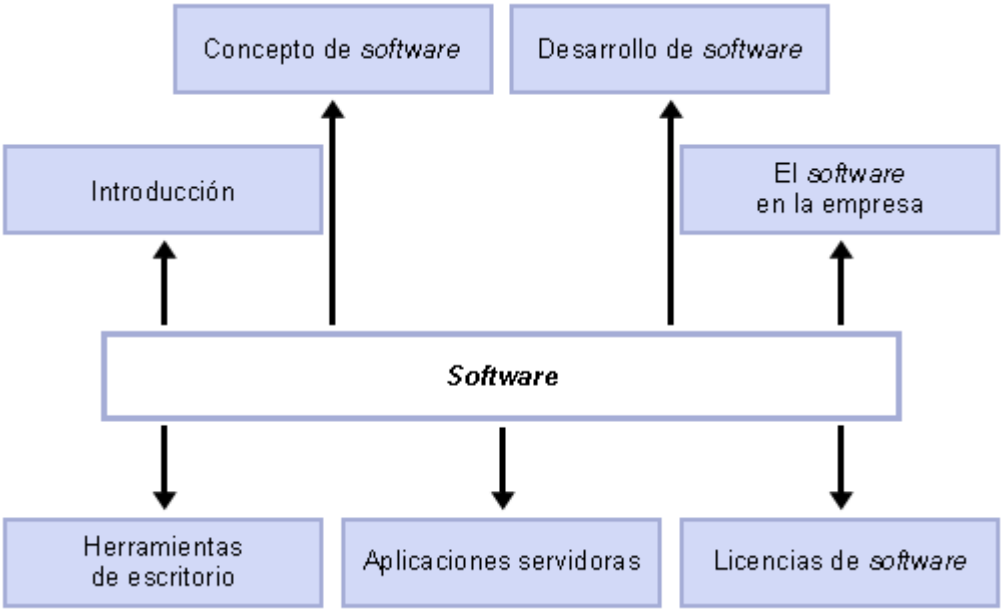
Software



Objetivos

Este módulo pretende introducir al alumno en el concepto de *software* y sus principales aplicaciones, así como en el desarrollo de las aplicaciones y programas.

A la finalización del estudio del módulo, el alumno será capaz de:

- Conocer una breve muestra de algunos de los lenguajes de programación más conocidos y utilizados, identificando las diferentes generaciones de los mismos.
- Identificar la necesidad de una ingeniería del *software* y las principales características de una herramienta CASE.
- Analizar la aplicación del *software* en los diferentes departamentos y áreas de una organización, identificando las principales funcionalidades de diferentes herramientas de escritorio y aplicaciones servidoras.
- Describir el concepto de licencias de *software* enumerando diferentes formas de piratería y ejemplos de posibilidades y opciones en la compra y uso de licencias.

Plan de trabajo

Módulo 4: *Software* (1,5 créditos)					
Títulos unidades	**Objetivos**	**Núcleos de conocimiento**	**Actividades**	**Lecturas**	**Tiempo**
Software	Describir el concepto de *software* y sus principales aplicaciones.	Concepto de *software*			2 h

Desarrollo de *software*	Describir el concepto de desarrollo de *software* para crear programas y aplicaciones. Examinar un breve mostruario de algunos de los lenguajes de programación más conocidos y utilizados. Identificar las diferentes generaciones de los lenguajes de desarrollo y distinguir las principales características de los lenguajes máquina, ensamblador y de alto nivel. Identificar la necesidad de una ingeniería del *software* y sus principales características. Describir las principales características de una herramienta CASE.	Introducción Lenguajes de programación Evolución de los lenguajes Lenguajes máquina, ensamblador y de alto nivel Introducción a la ingeniería del *software* Herramientas CASE		Miquel Barceló. "Lenguajes de programación" (módulo 3 del material en papel; pág. 15 a 19 y 35 a 40). Miquel Barceló. "Ejemplos de lenguajes orientados a objetos" (módulo 5 del material en papel; pág. 47 a 48). Miquel Barceló. "Introducción a la ingeniería del *software*" (módulo 5 del material en papel; pág. 12 a 17). Miquel Barceló. "Herramientas CASE" (módulo 5 del matertial en papel; pág. 29 a 32).	6 h
El *software* en la empresa	Analizar la aplicación del *software* en los diferentes departamentos y áreas de una organización.	Introducción	Debate	Juan Luis Cebrián. "Dónde está mi oficina". *La Red* (pág. 100-108).	3’5 h
Herramientas de escritorio	Identificar las principales funcionalidades de un procesador de texto, de un gestor de hojas de cálculo, de un editor de presentaciones, los organizadores personales y de los sistemas gestores de proyectos. Enumerar diferentes utilidades de *software*.	Procesadores de texto Hojas de cálculo Edición de presentaciones Organizadores personales Sistemas gestores de proyectos Otras aplicaciones	Actividad 1 Actividad 2 Actividad 3 Actividad 4		6 h
Aplicaciones servidoras	Describir las principales características de un servidor de mensajería	Mensajería y herramientas de colaboración	Actividad 5		3 h

	y colaboración, de un servidor de Internet y de una aplicación de cortafuegos y acceso a Internet. Identificar la necesidad de disponer de aplicaciones antivirus.	Servidores de Internet Cortafuegos y acceso a Internet Antivirus	Actividad 6	Miquel Barceló. "Los virus informáticos" (módulo 1 del material en papel; apartado 4.4).	
Licencias de *software*	Describir el concepto de licencias de *software*. Identificar las principales consecuencias del uso ilegal del *software* y enumerar las diferentes formas de piratería. Listar diferentes posibilidades y opciones en la compra y uso de licencias de *software.*	Introducción Consecuencias de la piratería del *software* Formas de piratería Políticas de compra y uso de licencias	Actividad 7 Actividad 8		2 h
					Total 22’5 h

Introducción

En los últimos tiempos, el *hardware* de computadoras y la comunicación entre éstas (redes de ordenadores-Internet) han experimentado un desarrollo vertiginoso.

Sin embargo, el aprovechamiento de dichos avances tecnológicos no sería óptimo si no se dispusiera del *software* adecuado. El ***software*** está compuesto por las diferentes **herramientas** y **programas** que trabajan en los ordenadores y nos permiten gestionar la información, automatizar tareas, monitorizar almacenes y cadenas de montaje, acceder a la información contable de la empresa, etc. Estas aplicaciones se programan con los llamados **lenguajes de programación**.

Con este fin se han diseñado varios tipos de lenguajes de programación, unos de propósito general y otros de aplicación específica en alguna de las áreas del ámbito informático.

La construcción de un programa o *software* sigue unos pasos y metodologías concretas para garantizar la programación y diseño óptimos y correctos del mismo. Este conjunto de metodologías, documentación y procesos se denominan **ingeniería del *software***.

Después de tratar el desarrollo del *software*, presentaremos diferentes tipos de **aplicaciones del *software* en las organizaciones**, incluyendo las herramientas de productividad personal y las aplicaciones corporativas.

El derecho al uso del *software* por parte de usuarios y compañías se adquiere mediante **licencias**, cuya problemática examinaremos al final del módulo.

Concepto de *software*

El *software* de un sistema informático está constituido por el **conjunto de programas** ejecutables en dicho sistema.

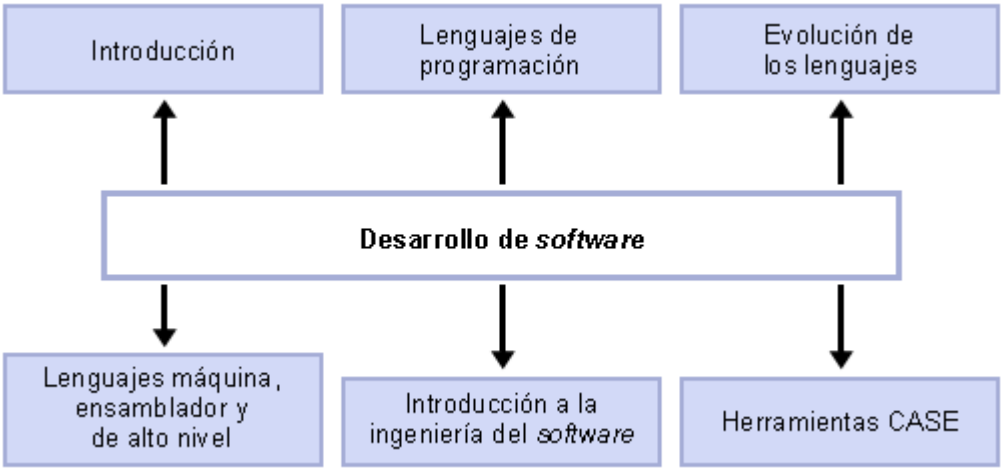
Dentro del *software* se incluyen el sistema operativo, las interfaces de usuario, los lenguajes de programación, las herramientas o utilidades, las aplicaciones de escritorio, los servidores de comunicación, etc.

La combinación del *hardware* (ordenadores, redes y sistemas de comunicación) con el *software* ha permitido **mejoras en ámbitos muy variados**:

- Negocios
- Medicina
- Educación
- Ciencia y tecnología
- Ingeniería y arquitectura
- Producción
- Logística y distribución
- Atención al cliente

Este conjunto de *software* se desarrolla utilizando los diferentes lenguajes de programación y siguiendo unas pautas y metodologías: la ingeniería del *software*.

Desarrollo de software



Introducción

Los programadores utilizan **lenguajes de programación** para elaborar paquetes de *software* como, por ejemplo, procesadores de palabras y hojas de cálculo (los cuales pueden ser utilizados por cualquier persona que no tenga experiencia en programación).

Los programas que el ordenador puede ejecutar tienen que estar en el **lenguaje nativo** de ese procesador. Es decir, cada instrucción debe estar en **código binario** y directamente **relacionada** con los **circuitos del procesador**.

Programar instrucciones completamente en un código binario es un proceso demasiado lento, difícil y sujeto a errores incluso para los programadores más hábiles y experimentados.

Los lenguajes de programación han sido diseñados para poder escribir instrucciones parecidas a un lenguaje humano, por lo general el inglés, que las computadoras pueden convertir a código binario mediante los llamados programas compiladores.

En resumen, los lenguajes de programación facilitan la programación del ordenador.

Los **compiladores** traducen las instrucciones de un **lenguaje de programación de alto nivel** a **código binario**, de bajo nivel, de forma similar a como una persona bilingüe podría traducir del francés al castellano, por ejemplo. Así, un compilador o **traductor** convierte los programas que el desarrollador elaboró utilizando un lenguaje de alto nivel en un **código fuente** y **código objeto**. El **código fuente** es el conjunto de instrucciones escritas en un lenguaje de programación; el **código objeto**, el conjunto de instrucciones binarias ya traducidas que puede ejecutar la

computadora.

Lenguajes de programación

> **Un lenguaje de programación es un conjunto de símbolos unido a un conjunto de reglas -que sirven para combinar dichos símbolos- y que se utilizan para expresar órdenes.**

Los lenguajes de programación están compuestos por un **léxico** (conjunto de símbolos permitidos o vocabulario), una **sintaxis** (reglas que indican cómo realizar las construcciones del lenguaje) y una **semántica** (reglas que permiten determinar el significado de cualquier construcción del lenguaje).

Para que una computadora pueda procesar un programa escrito en un determinado lenguaje de programación es necesario realizar antes una traducción del programa al lenguaje que entienda esa computadora según una serie de fases.

Evolución de los lenguajes

Veamos ahora cómo han surgido los lenguajes de programación.

Primera generación: lenguaje máquina

Cada computadora tiene sólo un lenguaje de programación que su procesador puede ejecutar; pues bien, éste es su **lenguaje nativo** o **lenguaje de máquina**.

Los programas en lenguaje máquina se escriben en el nivel más básico de operación de la computadora. Las instrucciones se codifican como una serie de unos (1) y ceros (0). Estos programas son complicados de leer y difíciles de escribir.

Segunda generación: lenguaje ensamblador

Para evitar que los programadores tuvieran que programar directamente en código binario o máquina, se desarrollaron unos **programas** para **traducir instrucciones a código de máquina**. Estos programas se llamaron **ensambladores**, puesto que leían las instrucciones que las personas podían entender en lenguaje ensamblador y las convertía al lenguaje máquina.

El lenguaje ensamblador también es de **bajo nivel**, ya que cada instrucción de este lenguaje corresponde a una instrucción de lenguaje maquinal.

Cada procesador posee su propio lenguaje ensamblador. Éste traduce el código fuente, línea por línea, a código de máquina y crea el archivo ejecutable del programa.

Tercera generación: lenguajes de alto nivel

Estos lenguajes son parecidos al inglés y facilitan el trabajo de los desarrolladores de *software*.

Existen muchos lenguajes de tercera generación como, por ejemplo, COBOL, BASIC, FORTRAN, C, PASCAL, etc.

Con estos lenguajes, los programadores pueden escribir en una sola instrucción lo equivalente a varias instrucciones complicadas de bajo nivel. De esta manera, no tienen que concentrarse en la operación interna del procesador, como sucede en los lenguajes de las generaciones anteriores, y pueden ocuparse mejor de la aplicación que están programando. Por ejemplo: un sistema de gestión de RR.HH., un sistema de nóminas, un sistema gestor de proyectos, etc.

Independientemente del lenguaje de alto nivel en que se escriba un programa, un compilador deberá traducirlo **al lenguaje de máquina** para que, de este modo, el procesador pueda ejecutarlo.

Cuarta generación: lenguajes orientados al usuario (4GL)

El *software* de estos lenguajes **genera de forma automática** la mayor parte de los **procedimientos de un programa**. Así pues, el programador indica lo que se debe hacer, no cómo hacerlo. Un programador que trabaje con un lenguaje de tercer nivel, como por ejemplo Pascal, escribe instrucciones de lo que se debe hacer y de cómo hacerlo.

