REVISTA Universidad EAFIT
Vol. 41. No. 137. 2005. pp. 60-76

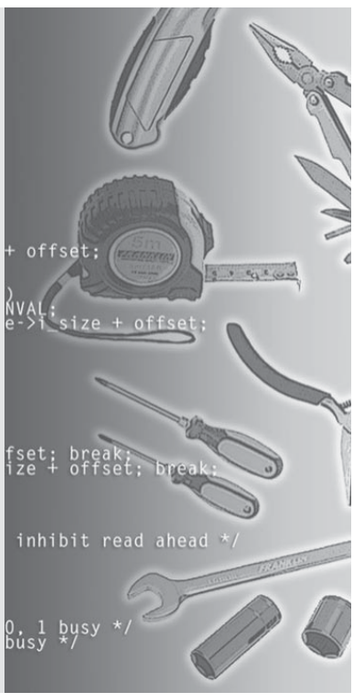
Un estudio comparativo de herramientas para el modelado con UML

Juan Bernardo Quintero
Ingeniero de Sistemas. Asistente del Grupo de Investigación en Ingeniería de Software de la Universidad EAFIT.
jquinte1@eafit.edu.co

Raquel Anaya de Páez
Ph.D. en Informática. Profesora del Departamento de Informática y Sistemas de la Universidad EAFIT y directora del Grupo de Investigación en Ingeniería de Software de la misma Universidad.
ranaya@eafit.edu.co

Juan Carlos Marín
Estudiante de último semestre de ingeniería de sistemas de la Universidad de Antioquia. Integrante del Grupo de Investigación en Ingeniería de Software de la Universidad EAFIT. Asesor de Calidad de Sequal Ltda.
auriga_marin@yahoo.es

Alex Bilbao López
Estudiante de último semestre de ingeniería de sistemas de la Universidad de Antioquia. Integrante del Grupo de Investigación en Ingeniería de Software de la Universidad EAFIT.
perseo_ab@yahoo.es



Recepción: 20 de abril de 2004 | Aceptación: 18 de junio de 2004

Resumen

El presente artículo caracteriza el entorno de las herramientas CASE de modelado y de manera especial aquellas que se apoyan en UML, como lenguaje de especificación para el modelado orientado a objetos. Los diferentes aspectos de las herramientas de modelado con UML se clasificaron en cuatro grupos: (a) Enfoque Procedimental, que describe el apoyo de la herramienta al proceso (b) Soporte al modelado arquitectónico, que analiza el grado de soporte de la herramienta para la definición de la arquitectura del sistema, (c) Apoyo al repositorio, que describe la manera como la herramienta soporta los servicios de almacenamiento, intercambio y recuperación de elementos y (d) Enfoque Funcional, que

agrupa características generales deseables de una herramienta de este tipo. Se seleccionaron cinco de las herramientas de modelado con UML más conocidas (ArgoUML, Rational Rose, WithClass, Together y Poseidon) con el fin de validar en ellas la aplicabilidad de los conceptos. El análisis se complementa con la caracterización de AR2CA, una herramienta que está siendo desarrollada por el grupo de Ingeniería de Software de la Universidad EAFIT.

Palabras Clave
UML
CASE
Repositorio
Ingeniería de Software
XML
XMI

A comparative study of UML modeling tools

Abstract
This article characterizes the environment of CASE modeling tools, especially those that are supported by UML, a specification language for object oriented modeling. The different aspects of UML modeling tools were classified in four groups: (a) Procedure approach, which describes the support of the tool to the process (b) Support to architectonic modeling, which analyzes the degree of support of the tool for the definintion of the architecture of the system, (c) Repository support, which describes the way the tool supports the services of storage, exchange, and element recovery and (d) Functional approach, which groups the desired characteristics of this type of tool. Five of the most well known UML modeling tools were chosen (ArgoUML, Rational Rose, WithClass, Together and Poseidon) with the purpose of validating in them the application of the concepts. The analysis is complemented with the characterization of AR2CA, a tool that is currently under development by the Software Engineering Group of Universidad EAFIT.

Key Words
UML
CASE
Repository
Software Engineering
XML
XMI

Introducción

En el desarrollo de sistemas de información existe el interés por mejorar la productividad y crear productos de *software* de alta calidad. Para ello, es indispensable establecer un entorno disciplinado y estructurado que genere una ventaja competitiva frente a las demás organizaciones y en el cual predomine el trabajo en equipo, para evitar resultados impredecibles.

La diversidad de consideraciones de orden organizacional, metodológico y tecnológico, involucradas en el desarrollo de sistemas, se articulan en lo que Pressman denomina una tecnología multicapa (Pressman, 2002), la cual está conformada por: (a) *el enfoque de calidad*, que establece los principios y lineamientos generales de la organización como estrategia para lograr la calidad tanto de proceso como del producto *software*; (b) *el proceso*, que establece un marco de trabajo para un conjunto de áreas claves que orientan la gestión del proyecto de desarrollo software; (c) *el método* o aproximación metodológica, que define la manera como se debe construir una solución *software* y debe estar acompañado por un lenguaje; (d) las *herramientas,* que ofrecen el soporte automatizado para la realización de las diversas tareas.

Las aproximaciones metodológicas de hoy en día (Hilera, 2003) (Larman, 2003), están altamente influenciadas por el enfoque de orientación por objetos; dichas aproximaciones permiten entender y analizar los fenómenos de la realidad e identificar abstracciones en el espacio de la solución que son consistentes con los elementos del espacio del problema (Meyer, 1999) (Booch, 2002). El lenguaje

de especificación o modelado, por su parte, representa el vehículo indispensable para la comunicación entre los participantes y para la representación de conceptos a lo largo de todo el proyecto, utilizando una semántica común (Meyer, 1999) (UM, 2003). El UML[1] es el lenguaje de mayor difusión que la OMG[2] ha convertido desde 1997 en el estándar para definir, organizar y visualizar los elementos que configuran la arquitectura de una aplicación, sea o no de *software* (OMG, 2003).

Además de la aproximación metodológica y el lenguaje, uno de los factores determinantes en el éxito de un proyecto de *software*, lo constituye la acertada selección de las herramientas de trabajo que se utilizarán durante su desarrollo, es por eso que en la actualidad la administración de proyectos atiende cuidadosamente la designación de herramientas como una tarea de vital importancia.

Las herramientas CASE[3] , están tomando cada vez más relevancia en la planeación y ejecución de proyectos que involucren sistemas de información, pues suelen inducir a sus usuarios a la correcta utilización de metodologías que le ayudan a llegar con facilidad a los productos de *software* construidos.

