TEMA

58

Ayudas automatizadas para el desarrollo del "software" (herramientas CASE). Tipos. Estructura. Prestaciones

Esteban Leyva Cortés

CUERPO DE PROFESORES DE ENSEÑANZA SECUNDARIA

ÍNDICE SISTEMÁTICO

1. INTRODUCCIÓN

El nombre genérico de herramientas ***CASE*** procede del inglés *Computer Aided Software Engineering*, que se puede traducir como *Ingeniería del Software Asistida por Ordenador*. Una herramienta *CASE* es una aplicación, o un conjunto de aplicaciones, que permiten desarrollar y mantener un proyecto. El popular dicho de *En casa de herrero, cuchillo de palo* se puede aplicar perfectamente al caso que estamos estudiando. Curiosamente las herramientas *CASE* han sido las últimas ayudas al diseño aparecidas para el ordenador, siendo muy anteriores las herramientas *CAD*[1].

Las ayudas automatizadas para el desarrollo del *software* pretenden facilitar el desarrollo de aplicaciones, haciendo hincapié en la fase de **análisis** del mismo, así como garantizar la generación de una documentación completa y automática de las aplicaciones.

Las principales características de una herramienta *CASE* son:

- **Entorno de desarrollo interactivo**: podemos trabajar de forma interactiva, es decir, sobre nuestro proyecto vamos creando directamente nuevos elementos, y viendo los resultados en el acto.
- **Tiempo de respuesta rápido**: cualquier modificación, o elemento que añadamos producirá una alteración inmediata del sistema.
- **Comprobación de errores**: el programa comprueba directamente la coherencia, consistencia e integridad de nuestro sistema.
- **Automatización de las tareas de desarrollo y mantenimiento**: las tareas repetitivas y rutinarias en el desarrollo de aplicaciones se realizan de forma automática mediante las especificaciones dadas por el usuario.
- **Potentes interfaces gráficas**: trabajamos directamente con diagramas, esquemas, etc., como si de un programa de diseño gráfico se tratase.

La implantación de estas herramientas es todavía deficitaria, ya que en muchos centros de desarrollo siguen sin disponer de este tipo de programas. Los escollos que se han de superar para alcanzar una mayor difusión son su alto precio, y la falta de formación de los técnicos en esta nueva filosofía. Sin embargo, es interesante resaltar que en proyectos de la administración, seguir una metodología de desarrollo es un requisito que el gobierno impone, por lo que el uso de las mismas se hace imprescindible.

La mayoría de las herramientas *CASE* comerciales aparecieron a finales de los años ochenta.

2. HISTORIA

El primer antecesor de las herramientas CASE aparece en los años setenta. En el proyecto ISDOS se desarrolla un lenguaje denominado PSL (*Problem Statement Language*: Lenguaje de sentencias de problemas) y PSA (*Problem Statement Analizer*: Analizador de sentencias de problemas). El primer lenguaje permitía a los usuarios definir problemas y el segundo asociaba los problemas de los usuarios con posibles soluciones. La finalidad del proyecto ISDOS consistía en crear un diccionario computerizado.

Hasta el año 1984 no encontramos la primera herramienta CASE tal como la conocemos hoy en día. Se trata de "Excelerator" y estaba programada para PC.

---

[1] *Computer Aided Design: Diseño asistido por ordenador.* Herramienta informática usada en Ingeniería para diseñar piezas, máquinas, edificios, etc.

3. PRESTACIONES

Las prestaciones que nos ofrece una herramienta *CASE* son:

- **Mayor calidad del software**: el uso de una buena metodología asegura programas más coherentes, cohesionados, y con menos errores.
- **Aumentar la productividad**: la automatización y comprobación automática de los errores redundan en un mayor rendimiento.
- **Disminuir los errores**: estas ayudas comprueban la coherencia del proyecto, su integridad, y consistencia de forma automática, por lo que se reducen considerablemente los fallos.
- **Reducir costes**: muchas tareas se realizan de forma automática, por lo que es más barato crear programas.
- **Facilitar el mantenimiento**: variando las especificaciones del proyecto, se genera de forma automática el nuevo sistema con su correspondiente documentación.
- **Mejorar la documentación de los proyectos**: la documentación se genera de forma automática empleando todas las características propias de la metodología que usemos.
- **Generación automática de código**: algunas *CASE* generan parte del código fuente del proyecto, e incluso la aplicación completa. De esta forma, la tarea de programación se reduce considerablemente.
- **Prototipado rápido**: podemos mostrar al cliente prototipos funcionales de nuestra aplicación, antes de desarrollar el proyecto. Así, puede hacerse una idea de cómo va a quedar el programa una vez ya finalizado.