En cambio, con los lenguajes 4GL, los usuarios finales escriben sus programas de manera sencilla para consultar una base de datos y para crear sistemas de información personales o departamentales.

Muchos de estos lenguajes disponen de una **interfaz gráfica** y sólo obligan al usuario o programador a usar instrucciones sencillas y fáciles de manejar.

Estos lenguajes convierten las citadas especificaciones en:

- Lenguajes de tercer nivel que posteriormente un programador puede refinar.
- Lenguaje máquina, directamente.

Ejemplos de lenguaje 4GL son la mayoría de las **herramientas CASE** de diseño de bases de datos, modelado de procesos, etc.

Podéis consultar el núcleo de conocimiento "Herramientas CASE" de este mismo apartado.

Quinta generación: lenguajes naturales

Los lenguajes naturales se asemejan más al lenguaje humano que sus antecesores, los lenguajes 4GL. Incluso algunos productos comerciales han empezado a incluir características del lenguaje natural. Cada vez hay más programas de bases de datos que pueden ser consultados utilizando herramientas de consulta en lenguaje natural.

Aunque estos lenguajes se encuentran en sus inicios, la mayoría de las herramientas de uso y trabajo con el ordenador tenderán a este tipo de lenguajes.

Las generaciones de lenguajes sucesivas se han ido aproximando al lenguaje natural.

Lenguajes máquina, ensamblador y de alto nivel

A continuación profundizaremos en las principales características de los lenguajes máquina, ensamblador y de alto nivel.

Lenguaje máquina

El lenguaje máquina es el único lenguaje que entiende directamente el ordenador.

Su estructura está completamente adaptada a los circuitos de la máquina y muy alejada de la forma de expresión y análisis de los problemas propios de los humanos. La programación en este lenguaje es complicada, de manera que se requiere un profundo **conocimiento** de la **arquitectura física del ordenador**.

Sin embargo, el código máquina hace posible que el programador utilice la totalidad de los recursos que ofrece el ordenador, con lo que se obtienen **programas muy eficientes en tiempo de ejecución y en ocupación de memoria**, dado que aprovechan al máximo los recursos existentes.

Las principales características del lenguaje máquina son las siguientes:

- Las instrucciones se expresan en **código binario**. Están codificadas en binario como cadenas de ceros y unos. Esta característica provoca que un programa en lenguaje máquina sea difícil de entender y, como consecuencia, difícil de modificar.
- Los datos se referencian por medio de las **direcciones de memoria** donde se encuentran (no aparecen nombres de variables o de constantes).
- Las instrucciones realizan **operaciones muy simples**. El programador debe ingeniárselas para expresar cada una de las operaciones que desea realizar en términos de las instrucciones elementales de las que dispone.
- Hay muy **poca versatilidad** para la redacción de las instrucciones, puesto que tienen un formato rígido en lo que respecta a la posición de los distintos campos (código de operación seguido de los campos dedicados a los operandos).
- El lenguaje máquina depende y está íntimamente **relacionado con la CPU** de cada ordenador. Por este motivo, los programas en lenguaje máquina no son transferibles de un modelo de ordenador a otro. Un programa en lenguaje máquina sólo se puede ejecutar en el procesador para el que está destinado.
- En un programa en lenguaje máquina **no pueden incluirse comentarios** que faciliten su legibilidad y el posterior mantenimiento por parte del mismo programador u otros.

El lenguaje ensamblador y casi todos los lenguajes simbólicos de alto nivel resuelven estas limitaciones.

Lenguaje ensamblador

El lenguaje ensamblador constituye el primer intento de sustitución del lenguaje por uno más cercano al utilizado por los humanos.

Como ya comentamos al mencionar la evolución del *software*:

- Estos lenguajes utilizan una **notación simbólica o mnemotécnica** para representar los códigos de operación. De esta forma, se evitan los códigos numéricos, tan difíciles de manejar. Los códigos mnemotécnicos están constituidos por abreviaturas de las operaciones en inglés (dado el origen anglosajón de los fabricantes). Así, por ejemplo, la suma se representa en la mayoría de los ensambladores con ADD.
- **Direccionamiento simbólico**. En lugar de utilizar direcciones binarias absolutas a la memoria, los datos pueden ser identificados con nombres como Importe, Población, A, Y, etc.
- Se permite el **uso de comentarios** entre las líneas de instrucciones, lo cual hace posible la redacción de programas más legibles.

Podéis consultar el apartado "Segunda generación: lenguaje ensamblador" del núcleo de conocimiento "Evolución de los lenguajes".

No obstante, el **lenguaje ensamblador** presenta la mayoría de los **inconvenientes** que tiene el lenguaje máquina: un **repertorio muy reducido de instrucciones**, el **rígido formato de las instrucciones**, la **baja portabilidad** y la fuerte **dependencia del *hardware***.

Este tipo de lenguajes hacen corresponder a cada instrucción en ensamblador una instrucción en código máquina. Esta traducción la lleva a cabo un programa traductor denominado ensamblador.

Dado que el lenguaje ensamblador está fuertemente condicionado por la arquitectura del ordenador que soporta, los programadores no suelen escribir programas de tamaño considerable en ensamblador, sino que utilizan este lenguaje para afinar partes importantes de programas escritos en lenguajes de más alto nivel.

El lenguaje ensamblador sigue siendo importante, ya que ofrece al programador el **control total de la máquina** y como resultado genera un **código compacto, rápido y eficiente.**

Lenguajes de alto nivel

Tal como tuvimos ocasión de ver en el apartado de la evolución del *software*, la siguiente generación de los lenguajes de programación tenían como objetivo **facilitar el trabajo del programador** permitiéndole desarrollar

aplicaciones independientes de la máquina, lo cual dio lugar a la aparición de los lenguajes de programación de alto nivel.

Podéis consultar el apartado "Tercera generación: lenguajes de alto nivel" del núcleo de conocimiento "Evolución de los lenguajes".

Estos lenguajes más evolucionados utilizan unas **instrucciones más fáciles de entender** y proporcionan **facilidades** para **expresar alteraciones del flujo de control** de una forma más intuitiva.

Las características fundamentales de estos lenguajes son las siguientes:

- Son **independientes de la arquitectura física** de la computadora, hecho que permite utilizar los mismos programas en computadoras de arquitecturas diferentes sin necesidad de conocer el *hardware* específico de la máquina.
- La ejecución de un programa en lenguaje de alto nivel requiere de una **traducción** del mismo al lenguaje máquina de la computadora en la que va a ser ejecutado. Una **sentencia en un lenguaje de alto nivel** da lugar, al ser traducida, a **varias instrucciones en lenguaje máquina**.
- Utilizan **notaciones cercanas** a las usadas por las **personas** en un determinado ámbito. Se pretende la máxima aproximación posible al lenguaje natural o al lenguaje algebraico. Las instrucciones están expresadas con texto, ya que es posible introducir comentarios en las líneas de las instrucciones y la escritura de programas por lo general está basada en reglas parecidas a las humanas.
- Se suelen incluir **potentes instrucciones** de uso frecuente que ofrece el lenguaje de programación. Por ejemplo, se suelen ofrecer funciones matemáticas de uso común (seno, coseno, conversión de entero a real, etc.), operadores específicos de entrada/salida, operadores de tratamiento de cadenas de caracteres, etc.

En la actualidad hay una gran cantidad de lenguajes de alto nivel en uso y se han desarrollado diferentes versiones de algunos de ellos. Esta heterogeneidad constituye el principal problema que presentan estos lenguajes.

Los lenguajes de alto nivel, a diferencia de los lenguajes máquina y ensamblador, no permiten aprovechar completamente los recursos internos de la máquina.

Todas estas características ponen de manifiesto un acercamiento a las personas y un alejamiento de la máquina. Por esta razón, los programas escritos en lenguaje de alto nivel no pueden ser interpretados directamente por la computadora, por lo que es necesario realizar previamente su traducción a lenguaje máquina.

Para ello, se deben utilizar unos programas **traductores** (desarrollados con antelación para cada computadora) que se encarguen de realizar dicho proceso de traducción.

Leed los siguientes textos de Miquel Barceló: "Lenguajes de programación" y "Algunos lenguajes de programación".

Lenguajes de programación

En realidad los primeros ordenadores eran máquinas "desnudas", es decir, sin ningún *software*. La programación se hacía directamente en binario. Para resolver esta incomodidad, se introdujo pronto la posibilidad de utilizar nombres simbólicos en los programas, que pasaron a escribirse en un lenguaje muy próximo al de la máquina, pero que ya permitía escribir con símbolos decimales y nombres simbólicos (palabras) para identificar las posiciones de memoria. Nacían así los lenguajes de programación.

Así pues, un lenguaje de programación es cualquier notación para la descripción de algoritmos y estructuras de datos, aunque suele exigirse que esté implementado en un ordenador.

El lenguaje más simple es el denominado *lenguaje máquina*, que utiliza como representación de datos e instrucciones los elementos de representación del sistema binario, y utiliza como única estructura de datos la palabra de memoria. En este caso, las únicas operaciones que pueden aparecer en los algoritmos son las

instrucciones que entiende la unidad central de proceso.

Ya se ha señalado lo engorroso que sería programar en tales condiciones o, lo que es lo mismo, con este lenguaje de programación tan limitado. Por ello se han diseñado nuevos lenguajes, más próximos al programador pero que exigen la presencia de un programa traductor (procesador de lenguaje) para convertir a lenguaje máquina el programa escrito en el lenguaje simbólico de alto nivel (que suele denominarse *programa fuente*).

Procesadores de lenguaje

Con ello surgieron los programas procesadores de lenguaje, que traducían directamente del lenguaje simbólico utilizado por el programador al lenguaje binario de la unidad central de proceso. En realidad, un procesador de lenguaje es un "metaprograma" que actúa sobre datos que, a su vez, son programas escritos en un lenguaje simbólico de alto nivel.

Si nos basamos en la manera como convierten un lenguaje de programación en lenguaje máquina, podemos establecer dos grandes familias de procesadores de lenguaje: los intérpretes y los traductores.

Intérpretes

Un programa intérprete es un procesador de lenguaje que convierte cada instrucción o expresión del programa redactado en lenguaje simbólico en la instrucción o instrucciones correspondientes del lenguaje máquina, y las ejecuta inmediatamente antes de proceder a la traducción de la instrucción siguiente del lenguaje simbólico fuente.

De hecho, un intérprete es realmente un simulador *software* que, para el programador, simula la existencia de una máquina con una unidad central de proceso capaz de entender el lenguaje simbólico de programación utilizado.

Traductores

Un programa traductor es un procesador de lenguaje que traduce a lenguaje máquina todas las expresiones o instrucciones de que consta el programa fuente, con lo cual genera un nuevo programa escrito en lenguaje máquina o, a veces, en un lenguaje intermedio, que recibe el nombre de *programa objeto*.

Podemos distinguir varios tipos de traductores de lenguajes:

1. **Ensambladores**

   Los ensambladores (*assembler*) son los que actúan sobre lenguajes de programación muy cercanos al lenguaje máquina. La traducción se hace símbolo a símbolo, efectuando una correspondencia directa entre las instrucciones del lenguaje ensamblador y las instrucciones del lenguaje máquina.

   En todo caso, se trata de lenguajes muy directamente ligados a cada máquina y por ello se dice que son de muy bajo nivel, porque su uso es muy poco generalizable.

2. **Compiladores**

   Los compiladores son los auténticos traductores, alejados ya de la servidumbre de la máquina. Los lenguajes que se compilan pueden tener sentencias, frases y expresiones complejas totalmente al margen de la concreción de un lenguaje máquina y un procesador en particular.

   De ahí su generalidad, ya que, para ejecutarlos en una máquina u otra, basta con disponer del programa traductor (el compilador), que traduce el lenguaje simbólico, bien a lenguaje máquina directamente o bien al ensamblador concreto de la máquina en cuestión. Suelen corresponder a los lenguajes llamados *de alto nivel*, por el mayor grado de simbolismo posible.

3. **Generadores**

   Otro nivel diferente lo presentan los procesadores de lenguaje de tipo traductor, que están orientados a generar un nuevo programa a partir de unas especificaciones en un lenguaje de especificación de problemas. Podríamos decir que cuando se utiliza un generador de programas lo que se hace es acogerse a unos módulos algorítmicos preexistentes, que el propio generador decidirá si tiene que utilizar o no, de acuerdo con unas especificaciones de lo que se supone que debe hacer el programa, de cómo son los datos de entrada y de cómo tienen que ser los datos de salida.

Características de los lenguajes de programación

Sintaxis y semántica

La sintaxis de un lenguaje de programación es la forma en que se escribe un programa utilizando este lenguaje. La

constituyen las reglas que indican cómo se escriben las frases (sentencias), las declaraciones de las estructuras de datos y el resto de las construcciones sintácticas del lenguaje.

La semántica de un lenguaje es el significado que se da a las diferentes construcciones sintácticas. Así, una misma estructura de datos simple, el número entero por ejemplo, tiene distintas representaciones sintácticas en diferentes lenguajes aunque su significado (su semántica) pueda ser el mismo: un número entero. Entre los elementos sintácticos fundamentales de un lenguaje de programación encontramos:

a. el conjunto de caracteres y separadores que utiliza para escribir las diferentes palabras o identificadores, con lo cual se configura el alfabeto del lenguaje;

b. los distintos símbolos de operaciones con que se enuncian las operaciones aritméticas y lógicas, así como operaciones especiales que dependen de las estructuras de datos utilizados;

c. las palabras reservadas en el lenguaje, y que el programador no tiene que utilizar como identificadores, ya que tienen un significado preciso en la semántica del lenguaje (nombre de las operaciones y operadores, nombre de las estructuras básicas para formar algoritmos, etc.);

d. la manera como puede potenciarse la legibilidad de un programa ampliando el texto con comentarios escritos en lenguaje natural humano;

e. el tipo de formato de escritura, que puede ser libre o fijo (en tal caso ciertos elementos tienen que escribirse obligatoriamente en determinadas posiciones concretas de la línea de escritura).