Todas las herramientas CASE prestan soporte a un lenguaje de modelado para acompañar la metodología y es lógico suponer, que un alto porcentaje de ellas soportan UML, teniendo en cuenta la amplia aceptación de este lenguaje y el valor conceptual y visual que proporciona, y su facilidad para extender el lenguaje para representar elementos particulares a determinados tipos de aplicaciones. El objetivo de este artículo es analizar las características de una herramienta CASE que apoya el modelado con UML y la manera como dichas características son satisfechas por algunas herramientas, incluyendo una que está siendo desarrollada en la Universidad EAFIT: AR2CA.

---

[1] *Unify Modeling Languaje*.

[2] *Object Management Group*.

[3] *Computer Aided Software Engineering* (Ingeniería de Software Asistida por Computador).

1. Evaluación de herramientas de modelado con UML

La toma de una acertada decisión a la hora de escoger una herramienta de modelado con UML, puede ser un factor importante para lograr la calidad de un proyecto. Por tal razón, se necesita tener un amplio conocimiento en aspectos como:

- Las herramientas que existen en el mercado, tanto comerciales como libres.
- Las características de una buena herramienta de modelado con UML.
- La manera como las herramientas satisfacen las necesidades de las personas que participan en un proyecto, para apoyar el proceso de Ingeniería de Software.

Para facilitar el entendimiento de los criterios a tener en cuenta al momento de evaluar una herramienta de modelado con UML, se han analizado cuatro aspectos en los que estas herramientas deben satisfacer resultados.

1.1 Enfoque Procedimental

El enfoque procedimental se refiere a la forma como las herramientas hacen uso de las metodologías para guiar al usuario a través de un proceso de Ingeniería de Software.

1.1.1 Apoyo Metodológico

Una herramienta de modelado con UML debe apoyar el uso de una o varias de las metodologías orientadas a objetos para el desarrollo de sistemas de información como RUP[4] , Métrica (Hilera, 2003), OMT[5] (Rumbaugh, 1997) entre otras; es importante que dicha herramienta garantice la unicidad y coherencia ya sea entre los diferentes modelos del sistema que representan vistas complementarias del sistema (estructurales, dinámicos, funcionales, etc.) o entre modelos que representan diferentes niveles de abstracción (de negocio, de análisis, de diseño, etc.). Esto permite guiar al usuario a través de etapas o fases permitiendo una trazabilidad entre diagramas de alto nivel que representan aspectos del dominio del problema y diagramas detallados que representan detalles de implementación (Shaw, 2003).

Para conseguir tal propósito, es importante que el navegador de la herramienta, muestre las etapas de la metodología o los modelos que propone la misma para cada etapa, fase o arquitectura, y de ser necesario, que le ayude al usuario a evolucionar o refinar un modelo al momento de pasar de una etapa a otra o de un aspecto o contexto arquitectónico a otro (Shaw, 2003) (Dewayne, 2003) (Garlan, 2003).

1.1.2 Soporte completo UML

Esta característica se refiere a la capacidad de las herramientas de construir todos los diagramas que propone UML, o por lo menos los más relevantes: *el diagrama de casos de uso*, que representa la funcionalidad o alcance del sistema; el *diagrama de clases*, que describe la estructura de objetos y sus relaciones; el *diagrama de interacción* (secuencia o colaboración) y, en algunos casos, el *diagrama de estados*, que visualiza el ciclo de vida de un objeto (OBD, 2003).

También es importante la versión que se soporta de UML y la manera como la herramienta cubre los conceptos tratados en ella. Aunque la OMG liberó recientemente la versión 2.0 de UML (OMG, 2003), todavía existen en el mercado muchas herramientas que trabajan con la versión 1.3.

1.1.3 Facilidad de extensión del lenguaje

A pesar de que UML proporciona riqueza sintáctica y semántica para la especificación de modelos, se presentan condiciones particulares de los ámbitos de desarrollo que no logran ser satisfechas. Para tal efecto, UML ofrece mecanismos de extensibilidad tales como restricciones, estereotipos y valores etiquetados (OMG, 2003), que deben ser contemplados e implementados por las herramientas CASE.

[4] *Rational Unified Process*.

[5] *Object Modeling Technique*.

Además, es deseable que dichas herramientas contemplen la posibilidad de edición de símbolos gráficos que permitan representar aspectos particulares de un dominio no contemplados por UML (OBD, 2003).

1.1.4 Modelado de datos

El modelado de la información persistente constituye uno de los factores importantes cuando se está construyendo un sistema de información. En términos prácticos dicha persistencia, estará soportada en bases de datos relacionales (Arce, 2003). Es por esto, que las herramientas de modelado con UML le deben permitir al usuario facilidades para definir el esquema relacional, bien sea a través de una interfaz para la construcción de diagramas entidad - relación o modelos de datos, o a través de la extensión de UML, modelando clases persistentes que se puedan implementar en motores relacionales (Dorsey, 1999).

Los modelos que se construyan utilizando la herramienta deben proveer la facilidad de generar scripts SQL, con las instrucciones DDL[6] para la creación de los objetos de la base de datos.

1.1.5 Autogeneración

Es una característica que podría mirarse desde dos sentidos, generar código a partir de modelos y generar modelos a partir de código.

Varias de las herramientas disponen de la generación de la definición estática (propiedades y signatura de los métodos) de las clases en diferentes lenguajes (como Java, C++) y para diferentes arquitecturas (como CORBA y J2EE).

También es recomendable que la herramienta genere el código que implementa la funcionalidad de mapeo (materialización y desmaterialización) entre el modelo de objetos y el modelo relacional, generalmente conocido como *esquema de persistencia* (Larman, 2003).

La generación de código puede presentar dos tipos de inconvenientes (Gomez, 2003):

- Cuando el código ya se ha generado y se desean realizar cambios sobre éste, la herramienta suele sobrescribir el código que el desarrollador ha introducido directamente.
- Al momento de generar el código, la herramienta CASE pone un gran número de líneas de comentario llamadas *polución*. Para solucionar esto en la actualidad, los poderosos editores de los lenguajes de programación (p.ej. El editor de WithClass y los recientes editores de C#) pueden colapsar y expandir las líneas de comentarios, mostrándole al usuario sólo lo que él quiere ver.

1.1.6 Ingeniería inversa

Mientras que un proceso clásico de Ingeniería de Software, conduce al usuario desarrollador a través de unas etapas para que, a partir de unos modelos, llegue a un producto de *software* construido, la Ingeniería Inversa parte del producto y lo transforma en modelos (Sánchez, 2003).

La generación de diagramas a partir de código puede tener dos vertientes: La generación a partir de lenguajes OO, o a partir de lenguajes estructurados.