4. TIPOS

Existen muchas clasificaciones de herramientas, al igual que existen muchos tipos de herramientas CASE. Atendiendo a la función y a la clasificación que realiza Pressman[2] destacaremos los siguientes tipos:

- **Herramientas de ingeniería de procesos de negocio**: supone un marco en el que encuadrar la información de negocio cuando se transfiere entre distintas entidades. De esta manera es posible estudiar la forma en que se relacionan los objetos de un negocio y el flujo de los mismos entre las diferentes entidades que lo conforman.
- **Modelado de procesos**: posibilitan crear un modelo sobre procesos (tanto de negocio como software) de manera que podemos entender mejor el funcionamiento de un negocio o proceso.
- **Herramientas de planificación de proyectos**: a su vez, en esta categoría podemos distinguir entre:
  * **La estimación de costes y esfuerzos**: la herramienta CASE permite calcular el coste del proyecto, número de personas necesarias, duración aproximada del mismo, etc.
  * **Planificación de proyectos**: permite al ingeniero del proyecto estudiar las tareas a realizar, las dependencias entre ellas, aplicarle el paralelismo posible, estudiar diferentes alternativas de realización del mismo, etc.

---

[2] Pressman, Roger S.: *Ingeniería del Software. Un enfoque práctico*. 5ª edición. McGraw-Hill. 2002.

- **Herramientas de análisis de riesgos**: con la ayuda de esta herramienta el gestor del proyecto puede estudiar los diferentes riesgos, formas de evitarlos, posibles soluciones, etc.
- **Herramientas de gestión de proyectos**: permite realizar un seguimiento exhaustivo del progreso del proyecto.
- **Herramientas de seguimiento de requisitos**: consiste en una ayuda que permite almacenar en una base de datos todos los requisitos del cliente agrupados por categorías, así resulta bastante sencillo comprobar los requisitos del cliente en cualquier momento y realizarle un seguimiento.
- **Herramientas de métricas**: las métricas posibilitan un estudio de la calidad del software o del proceso de desarrollo del mismo.
- **Herramientas de documentación**: existen herramientas para desarrollar la documentación relativa a un proceso o producto. Tenga en cuenta que se estima que aproximadamente entre el 20% y 30% del esfuerzo de desarrollo de un proyecto se dedica a la documentación.
- **Herramientas de control de calidad**: la mayoría de estas herramientas permiten, a través de métricas, comprobar la calidad de un proceso o producto.
- **Herramientas de gestión de bases de datos**: actualmente las herramientas CASE pueden evolucionar a partir de los SGBDR[3] o SGBDOO[4], dicho de otra manera, contienen una base de datos (repositorio) sobre la cual trabaja. Es por lo que una herramienta CASE debe contener una herramienta de gestión de bases de datos.
- **Herramientas de análisis y diseño**: posibilitan la creación de modelos del producto a desarrollar.
- **Herramientas PRO/SIM**: permiten crear prototipos y simulaciones de un sistema antes de su implementación. Estas herramientas generan diseños esquemáticos en lenguajes como C o ADA.
- **Herramientas de desarrollo y diseño de interfaz**: con estas herramientas podemos diseñar y desarrollar el interfaz con el usuario. Actualmente están siendo desplazamos por las herramientas de construcción de prototipos.
- **Herramientas de construcción de prototipos**: existen una gran variedad de software para generar prototipos de nuestras aplicaciones como generadores de pantallas, generadores de informes, etc.
- **Herramientas de programación**: dentro de esta categoría tenemos compiladores, editores, intérpretes, depuradores, entornos de programación gráfica, lenguajes de acceso a bases de datos, etc.
- **Herramientas de desarrollo Web**: disponemos de una amplia selección de programas como ayuda a la generación de código, generadores de formularios, menús, animaciones, etc.
- **Herramientas de integración y pruebas**: podemos subdividirlas en:
  * **Adquisición de datos**: obtienen los datos que se van a emplear en una prueba.
  * **Medida estática**: estudian el código fuente de un programa sin llegar a ejecutarlo.