Un buen lenguaje debe a su sintaxis el grado de legibilidad y la facilidad para ser utilizado en la escritura de programas, así como la sencillez o la complejidad del compilador que finalmente tiene que traducirlo a lenguaje máquina. El objetivo principal es conseguir un lenguaje que no sea ambiguo; esta necesidad de eliminar la ambigüedad es la razón fundamental por la cual el lenguaje que utilizamos los seres humanos no es adecuado para programar un ordenador.

Universalidad de los lenguajes

La característica de universalidad de un lenguaje de programación merece una mención aparte.

Se dice que un lenguaje es universal cuando su estructura sintáctica y semántica permite expresar cualquier programa posible.

En realidad, los lenguajes se consideran más o menos universales; la adecuación específica de un lenguaje a un tipo concreto de problemas (gestión, cálculo científico, etc.) va en detrimento de su universalidad, entendiéndola como una adecuación cualitativa.

En todo caso, la presencia de varios lenguajes con vocación de universalidad plantea el problema de si un programa escrito en un lenguaje de programación A puede también ser escrito en un lenguaje de programación B, y si estos programas son totalmente equivalentes.

Dos programas serán equivalentes si, respondiendo al mismo conjunto de datos de entrada, proporcionan el mismo conjunto de datos de salida; es decir, si implementan de manera totalmente igual la función algorítmica del programa.

Este problema se estudia con la ayuda de máquinas abstractas o autómatas como la conocida máquina de Turing, descrita por Alan Turing en 1936.

El efecto práctico del estudio de estos lenguajes universales simples y de las máquinas abstractas lleva a la conclusión de que cualquier lenguaje de programación que pueda ser razonablemente utilizado en la práctica es, sin ninguna duda, un lenguaje universal, siempre que se considere que no hay límites en la capacidad de almacenaje ni en el tiempo de ejecución.

De hecho, las diferencias entre los lenguajes de programación no son diferencias cuantitativas sino, esencialmente, diferencias de tipo cualitativo, que indican el grado de elegancia, facilidad y efectividad con que pueden utilizarse para abordar determinados tipos de problemas.

Por esta razón, se habla de lenguajes para programación científica (como el Fortran) y de otros claramente orientados a la gestión (como el Cobol) y a otras aplicaciones, aunque esta caracterización es solamente cualitativa.

En realidad, la visión actual de la programación está orientada según la idea de la universalidad de los diferentes lenguajes. Una vez conocidas de manera general las estructuras de datos y las estructuras algorítmicas posibles, la programación de un problema tiene que realizarse siempre siguiendo el proceso de diseño descendente por refinamientos sucesivos, típico de la programación estructurada. Esto hace que se supere la reducida visión de antes, en que se "programaba" en un lenguaje determinado, para pasar a la visión actual según la cual se diseña un

programa que posteriormente puede codificarse en un lenguaje u otro. Las facilidades que dé un determinado lenguaje en la etapa de codificación son simplemente de tipo cualitativo.

La máquina de Turing

La máquina de Turing dispone de una sola estructura de datos: un vector lineal denominado *cinta* (*tape*), con un único carácter en cada uno de sus elementos. Hay una sola variable, que es el *cabezal de lectura* (*read head*), que apunta a un elemento del vector cinta.

El programa que controla la máquina de Turing puede realizar muy pocas operaciones:

- se pueden leer o modificar los elementos de la cinta;
- el programa puede proceder a hacer bifurcaciones de acuerdo con el carácter leído en la cinta, así como hacer bifurcaciones de tipo incondicional (tipo GO TO);
- la posición del cabezal de lectura puede modificarse de forma que apunte al componente de la cinta inmediatamente a la derecha o a la izquierda de la posición en un momento dado;
- la capacidad de almacenaje de la cinta es ilimitada.

A pesar de esta simplicidad, la máquina de Turing resulta muy útil, aunque entre sus operaciones primitivas no dispone de la posibilidad de efectuar operaciones aritméticas. Puede demostrarse que cualquier algoritmo puede expresarse como un programa para la máquina de Turing y por ello puede afirmarse que el lenguaje con que se programa la máquina de Turing es un lenguaje universal. Se trataría del lenguaje universal más simple.

Algunos lenguajes de programación

El número de lenguajes de programación se cuenta ya por centenares o, quizá, por millares.

Junto a los lenguajes de tipo algorítmico tradicionales, se han desarrollado también lenguajes orientados a la consulta de bases de datos. Se trata de los lenguajes de interrogación (*query*), a veces llamados *lenguajes de cuarta generación* (4 GL). En el fondo son lenguajes de especificación de problemas a partir de los cuales se genera automáticamente el programa que resuelve la consulta deseada.

Otra familia la constituyen los lenguajes que no son de tipo algorítmico, como los utilizados en el campo de la inteligencia artificial (el Prolog, por ejemplo).

A continuación comentamos algunas características de los lenguajes algorítmicos más utilizados.

Fortran

El lenguaje Fortran (*FORmula TRANslation* o traducción de fórmulas) se utiliza muy a menudo en aplicaciones de cálculo científico e ingeniería.

A causa de su origen, no es un lenguaje estructurado, pero se han hecho extensiones que introducen las nociones de la programación estructurada, como pasa con la versión denominada *Ratfor*. Para escribir las expresiones, el lenguaje Fortran utiliza una notación muy próxima a la que se usa en el campo de las matemáticas. Trata las expresiones aritméticas con órdenes que las evalúan y asignan el resultado como nuevos valores de las variables. Incluye tipos de datos elementales, matrices, subprogramas y elementos de control de secuencia elementales, así como una estructura repetitiva.

Algol

Si bien ha caído en desuso, el Algol (*ALGOrithmic Language* o *ALGebraic Oriented Language*) es un lenguaje algorítmico u orientado al álgebra, importante por haber introducido por vez primera conceptos que se han revelado básicos en la programación, principalmente la idea de estructuración.

Se llegó a utilizar como un lenguaje internacionalmente adoptado para la descripción de algoritmos matemáticos.

Cobol

El Cobol (*COmmon Bussines Oriented Language* o lenguaje orientado a los negocios) se utiliza ampliamente en el campo de las aplicaciones de gestión.

Su sintaxis permite que los programas puedan leerlos fácilmente personas no formadas técnicamente y con pocos

conocimientos de inglés, idioma en el que basa su vocabulario. Es de destacar el aspecto de gestión de estructuras de datos externos como son los ficheros, tan necesarias en las aplicaciones de gestión para las cuales está diseñado.

La entidad que regula su evolución y establece su estándar (CODASYL) lo ha estructurado en forma modular, con un núcleo y un conjunto de varios módulos, cada uno de ellos con tres niveles de implementación. Así, puede implementarse en una gran variedad de máquinas; de ahí, el elevado grado de portabilidad de las aplicaciones escritas en este lenguaje.

PL/1

El PL/1 o "lenguaje de programación 1" (*Program Language I*) es un lenguaje que quiere ser útil tanto en los problemas de tipo científico como en los de gestión.

Su diseño se basa mucho en otros lenguajes como el Fortran, el Cobol y el Algol. Del Algol adopta la estructura de bloques y las instrucciones de control de secuencia estructuradas; del Cobol, la riqueza en la gestión de ficheros y la declaración de datos tipo PICTURE, y del Fortran, la forma concisa y simple de las instrucciones y el mecanismo de transmisión de parámetros, entre otros elementos.

Basic

El Basic (*Beginners All-Purpose Symbolic Instruction Code* o código de instrucciones simbólicas para principiantes y toda clase de propósitos) pretende ser un código de instrucciones simbólicas orientadas a cualquier tipo de problema y muy fácil de aprender.

La fama de este lenguaje procede de haber sido el procesador de lenguaje de tipo intérprete que se ha implementado en la mayoría de los microordenadores.

Aunque es más adecuado para el cálculo científico, con una forma parecida a la del Fortran, en los últimos años se ha utilizado para desarrollar aplicaciones de gestión en microordenadores. La ausencia de un estándar complica la portabilidad de programas entre una máquina y otra. El hecho de que sea tan fácil de aprender y la ausencia total de mecanismos de estructuración han llevado, en los últimos años, a la aparición de un gran número de programadores especialistas en Basic, que parecen irremediablemente perdidos para el proceso de racionalización de la actividad de programar que se inicia en la programación estructurada.

Lisp

Diseñado inicialmente por John McCarthy en el MIT hacia 1960, el Lisp (*LISt Processing*) es un lenguaje creado especialmente para procesar listas. Una lista en Lisp es una serie de elementos separados por un espacio en blanco e incluidos entre paréntesis. Un elemento puede ser simple (átomo) o puede ser una lista. Una subrutina se considera una herramienta que define una función en sentido análogo al matemático, estableciendo una correspondencia entre un conjunto de valores a la entrada y un conjunto de valores a la salida. Hechos destacables son la equivalencia formal entre programas y datos, todos ellos en forma de listas, lo cual permite que una estructura de datos se ejecute como un programa, o que un programa se modifique a sí mismo considerándose como datos. Otro elemento nuevo en su diseño es el uso tan extendido de la recursividad. Con el Lisp aparece el concepto de "recogida de basuras" (*garbage collection*) para el aprovechamiento de la zona de almacenaje. Se utiliza en el área de la inteligencia artificial: robótica, tratamiento del lenguaje natural, prueba de teoremas, sistemas inteligentes, etc. También es importante por haber dado origen al Logo, que se emplea a menudo en el campo de la informática educativa.

RPG

El RPG (*Report Program Generator* o programa generador de listados) es en realidad un generador de programas que lee unas especificaciones y adapta los módulos de un ciclo fijo de programa para la realización de las entradas y salidas indicadas en las instrucciones. Apareció con los pequeños ordenadores para gestión, como el Sistema/3 de IBM, a finales de los sesenta. Modificaciones posteriores (las tarjetas de instrucciones tipo C y las versiones RPG III y RPG 400) lo configuran con la posibilidad de desarrollar algoritmos específicos, mediante el uso de variables binarias conocidas como interruptores, y también de gestionar bases de datos.

Pascal

El lenguaje Pascal fue diseñado por el profesor Niklaus Wirth y publicado en 1971, aunque su primera implementación es de 1973. Su origen era servir como lenguaje base para la enseñanza de la programación desde un punto de vista sistemático, introduciendo en el mismo lenguaje los conceptos de la programación estructurada. Su nombre es un homenaje a Blaise Pascal. Se utiliza mucho en la enseñanza, en la construcción de *software* y en muchas otras áreas de aplicación.

Se trata de un lenguaje que presenta estructura de bloques e implementa la mayoría de los conceptos de la programación estructurada, tal como se han descrito en este texto. Incluso la notación utilizada anteriormente para

la descripción de las estructuras de control de secuencia se basa claramente en el Pascal. Este lenguaje ha ejercido una gran influencia en la creación de nuevos lenguajes como el Ada y el Concurrent Pascal. La principal innovación, junto con la estructuración, es la introducción de los tipos de datos y, sobre todo, que el programador pueda describir nuevos tipos de datos. Los subprogramas adoptan la forma de *function*, si devuelve un único valor, o *procedure*, si modifica los parámetros recibidos o las variables globales. En realidad, una *procedure* es un bloque estructural completo, ya que el Pascal implementa este concepto tomado con toda seguridad del Algol.

Ada

A principios de los setenta, el Departamento de Defensa de Estados Unidos abordó el estudio de las especificaciones necesarias para un lenguaje de programación de uso en aplicaciones en las que uno o varios ordenadores formaran parte de un sistema más amplio, como un aeroplano, un navío o sistemas de comunicaciones. Las especificaciones (versión *Tinman*) utilizadas en 1976 y 1977 demostraron que ninguno de los lenguajes existentes era adecuado. Se estableció una competición internacional de diseño, que acabó con la selección, en 1979, del lenguaje que se llamó Ada en honor de Ada Lovelace, considerada la primera programadora de la historia.

El Ada deriva en muchas cosas del Pascal, pero tiene más posibilidades, en particular la ejecución concurrente de tareas, el control de tareas en tiempo real, la gestión de excepciones y tipos abstractos de datos.

Un programa hecho con el Ada se considera una colección de componentes, denominados *paquetes* (*packages*). Un paquete puede ser un tipo abstracto de datos o un conjunto de objetos compartidos entre programas, aunque en general un paquete contiene un conjunto integrado de definiciones de tipos, objetos de datos y los subprogramas para manipular estos objetos. El programador que trabaja con el Ada dispone de una biblioteca (*program library*) de paquetes, y su primer objetivo es construir un nuevo programa utilizando algunos de estos paquetes en vez de proceder a construirlo de manera aislada.

C

El lenguaje C está en la base del sistema operativo UNIX y lo diseñaron Dennis Ritchie y Brian W. Kernighan a partir del llamado *lenguaje B*, desarrollado por Ken Thompson en 1970.

Se trata de un lenguaje de propósito general con estructuras de datos y de control de secuencia al estilo del Pascal, y con un conjunto potente de operadores. Tiene muy pocas restricciones y por ello es laxo en la gestión de tipo de datos. Un elemento importante del lenguaje es que trabaja también a "bajo nivel", es decir, trabaja directamente con direcciones de memoria y punteros, con lo cual obtiene rendimientos de ejecución cercanos a los que se consiguen con los lenguajes de tipo ensamblador.

Se utiliza fundamentalmente para el desarrollo de *software* de base y para la mayoría de los paquetes de aplicación en el campo de los miniordenadores. Se empieza a introducir en las aplicaciones para microordenadores que hasta ahora se escribían directamente en ensamblador. La ventaja de la portabilidad añadida a la eficiencia le augura un gran futuro como lenguaje de programadores especializados en *software*.