La transformación a partir de programas OO se realiza con el propósito de documentar, o bien, reconstruir modelos con base en la versión en producción de un sistema. En este caso es ideal, que además de la generación del diagrama de clases, se puedan también generar diagramas de interacción (colaboración y secuencias) y diagramas de estados (OBD, 2003). Esto puede ser posible a partir de archivos de rastreo durante la ejecución de la aplicación, extrayendo los mensajes enviados entre clases para los diagramas de interacción y los cambios en los estados de una clase para los diagramas de estado. Los modelos así obtenidos son fuente valiosa para la revisión, transformación y generación de la versión en una nueva iteración del proceso del software (Mejía, 2003).

Para aplicaciones construidas en lenguajes no objetuales (C, VB, COBOL, PASCAL y SQL), se hace interesante la posibilidad de realizar ingeniería inversa, facilitando ya sea la migración de sistemas con fines de actualización o la identificación de aspectos estructurales y relacionales de los sistemas difíciles de leer desde el código.

[6] *Data Definition Language*.

1.1.7 Métricas

Medir la bondad de un conjunto de modelos que representan un sistema suele ser una tarea de mucha utilidad para construir sistemas viables, de fácil mantenimiento y de buen rendimiento (McConnell, 1997).

Las métricas son generalmente presentadas en reportes con información como: el número de atributos y métodos por clases, el índice de métodos públicos, privados y protegidos, el número de líneas de código por método, el grado de acoplamiento[7] y el grado de cohesión[8] (OBD, 2003).

1.1.8 Apoyo a lenguajes formales

Un lenguaje formal es una técnica de base matemática para describir las propiedades de un sistema, con el propósito de evitar la ambigüedad y minimizar la interpretación subjetiva de la especificación de un sistema.

OCL[9] es el lenguaje formal adoptado alrededor de UML; es utilizado para especificar las invariantes, precondiciones y postcondiciones y otras restricciones de modelos orientados a objetos con el fin de otorgar mayor formalidad a las especificaciones y aspectos visuales de UML (Moreno, 2003).

Integrar un lenguaje formal en una herramienta CASE, es una de las características sobresalientes que distinguen una herramienta CASE robusta. El lenguaje formal dota los diagramas con una semántica precisa y consistente, lo cual permite que la generación de código se pueda hacer de manera completa.

1.1.9 Actualización

El mejoramiento continuo es una práctica de la mayoría de herramientas de modelado con UML. Tanto las herramientas comerciales como las libres, frecuentemente liberan versiones de actualización que proveen nuevas funcionalidades, mejor rendimiento y soporte a nuevas tendencias metodológicas o a nuevas versiones de UML.

1.2 Soporte al modelado Arquitectónico

Con la revisión de este aspecto se pretende evaluar la capacidad que tienen las herramientas de modelado UML para apoyar la definición de la arquitectura de un sistema. La arquitectura es el elemento clave de diseño que permite establecer acuerdos de alto nivel referentes a la forma del sistema (módulos y subsistemas) y sus relaciones (Garlan, 2003).

El énfasis en componentes y arquitecturas es una de las aproximaciones para desarrollo de software más prometedoras (Anaya, 2000), es por lo tanto importante analizar el soporte que una herramienta CASE provee para la definición de arquitecturas.

1.2.1 Soporte a ADL

Un Lenguaje de Definición de Arquitecturas (ADL) proporciona notaciones para descomponer un sistema en componentes y conectores y especificar cómo se combinan estos elementos para formar cierta configuración, teniendo la ventaja de ofrecer una notación descriptiva directa para las principales abstracciones arquitectónicas. De este modo, los arquitectos pueden definir la estructura de sus sistemas evaluando posibles cambios, y guiar su implementación evitando la ambigüedad.

Entre algunos de los ADL's más conocidos están: WRIGHT, Aesop, Darwin, Rapide y UniCon (Georgas, 2003). Una de las características sobresalientes de la versión de UML 2.0 es su extensión como lenguaje ADL.

Es importante que las herramientas CASE den un soporte adecuado a la definición de arquitecturas, de manera básica, facilitando la descripción y consistencia de vistas de acuerdo al modelo de 4+1 vistas (Kruchtem, 1995) de UML o, de forma avanzada, proveyendo soporte para conceptos propios de un ADL (OOPSLA 99, 2003) (Egyed, 2000).

---

[7] Interdependencia existente entre los módulos.

[8] Relación funcional de los elementos de un módulo.

[9] *Object Constraint Language*.

1.2.2 Apoyo al modelado por capas

El estilo o patrón de arquitectura más utilizado es el que especifica un sistema estableciendo una clara separación de la funcionalidad de éste a través de capas o niveles. El patrón de arquitectura por capas establece como mínimo la definición de tres capas: la capa de *presentación*, que es la parte del sistema que ofrece interface al entorno; la capa de *lógica del negocio* que determina el comportamiento de la aplicación; y la capa de *datos*, que administra la información persistente que maneja el sistema (Clements, 2003). El nivel de aceptación de esta arquitectura se debe a los beneficios que proporciona como: escalabilidad del sistema a través de la fácil distribución de componentes del sistema, confiabilidad por medio de implementación de múltiples niveles de redundancia, flexibilidad ante cambios debido al bajo acoplamiento entre la lógica de presentación y la lógica del negocio, y soporte a la reusabilidad por la identificación de elementos con una clara funcionalidad (LS, 2003).

Es importante que la herramienta CASE permita la visualización de la arquitectura de alto nivel del sistema por medio de paquetes y sus relaciones de dependencia y que además, provea soporte especializado en cada una de las capas de la arquitectura como se describe a continuación:

• **Servicios de presentación.** Tal como se mencionó, la capa de presentación contempla los componentes que especifican la interacción del sistema con el entorno (usuario, sistemas externos, dispositivos, etc.), los elementos propios de esta capa deben poderse representar haciendo uso del estereotipo, con su respectivo ícono asociado, establecido por UML para especificar objetos de tipo frontera (<<*boundary*>>). Es posible que algunas herramientas provean el soporte a las extensiones de UML para manejar los componentes de la capa de presentación; por ejemplo, para el modelado de aplicaciones web (Conallenn, 1999).

• **Servicios de lógica del negocio.** Las reglas del negocio pueden normalizarse de la misma manera como se hace con los datos, para disminuir costos y mantenimientos, incrementar productividad y reducir esfuerzos. Por esto, el proceso de administración de las reglas del negocio, por parte de las herramientas CASE, debería caracterizarse por facilitar la unicidad, consistencia e independencia en la implementación de las reglas, con el fin de apoyar el sostenimiento de la integridad del sistema (CLARION, 2003).