---

[3] SGBDR: Sistema Gestor de Bases de Datos Relacionales.

[4] SGBOO: Sistema Gestor de Bases de Datos Orientadas a Objetos.

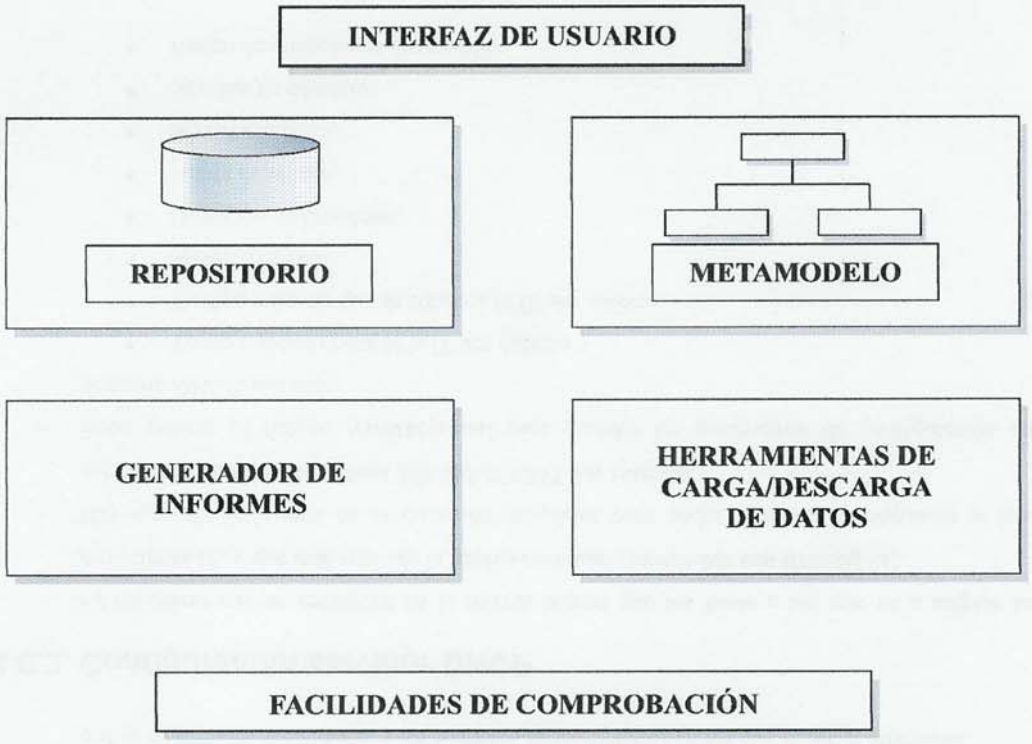
* **Medida dinámica**: estudian el código fuente de un programa mientras se ejecuta.
* **Simulación**: emula las funciones de elementos hardware o externos no existentes para la prueba.
* **Gestión de pruebas**: permiten monitorizar todas las pruebas realizadas a un sistema, coordinar las mismas y controlar cada prueba de forma eficiente.
* **De funcionalidad cruzada**: realizan una o más tareas de las anteriormente citadas.

- **Herramientas de reingeniería**: se encargan del mantenimiento del software. Dentro de estas herramientas podemos diferenciar entre:
  * **Herramientas de ingeniería inversa para producir especificaciones**: a partir del código fuente de un programa, genera información sobre el análisis y diseño del programa.
  * **Herramientas de reestructuración y análisis de código**: a partir de la sintaxis del programa obtenemos un gráfico del flujo de control del mismo.
  * **Herramientas de reingeniería para sistema en línea**: permiten modificar bases de datos usadas en línea.