Prolog

El Prolog se presenta como un lenguaje de los que se han venido en llamar *de quinta generación*. Se trata de un lenguaje declarativo y no algorítmico en el que se establecen reglas y hechos, de los cuales el propio procesador de lenguaje deducirá nuevos hechos a partir de la petición de que se investigue la veracidad o no de una determinada declaración.

Los programas escritos en Prolog requieren pocas líneas, ya que sólo tiene que darse la declaración de hechos y reglas que definen el problema, y el propio Prolog se encarga de organizar el proceso de tratamiento que debe seguirse para determinar la respuesta solicitada.

El Prolog se utiliza mucho en el campo de la inteligencia artificial y en la construcción de sistemas expertos. Un *sistema experto* persigue la reproducción correcta del comportamiento de un experto humano en un dominio determinado. Por ello, tiene que proceder imitando el razonamiento humano en la deducción y la inducción, y establecer las estrategias adecuadas de búsqueda de nuevas relaciones. Esto lo hace a partir de los conocimientos suministrados por un experto humano, que forman la denominada *base de conocimientos*, es decir, la memoria a largo plazo que contiene las reglas del problema. A partir de los datos iniciales, el sistema va elaborando una base de hechos, es decir, la memoria a corto plazo que contiene los hechos establecidos o que se han ido deduciendo. El proceso de averiguación de nuevos hechos para añadir a la base de hechos lo realiza el *motor de inferencia,* que es el mecanismo lógico por el cual un sistema experto "razona". Los dominios de aplicación del Prolog son los típicos de la inteligencia artificial, ya mencionados al tratar del Lisp.

Lenguajes RAD

En los últimos años han aparecido una serie de lenguajes muy fáciles de utilizar y que poco a poco unen las ventajas de la orientación a objetos a la denominada *programación visual*. Nacidos en el mundo de la

microinformática para ayudar a la realización de programas con interfaz visual bajo el paradigma WIMP (*windows*, *icons, mouse* y *pop-up menu*, es decir, ventanas, iconos, ratón y menús desplegables), constituyen la forma moderna y más evolucionada de lo que se ha denominado RAD (*Rapid Application Development*, desarrollo rápido de aplicaciones).

De entre los muchos ejemplos disponibles, puede ser procedente destacar los de Powerbuilder, Visual Basic o Delphi, posiblemente los más utilizados actualmente.

En todos los casos, con un apoyo previo de base de datos, se trata de diseñar interactivamente un formulario que será atendido por uno o más módulos. En estos formularios es posible utilizar nuevos tipos de datos típicos de los entornos visuales: botones de activación, cuadros de listas (*list box*), cuadros de diálogo (*dialog box*), cuadros de texto (*text box*), etc.

Hay dos tipos de traductores de lenguajes de alto nivel que vamos a considerar: los compiladores y los intérpretes.

Traductores. Compiladores e intérpretes

Como un ordenador puede interpretar y ejecutar sólo el código máquina, existen programas especiales, denominados **traductores**, que traducen programas escritos en un lenguaje de programación al lenguaje máquina de la computadora.

Un traductor es un metaprograma que toma como entrada un programa (o parte de un programa) escrito en lenguaje simbólico -alejado de la máquina- denominado **programa fuente** y proporciona como salida otro programa semánticamente equivalente y escrito en un lenguaje comprensible por el *hardware* de la computadora que recibe el nombre de **programa objeto**.

A continuación se detallan dos tipos de traductores: los **compiladores** y los **intérpretes**.

- Un **compilador** traduce completamente en programa fuente, con lo que genera un programa objeto (semánticamente equivalente) escrito en lenguaje máquina. El programa fuente suele estar contenido en un fichero, y el programa objeto puede almacenarse como otro fichero en memoria masiva para ser ejecutado más adelante, sin necesidad de volver a realizar la traducción. Una vez traducido un programa, su ejecución es independiente de su compilación.
- Un **intérprete** permite que un programa fuente escrito en un determinado lenguaje sea traducido y ejecutado directamente sentencia a sentencia por la computadora. El intérprete capta una sentencia fuente, la analiza y la interpreta, lo que da lugar a su ejecución inmediata. En este caso no se crea ningún fichero o programa objeto almacenable en memoria masiva para posibles ejecuciones futuras.

Los lenguajes de programación y los traductores facilitan la comunicación entre el programador y la máquina.

Introducción a la ingeniería del *software*

Los problemas de la construcción de grandes sistemas *software* son innumerables y semejantes, en muchos aspectos, a los que surgen en cualquier proyecto de ingeniería (planificación, administración, control de calidad, selección de herramientas y personal, etc.). De ahí nace la expresión **ingeniería del *software*** para englobar una serie de principios de ingeniería que posibiliten la construcción de *software* con garantías de calidad, de forma eficiente y con costes económicos.

La ingeniería del *software* abarca un conjunto de tres elementos clave:

- **Métodos**
- **Herramientas**
- **Procedimientos**

Estos elementos facilitan al gestor el control del proceso de desarrollo de *software*, así como el suministro de las bases para construir un *software* de alta calidad de una forma productiva.

Los **métodos** de la ingeniería del *software* suministran el "cómo" construir técnicamente el *software*. Éstos abarcan un amplio espectro de tareas que incluyen:

- La planificación y estimación de proyectos
- El análisis de los requerimientos
- El diseño
- La codificación
- Las pruebas
- El mantenimiento del *software* producido

Las **herramientas de la ingeniería del *software*** suministran un soporte automático o semiautomático para dichos métodos. En la actualidad hay herramientas para soportar cada uno de los métodos que hemos mencionado antes.

Los **procedimientos** de la ingeniería del *software* son la cola que pega los métodos y las herramientas facilitando, así, un desarrollo racional y oportuno del *software* de computadoras.

Dichas herramientas definen la secuencia en la que se aplican los métodos, las entregas (documentos, informes, formas, etc.) que se requieren, los controles que ayudan a asegurar la calidad y coordinar los cambios y las guías que facilitan a los gestores del *software* el establecimiento de su desarrollo.

Una metodología de diseño, desarrollo e implantación de soluciones informáticas también define un conjunto de **documentos** que hay que generar:

- Especificación del sistema
- Planificación del proyecto
- Especificación de los requerimientos del *software*
- Diseño preliminar
- Diseño detallado
- Código fuente
- Resumen de la verificación y validación
- Manual de usuario
- Legado del proyecto. Informe de errores
- Informe de mantenimiento.

Es importante tener presentes los elementos clave de la ingeniería del *software* cuando encargamos *software* a medida para nuestra organización o departamento.

Leed el texto de Miquel Barceló: "Introducción a la ingeniería del *software*".

Introducción a la ingeniería del *software*

Intentos de definición de la ingeniería del *software*

La expresión **ingeniería del *software*** se utilizó por primera vez en el año 1968, como tema de una conferencia organizada por la división de asuntos científicos de la OTAN. En dicha conferencia se llegó a la conclusión de la necesidad de una nueva disciplina científica con una gran orientación práctica que superara las limitaciones que la ciencia informática (*computer science*) había encontrado a la hora de realizar productos de *software* grandes y complejos. Ya entonces surgió la idea de aplicar técnicas típicas de la actividad de ingeniería a la construcción de *software* para incrementar, al mismo tiempo, tanto la productividad como la calidad del proceso de desarrollo de aplicaciones informáticas, así como para hacer más fácil el inevitable proceso de su mantenimiento.

Ya en 1969, en una nueva conferencia sobre el tema, **Fritz Bauer** caracterizaba la ingeniería del *software* como "el establecimiento y uso de principios de ingeniería robustos, orientados a obtener económicamente *software* que sea fiable y funcione con eficiencia sobre máquinas reales".

Con el tiempo han surgido otras definiciones de ingeniería del *software*, como la proporcionada por **H.D. Mills** en

1980, que la considera "un conjunto de disciplinas y procedimientos para la construcción y el mantenimiento de *software* viable y económico".

Es especialmente interesante la definición proporcionada en 1981 por **Barry W. Boehm**, el cual, tras analizar los conceptos de *software* e ingeniería define la ingeniería del *software* parafraseando la definición de ingeniería del diccionario Webster pero aplicándola concretamente al aprovechamiento del equipo informático y los ordenadores en concreto: "La aplicación de la ciencia y las matemáticas por la cual las capacidades de los ordenadores se hacen útiles a los humanos por medio de los programas de ordenador, los procedimientos y la documentación que los acompaña".

También encontramos un resumen adecuado del alcance de la ingeniería del *software* en la escueta y concisa definición de una famosísima asociación de profesionales: "Una perspectiva sistemática del desarrollo, la operación, el mantenimiento y la retirada del *software*" (IEEE, Institute of Electrical and Electronics Engineers, *Standard Glossary of Software Engineering Terminology,* 1983). Hay que destacar la incorporación de una fase necesaria de "retirada" o sustitución del *software*, que se añade a otras más fácilmente reconocidas: desarrollo, utilización y mantenimiento.

En todos los casos, la ingeniería del *software* se presenta como un enfoque sistemático y profesional que quiere producir *software* de calidad con el coste económico más bajo posible y cumpliendo los plazos de tiempo establecido.

La calidad de este *software* tiene que entenderse en el sentido de que no sólo sea útil a los usuarios, eficiente en el uso del *hardware* y no incorpore errores, sino que también permita un fácil mantenimiento.

Especificidad de la ingeniería del *software*

Evidentemente la creciente importancia de los costes del *software* informático y, principalmente, de su mantenimiento justifican la necesidad de un enfoque sistemático y profesional en la construcción de aplicaciones y sistemas informáticos. Pero, además, hay razones específicas que surgen de la particularidad del proceso de desarrollo y mantenimiento del *software*.

De hecho, los proyectos informáticos de construcción de *software* presentan unas características diferenciales que impiden gestionarlos con los mismos métodos y técnicas con que se gestionan otros proyectos de ingeniería.

La diferencia más significativa reside en el hecho de que, en los proyectos informáticos, **el diseño y la construcción del sistema son un solo proceso** y no hay una fase de diseño previa a la construcción o fabricación.

En realidad, la construcción de *software* es un mero proceso de diseño, pero el que este diseño adquiera finalmente la forma de una especificación final en un lenguaje ejecutable por ordenador asimila dicho diseño en un proceso que incluye también la fabricación, de modo que la comprobación de la calidad del *software* se hace francamente compleja y difícil.

Objetivos de la ingeniería del *software*

Hay dos objetivos centrales en la ingeniería del *software*: por una parte, obtener un producto de *software* de **calidad** (económico, fiable, eficiente y de uso y mantenimiento fáciles), y, por otra parte, construir ese *software* mediante un **proceso correcto y adecuado** que signifique una gestión eficaz de los recursos puestos a disposición del ingeniero de *software* para la construcción de un nuevo sistema informático: personas y recursos informáticos.

En definitiva, la ingeniería del *software* debe atender a la elaboración del *software* y a la gestión del proyecto que lleva a dicha elaboración.

Componentes de la ingeniería del *software*

Para conseguir los dos objetivos centrales de la ingeniería del *software*, Boehm analiza la existencia de tres componentes fundamentales: las **relaciones humanas** (*human relations*), la **ingeniería de recursos** (*ressource engineering*) y la **ingeniería de programas** (*program engineering*).

En lo que respecta al **producto**, las relaciones humanas se concretan en la necesidad de que el *software* producido sea fácil de utilizar y satisfaga necesidades reales; la ingeniería de recursos exige que el producto sea ajustable y adaptable y que su eficiencia proceda de un equilibrio adecuado entre los costes de producirlo y los beneficios obtenidos de utilizarlo; la ingeniería de programas se concreta en el hecho de que el *software* esté especificado con precisión de una manera completa, consistente, verificable y realizable, además del hecho de que sea correcto, así como intrínsecamente adaptable o modificable, por ejemplo, porque utiliza técnicas como la estructuración y porque garantiza la legibilidad y fácil comprensión de diseños y resultados.

En cuanto al **proceso**, el aspecto de las relaciones humanas representa planificación, organización, dirección y

control del proceso de construcción del *software* y su posible automatización; la ingeniería de recursos presupone la adecuada estimación previa a la planificación y el correcto análisis de costes y beneficios, así como el control de los hitos o puntos cruciales del proyecto y el cumplimiento de plazos y presupuestos; por último, la ingeniería de programas exige la validación de la factibilidad del proyecto y de sus requerimientos y especificaciones, además de la verificación y validación del diseño, la programación y la misma integración del sistema y de su implementación y los procedimientos establecidos para su mantenimiento.

Para un especialista como Mills, todo esto se traduce en la incorporación y el dominio de técnicas procedentes de tres entornos conceptuales complementarios: el diseño, el desarrollo y la gestión.

En el **diseño** (*software engineering design*) tiene que atenderse a los temas más teóricos propios de la ciencia informática, como son los esquemas básicos de composición de algoritmos, la estructuración de datos y programas o el control de procesos múltiples y asíncronos en los sistemas distribuidos y en tiempo real.

En el **desarrollo** (*software engineering development*) se agrupan los procesos prácticos del desarrollo de *software* desde el punto de vista de las herramientas y técnicas puestas a disposición de las personas responsables de construir el producto final: entornos de desarrollo (lenguajes, ayudas, herramientas, etc.), procedimientos ordenados de desarrollo progresivo (*top down*) y técnicas formales de documentación de productos de *software*.

Por último, en la **gestión** (*software engineering management*) se agrupan todas las prácticas necesarias para valorar, planificar y controlar el trabajo de construir *software* y, en definitiva, gestionar el proyecto informático consistente en la elaboración de una determinada aplicación informática.

Principales etapas y actividades en el desarrollo del *software*

De una manera genérica, todo proceso de construcción de *software* pasa por etapas sucesivas cuyo nombre, contenido y detalle ha ido variando a lo largo del tiempo según los diferentes métodos utilizados. Si se une el inevitable mantenimiento y la retirada del *software* al desarrollo, se habla con más frecuencia de ciclo de vida que de "etapas", para marcar este carácter evolutivo y perecedero de cualquier aplicación informática.

