debe disponer de una opción para mover o copiar elementos de un paquete a otro.

RF44. Se debe disponer de una panel de propiedades donde se muestren todas las características de los elementos seleccionados.

RF45. Se debe disponer de un panel donde se muestre todo el diagrama completo que está actualmente en edición, en miniatura de modo que se pueda navegar con facilidad sobre él.

RF46. Los elementos de los diagramas como Clases, Interfaces y demás pueden ser redimensionados al antojo del usuario, pero siempre deben guardar proporción con respecto a los demás en cuanto a tamaño de la letra.

RF47. El área de trabajo para los diagramas se puede agrandar o achicar dependiendo de las necesidades del usuario, se puede tener un límite para el tamaño, siempre y cuando sea lo suficientemente grande para que el usuario pueda diseñar libremente.

RF48. Se debe disponer de vistas personalizadas, cuadricula, guías, tooltips, etc.

RF49. Los elementos pueden contraerse o expandirse, de modo que se muestre únicamente la sección del nombre, esto ayuda a la visualización rápida del diagrama o cuando el cuerpo del elemento no es realmente importante.

RF50. Se debe disponer de una herramienta de selección de varios componentes en caso de querer editar o eliminar varios al tiempo.

RF51.

1. 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, Victoria. Actualmente la empresa se ha extendido a más de 60 países con su producto insignia llamado Enterprise Architect. [↑](#footnote-ref-1)
2. 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-2)
3. XML Metadata Interchange o XMI por sus siglas en inglés, es una especificación para el intercambio de diagramas que fue escrita como una manera de compartir modelos UML entre diferentes herramientas de modelado. En versiones anteriores se usaba XML, pero este Schema no decía nada acerca de cómo se debía representar el modelo. La especificación fue desarrollada mediante un nuevo Schema XML que permite la construcción una representación SVG (Scalable Vector Graphics) y típicamente es utilizada solamente por quienes desarrollan herramientas de modelado UML. [↑](#footnote-ref-3)
4. La arquitectura dirigida por modelos (Model Driven Architecture o MDA) es un acercamiento al diseño de software, propuesto y diseñado por el Object Management Group. MDA se ha concebido para dar soporte a la ingeniería dirigida a modelos de los sistemas de software. MDA es una arquitectura que proporciona un conjunto de guías para estructurar especificaciones expresadas como modelos. [↑](#footnote-ref-4)