5. ESTRUCTURA

Una herramienta *CASE* se divide en:

- **Interfaz de usuario**: está conformado por editores de textos, y herramientas de diseño gráfico que permiten definir diagramas, esquemas, etc., de las diferentes metodologías de desarrollo. El interfaz de usuario es utilizado por todos los demás componentes de la herramienta.
- **Repositorio o diccionario**: almacena los elementos definidos o creados por la herramienta *CASE* (diagramas de flujos de datos, diseños de pantalla, diagramas entidad-relación, algoritmos, etc.), con sus datos correspondientes. Es la base de datos central de nuestro proyecto. Amplía el concepto de diccionario de datos. Es el componente más importante de una herramienta *CASE*. La existencia del repositorio tiene muchas ventajas como, por ejemplo, evitamos que existan datos no definidos, datos duplicados, etc.
- **Metamodelo**: este componente no está siempre visible al usuario. Constituye el marco en el que se definen las diferentes técnicas, y metodologías soportadas por la herramienta.
- **Generador de informes**: permite obtener información sobre la aplicación que estamos desarrollando. Toda la base de esta documentación se extrae directamente desde el repositorio.
- **Herramientas para cargar/descargar datos**: es un medio de comunicación con otras herramientas *CASE*. Podemos, por ejemplo, cargar el repositorio con datos de otros sistemas, generar esquemas de bases de datos, programas, que alimenten a otros sistemas, etc.
- **Facilidades de comprobación**: verifica la exactitud, integridad, y consistencia de los esquemas desarrollados por el usuario de la aplicación.



6. CICLO DE VIDA TRADICIONAL FRENTE AL CICLO DE VIDA *CASE*

El uso de las *CASE* no significa sólo una ayuda para las tareas de ingeniería, sino que el uso de estas herramientas supone un **importante giro en la filosofia** de la creación de aplicaciones. Vamos a comparar el ciclo de vida tradicional de una aplicación con el ciclo de vida usando una herramienta *CASE*.

En el ciclo de vida tradicional, hasta que la aplicación no está completamente terminada no podemos ver realmente cómo va a quedar. De esta manera podemos tener problemas ya que, normalmente, el cliente que encarga la aplicación no suele saber lo que realmente necesita. Este problema se soluciona en las herramientas *CASE* con el uso del **prototipado**, con lo que podemos generar un modelo previo para que el cliente lo pruebe, antes de terminar la aplicación completa. Además, este prototipo permite la **iteración**, es decir, podemos modificarlo y ver inmediatamente los resultados.

En una aplicación desarrollada de la forma tradicional la mayor parte del **esfuerzo** se emplea en la **programación**. Sin embargo, en una aplicación creada mediante una herramienta *CASE*, las fases de **análisis** y **diseño** son las que requieren mayor dedicación.

La **codificación** del programa, en la metodología tradicional, se realiza de forma **manual**. El programador va creando el programa instrucción a instrucción. Si empleamos una ayuda para el diseño, el código del programa lo genera **automáticamente** la herramienta.

La **documentación** que requiere un proyecto informático se genera **automáticamente** en las herramientas *CASE*. En el sistema tradicional, somos nosotros quienes hemos de crear **manualmente** la documentación.

Para **comprobar** que nuestra aplicación tiene un diseño correcto hemos de realizar **pruebas** del software. En una herramienta *CASE* el diseño se comprueba **automáticamente**.

El **mantenimiento** de nuestro programa en la metodología tradicional pasa por la alteración del **código**. En una herramienta *CASE* el mantenimiento se realiza modificando las especificaciones de la aplicación.

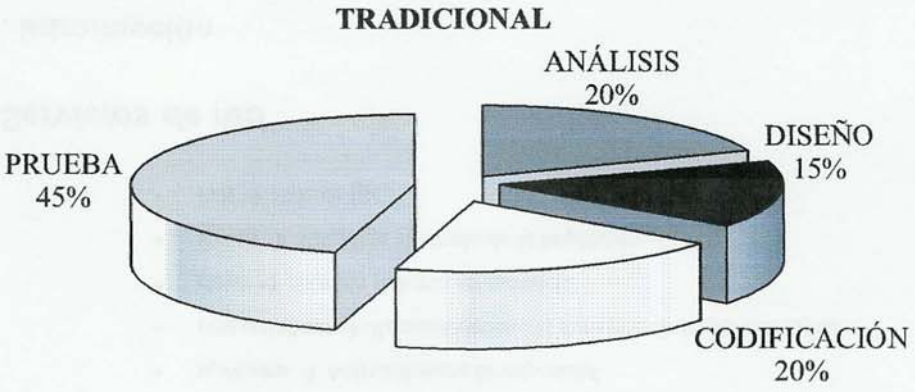
Todas las ideas anteriormente citadas las podemos resumir en la tabla siguiente:

Tradicional	*CASE*
Desarrollo tradicional.	Prototipo rápido e iterativo.
Énfasis en la codificación y programación de la aplicación.	Énfasis en las fases de análisis y diseño.
Codificación manual.	Generación automática del código.
Documentación manual.	Generación automática de la documentación.
Pruebas de software.	Comprobación automática del diseño.
Mantenimiento del código de la aplicación.	Mantenimiento de las especificaciones del diseño.

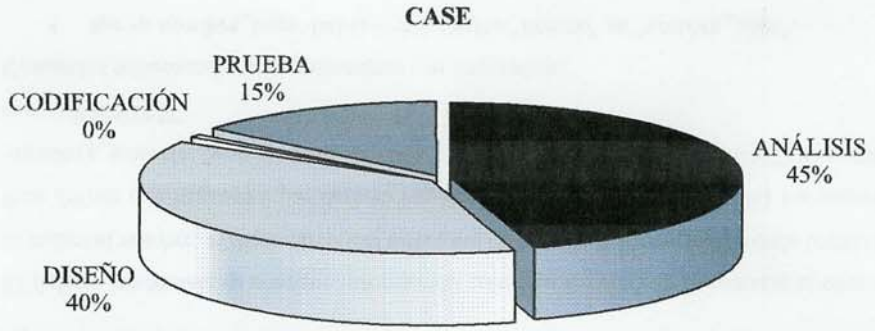
Como hemos citado anteriormente, el empleo de las herramientas *CASE* va a modificar la distribución tradicional del **esfuerzo** en la fase de creación de una aplicación, de forma que la mayor parte del trabajo se va a emplear en la fase de análisis y diseño de la aplicación. Podemos realizar una comparativa de la distribución del esfuerzo si usamos una herramienta *CASE* o si empleamos una herramienta tradicional.

Como podemos comprobar, en la metodología clásica el esfuerzo dedicado al análisis y el diseño representa menos del 35%, lo que se traduce en que se pasa rápidamente a la fase de codificación.

Esta distribución del esfuerzo nos lleva a sistemas mal especificados y de alto coste. En un sistema de este tipo se produce fácilmente la propagación de errores, y es muy difícil modificar los requisitos iniciales de la aplicación. Además, dan lugar a proyectos mal documentados.



En el caso de la metodología *CASE*, el esfuerzo se centra en el análisis y el diseño. Es un hecho significativo que la suma del esfuerzo entre análisis y el diseño sea del 85%. La codificación se realiza de forma automática, lo que explica que se le dedique el 0% del tiempo.



Una aplicación creada con una herramienta *CASE* permite automatizar el análisis y el diseño, y la comprobación automática de errores. Esto redunda en un incremento de la productividad, y una disminución de los errores. A su vez, las aplicaciones quedan perfectamente documentadas.

7. INTEGRACIÓN DE LAS HERRAMIENTAS *CASE*

Lógicamente, nuestra herramienta *CASE* no la vamos a utilizar de forma aislada. Esta herramienta hemos de integrarla con el resto de las aplicaciones que usamos para crear la aplicación.