Un esquema sencillo y clásico de este ciclo de vida es el que forman:

a. el **análisis** de los requerimientos, funciones y objetivos del sistema y la elaboración de las especificaciones correspondientes;

b. el **diseño** de una solución, lógica y técnica, que satisfaga las especificaciones;

c. la **implementación** final del sistema, concretada en forma de procedimientos y, sobre todo, en la codificación de programas escritos ya en lenguaje fuente;

d. la **prueba** del sistema;

e. la **instalación**, la operación y el **mantenimiento** del sistema hasta que se decida retirarlo.

Distribución de costes en el desarrollo del *software*

Varios autores han analizado la distribución del coste global del desarrollo de un proyecto informático y, aunque las proporciones exactas pueden depender de cada sistema concreto y del método utilizado para construirlo, se aceptan como referencia general las cifras establecidas a finales de los años setenta por autores como Boehm y Brooks: un 40% del coste de desarrollar una aplicación informática se emplea en las etapas de análisis y diseño; sólo un 20%, en la etapa de codificación, y el resto, un 40%, queda para la parte correspondiente a las pruebas.

Desglose del porcentaje de esfuerzo en las etapas del ciclo de vida:

- análisis y diseño 40%
- codificación 20%
- pruebas 40%

Por supuesto, el mantenimiento, uno de los costes más elevados del *software*, se ha dejado al margen de esta distribución, que se ocupa sólo de los costes del desarrollo hasta la puesta en marcha inicial, aunque ya se ha indicado que el mantenimiento suele representar el doble del coste del desarrollo, si bien se extiende a lo largo de un periodo de tiempo mucho más dilatado.

La figura 2 muestra un diagrama de la evolución del uso de recursos de personal a lo largo de un proyecto informático, con indicación de dos grandes etapas: desarrollo y mantenimiento, de momento separados de la puesta en marcha inicial. El diagrama es esquemático y, en la etapa de mantenimiento, es posible añadir varios picos

puntuales causados por cambios de envergadura que una aplicación informática puede experimentar más de una vez en su vida.

Origen e importancia de los errores en el *software*

Estudios como los de T.C. Jones a finales de los años setenta han establecido que sólo un 30% de los errores presentes en el *software* proceden de la etapa de codificación. Se considera que un 15% de los errores procede de la fase de análisis y diseño funcional; un 20% más, de la fase de diseño lógico, y hasta un 35% son errores que tienen el origen en insuficiencias de la documentación, incomprensiones y malentendidos y otras causas a lo largo de todo el ciclo de desarrollo.

Origen de los errores de *software*:

15% de la fase de análisis

20% de la fase de diseño

30% de la fase de codificación

35% por documentación insuficiente, incomprensiones y malentendidos

Boehm ha mostrado que la importancia de los errores y el coste asociado a corregirlos difiere según la etapa en que se descubren. Tomando como base el coste de corregir un error de análisis descubierto en la etapa de análisis de requerimientos y elaboración de especificaciones, la corrección de un error detectado en la fase de diseño cuesta 7 veces más. El coste de corrección es 10 veces más alto si se detecta en la etapa de codificación, entre 20 y 50 veces más si se descubre en la fase de pruebas y hasta 100 veces más si se detecta ya en la fase final de operación y mantenimiento.

Por este motivo existe la necesidad de establecer métodos y procesos de desarrollo de *software* que permitan detectar los inevitables errores cuanto antes mejor. Encontramos un ejemplo claro de esto en la reciente tendencia a experimentar con prototipos provisionales para detectar, sobre todo, los errores creados en la fase de análisis de requerimientos y elaboración de especificaciones que no suelen detectarse hasta las pruebas finales de integración o aceptación del sistema.

Herramientas CASE

La ingeniería del *software* asistida por computadora, conocida normalmente como **CASE** (*Computer Aided Software Engineering*), es un conjunto de herramientas que permiten la **automatización del proceso de producción de desarrollo de *software***.

El objetivo principal de una aplicación CASE es proporcionar un conjunto de herramientas bien integradas que ahorren trabajo, enlazando y automatizando todas las fases del ciclo de ciclo del *software*.

Leed el texto de Miquel Barceló, "Herramientas CASE".

CASE: ingeniería del *software* asistida por ordenador

Herramientas CASE

CASE (*Computer Aided Software Engineering*), es decir, la ingeniería de *software* asistida por ordenador, incluye el conjunto de herramientas y métodos asociados que pueden servir de ayuda en el proceso de construcción del *software* a lo largo de su ciclo de vida.

El término CASE, popularizado en la década de los ochenta, designa una tecnología que no es nueva –las **ayudas automatizadas** de las metodologías de desarrollo de *software*– y representa la evolución de las primeras ayudas (editores, compiladores, etc.) que formaron los entornos de desarrollo de *software* de los años setenta.

La novedad del CASE se hizo evidente cuando se incorporaron a las herramientas de ayuda al diseño informático nuevas funcionalidades gráficas con las cuales se hizo posible representar, modificar y mantener actualizados los **diagramas gráficos** que se han convertido en clásicos y se utilizan en la mayor parte de las metodologías de desarrollo de *software*.

La primera herramienta del nuevo CASE apareció cuando la empresa McDonnell Douglas utilizó su experiencia en

el **CAD** (*Computer Aided Design*, diseño ayudado por ordenador, generalmente aplicado a la ingeniería) para obtener el primer producto de ayuda al diseño informático en el aspecto gráfico, el STRADIS/DRAW.

Aun siendo importante la posibilidad de crear, modificar y mantener una documentación gráfica adecuada, las verdaderas herramientas CASE surgieron cuando los diagramas se incorporaron a una base de datos global del diseño (diccionario de datos o **repositorio**), que también mantenía detalles de los elementos de datos y de la lógica de los procesos. Este diccionario de datos es, en realidad, una base de datos que incorpora un modelo en el que se incluyen restricciones de integridad que garantizan un análisis correcto de la compleción y consistencia del diseño.

Este conjunto de funcionalidades se presenta, por ejemplo, en sistemas como EXCELERATOR, System Architect, EasyCASE, IEF (*Information Engineering Facility*), IEW (*Information Engineering Workbench*), Teamwork, VAW (*Visible Analyst Workbench*) y muchos más que han alcanzado una difusión aún más amplia al poder ejecutarse desde estaciones de trabajo y, también, en ordenadores personales de coste bajo, gracias al reciente aumento de potencialidades gráficas y de proceso de la microinformática.

Variedades del CASE

En la literatura especializada hay varias subdivisiones de CASE no siempre coherentes y que posiblemente proceden de una voluntad comercial más que de la realidad tecnológica. En todo caso, se habla de CASE alto (*upper CASE*), medio (*middle CASE*) y bajo (*lower CASE*) según el nivel de abstracción y generalidad, aunque a menudo la denominación exprese simplemente si la herramienta CASE cubre aspectos de la construcción del *software* presentes en las primeras etapas del ciclo de vida (alto) o las últimas (bajo).

Otra distinción posible se establece entre herramientas (***toolkits***), supuestamente aplicables a una sola tarea del desarrollo del *software*, y talleres de trabajo (***workbench***) que se presentan como una colección de herramientas integradas que ayudan a todo el proceso de desarrollo de *software*: análisis, diseño e implementación.

Generadores de lenguaje

La mayoría de las herramientas CASE se aplican adecuadamente a las fases de análisis y diseño, pero algunos de los sistemas CASE más completos incorporan generadores de lenguajes e incluyen la posibilidad de **generar el código fuente de la aplicación de una manera automática**.

Con ello, además de ser una ayuda valiosísima al proceso de construcción del *software* y permitir la elaboración y el mantenimiento de la documentación, consiguen la eliminación de buena parte del trabajo de programar y producen incluso el *software* ejecutable.

A veces, la herramienta CASE se reduce a ser un generador que utiliza las especificaciones procedentes de otra herramienta CASE, como puede pasar con la familia de generadores del *APS Development Center*, que puede generar el Cobol a partir, por ejemplo, de las especificaciones que le proporciona EXCELERATOR. Otros sistemas, como IEW, incluyen su propio generador.

Técnicas más frecuentes

En lo que respecta al análisis y diseño de sistemas, predominan las herramientas que implementan los **diagramas de flujo de datos** (DFD*, Data Flow Diagrams*) del análisis y diseño estructurado de Constantine, Yourdon y De Marco y también en la variante de Gane/Searson.

En cuanto al diseño de datos, se utiliza a menudo el **diagrama de entidad-relación de Chen** (ERS*, Entity-Relationship Diagrams*) o la versión más clásica de los diagramas de Bachman para convertirlos finalmente a su equivalente en el modelo de bases de datos relacionales. También, para nuevas metodologías como MERISE, se implementan los **diagramas de la historia de vida de una entidad** (ELH*, Entity-Life-History Diagrams*).

Otras técnicas de diseño muy populares y utilizadas por las herramientas CASE son los diagramas de descomposición y jerarquía de funciones (HD*, hierarchical diagram*); los diagramas de estructura de módulos de la programación modular y el diseño estructurado (SC, *Structured Chart*); los diagramas de estructura de datos de la metodología Jackson (DSD, *Data Structure Diagrams*), o los grafos de diseño de diálogos (IDS, *Invocation Dialog Structure*) y los **diseños de pantallas y listados**, entre otros modelados gráficos muy empleados en la ingeniería del *software*.

Herramientas IPSE

La ingeniería del *software* tiene por objetivo central la obtención de un producto (*software*) de calidad por medio de un proceso correcto. Generalmente se había reservado el término CASE para las herramientas que ayudan a establecer el producto, mientras que las herramientas automatizadas que también intervienen en el proceso, es decir en la gestión del proyecto informático de construcción del *software*, recibían el nombre de IPSE (*Integrated Project Support Environment)* o **entornos integrados de soporte de proyectos**. Pero en los últimos años, el término CASE se va convirtiendo en omnicomprensivo y puede decirse que en muchos casos se ha convertido ya

en sinónimo de IPSE, al cual, en cierta manera, incluye.

Las herramientas IPSE incluyen, además de las funcionalidades típicas del CASE para el análisis y el diseño, métodos de estimación de la carga de trabajos, de planificación de las tareas de un proyecto y de control de la ejecución y el desarrollo de este proyecto. También suelen incluir subsistemas de control de las versiones sucesivas de la documentación. El ejemplo más completo es el sistema MAESTRO, aunque en otros casos las herramientas CASE pueden presentar una interfaz directa con otros productos de *software* ya conocidos en la gestión de proyectos de ingeniería, como pasa con DESIGN/1, que utiliza el sistema de gestión de proyectos METHOD/1.

Metodologías CASE

Cuando las herramientas CASE soportan íntegramente el procedimiento, la documentación y la organización de un método concreto para el desarrollo de *software*, suele hablarse de **metodologías CASE**. Aunque la metodología sea siempre anterior en el tiempo, suele ser su ayuda informatizada CASE la que la hace popular y favorece su divulgación y, a menudo, constituye la razón más sólida para su implantación. Las metodologías más utilizadas en los sistemas CASE son las que se basan en los diagramas de flujo de datos para describir análisis y diseño y en los diagramas entidad-relación para describir los datos, como pasa con las de Constatine/Yourdon, De Marco, Gane/Searson o Yourdon (consultad la bibliografía) y sus extensiones en las aplicaciones en tiempo real (Ward/Mellor).

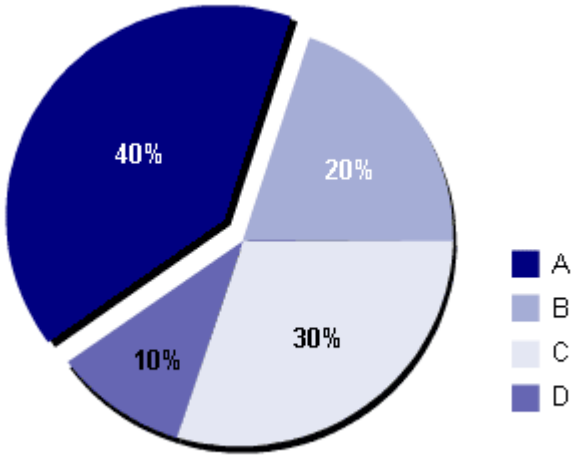
De hecho, es un error considerar el CASE como una metodología. En realidad, se trata sólo de las **ayudas mecanizadas a una metodología**, y, por tanto, como se ha dicho, es una herramienta y ya está. Pero el hecho de que las metodologías de construcción de *software* no estén muy difundidas en la práctica profesional ha llevado a los vendedores de herramientas CASE a poner énfasis también en la metodología implícita que comportan.

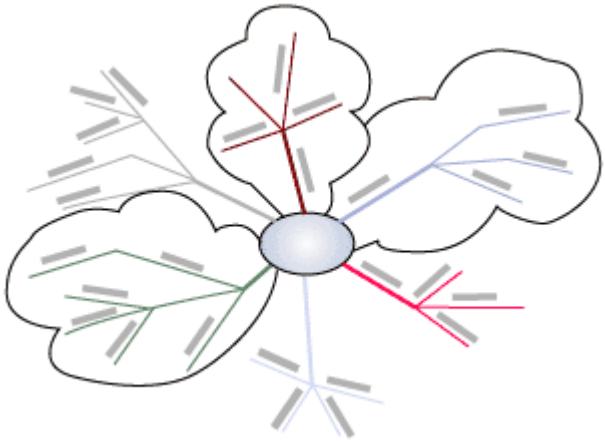
Existe una gran variedad de herramientas CASE en el mercado, cada una con una orientación diferente y aplicable a una o más fases del ciclo de vida del desarrollo de *software*. Algunas sólo apoyan al ingeniero o analista en el diseño de diagramas para modelizar procesos o flujos de datos; otras ofrecen prestaciones incluso para generar código o bases de datos.