Es en este punto en el que lenguajes como el OCL contribuyen enormemente a describir reglas, restricciones y particularidades de los algoritmos que reflejan la lógica del negocio (GENTLEWARE, 2003).

• **Servicios de persistencia.** La capa de persistencia administra la información que debe perdurar en el sistema, y para este conjunto de componentes, los servicios prestados por las herramientas de modelado le deben permitir al usuario construir diagramas de bases de datos o extender el UML, de tal manera que pueda representar las clases persistentes, haciendo uso de estereotipos.

1.3 Apoyo al repositorio

Los repositorios son herramientas que permiten centralizar, administrar y gestionar las versiones o estados de un proyecto en el que se requieren revisiones frecuentes. En el desarrollo de sistemas usando herramientas CASE, se hace relevante la presencia de un repositorio, no sólo si la herramienta es multiusuario, sino en general para la organización de los cambios reflejados en el refinamiento de los modelos, en el avance de las etapas o en la colaboración entre usuarios (MACPRO, 2003).

Para resaltar la relevancia del apoyo al repositorio por parte de una herramienta CASE, abordamos cuatro aspectos: (a) la robustez de la herramienta a partir del uso de un repositorio, (b) las herramientas de administración, (c) la colaboración entre usuarios y (d) el intercambio.

1.3.1 Robustez

La robustez se refiere a la capacidad de los sistemas de reaccionar apropiadamente ante condiciones excepcionales (Meyer1999), como casos no previstos en la especificación del sistema (por ejemplo, fallas

eléctricas, entrada errónea de datos, incluso la corrupción por virus); y la capacidad de recuperarse ante tales situaciones, protegiendo de forma segura la información almacenada en un repositorio.

La información que se almacena en el repositorio suele ser de mucha importancia, por corresponder a los modelos que representan los sistemas de información. La confiabilidad de la plataforma se constituye entonces en un elemento relevante, puesto que no se pueden arriesgar valiosas horas de trabajo por la pérdida de información o la corrupción de un modelo, producto de la caída del sistema.

1.3.2 Herramientas de administración

La administración en una herramienta CASE, podría clasificarse en cuatros categorías:

• **De repositorio (Dorsey, 1997).** El administrador de la plataforma CASE, necesita realizar tareas como, creación, actualización y borrado del repositorio, mantenimiento de usuarios, copias de seguridad (*export* e *import* del repositorio), recolección de estadísticas y manejo de las extensiones que usan los usuarios en los archivos, al momento de la autogeneración.

• **De ejecución de proyectos.** El administrador del proyecto, necesita realizar tareas como: identificación de riesgos, planeación del proyecto, asignación de tareas a personas, asignación de fechas y recursos a tareas, seguimiento de las tareas o fases de un proyecto.

• **De áreas de trabajo (ORACLE, 2002).** Las áreas o espacios de trabajo (*workarea* o *workspace*) son la manera en que la herramienta facilita la administración de todos los objetos sobre los que puede trabajar un usuario, tanto los objetos de su propiedad, como los que otros usuarios le compartieron. La administración de las áreas de trabajo las realiza cada usuario a través de un navegador de objetos del repositorio. Con frecuencia existe un área de trabajo global compartida, en la que se ubican los objetos a los que todos los usuarios tienen acceso.

• **De aplicaciones.** En esta categoría se ubican tareas como la administración de versiones, el manejo de dependencias y la comparación entre modelos.

La comparación entre los modelos y la implementación, presenta dos variantes:

- Reconciliación. Consiste en que los modelos se actualicen con base en los cambios realizados en la versión implementada (forma de la ingeniería inversa, cuando ya existe el modelo que se va a reconstruir).

- Sincronización. Consiste en pasar los cambios realizados en un modelo, a la implementación.

1.3.3 Colaboración entre usuarios

En la ejecución de un proyecto de Ingeniería de Software participan muchos roles (analistas, arquitectos, diseñadores, revisores, administradores, etc.). Cada uno puede necesitar al mismo tiempo, información del mismo artefacto, por tanto es imprescindible compartir información. Esto significa que las herramientas CASE que soportan el acceso multiusuario deben poseer estrategias de control de concurrencia, ya sea soportados por una base de datos, o esquemas de concurrencia más sofisticados como los propuestos para el desarrollo de sistemas distribuidos.

Una colaboración representa la relación de interacción entre usuarios para la realización de las tareas de cada uno con el sistema. Las herramientas de modelado con repositorio multiusuario pueden asistir en el establecimiento de dichas relaciones entre usuarios, a partir del apoyo metodológico y de la especificación de las responsabilidades de los usuarios (Holmer, 2003) (Villanova, 2003).

Para facilitar dicha colaboración entre usuarios, las herramientas proporcionan funcionalidades para: definir roles, otorgar y revocar privilegios para cada rol establecido, de tal manera que se puedan compartir objetos del repositorio.

1.3.4 Intercambio

Es la posibilidad de que la información de los proyectos, guardada por una herramienta en cierto formato, pueda ser leída por otras herramientas CASE, que acepten dicho formato, para luego ser usada en otros desarrollos.

Si la herramienta CASE soporta UML, la funcionalidad de intercambio de modelos con otras herramientas CASE se convierte en una meta alcanzable, teniendo en cuenta la estandarización sintáctica y semántica que persigue dicho lenguaje.

En esta línea de estandarización alrededor de UML, la OMG definió un estándar de intercambio llamado XMI[10] , que basado en el metamodelo de UML, sirve para el intercambio de metadatos utilizando XML[11].

1.4 Enfoque funcional

El tercer y último enfoque, estudia las utilidades adicionales que le ayudan al usuario a desarrollar un sistema de información con mayor o menor facilidad.

1.4.1 Versionamiento

Las iteraciones en el proceso de Ingeniería de Software, proponen que se construyan versiones de los artefactos y que, de forma incremental, se llegue a un producto maduro. Por tal motivo, es importantísimo que la herramienta de modelado, le permita al usuario guardar versiones del sistema o de sus artefactos, de tal manera que cuando comience otra iteración, la versión previa esté disponible.

1.4.2 Navegación

Con frecuencia los usuarios de las herramientas de modelado con UML, necesitan acceder directamente a la información de una clase o método, o también revisar los diagramas que han construido. Para facilitar éstas y otras labores, las herramientas deben estar dotadas de un navegador de objetos del repositorio, que presente toda la información de los modelos en una forma organizada, garantizando el rápido acceso a cualquier información.

