Herramientas CASE

Las **Herramientas CASE** (***C**omputer **A**ided **S**oftware **E**ngineering*, Ingeniería de Software Asistida por Ordenador) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de dinero. Estas herramientas nos pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, calculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras.

Ya en los años 70 un proyecto llamado ISDOS diseñó un lenguaje y un producto que analizaba la relación existente entre los requisitos de un problema y las necesidades que éstos generaban, el lenguaje en cuestión se denominaba PSL (*Problem Statement Language*) y la aplicación que ayudaba a buscar las necesidades de los diseñadores PSA (*Problem Statemente Analyzer*).

Aunque ésos son los inicios de las herramientas informáticas que ayudan a crear nuevos proyectos informáticos, la primera herramienta CASE fue Excelerator que salió a la luz en el año 1984 y trabajaba bajo una plataforma PC.

Las herramientas CASE alcanzaron su techo a principios de los años 90. En la época en la que IBM había conseguido una alianza con la empresa de software AD/Cycle para trabajar con sus mainframes, estos dos gigantes trabajaban con herramientas CASE que abarcaban todo el ciclo de vida del software. Pero poco a poco los mainframes han ido siendo menos utilizados y actualmente el mercado de las Big CASE ha muerto completamente abriendo el mercado de diversas herramientas más específicas para cada fase del ciclo de vida del software.

Objetivos

1. Mejorar la productividad en el desarrollo y mantenimiento del software.
2. Aumentar la calidad del software.
3. Mejorar el tiempo y coste de desarrollo y mantenimiento de los sistemas informáticos.
4. Mejorar la planificación de un proyecto
5. Aumentar la biblioteca de conocimiento informático de una empresa ayudando a la búsqueda de soluciones para los requisitos.
6. Automatizar, desarrollo del software, documentación, generación de código, pruebas de errores y gestión del proyecto.
7. Ayuda a la reutilización del software, portabilidad y estandarización de la documentación
8. Gestión global en todas las fases de desarrollo de software con una misma herramienta.
9. Facilitar el uso de las distintas metodologías propias de la ingeniería del software

Clasificación

Aunque no es fácil y no existe una forma única de clasificarlas, las herramientas CASE se pueden clasificar en base a los parámetros siguientes:

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

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

- *Upper CASE*, herramientas que ayudan en las fases de planificación, análisis de requisistos y estrategia del desarrollo, usando diagramas UML.
- *Middle CASE*, herramientas para automatizar tareas en el análisis y diseño de la aplicación.
- *Lower CASE*, herramientas que semiautomatizan la generación de código, crean programas de detección de errores, soportan la depuración de programas y pruebas. Además automatizan la documentación completa de la aplicación.

Por funcionalidad podríamos diferenciar algunas como:

- Herramientas de generación semiautomática de código.
- Editores UML.
- Herramientas de Refactorización de código.
- Herramientas de mantenimiento como los sistemas de control de versiones

Herramientas CASE

Introducción

CASE, o Computer-Aided Software Engineering es un termino que ha estado por décadas. Este puede ser generalmente aplicado a cualquier sistema o colección de herramientas que ayuda a automatizar el proceso de diseño y desarrollo de software. Compiladores, editores estructurados, sistemas de control de código fuente, y herramientas de modelado son todas, estrictamente hablando herramientas CASE. Ellas impiden a los programadores tratar tan directamente con el hardware y les permiten trabajar en un alto nivel de abstracción en la definición de un sistema de software que entonces será construido.

Sistemas CASE

Hay generalmente tres tipos de sistemas CASE: Herramientas de Diseño, Ambientes de Construcción e Híbridos. Algunas de estas herramientas vienen por default en ambientes UNIX, como aquellas herramientas y utilidades que sirven para editar y compilar software.
Este tipo de herramientas (make, cvs/rcs, gcc, Text/groff) que vienen con UNIX son herramientas de desarrollo base, pero los sistemas CASE generalmente no se enfocan en el codificado/escritura/compilado, en vez de esto se encargan del proceso de diseño,