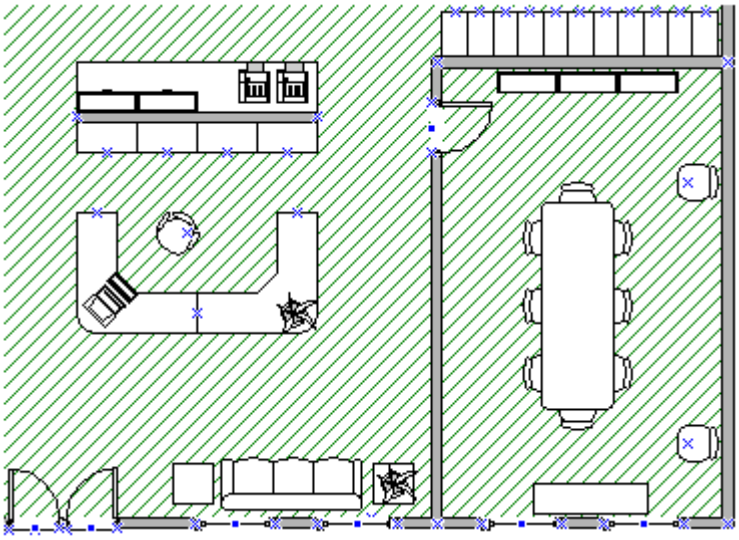
La mayoría de los diagramas y documentos que generan estas herramientas se utilizan como documentación en el diseño y desarrollo de las soluciones informáticas. Algunos de los modelos que permiten generar dichas herramientas son los siguientes:

Podéis consultar el núcleo de conocimiento "Introducción a la ingeniería del *software*" de este mismo apartado.

- **Diagramas genéricos**. Este tipo de diagramas se utiliza para crear diseños y esquemas para *brainstormings*, planificación, comunicación, generar la documentación de apoyo a un proyecto informático, etc.

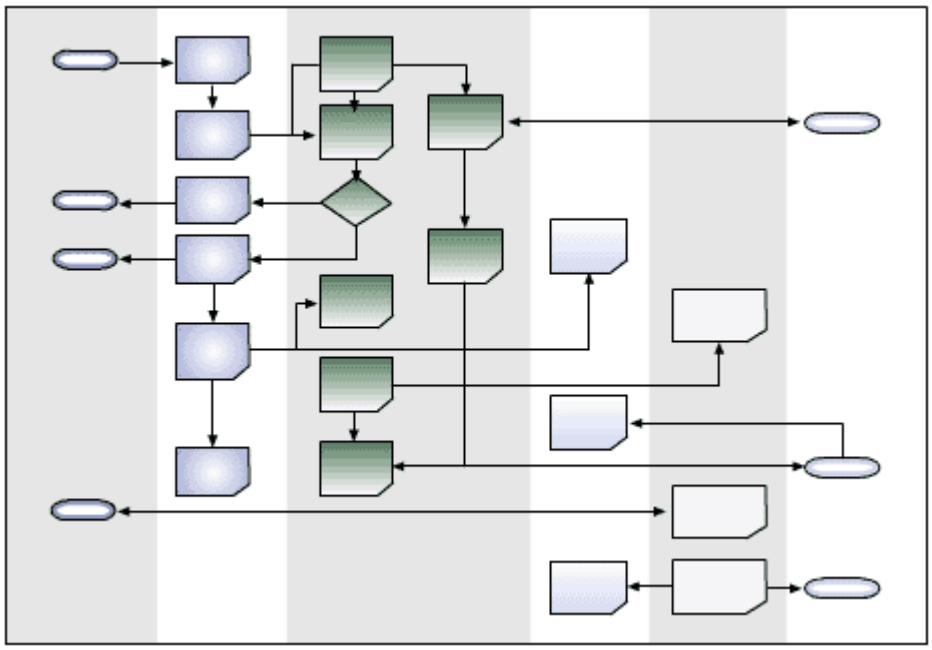






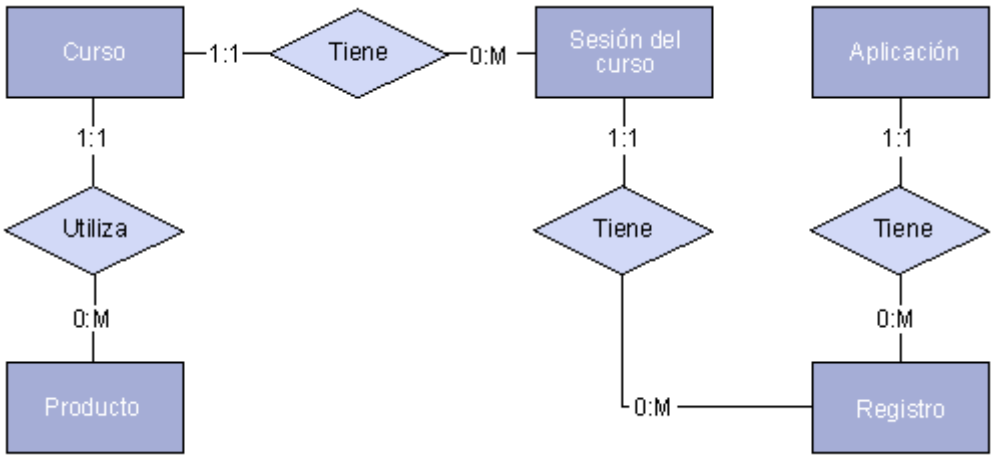
Ejemplo de diagramas de gráficos por sectores, mapas metales y planos de oficinas

- **Diagramas de auditoría**. Este tipo de diagramas se utilizan para documentar y analizar procesos que incluyen transacciones financieras y gestión de inventarios.



Ejemplo de diagrama de auditoría

- Diagramas de **modelos conceptuales** de bases de datos. Existen diferentes notaciones para **modelizar "conceptualmente"** bases de datos. La mayoría de las herramientas ofrecen prestaciones para diseñar estos modelos y generar a partir de éstos los **modelos relacionales** de las bases de datos. Algunas van incluso más allá y generan la base de datos para diferentes **sistemas gestores** de bases de datos. Algunas de estas notaciones son la notación Bachman, Chen ERD, etc.



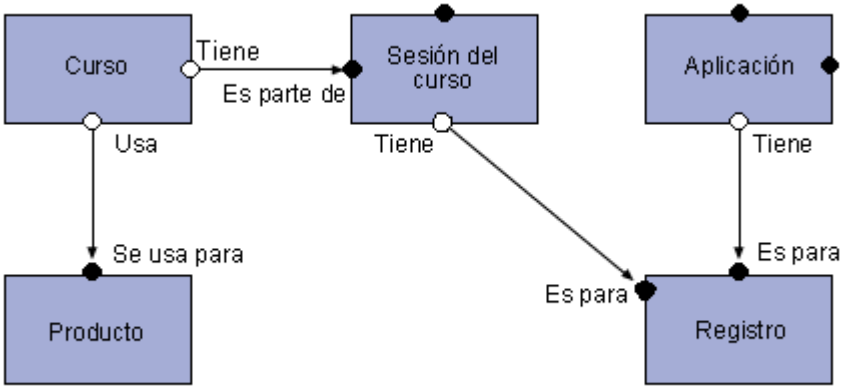


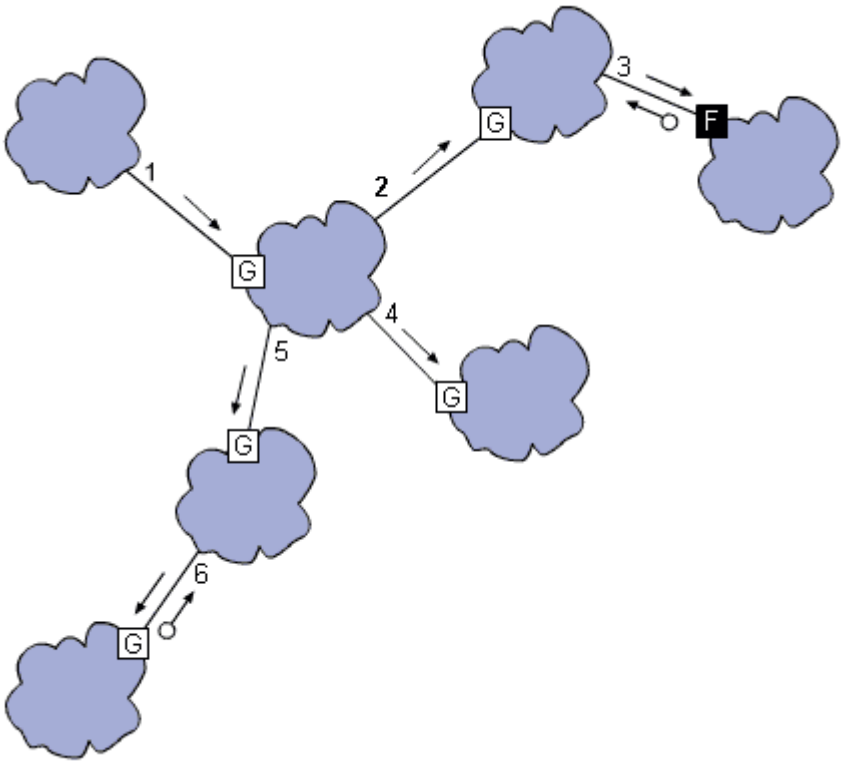
Diagrama de diseño conceptual de bases de datos

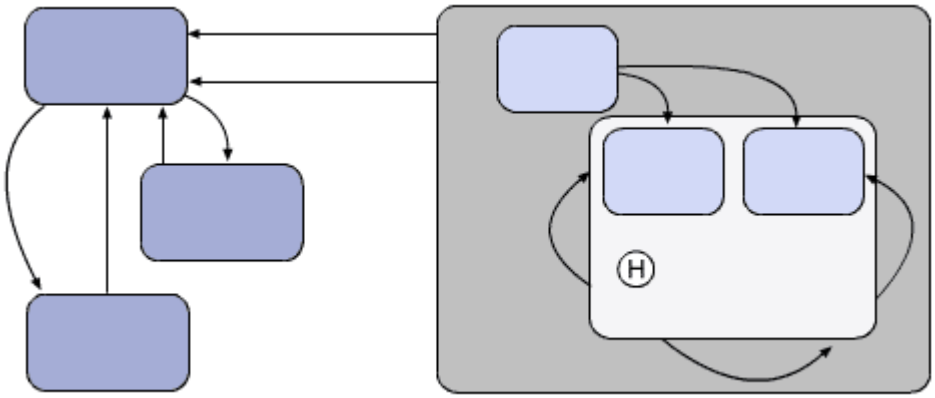
Podéis consultar el núcleo de conocimiento "Definición conceptual de una base de datos" del apartado "Bases de datos" del módulo "Gestión de datos".

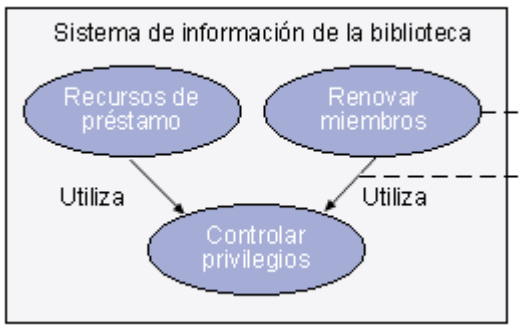
Podéis consultar el núcleo de conocimiento "Generación de bases de datos relacionales" del apartado "Bases de datos" del módulo "Gestión de datos".

Podéis consultar el núcleo de conocimiento "Sistemas gestores de bases de datos" del apartado "Bases de datos" del módulo "Gestión de datos".

- Diagramas de modelos **orientados a objetos y notación UML**. Siguiendo diferentes notaciones (Booch OOD, los casos de uso de Jacobson, la notación ERD Martin, el lenguaje de modelado ROOM o la metodología OMT de Rumbaugh) este tipo de diagramas permiten diseñar aplicaciones orientadas a objetos. Al igual que en el caso de los diagramas de bases de datos, la mayoría de las herramientas CASE disponen de **funciones para generar código de programación a partir de los diseños**.







Diagramas de diseño orientado a objetos según diferentes notaciones

- Diagramas **conceptuales de *web sites***. Utilizados en el diseño y desarrollo de webs, este tipo de diagramas permite esquematizar la organización en páginas de una web, la navegación entre ellas, su contenido, etc.