1.4.3 Manejo de diagramas

Para la manipulación de diagramas existen tres labores fundamentales que se deben tener en cuenta:

• **Visualización (ORACLE, 2002).** Al momento de visualizar un diagrama, hay varias funciones que resultan de mucha utilidad para el usuario, como:

- *Zoom in y Zoom out*: Para acercar o alejar el diagrama.
- *Refrescar*: Para que las clases o componentes del diagrama se actualicen con base en el repositorio.
- *Ajustar*: Para que el área de visualización, sea ocupada totalmente por el diagrama o parte de éste.
- *Autolayout*: Para arreglar la presentación de los objetos dentro del diagrama, de tal manera que estos queden lo mejor dispuestos posible.
- *Show y Hide*: Para mostrar u ocultar información y adornos de los diagramas, y de esta forma no saturarlos, dependiendo del nivel de detalle deseado.

• **Impresión.** La herramienta debe tener características wysiwyg[12] , que le permiten el usuario ver el documento al momento del procesamiento como saldrá impreso, es decir la Vista Previa. También suelen necesitarse funcionalidades como la impresión en múltiples páginas o a escala, para ajustar el diagrama al número de páginas deseadas.

• **Exportación.** Poder manipular los diagramas construidos con la herramienta de modelado, se convierte en una posibilidad que resulta bastante útil. Por esto, la exportación de los diagramas a los formatos gráficos más populares como GIF, JPEG o BMP, se presenta como una característica deseable para las herramientas de modelado.

1.4.4 Edición de código

Para que la integración entre los modelos y su representación en un lenguaje de programación sea mayor, la herramienta de modelado debe disponer de editores de código integrados, que le faciliten al usuario la edición del código fuente de las clases, los métodos y todos los artefactos que involucre la solución ya implementada.

---

[10] *eXtensible Markup Language Interchange*.

[11] *eXtensible Markup Languaje*.

[12] *What You See Is What You Get* (lo que ves es lo que recibes).

Es conveniente que las herramientas de modelado tengan integración directa con entornos avanzados de desarrollo, conocidos como IDE[13] .

La inclusión de un editor de XML en la herramienta, se presenta como una utilidad muy atractiva para manipular la información de los modelos construidos con la herramienta de modelado con UML. "Además, como XMI utiliza XML para representar la información del modelo, un grupo de soluciones basadas en XML estarán inmediatamente disponibles, tales como las hojas de estilo (stylesheets) XSL para la presentación basada en navegadores y las herramientas de interrogación (*query tools*) XQL para funciones de búsqueda."(13).

2. Análisis de herramientas UML

Existen en la actualidad muchas comunidades en Internet que se dedican a tratar temas acerca de UML, al punto que se encuentran algunos meta-sitios[14] en donde se organizan las direcciones que nos brindan información de UML. Algunas de estas comunidades en Internet, se dedican a realizar y mantener páginas en las que listan las herramientas de modelado con UML, con sus principales características, incluso su precio y vínculos, donde puede ampliarse la información o comprarlas. Los dos sitios que consideramos más representativos son:

- http://www.objectsbydesign.com/tools/umltools_byCompany.html.
- http://www.uax.es/uax/oop/oo_ooa_ood_tools.html

Con el fin de evaluar los aspectos característicos ya mencionados, se realiza un análisis comparativo entre seis herramientas UML:

(a) **ArgoUML,** una de las herramientas libres para proyectos académicos y de investigación, descargada completamente para su evaluación desde la página de la Tigris Organization. (http://argouml.tigris.org)

(b) **Rational,** plataforma creada por Booch, Rumbaugh y Jacobson, miembros de la OMG, evaluada a través de la guía de usuario y una versión de prueba descargada desde la página de IBM Rational Software. (http://www.rational.com)

(c) **WithClass,** herramienta para proyectos interdisciplinarios, evaluada a través de un demo y descargada desde la página de Microgold Company. (http://www.microgold.com)

(d) **Together,** plataforma de Borland que se ajusta a grandes proyectos de ingeniería de software, evaluada a través de la guía de usuario y documentación adicional proporcionada por Borland Software Corporation. (http://www.borland.com/together/index.html)

(e) **Poseidon,** herramienta de Gentleware que se concibe como la versión comercial y mejorada de ArgoUML y evaluada sobre la versión Comunity Edition 2.0. (http://argouml.tigris.org)

(f) **AR2CA,** (Arquitectura de Refinamiento y Recuperación de Componentes de Análisis), un caso específico de estudio de una herramienta que se encuentra actualmente en construcción, que representa un esfuerzo de investigación apoyado por algunas industrias de software[15].

En AR2CA subyace una aproximación metodológica (Anaya, 1999) que define un proceso de desarrollo en el que se detectan los niveles por los que evoluciona un componente (denominado componente de negocio) en su proceso de construcción. Cada nivel, a su vez, permite modelar diferentes aspectos o vistas del componente.

Los diagramas definidos en AR2CA tienen tres características fundamentales: *la vista*, que describe la dimensión de modelado que representa el diagrama (de contexto, estructural, dinámica, del proceso); *el nivel*, que representa el grado de abstracción del diagrama (requisitos, especificación, implementación)

[13] *Integrated Development Environment*.

[14] Sitios que tratan de los sitos.

[15] PSL S.A, MVM S.A.

y *el alcance*, caracteriza el grado de generalidad que el diagrama representa (alcance dominio o alcance proyecto).

(http://dis.eafit.edu.co/areas/IngSoft/manejoVersiones.htm)

Los resultados de este análisis, en el que se incluyen las seis herramientas estudiadas, se sintetizan en la tabla comparativa (ver Anexo 1).

Conclusiones

Al momento de iniciar un proyecto de Ingeniería de Software, es imprescindible tomarse el tiempo necesario para determinar cual de las herramientas disponibles en el mercado suplen las necesidades que el proyecto plantea.

Este trabajo realizó una caracterización de las herramientas de modelado UML teniendo en cuenta los aspectos procedimentales, de arquitectura, de apoyo al repositorio y de funcionalidad. Esta caracterización es una guía útil para aquellas empresas desarrolladoras de software en el momento de seleccionar una herramienta CASE. La adopción de una herramienta de este tipo debe realizarse una vez la empresa tenga claridad tanto del proceso de desarrollo como de la aproximación metodológica a seguir y de la manera como la herramienta CASE contribuirá a apoyar dichos elementos.

Los requerimientos de una herramienta CASE crecen de manera acelerada. Las herramientas rápidamente quedan desactualizadas por la definición de nuevos estándares para UML (1.3, 1.5, 2.0)

El costo de la herramienta es directamente proporcional a la funcionalidad que ésta ofrece. Las universidades o grupos de investigación están liderando proyectos de construcción de herramientas de dominio público.

AR2CA es una herramienta en construcción que proyecta una funcionalidad oportuna y apropiada de acuerdo con los intereses metodológicos actuales en el desarrollo de software con énfasis en la reutilización.