refinamiento, documentación, construcción y administración de versiones necesarias para desarrollar y administrar un sistema o paquete de software. En un ambiente de un gran equipo o un gran paquete donde usted puede tener cinco versiones de este paquete en varios estados de desarrollo y/o desplegándose en cinco arquitecturas de hardware diferentes, suportando de tres a cuatro versiones de sistemas operativos, los procesos de trabajo son complejos.
Herramientas de diseño CASE auxilian grandes equipos de ingenieros en la especificación de sistemas de software y ayudan a automatizar la escritura de arquitecturas, documentación, y además integrar automáticamente esas piezas generadas en el IDE del desarrollador
Muchas herramientas CASE utilizan el Lenguaje de Modelado Unificado (UML) desarrollador por Grady Booch, Jim Rumbaugh, e Ivar Jacobsen. Su compañia, Rational Software es una de la más conocidas en sistemas CASE. La disponibilidad de UML ha revolucionado la habilidad de los ingenieros de software para crear especificaciones de sistemas que pueden ser relativamente fácil de traducir en código mantenible y que funcione.
Hay herramientas CASE para casi todo tipo de especialización que uno puede pensar, de diseño de base de datos a data warehousing, de generación de documentación a desarrollo de sistemas embedidos como teléfonos celulares.
Herramientas de construcción auxilian equipos grandes en la construcción y administración de liberación de paquetes de software.
Herramientas híbridas son un nuevo fenómeno, aplicación Servicios Web para crear un sistema distribuido que puede manejar múltiples estilos de desarrollo y la flexibilidad de agregar nuevas herramientas y servicios sin mucho trabajo. Buenos ejemplos incluyen Sourceforge, Collab.NET, y todas sus variantes.

Ejemplos de Herramientas CASE

8.2.3.1. Herramientas Abiertas

- Umbrello
- ArgoUML
- Gaphor

8.2.3.2. Herramientas Comerciales/Cerradas

- Rational Rose
- Together
- System Architect
- Visual Paradigm
- Poseidon

1. Introducción

Hoy en día, muchas empresas se han extendido a la adquisición de herramientas CASE (Ingeniería Asistida por Computadora), con el fin de automatizar los aspectos clave de todo el proceso de desarrollo de un sistema, desde el principio hasta el final e incrementar su posición en el mercado competitivo, pero obteniendo algunas veces elevados costos en la adquisición de la herramienta y costos de entrenamiento de personal así como la falta de adaptación de la herramienta a la arquitectura de la información y a las metodologías de desarrollo utilizadas por la organización. Por otra parte, algunas herramientas CASE no ofrecen o evalúan soluciones potenciales para los problemas relacionados con sistemas o virtualmente no llevan a cabo ningún análisis de los requerimientos de la aplicación.

Sin embargo, CASE proporciona un conjunto de herramientas semiautomatizadas y automatizadas que están desarrollando una cultura de ingeniería nueva para muchas empresas. Uno de los objetivos más importante del CASE (a largo plazo) es conseguir la generación automática de programas desde una especificación a nivel de diseño.

Ahora bien, con la aparición de las redes de ordenadores en empresas y universidades ha surgido en el mundo de la informática la tecnología cliente / servidor. Son muchas de las organizaciones que ya cuentan con un número considerable de aplicaciones cliente / servidor en operación: Servidores de Bases de Datos y Manejadores de Objetos Distribuidos. Cliente / servidor es una tecnología de bajo costo que proporciona recursos compartidos, escalabilidad, integridad, encapsulamiento de servicios, etc. Pero al igual que toda tecnología, el desarrollo de aplicaciones cliente / servidor requiere que la persona tenga conocimientos, experiencia y habilidades en procesamiento de transacciones, diseño de base de datos, redes de ordenadores y diseño gráfica de interfase.

El objeto de estudio está centrado en determinar ¿cuáles son las influencias de las herramientas CASE en las empresas desarrolladoras de sistemas de información cliente / servidor? Y ¿cuáles son las tendencias actuales de las empresas fabricantes de sistemas cliente / servidor?.
A continuación, en el siguiente artículo ahondaremos más en el propósito general de las Herramientas CASE y el impacto que puede ocasionar el uso de las mismas en una empresa.

2. Herramientas Case

De acuerdo con Kendall y Kendall la ingeniería de sistemas asistida por ordenador es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo, su objetivo es acelerar el proceso para el que han sido diseñadas, en el caso de CASE para automatizar o apoyar una o mas fases del ciclo de vida del desarrollo de sistemas.