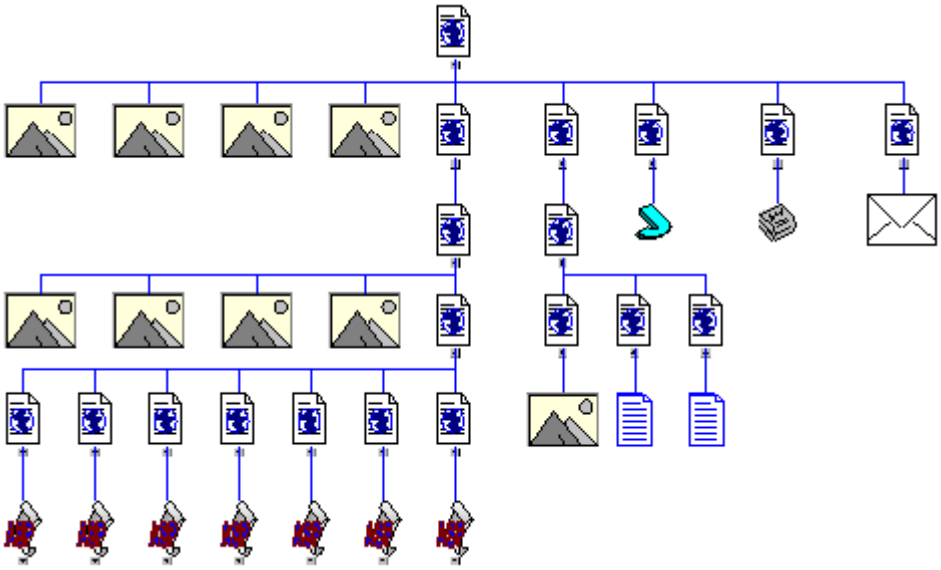
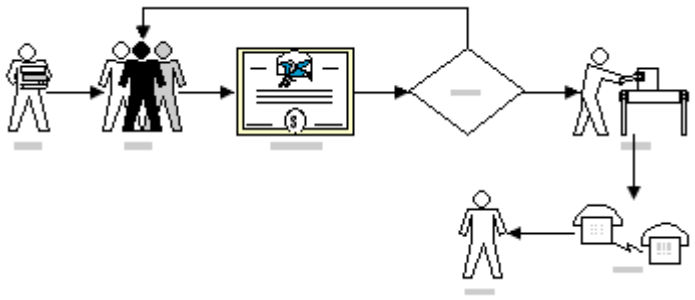
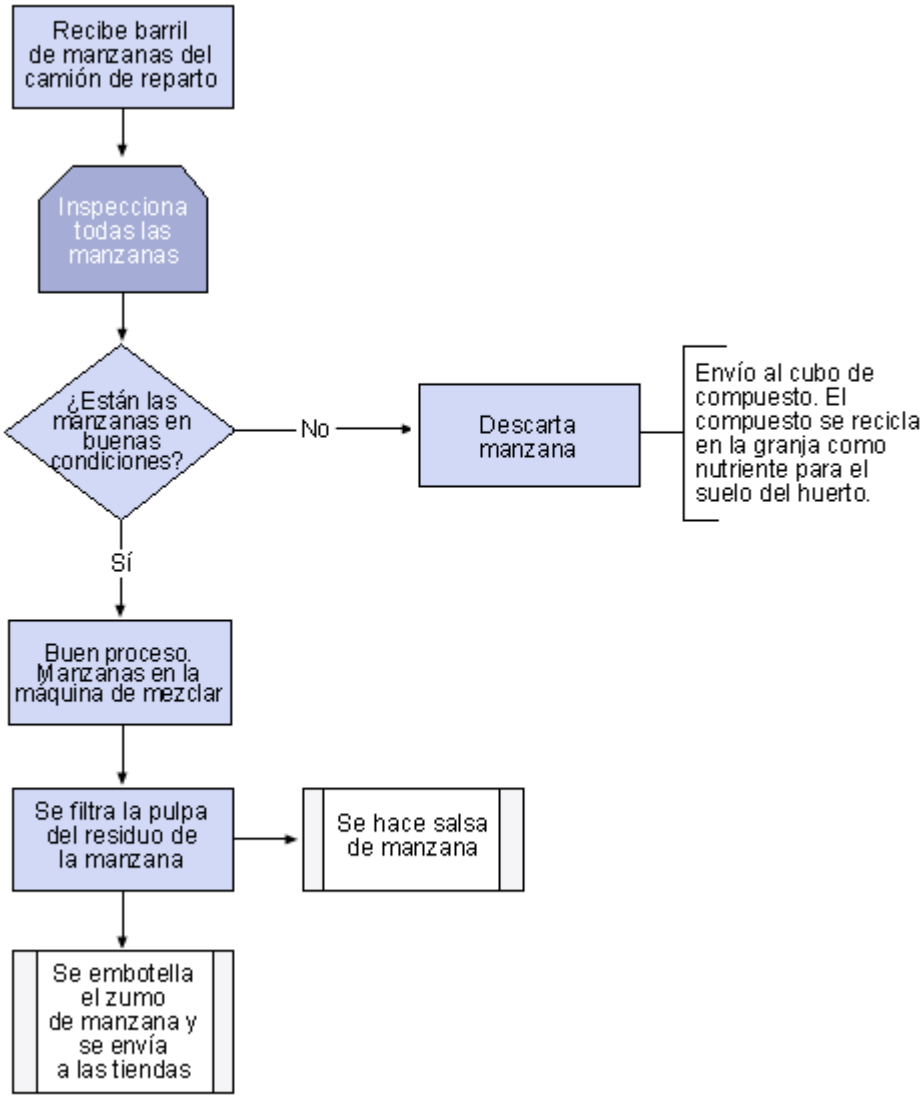


Diagrama de diseño de una *web site*

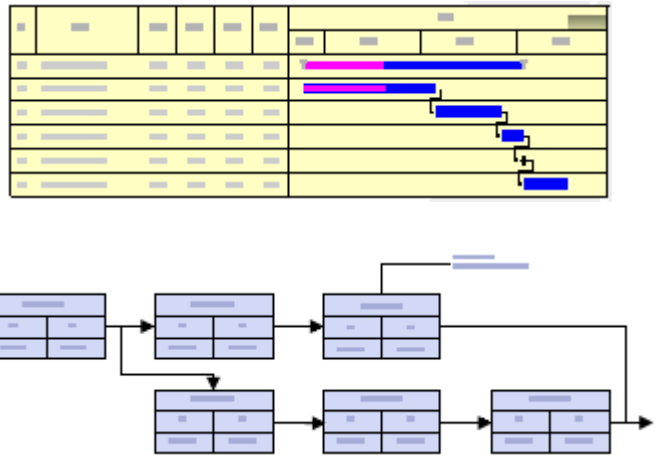
- Diagramas de **procesos y flujos de datos**. Permiten diseñar y reflejar los procesos de negocio y los flujos de datos y documentos de los diferentes departamentos de una compañía y la interacción entre éstos. Las herramientas CASE avanzadas disponen de funcionalidades para **simular** el comportamiento de dichos procesos y prever situaciones futuras modificando los diferentes parámetros de los procesos. Un método de generación de modelos de flujos de datos es el de Gane-Sarson.





Diagramas de procesos de flujo de trabajo

- Diagramas **GANTT y PERT**. Se utilizan para planificar proyectos y procesos. Existen herramientas específicas, los sistemas gestores de proyectos, para tratar este tipo de información.



Diagramas Gantt y PERT

- **Diagramas de redes**. Este tipo de diagramas permite crear y diseñar el diagrama tanto lógico como físico de una **red de ordenadores y dispositivos**. Una vez diseñados los citados diagramas, algunas aplicaciones permiten guardar esta información en bases de datos y/o realizar simulaciones de tráfico por la red, etc.

Podéis consultar el núcleo de conocimiento "Tipología de redes" del apartado "Redes" del módulo "Redes y transmisión de datos".

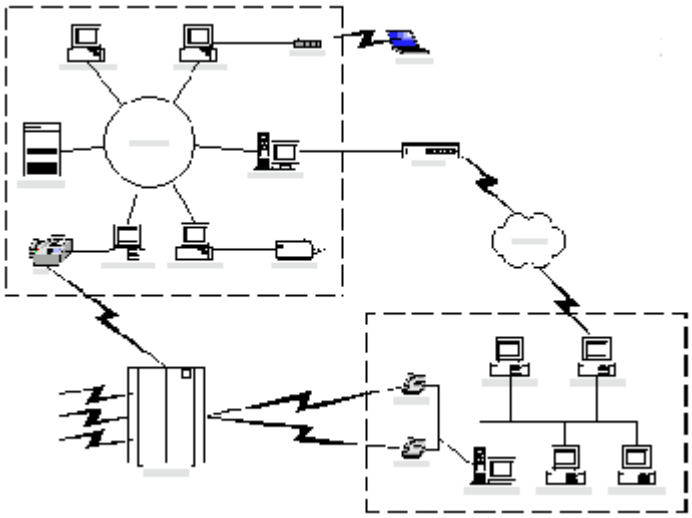


Diagrama lógico de redes e infraestructuras

- Diagramas de **interfaz de usuario, formularios e informes**. La mayoría de los programas y aplicaciones disponen de una interfaz gráfica para el usuario. Cuando trabajamos con un programa introducimos datos, leemos información, seleccionamos opciones de menús... Durante la fase de diseño de un *software*, se utiliza este tipo de diagramas para definir cómo serán sus diferentes interfaces. Muchas herramientas CASE ofrecen funcionalidades para diseñar formularios de entrada de datos e informes, y más adelante podrán generar esos formularios e informes en código de programación de diferentes lenguajes.

Diseño de formulario de interfase de usuario

Las herramientas CASE facilitan la producción automatizada de *software*.

El *software* en la empresa

La aplicación del *software* en las organizaciones es muy amplia; sólo depende de las posibilidades que se generen.

> **En el mercado hay miles de aplicaciones y *software* estándar que resuelven y dan solución a los principales problemas que tienen las compañías.**

Estas aplicaciones pueden ser desde aplicaciones de nóminas o gestión de recursos humanos hasta sistemas de control de la producción, la monitorización de cadenas de montaje o la gestión de proyectos, pasando por los sistemas internos de la compañía que garantizan la existencia de bases de datos corporativas, servicios de mensajería y colaboración o la publicación de aplicaciones en Internet.

> **A este abanico de posibilidades hay que añadir las infinitas posibilidades existentes para desarrollar soluciones a medida.**

Estas soluciones se construyen utilizando los diferentes **lenguajes de programación**, y usando **herramientas CASE** como apoyo a las diferentes metodologías enmarcadas en la **ingeniería del *software***.

Podéis consultar el núcleo de conocimiento "Lenguajes de programación" del apartado "Desarrollo de *software*" de este mismo módulo.

Podéis consultar el núcleo de conocimiento "Herramientas CASE" del apartado "Desarrollo de *software*" de este mismo módulo.

Podéis consultar el núcleo de conocimiento "Introducción a la ingeniería del *software*" del apartado "Desarrollo de *software*" de este mismo módulo.

Si quisiéramos **clasificar** todo este *software* o aplicaciones en categorías, podríamos utilizar **diferentes criterios**:

- Por **departamento funcional**. Aplicaciones de recursos humanos, finanzas, producción, gestión de clientes, gestión de contenido, gestión documental, etc.
- Por **familia de soluciones**. Soluciones de negocios entre empresas (B2B), soluciones de negocios empresa-cliente (B2C), las soluciones de optimización de procesos de la compañía y servicios al empleado (B2E) o las soluciones de negocios entre empresas y gobierno (B2G), por ejemplo.
- Por **aplicaciones cliente o servidor**. En el primer caso, existen programas que se ejecutan en el ordenador del usuario, mientras que en el segundo, se cuenta con aplicaciones alojadas en una máquina servidora a la que acceden todas las máquinas de los usuarios.

De esta manera, se podrían generar otras clasificaciones.

Dado el objetivo de esta asignatura, servir como introducción a las tecnologías de la información, no se cubrirán ni las soluciones de negocio departamentales ni las soluciones basadas en tecnologías Internet (B2B, B2C, etc.).

A continuación nos disponemos a presentar dos tipos de aplicaciones *software*: las herramientas de escritorio y las aplicaciones servidoras:

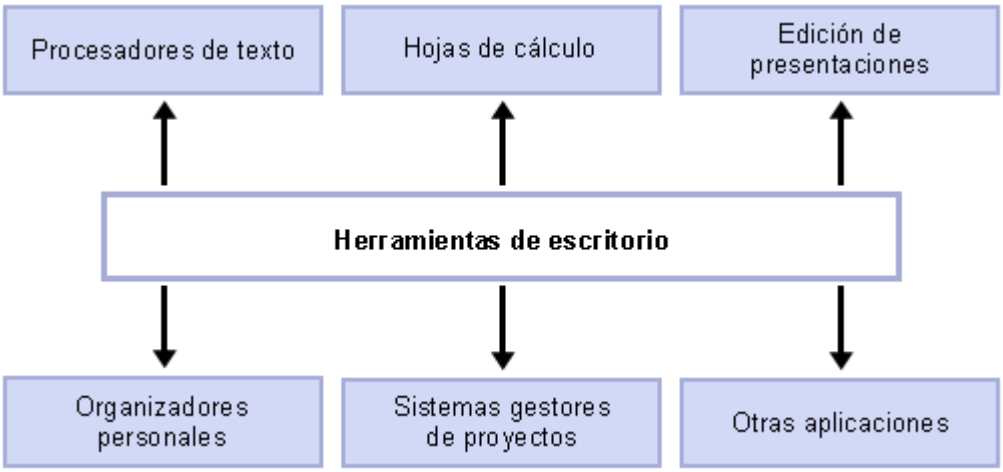
- **Aplicaciones de escritorio**. En esta categoría se consideran todas aquellas aplicaciones de uso genérico en el trabajo de cualquier usuario, tanto desde el punto de vista personal como profesional: procesadores de texto, hojas de cálculo, gestores de proyectos, etc.
- **Aplicaciones servidoras corporativas**. Se cubrirán en este apartado todos los servidores básicos que sustentan los sistemas de información de la compañía y que permiten el trabajo con herramientas de comunicación y colaboración, bases de datos corporativas, etc.

La aplicación del *software* en las compañías ha supuesto muchos cambios, tanto en la cultura organizacional de las mismas, la aparición del teletrabajo, las reuniones de trabajo virtuales, la formación a distancia, la automatización de la mayoría de los procesos, etc.

Juan Luis Cebrián (1999). "Dónde está mi oficina". En: *La red*. Madrid: Taurus.

No obstante, el impacto social de esta aplicación no es objeto de la presente asignatura, de manera que no profundizaremos en ese tema.

Herramientas de escritorio



Procesadores de texto

Una de las principales herramientas de productividad personal, quizá la más importante, es el procesador de texto.

Antes de que se crearan los procesadores de texto era común que se utilizaran **editores de texto.** A pesar de que éstos no contaban con todas las características de formateo que nos proporciona un procesador de palabras, era más fácil crear un documento en un editor que en una máquina de escribir, ya que el documento se almacena en un disco y se puede utilizar tantas veces como sea necesario, realizando las modificaciones necesarias sin tener que volver a escribir todo el documento.