El proyecto de construcción de una herramienta CASE como AR2CA es un proyecto ambicioso, y en algunos momentos desalentador, por la alta complejidad de la funcionalidad involucrada, la rápida evolución de las tecnologías asociadas y la frecuente evolución de los estándares involucrados (UML, XMI). No obstante, contar con un proyecto nacional de esta naturaleza, provee un entorno didáctico excelente para la aplicación de tecnologías de punta y para el estudio de enfoques avanzados de desarrollo de software, como componentes y aspectos.

Bibliografía

Anaya de Páez, Raquel (1999). *Desarrollo y gestión de componentes reutilizables en el marco de Oasis*. Valencia, 1999. Tesis Doctoral. Universidad Politécnica de Valencia. Departamento de Sistemas Informáticos y Computación.

_____ (2000). "Desarrollo Basado en componentes utilizando UML". En: BRISABOA, Nieves R. *Ingeniería de Software en la década del 2000*. pp. 23-50.

Arce M., Rafael (2003). *Diseño y explotación de las bases de datos relacionales*. Escuela de geografía, Universidad de Costa Rica. (En línea) (Consulta: noviembre 29 de 2003). http://ns.fcs.ucr.ac.cr/~geografia/mapinfo5.html

Booch, Grady ; Rombaugh, James y Jacobson, Ivar (2002). *El lenguaje unificado de modelado*. España : Adyson Wesley, 432 p.

Clarion software (2003). *Clarion Business Rules Manager Application Guide*. SoftVelocity. (En línea) (Consulta: diciembre 5 de 2003) http://www.softvelocity.com/products/applicationguides.htm.

Clements, Paul; NORTHROP, Linda (20003). *Software Architecture: An executive Overview. Technical Report CMU/SEI-96-TR-003*. (En línea) (Consulta: julio 15 de 2003) http://www.sei.cmu.edu.

Conallenn, Jim (1999). *Building Web Applications with UML*. 2 ed. España: Addison Wesley. 496 p.

Dewayne E. Perry ; WOLF, Alexander L. (2003) *Foundations for the Study of Software Architecture*. ACM SIGSOFT. (En línea) (Consulta: noviembre 30 de 2003) http://www.fi.uba.ar/materias/7572/review0.pdf

Dorsey, Paul ; Koletzke, Peter (1997). *Manual de Oracle Designer*. España : Mac Graw Hill. 556 p.

Dorsey, Paul ; Hudicka, Joseph R. (1999). *Oracle 8: Diseño de bases de datos con UML*. México: Mcgraw Hill Interamericana. 394 p.

Egyed, Alexander ; Hilliard, Rich (2000). *Architectural Integration and Evolution in a Model World*. Proceeding of 4th International Workshop on Software Architecture. Limerick, Ireland. 2000.

Garlan, David ; Shaw Mary (2003). An *Introduction to Software Architecture. School of Computer Science*. Carnegie Mellon University, Pittsburgh. (En línea) (Consulta: noviembre 29 de 2003). http://www.fi.uba.ar/materias/7572/review0000.pdf

Gentleware (2003). *User Guide: OCL*. En: Poseidon 2.0, Online Help.

Georgas, John (2003). *Recommendations for Architecture-Centric Software soport self-adaptive behavior*. (En línea) (Consulta: diciembre 12 de 2003). http://sunset.usc.edu/gsaw/gsaw2003/s8a/bo_georgas.pdf

Gómez del Moral, Martín ; Consuegra, Diego (2003). *Tendencia de futuro de las herramientas UML*. (En línea) Universidad Politécnica de Valencia (Consulta: Septiembre 30 de 2003). http://www.diatel.upm.es/malvarez/UML/Tendencias.html.

Hilera, José R. (2003). *Metodología MÉTRICA Orientada a Objetos*. NOVÁTICA. (En linea) (Consulta: diciembre 12 de 2003). http://www.cc.uah.es/hilera/ingsoft.htm.

Holmer, Torsten; Schümmer, Jan (2003). *A Tool for Co-operative Program Exploration*. (En línea) (Consulta: diciembre 1 de 2003). http://prog.vub.ac.be/~imichiel/ecoop2000/workshop/subm_papers/holmer.pdf

Kruchtem, P. (1995). "The 4+1 View Model of Architecture". En: IEEE Software. Noviembre 1995. Vol 12. No. 6. pp. 42-50.

Larman, Craig (2003). *UML y Patrones: Una Introducción al Análisis y al Diseño Orientado a Objetos y al Proceso Unificado*. Madrid : Prentice Hall. 590 p.

Lagahs systems (2003). *El modelo de tres capas o "Three Tier"*. (En línea) (Consulta: septiembre 01 de 2003). http://www.lagash.com/papers/paper_threetier.html

Macprogramadores (2003) *Los repositorios : ¿Qué es un repositorio?.* (En línea) (Consulta: diciembre 15 de 2003). http://www.macprogramadores.org/beos/tutoriales/THD/herramientasgnu/repositorios/intro.html

Mc Connell, Steve (1997). *Desarrollo y gestión de proyectos informáticos*. España: McGraw Hill Interamericana. 691 p.

Mejía Álvarez, Pedro (2003). *Reingeniería de software*. Departamento de Ingeniería Eléctrica CINVESTAV-IPN, México. (En línea) (Consulta: noviembre 30 de 2003). http://delta.cs.cinvestav.mx/~pmejia/softeng/Cap34-1.ppt

Meyer, Bertrand (1999). *Construcción de software orientado a objetos*. España: Prentice Hall, 1999. 1.198 p.

Moreno García, María (2003). *Técnicas Formales de Especificación*. Universidad de Salamanca, España. (En línea) (Consulta: diciembre 12 de 2003): http://lisisu02.usal.es/~mmoreno/ASTema7.pdf

Objects by Design (2003). *Choosing a UML Modeling Tool.* (En línea) (Consulta: septiembre 30 de 2003).. http://www.objectsbydesign.com/tools/modeling_tools.html.

OMG (2003). *UML 2.0 : Draft Specification*. (En línea) (Consulta: noviembre 29 de 2003). http://www.uml.org/uml

_____ *UML: Specification nearing completion*. (En línea) (Consulta: diciembre 2 de 2003). http://www.omg.org/uml

OOPSLA99 (2003). *¿Is UML an Architectural Description Language?*. (En línea) (Consulta: diciembre 12 de 2003). http://www.acm.org/sigplan/oopsla/oopsla99/2_ap/tech/2d1a_uml.html

Oracle Corporation (2002). *Design Editor: View Menu*. (En**:** Oracle 9i Designer, Online Help. 2002).