Cuando se hace la planificación de la base de datos, la primera etapa del ciclo de vida de las aplicaciones de bases de datos, también se puede escoger una herramienta CASE (Computer-Aided Software Engineering) que permita llevar a cabo el resto de tareas del modo más eficiente y efectivo posible. Una herramienta CASE suele incluir:

Un diccionario de datos para almacenar información sobre los datos de la aplicación de bases de datos.
Herramientas de diseño para dar apoyo al análisis de datos.
Herramientas que permitan desarrollar el modelo de datos corporativo, así como los esquemas conceptual y lógico.
Herramientas para desarrollar los prototipos de las aplicaciones.
El uso de las herramientas CASE puede mejorar la productividad en el desarrollo de una

aplicación de bases de datos.

3. Historia

En la década de los setenta el proyecto ISDOS desarrolló un lenguaje llamado "Problem Statement Language" (PSL) para la descripción de los problemas de usuarios y las necesidades de solución de un sistema de información en un diccionario computarizado. Problem Statement Analyzer (PSA) era un producto asociado que analizaba la relación de problemas y necesidades.
Pero la primera herramienta CASE como hoy la conocemos fue "Excelerator" en 1984, era para PC. Actualmente la oferta de herramientas CASE es muy amplia y tenemos por ejemplo el EASYCASE o WINPROJECT. (Monografías.com)

4. Tecnología Case

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información y se plantean los siguientes objetivos:

Permitir la aplicación práctica de metodologías estructuradas, las cuales al ser realizadas con una herramienta se consigue agilizar el trabajo.
Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
Simplificar el mantenimiento de los programas.
Mejorar y estandarizar la documentación.
Aumentar la portabilidad de las aplicaciones.
Facilitar la reutilización de componentes software.
Permitir un desarrollo y un refinamiento visual de las aplicaciones, mediante la utilización de gráficos.
Automatizar:

Ø El desarrollo del software
Ø La documentación
Ø La generación del código
Ø El chequeo de errores
Ø La gestión del proyecto

Permitir:
Ø La reutilización del software
Ø La portabilidad del software
Ø La estandarización de la documentación

5. Componentes de una herramienta case

De una forma esquemática podemos decir que una herramienta CASE se compone de los siguientes elementos:

Repositorio (diccionario) donde se almacenan los elementos definidos o creados por la herramienta, y cuya gestión se realiza mediante el apoyo de un Sistema de Gestión de Base de Datos (SGBD) o de un sistema de gestión de ficheros.
Meta modelo (no siempre visible), que constituye el marco para la definición de las técnicas y metodologías soportadas por la herramienta.
Carga o descarga de datos, son facilidades que permiten cargar el repertorio de la herramienta CASE con datos provenientes de otros sistemas, o bien generar a partir de la propia herramienta esquemas de base de datos, programas, etc. que pueden, a su vez, alimentar otros sistemas. Este elemento proporciona así un medio de comunicación con otras herramientas.
Comprobación de errores, facilidades que permiten llevar a cabo un análisis de la exactitud, integridad y consistencia de los esquemas generados por la herramienta.
Interfaz de usuario, que constará de editores de texto y herramientas de diseño gráfico que permitan, mediante la utilización de un sistema de ventanas, iconos y menús, con la ayuda del

ratón, definir los diagramas, matrices, etc. que incluyen las distintas metodologías.
6. Estructura general de una herramienta case

La estructura CASE se basa en la siguiente terminología:

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

En las últimas décadas se ha trabajado en el área de desarrollo de sistemas para encontrar técnicas que permitan incrementar la productividad y el control de calidad en cualquier proceso de elaboración de software, y hoy en día la tecnología CASE (Computer Aided Software Engineering) reemplaza al papel y al lápiz por el ordenador para transformar la actividad de desarrollar software en un proceso automatizado.

La tecnología CASE supone la –informatización de la informática—es decir –la automatización del desarrollo del software--, contribuyendo así a elevar la productividad y la calidad de en el desarrollo de los sistemas de información de forma análoga a lo que suponen las técnicas CAD/CAM en el área de fabricación.
En este nuevo enfoque que persigue mejorar la calidad del software e incrementar la productividad en el proceso de desarrollo del mismo, se plantean los siguientes objetivos:
< Permitir la aplicación práctica de metodologías, lo que resulta muy difícil sin emplear herramientas.
< Facilitar la realización de prototipos y el desarrollo conjunto de aplicaciones.
< Simplificar el mantenimiento del software.