Hay diferentes estrategias que pueden servir para integrar nuestra herramienta *CASE*:

- **Sin integración**: cada herramienta se usa de forma independiente. El único enlace que existe entre ellas se realiza mediante documentos impresos con información sobre nuestro proyecto.
- **Intercambio de datos**: el enlace, entre los distintos módulos, ya no se hace mediante documentos impresos, sino mediante ficheros. De esta forma evitamos los posibles errores al copiar, e interpretar la documentación. Casi todas las herramientas *CASE* permiten exportar nuestros datos a un fichero. Este archivo puede encontrarse sin formato, o bien, puede emplear el formato de otra aplicación. Este modelo tiene muchos inconvenientes porque pueden quedarse algunos archivos desfasados, además, cada vez que modifiquemos la aplicación es necesario reimportar todos los datos en los módulos en los que afecten esas alteraciones. Por si esto fuera poco, sólo podemos transferir los datos en un sentido. Por lo tanto, es muy difícil mantener la integridad, coherencia y cohesión de los datos. La única ventaja que tiene este modelo es que tenemos libertad de elegir diferentes programas para realizar cada una de las diferentes tareas.
- **Acceso común a herramientas**: además de los datos, integramos las diferentes herramientas conjuntamente, en un entorno integrado de desarrollo, o en un entorno multitarea. Con un botón, o una opción, se pasan automáticamente los datos de un módulo a otro. Esta solución es mejor que la anterior ya que es más cómodo exportar los datos de un módulo a otro.
- **Integración de datos**: en este caso el único elemento que se encuentra integrado son los datos. Poseemos dos opciones distintas:
  * **Gestión común de datos**: usamos una única base de datos a la que acceden todos los módulos.
  * **Datos compatibles**: cada herramienta es compatible con las demás, por lo que suelen ser del mismo fabricante, o atiende a algún estándar ampliamente usado.
- **Integración total**: cada herramienta puede gestionar de forma independiente y autónoma los metadatos. Además cada módulo ha de ser capaz de controlar al resto. Es decir, cada herramienta puede solicitar información, comprobación de errores, o la realización de cualquier tarea a otro módulo cuando lo estime oportuno.

8. HERRAMIENTAS *CASE* COMERCIALES

- **Easy *CASE***: herramienta para *PC* bajo *Windows 3.11* o *95*. Fabricado por *Evergreen CASE Tools Inc*. Tal como su nombre apunta, es muy sencilla de usar. Ocupa poco espacio. Si a esto le unimos que es barata, y no requiere un ordenador muy potente, podemos entender por qué es tan popular. Es una herramienta del tipo **frontal**. Tiene facilidades para diseñar bases de datos.
- **Microsoft Project 2000**: es un administrador de cualquier tipo de proyectos, no sólo informáticos. Es, por lo tanto, una **herramienta de gestión**. Es un programa muy completo, que se complementa con *Microsoft Project Central*. Esta es una aplicación servidora, que permite el acceso remoto al proyecto tanto al director del proyecto, como a los demás colaboradores. Funciona bajo el sistema operativo *Windows*.
- **Excelerator**: es una utilidad disponible en muchas plataformas (*PC, SUN y APOLLO*). Es muy sencilla de usar, mediante menús. Dispone de un *diccionario de proyectos*, que es el corazón de este sistema.
- **Structured architecture**: disponible para ordenadores *PC*. Podemos disponer de una versión educacional de sencillo manejo, y bajo precio. Sólo podemos usar la metodología de *DeMarco*. El sistema gira en torno al repositorio, que aquí recibe el nombre de **enciclopedia**.
- **Oracle *CASE***: la importancia de los sistemas gestores de bases de datos de la empresa Oracle ha propiciado la aparición de una ayuda para la ingeniería del software específica para este programa.

9. INGENIERÍA INVERSA

La ingeniería inversa es lo opuesto a la ingeniería del software. Si la ingeniería del software sirve para planificar, gestionar y aplicar una metodología a un proyecto durante su desarrollo y preparación, en la ingeniería inversa, a partir del proyecto concluido, extraemos los modelos de datos, especificaciones, etc. Esa información se exporta al repositorio, y se reconstruye la aplicación. Así podemos documentarla correctamente, y mantenerla de una manera sencilla.

Algunas *CASE* permiten realizar funciones de ingeniería inversa. Por supuesto que lo ideal es usar una herramienta *CASE* durante el desarrollo del proyecto pero, para aplicaciones antiguas, es más sencillo usar una de estas utilidades para documentarla, y reformarla de una forma coherente, que empezar de cero.

Con la aplicación ya reformada, podemos trabajar empleando las ventajas de estas herramientas.