_____ (2002a). *Repository Management: Workareas*. En**:** Oracle 9i Designer, Online Help. 2002.

Pressman, Roger S. (2002). *Ingeniería del Software: Un enfoque práctico*. España: Mcgraw-Hill Interamericana. 824 p.

RumbaugH, James; Blaha, Michel ; Premerlani, William y otros (1997). *Modelado y diseño orientado a objetos*. España: Prentice Hall. 643 p.

Sanchez de La Cruz, María Mar; Del Casar Tenorio, Mar Sol (2003). *Reingeniería de bases de datos relacionales*. Universidad de Castilla, La Mancha. (En línea) (Consulta: septiembre 30 de 2003). http://alarcos.inf-cr.uclm.es/doc/bbddavanzadas/reingenieria.pdf

SHAW, Mary (2003). *The Coming-of-Age of Software Architecture Research.* Institute for Software Research International, Carnegie Mellon University, Pittsburgh. (En línea) (Consulta: 29 Noviembre de 2003). http://www.fi.uba.ar/materias/7572/research.pdf

Universidad de Murcia (2003). *¿Por qué el empleo de una metodología de desarrollo de software?*. Facultad de Informática. Murcia. (En línea) (Consulta : noviembre 25 de 2003). http://www.um.es/~eutsum/escuela/Apuntes_Informatica/Sigef/2sigef.html

_____ (2003a). *Metodologías de desarrollo de software*. Facultad de Informática (En línea) (Consulta: noviembre 25 de 2003). http://dis.um.es/~barzana/Curso03_04/lagp2003_04_2.html

Villanova, M.; Belkhatir, N.; Martin, H. (2003). *A process environment supporting web multimedia Information systems*. Francia (En línea) (Consulta: diciembre 5 de 2003). http://www-lsr.imag.fr/Les.Groupes/sigma/articles/IJCA02.pdf

Anexo 1. Tabla comparativa de herramientas CASE

PROPIEDADES	ARGOUML argouml.tigris.org	RATIONAL ROSE www.rational.com	WITHCLASS www.microgold.com	TOGETHER www.borland.com	POSEIDON www.gentlewere.com	AR$_2$CA www.eafit.edu.co
ASPECTOS GENERALES						
Observaciones Generales	Navega de acuerdo con diferentes perspectivas de modelado y de metodología de desarrollo.	Provee productos de UML para los lenguajes comunes de la industria para especificación, visualización, construcción y documentación de los artefactos de los sistemas software.	Utiliza perspectiva objeto relacional con diferentes metodologías de desarrollo y modelado.	Robusta y sólida plataforma de Borland fácilmente integrable a JBuilder. Usa refactoring y patrones. Apoya procesos de testing a través de un marco apropiado. Es extensible a necesidades de estándar de la organización.	Herramienta de Gentleware que genera código en Java usando plantillas, realiza ingeniería inversa y sincronización de código en Java. Se puede considerar la versión comercial y mejorada de ArgoUML Apoya el proceso de documentación de diagramas con UMLdoc.	Herramienta de apoyo a la reutilización de modelos
Ámbito de Utilización	Sus características se ajustan a **Proyectos académicos y de investigación.**	Se enmarca dentro del desarrollo de modelado para fines académicos, investigativos y comerciales.	Sus características se apoyan en el **desarrollo orientado o objetos** , para proyectos académicos, investigativos y de comercialización.	Se a justa a grandes proyectos de ingeniería de software en los que la confiabilidad y administración de la información sean relevantes.	Sus características se apoyan en el **desarrollo orientado o objetos** , para proyectos académicos, investigativos y de comercialización.	Utilizada en **proyectos académicos** pretende convertirse en una herramienta utilizada por las industrias de software.
Plataforma	**Independiente de la Plataforma.** Requiere Máquina Virtual Java 1.3 o superior.	Sistema Operativo Windows 98, 98 SE, ME, NT 4.0 , XP	Sistema operativo Windows.	Funciona sobre una máquina virtual de Java.	**Independiente de la plataforma**. Funciona sobre una máquina virtual de Java 1.3 o superior.	**Independiente de la plataforma**. Funciona sobre una máquina virtual de Java 1.3 o superior.
Precio	Libre	US. $6115, $3490 y $2615, de acuerdo la versión Enterprice Edition, Professional o Modeler respectivamente.	US. $250.00 También posee un *trial* de 30 días.	Su precio esta entre US$ 3.000 y US$ 6.000 de acuerdo con las opciones seleccionadas.	US.$600.00 Además tiene una edición libre con las funcionalidades básicas.	Libre

PROPIEDADES	ARGOUML	RATIONAL ROSE	WITHCLASS	TOGETHER	POSEIDON	AR$_2$CA
ENFOQUE PROCEDIMENTAL						
Apoyo Metodológico	No apoya una metodología específica	UML, Booch y OOSE; todas ellas apoyadas en RUP. Soporta MVC.	OMT, UML, Booch, Martin-Odell, Coad-Yourdon.	Refactoring, testing Framework, desarrollo de plantillas y patrones. Apoya el desarrollo de *Web Applications* y *WService*.	No apoya una metodología específica. A través del browser sugiere cierta organización de proyecto	Soporte al desarrollo de software basado en componentes y reutilización.
Soporte completo UML	**UML 1.3**	**UML 1.3**	**UML 1.3**	**UML 1.1, 1.3 y 1.4**	**UML 2.0**	Soporte a **UML 1.0** **Se adoptará UML 2.0**
Ingeniería inversa	Módulos en Framework para Java y .Jar	C++, VB, COM, código ADA, J2EE, Corba/IDL, MIDL,	C++, Java, Delphi, VB, IDL , Perl, PHP, C# and VB.NET	Puede Generar todo el modelo de un archivo completo	Para C#, VB.NET, Java y PHP	No hay Ingeniería Inversa
Métricas	No hay métricas	Apoya la utilización de métricas	Apoya la utilización de métricas	Apoya la utilización de métricas	No hay métricas	No apoya el cálculo de métricas.
Apoyo a Lenguajes Formales	OCL	OCL	No hay apoyo a Lenguajes Formales	OCL	OCL	La versión actual está soportada en el lenguaje OASIS (ANAYA2000).
Actualización	Posee servicio de Actualización	Posee servicio de Actualización	Posee servicio de Actualización	Posee servicio de Actualización	Posee servicio de Actualización	Actualmente en desarrollo.
SOPORTE AL MODELO ARQUITECTÓNICO						
Soporte ADL	A través de UML.	A través de UML.	A través de UML.	A través de UML.	A través de UML.	Evolución del lenguaje OASIS como ADL (ANAYA2000).
Servicios de Presentación	Extensiones de UML	Extensiones de UML	Apoya la utilización de otros Diagramas	Extensiones de UML	Extensiones de UML	A través de la definición de estereotipos
Servicios de lógica del negocio	Soporte de edición de restricciones con OCL.	Chequeo de la sintaxis UML.	A través de la organización según las diferentes metodologías y funcionalidad de browser.	Apoyo a la validación sintáctica y semántica de UML. Auditoria de proyectos. Restricciones usando código.	Edición de restricciones usando OCL. Control de validez de sintaxis de los modelos a partir de las reglas de UML	A través de paquetes con estereotipo específico que poseen una semántica propia
Servicios de persistencia	A través de la definición de estereotipos, no genera SQL.	A través de la definición de estereotipos	A través de la definición de estereotipos	A través de la definición de estereotipos	A través de la definición de estereotipos, no genera SQL.	A través de paquetes con estereotipo específico que poseen una semántica de almacenamiento.