Mejorar y estandarizar la documentación.
Aumentar la portabilidad de las aplicaciones.
Facilitar la reutilización de componentes de software
Permitir un desarrollo y un refinamiento (visual) de las aplicaciones, mediante la utilización de controles gráficos (piezas de código reutilizables).
8. Integración de las herramientas case en el futuro

Las herramientas CASE evolucionan hacia tres tipos de integración:

La integración de datos permite disponer de herramientas CASE con diferentes estructuras de diccionarios locales para el intercambio de datos.
La integración de presentación confiere a todas las herramientas CASE el mismo aspecto.
La integración de herramientas permite disponer de herramientas CASE capaces de invocar a otras CASE de forma automática.
9. Clasificación de las herramientas case

No existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase determinada. Podrían clasificarse atendiendo a:
- Las plataformas que soportan.
- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- La arquitectura de las aplicaciones que producen.
- Su funcionalidad.
CASE es una combinación de herramientas software (aplicaciones) y de metodologías de desarrollo :
1. Las herramientas permiten automatizar el proceso de desarrollo del software.
2. Las metodologías definen los procesos automatizar.
Una primera clasificación del CASE es considerando su amplitud :

TOOLKIT: es una colección de herramientas integradas que permiten automatizar un conjunto de tareas de algunas de las fases del ciclo de vida del sistema informático: Planificación estratégica, Análisis, Diseño, Generación de programas.
WORKBENCH: Son conjuntos integrados de herramientas que dan soporte a la automatización del proceso completo de desarrollo del sistema informático. Permiten cubrir el ciclo de vida completo. El producto final aportado por ellas es un sistema en código ejecutable y su documentación.
Una segunda clasificación es teniendo en cuenta las fases (y/o tareas) del ciclo de vida que automatizan:
UPPER CASE: Planificación estratégica, Requerimientos de Desarrollo Funcional de Planes Corporativos.
MIDDLE CASE: Análisis y Diseño.
LOWER CASE: Generación de código, test e implantación

10. Características Deseables De Una Case

Una herramienta CASE cliente / servidor provee modelo de datos, generación de código, registro del ciclo de vida de los proyectos, comunicación entre distintos ingenieros. Las principales herramientas son KnowledgeWare's Application Development Workbench, TI's, Information Engineering Facility (IEF), y Andersen Consulting's Foundation for Cooperative Processing.
Deberes de una herramienta CASE Cliente / servidor:
Ø Proporcionar topologías de aplicación flexibles. La herramienta debe proporcionar facilidades de construcción que permita separar la aplicación (en muchos puntos diferentes) entre el cliente, el servidor y más importante, entre servidores.
Ø Proporcionar aplicaciones portátiles. La herramienta debe generar código para Windows, OS/ 2, Macintosh, Unix y todas las plataformas de servidores conocidas. Debe ser capaz, a tiempo de corrida, desplegar la versión correcta del código en la máquina apropiada.
Ø Control de Versión. La herramienta debe reconocer las versiones de códigos que se ejecutan en los clientes y servidores, y asegurarse que sean consistentes. También, la herramienta debe ser capaz de controlar un gran número de tipos de objetos incluyendo texto, gráficos, mapas de bits, documentos complejos y objetos únicos, tales como definiciones de pantallas y de informes, archivos de objetos y datos de prueba y resultados. Debe mantener versiones de objetos con niveles arbitrarios de granularidad; por ejemplo, una única definición de datos o una agrupación de módulos.
Ø Crear código compilado en el servidor. La herramienta debe ser capaz de compilar automáticamente código 4GL en el servidor para obtener el máximo performance.
Ø Trabajar con una variedad de administradores de recurso. La herramienta debe adaptarse ella misma a los administradores de recurso que existen en varios servidores de la red; su interacción con los administradores de recurso debería ser negociable a tiempo de ejecución.
Ø Trabajar con una variedad de software intermedios. La herramienta debe adaptar sus comunicaciones cliente / servidor al software intermedio existente. Como mínimo la herramienta debería ajustar los temporizadores basándose en, si el tráfico se está moviendo en una LAN o WAN.
Ø Soporte multiusuarios. La herramienta debe permitir que varios diseñadores trabajen en una aplicación simultáneamente. Debe gestionarse los accesos concurrentes a la base de datos por diferentes usuarios, mediante el arbitrio y bloqueos de accesos a nivel de archivo o de registro.
Ø Seguridad. La herramienta debe proporcionar mecanismos para controlar el acceso y las modificaciones a los que contiene. La herramienta debe, al menos, mantener contraseñas y permisos de acceso en distintos niveles para cada usuario. También debe facilitar la realización automática de copias de seguridad y recuperaciones de las mismas, así como el almacenamiento de grupos de información determinados, por ejemplo, por proyecto o aplicaciones.
Ø Desarrollo en equipo, repositorio de librerías compartidas. Debe permitir que grupos de programadores trabajen en un proyecto común; debe proveer facilidades de check-in/ check-out registrar formas, widgets, controles, campos, objetos de negocio, DLL, etc.; debe proporcionar un mecanismo para compartir las librerías entre distintos realizadores y múltiples herramientas; Gestiona y controla el acceso multiusuario a los datos y bloquea los objetos para evitar que se pierdan modificaciones inadvertidamente cuando se realizan simultáneamente.