10. ELECCIÓN DE UNA HERRAMIENTA *CASE*

A la hora de elegir una herramienta *CASE*, en el mercado encontraremos muchas posibilidades. Algunas de las variables que hemos de sopesar antes de decantarnos por una ayuda para la ingeniería han de ser las siguientes:

- **Envergadura del proyecto**: lógicamente cuando mayor, o más compleja sea la aplicación a desarrollar, más completa ha de ser la herramienta a usar. Si bien para un proyecto pequeño nos servirá una ayuda que imprima algunos diagramas, y alguna documentación básica. Para un proyecto complejo necesitaremos una buena ayuda para el ingeniero del software.

- **Fecha de entrega de la aplicación**: si el tiempo apremia, es mejor usar una herramienta de tipo **dorsal**, que dará sus frutos más rápidamente si disponemos de un modelo de datos, y procesos bien definidos.

- **Tamaño de los archivos de datos**: en el caso de que el programa a diseñar vaya a generar ingentes cantidades de datos, es recomendable usar una ayuda para el diseño que permita el diseño de base de datos.

- **Personal que va a usar la herramienta**: hemos de tener en cuenta la formación *"real"* de los integrantes del proyecto, si vamos a usar el acceso remoto a la herramienta, si vamos a trabajar en grupo, y además según la función que realice cada colaborador, hemos de permitirle diferentes accesos, y el uso de diferentes módulos.

- **Presupuesto del proyecto**: desgraciadamente las *CASE* son bastante caras, por lo que el coste de la adquisición de este producto es uno de los factores determinantes en la elección de la misma.

- **Posibilidad de uso en el futuro**: quizás ahora podamos bastarnos con una ayuda sencilla, pero en el futuro puede quedársenos pequeña, con lo que tendríamos que comprar una *CASE* nueva, y transvasar todos los datos de la herramienta antigua a la nueva, con la consiguiente pérdida de tiempo.

- **La plataforma que se va a emplear**: hemos de tener en cuenta la potencia de los ordenadores donde vamos a instalar nuestro proyecto, y el sistema operativo que usaremos. Desgraciadamente, estas utilidades son auténticas devoradoras de recursos, es decir, son muy exigentes en cuanto a la máquina en la que funcionan.

11. TENDENCIAS EN LAS *CASE*

Este tipo de utilidades son relativamente nuevas, se encuentran en constante desarrollo. Todavía son muchas las innovaciones con las que nos van sorprendiendo las empresas dedicadas al *CASE*. A continuación mencionaremos algunas de las más interesantes:

- **Cliente/Servidor**: la mayoría de estas aplicaciones funcionan bajo un entorno cliente/servidor, pero eso no quiere decir que estén preparadas para funcionar de esta forma. Podemos distinguir dos claras vertientes:

  * **Herramientas que funcionan como cliente/Servidor**: la herramienta *CASE* corre bajo un entorno *cliente/servidor*.

  * **Herramientas para diseñar aplicaciones que usen un entorno cliente/servidor**: los programas que diseñemos con la utilidad van a ser programas del tipo *cliente/servidor*.

- **Multiplataforma**: podemos usar muchos sistemas operativos, ordenadores, interfaces gráficas, metodologías, entornos programación, etc. Este es uno de los avances más interesantes, que aumentarán considerablemente la portabilidad, y difusión de estas utilidades.

- ***CASE* orientados a objetos**: actualmente ninguna *CASE* cubre todo el ciclo de desarrollo de una aplicación orientada al objeto, pero sí van apareciendo utilidades que cubren uno o varios ciclos de vida de una aplicación orientada al objeto. La mayoría de estas generan código en *C++*.

- ***CASE* para trabajo en grupo**: su principal función, más que trabajar el proceso de desarrollo de un proyecto, se centra en coordinar e integrar grupos de trabajo participantes en un proyecto.

Es un poco arriesgado vaticinar los derroteros por los que se moverán estas aplicaciones en el futuro. Algunas de las posibles innovaciones que veremos próximamente, referentes a este tema se pueden basar en el uso de la *realidad virtual*, los *sistemas multimedia*, o la *inteligencia artificial*, entre otros.

De todas formas, la presencia de estas utilidades en centros de formación, y de desarrollo del software es cada vez más patente y necesaria.