PROPIEDADES		ARGOUML argouml.tigris.org	RATIONAL ROSE www.rational.com	WITHCLASS www.microgold.com	TOGETHER www.borland.com	POSEIDON www.gentlewere.com	AR$_2$CA www.eafit.edu.co
APOYO AL REPOSITORIO							
Robustez		Mediana. No realiza auto guardado.	No presenta auto guardado. Depende de la acción propia de guardado ejecutada por el usuario.	No presenta autoguardado regular. Depende de la acción propia de guardado ejecutada por el usuario.	Es una herramienta de gran robustez. Presenta auto guardado y otras funciones para control de excepciones.	Tiene funcionalidades de control sobre acciones inesperadas: autoguardado, fallos y errores.	No presenta autoguardado.
Herramientas para Administrar	Repositorio	Posee un sistema de archivos usual. El almacenamiento es local y no es considerado repositorio.	Administración de repositorio con acceso multiusuario.	Administración de repositorio. Administración de usuarios.	Administración repositorio local y red Funcionalidad multiusuario. Administración de versiones concurrentes.	Administración multiusuario para la versión empresarial. Apoyo de persistencia y concurrencia en repositorio co mún por proyectos.	Actualmente en construcción el repositorio multiusuario.
	Proyectos	Administración del diseño y la aplicación en desarrollo.	Evaluación y control de edición de diagramas y proyectos. Administración de proyectos.	Control de diseño y evaluación de proyectos, Despliegue de información y edición de los diagramas y elementos de los proyectos en cada modelo.	Administración de proyectos. Permite correr múltiples instancias.	Configuración de las propiedades del proyecto. Apoyo de documentación del proyecto.	Permite definición de proyectos y usuarios responsables.
	Áreas de trabajo	Corrección automática de errores sobre el área de diagramación. La funcionalidad de listado de acciones.	Generación de reportes por uso, instancias, accesos de violación y documentación. Integración documentación y archivos vía URL.	Visualización de barras de herramientas y navegador. Control de paquetes, clases y funciones en cada diagrama a relacionar. Control de metodologías para el diseño.	Visualización de páneles de edición, exploración, diseño, visor de código.	Barra de menú y herramientas, panel de navegación con tipos diferentes de vistas estructurales y de arquitectura.	Browser en el que se distingue los modelos del proyecto, modelos del dominio y catálogo de términos
	Aplicación	Representación alternativa del diseño de manera objetual, código origen y textual.	Consistencia de modelos del sistema software.	Su fortaleza a nivel de aplicación es el apoyo a otras metodologías previas a RUP.	Comparación entre modelos e implementación. Sincroniza código y modelo.	Actualización de diagramas para ingeniería inversa.	Genera plantillas XML con la documentación del modelo
Colaboración entre usuarios		Es una herramienta StandAlone por lo que no aplica la colaboración entre usuarios.	Soporte multiusuario. Modelos compartidos y proyectos para desarrollo individual.	Soporte de archivos compartidos.	Administración de repositorio. Administración y auditoria de proyectos.	Configuración de propiedades de usuarios de repositorio por proyecto para versión empresarial	Por desarrollar funcionalidad para control de concurrencia
Intercambio		XMI como mecanismo estándar.	XMI como mecanismo estándar.	No genera XMI	Intercambio con bases de datos JDBC. XMI como mecanismo estándar.	XMI como mecanismo estándar Compatibilidad con proyectos *Rational 1.6*.	Actualmente se desarrolla la funcionalidad de intercambio con XMI

PROPIEDADES		ARGOUML argouml.tigris.org	RATIONAL ROSE www.rational.com	WITHCLASS www.microgold.com	TOGETHER www.borland.com	POSEIDON www.gentlewere.com	AR$_2$CA www.eafit.edu.co
ENFOQUE FUNCIONAL							
Autogeneración		Java, C++, PHP.	Java, J2EE, ANSI C++, Visual C++, VB, CORBA IDL, MIDL y XML.	C++, C#, Java, Delphi, VB, VB.NET.	VB, VB.NET, CORBA IDL, C++.	Java, HTML, C++ y XML. Plugins para C#, CORBA IDL.	No genera código aún.
Versionamiento		No aplica	Rose ClearCase y Rose ClearCase LT	CVS.	CVS, PVCS.	Administración propia. Propiedades proyecto, versión y actualización.	Se proyecta controlar el manejo de versiones de los modelos.
Navegación		Tres vistas	Proyectos. Diagramas.	Repositorio. Diagramas.	Proyectos. Módulos - Componentes	Modelos	En el browser se puede navegar en modelos del Proyecto o del Dominio identificados por niveles y vistas.
Manejo de diagramas	**Visualización**	Zoom in/out Refresh. Ajuste. Autolayout.	Discriminación de colores en diagramas. Zoom in/out	Zoom in/out Refresh. Ajuste. Autolayout.	Autolayout. Zoom in/out. Refresh. Show/Hide.	Autolayout. Zoom in/out. Show/Hide. Copy/ Cut /Paste.	Copy/ Cut /Paste.
	Impresión	Configuración de página.	Funcionalidad *WYSIWYG*, con vista previa.	Diagrama, página activa o región seleccionada.	Permite impresión de Diagramas y paginación	Permite Impresión de Diagramas desde la Edición Empresarial	Facilidad de vista previa
	Exportación	GIF,PS, Encapsulated PS, PGML y SVG.	GIF, JPEG, BMP, y otros.	GIF, JPEG, BMP, y otros.	BMP, WMF, GIF y SVG.	JPG, JPEG, BMP, WMF, GIF y SVG.	GIF, PS, Encapsulated PS, PNG, JPEG, WMF y SVG.
Edición de código		Java	Visual Basic, Java	Add-in para otros editores	**Java** Eclipse y JBuilder	OCL para JAVA.	Editor de código XML