11. Factores asociados a la implantación de las herramientas case

La difusión de las innovaciones en esta área ha comenzado a estudiarse a partir de los años 1940. Por ello, existen estudios teóricos al respecto, realizándose evaluaciones, adopción e implementación tecnológica.

Existe un amplio cuerpo de investigaciones disponibles sobre la adopción de innovaciones. Muchos de los estudios sobre innovación se han analizado bajo dos perspectivas: adopción y difusión (Kimberly, 1981). Mientras unos estudios usan la perspectiva de la adopción para evaluar la receptividad y los cambios de la organización o sociedad por la innovación, otros usan la perspectiva de la difusión para intentar entender por qué y cómo se difunde y qué características generales o principales de la innovación son aceptadas.

12. Conclusión

Sin lugar a dudas las herramientas CASE han venido a revolucionar la forma de automatizar los aspectos clave en el desarrollo de los sistemas de información, debido a la gran plataforma de seguridad que ofrecen a los sistemas que las usan y es que éstas, brindan toda una gama de componentes que incluyen todas o la mayoría de los requisitos necesarios para el desarrollo de los sistemas, han sido creadas con una gran exactitud en torno a las necesidades de los desarrolladores de sistemas para la automatización de procesos incluyendo el análisis, diseño e implantación.

Las Herramientas CASE se clasifican por su amplitud en: TOOLKIT, WORKBENCH además también se pueden dividir teniendo en cuenta las fases del ciclo de vida que automatizan: UPPER CASE, MIDDLE CASE, LOWER CASE.

Debido a la gran demanda que tienen las CASE su exigencia en cuanto a su uso ha ido aumentando, por lo que toda CASE debe entre otras cosas:

Proporcionar topologías de aplicación flexibles
Proporcionar aplicaciones portátiles
Brindar un Control de versión
Crear código compilado en el servidor
Dar un Soporte multiusuario
Ofrecer Seguridad
Desde que se crearon éstas herramientas (1984) hasta la actualidad, las CASE cuentan con una credibilidad y exactitud que tienen un reconocimiento universal, siendo usadas por cualquier desarrollador y / o programador que busca un resultado óptimo y eficiente, pero sobre todo que busca esa minuciosidad necesaria de los procesos y entre los procesos.

13. Bibliografía

Analisis Y Diseño De Sistemas
3ª. Edición
Kendall & Kendall
Páginas 15.16.17.18

http://ceds.nauta.es/Catal/Products/caselist2.htm
http://www3.uji.es/~mmarques/f47/apun/node75.html
www.monografias.com
http://www.iscmolina.com/Herramientas%20CASE.html

Herramientas CASE

Cuando se hace la planificación de la base de datos, la primera etapa del ciclo de vida de las aplicaciones de bases de datos, también se puede escoger una herramienta CASE (Computer-Aided Software Engineering) que permita llevar a cabo el resto de tareas del modo más eficiente y efectivo posible. Una herramienta CASE suele incluir:

- Un diccionario de datos para almacenar información sobre los datos de la aplicación de bases de datos.
- Herramientas de diseño para dar apoyo al análisis de datos.
- Herramientas que permitan desarrollar el modelo de datos corporativo, así como los esquemas conceptual y lógico.
- Herramientas para desarrollar los prototipos de las aplicaciones.

El uso de las herramientas CASE puede mejorar la productividad en el desarrollo de una aplicación de bases de datos. Y por productividad se entiende tanto la eficiencia en el desarrollo, como la efectividad del sistema desarrollado. La eficiencia se refiere al coste, tanto en tiempo como en dinero, de desarrollar la aplicación. La efectividad se refiere al grado en que el sistema satisface las necesidades de los usuarios. Para obtener una buena productividad, subir el nivel de efectividad puede ser más importante que aumentar la eficiencia.