Entre las características que tiene un editor de textos se encuentra la capacidad de escribir el texto y modificarlo posteriormente (por ejemplo, insertando o eliminando caracteres, moviendo o copiando una porción del documento a otra parte del mismo, buscando la ocurrencia de alguna palabra o frase y, si se desea, reemplazándola por otra palabra o frase).

Los editores de texto también han sido y siguen siendo muy utilizados para la escritura de programas.

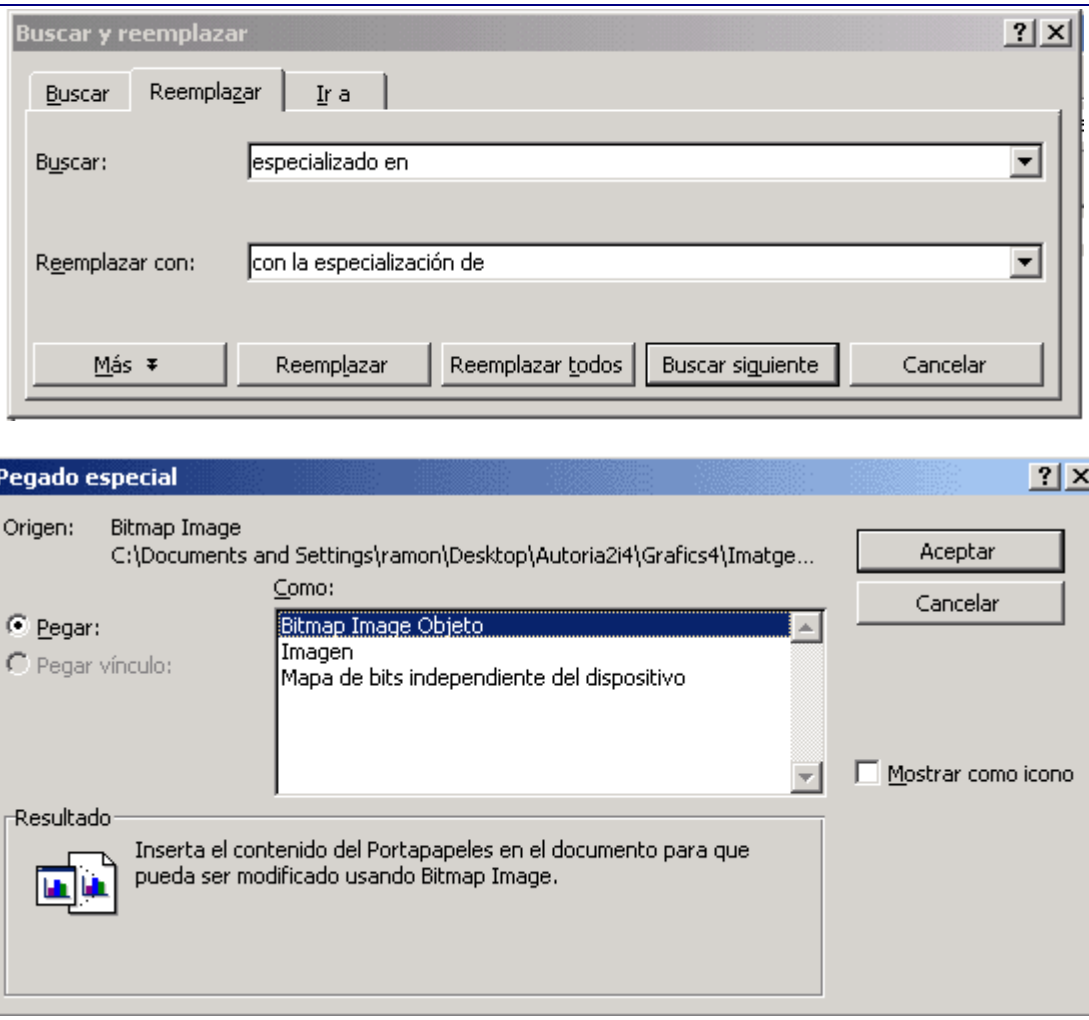
Todos los sistemas operativos proporcionan un editor de textos. Dos productos del mercado son WordPerfect y Ms Word.

Los **paquetes de procesamiento de textos** ayudan al usuario a efectuar **funciones** como la entrada y edición de texto, el formato, la revisión y el almacenamiento del documento en disco y la impresión del documento.

- **Entrada de texto**. La captura en un procesador de textos es similar a utilizar una máquina de escribir, pero de manera más sencilla. Al ir tecleando en el teclado, el texto aparece en la pantalla y queda almacenado en la memoria. Como la memoria no es un medio de almacenamiento permanente, es importante guardar con frecuencia el texto para tener una copia en nuestro disco.

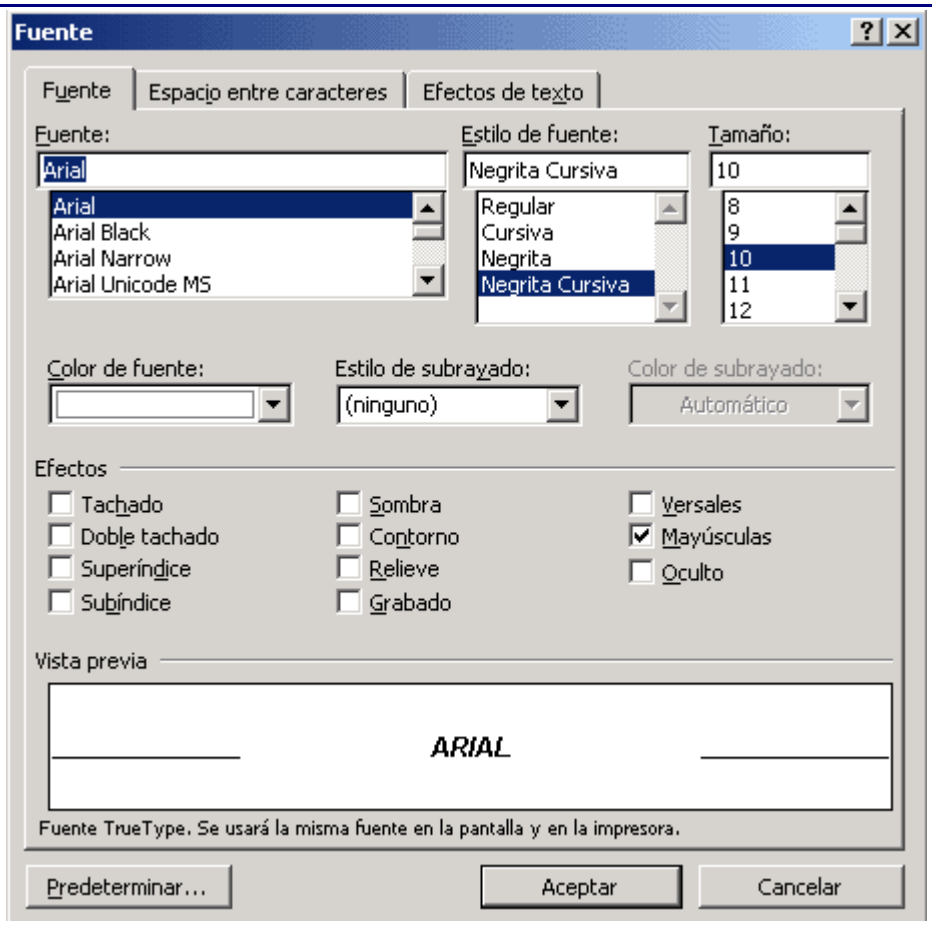
  Podéis consultar el núcleo de conocimiento "Dispositivos de almacenamiento de datos" del apartado "El almacenamiento externo de la información" del módulo "Gestión de datos".

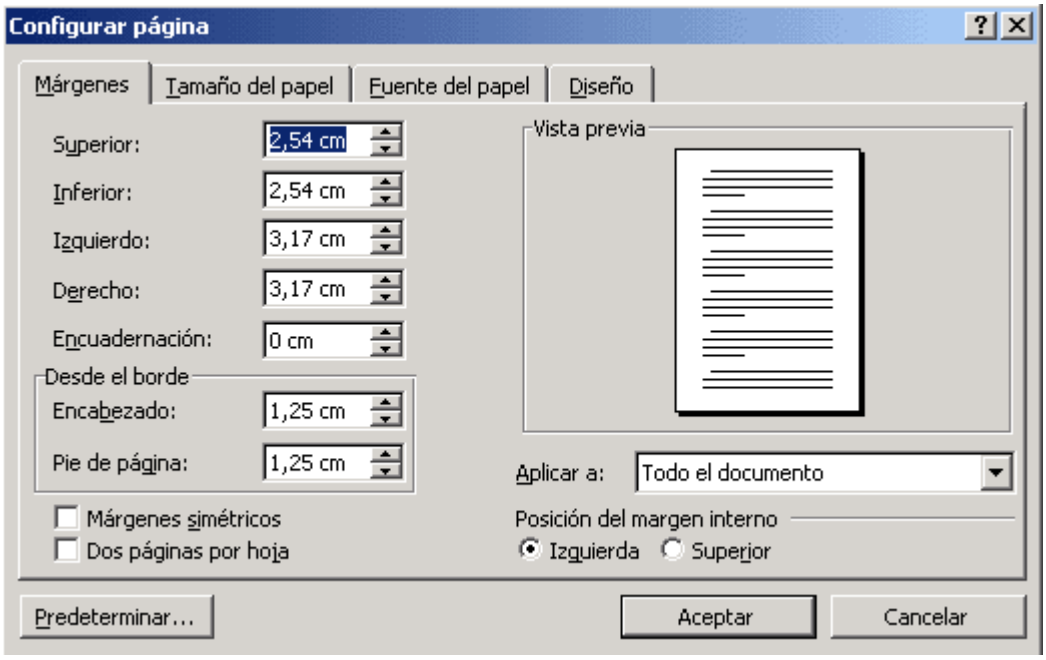
- **Edición de textos**. Todos los procesadores de textos permiten escribir el texto y darle una presentación adecuada para ser impreso en papel. Si se trabaja con un procesador de textos moderno, éste contiene la característica **WYSIWYG** (siglas de *What You See Is What You Get*, 'lo que se ve es lo que obtiene'), de manera que podemos acomodar el texto en la pantalla con la confianza de que ésta será su presentación en papel. Un procesador de textos dispone de funciones de desplazamiento a diferentes partes del documento utilizando el teclado, el ratón o por medio de menús solicitando una búsqueda. También permite insertar texto en cualquier lugar del documento o eliminarlo. Permite, además, seleccionar una parte de texto y cambiarla o copiarla a otro lugar del documento, encontrar la ocurrencia de alguna palabra o frase, etc.



Herramientas de reemplazo de texto en el documento y de copiado de información

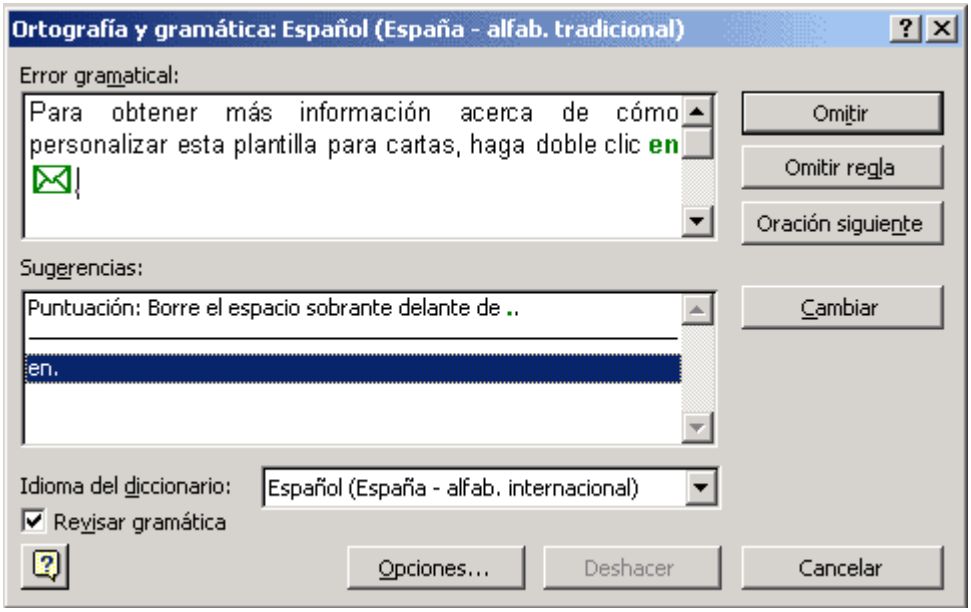
- **Formato de texto**. Disponen de funcionalidades para controlar el formato de los caracteres, líneas y párrafos de todo el documento, tales como la fuente del texto (Arial, Times New Roman, Helvética, etc.), el tamaño, la forma (subrayado, negrita o cursiva), el color, los márgenes, la orientación de la página, el contenido, el tamaño y estilo de los encabezados y pies de página, etc.





Herramientas de definición de diferentes aspectos del formato del documento

- **Revisión ortográfica**. Los procesadores de textos proporcionan herramientas que revisan la ortografía del texto. Estos correctores de ortografía buscan cada palabra que se encuentra en el documento y la revisan mediante su búsqueda en un diccionario. Si la palabra coincide con la del diccionario, eso significa que ha sido escrita correctamente, y procede a revisar la siguiente. Cuando la palabra no se encuentra en el diccionario, puede ser porque fue escrita incorrectamente o porque simplemente no está registrada en el diccionario (el usuario, en este caso, podrá agregarla si así lo desea). Algunos paquetes incluyen **correctores de gramática y estilo** que revisan, por ejemplo, que alguna palabra no se repita demasiadas veces y que no se cometan errores gramaticales y de puntuación. También incluyen herramientas que proporcionan **sinónimos** de las palabras para ayudar al usuario durante la redacción del texto.



Herramientas de selección del idioma del documento y revisión ortográfica

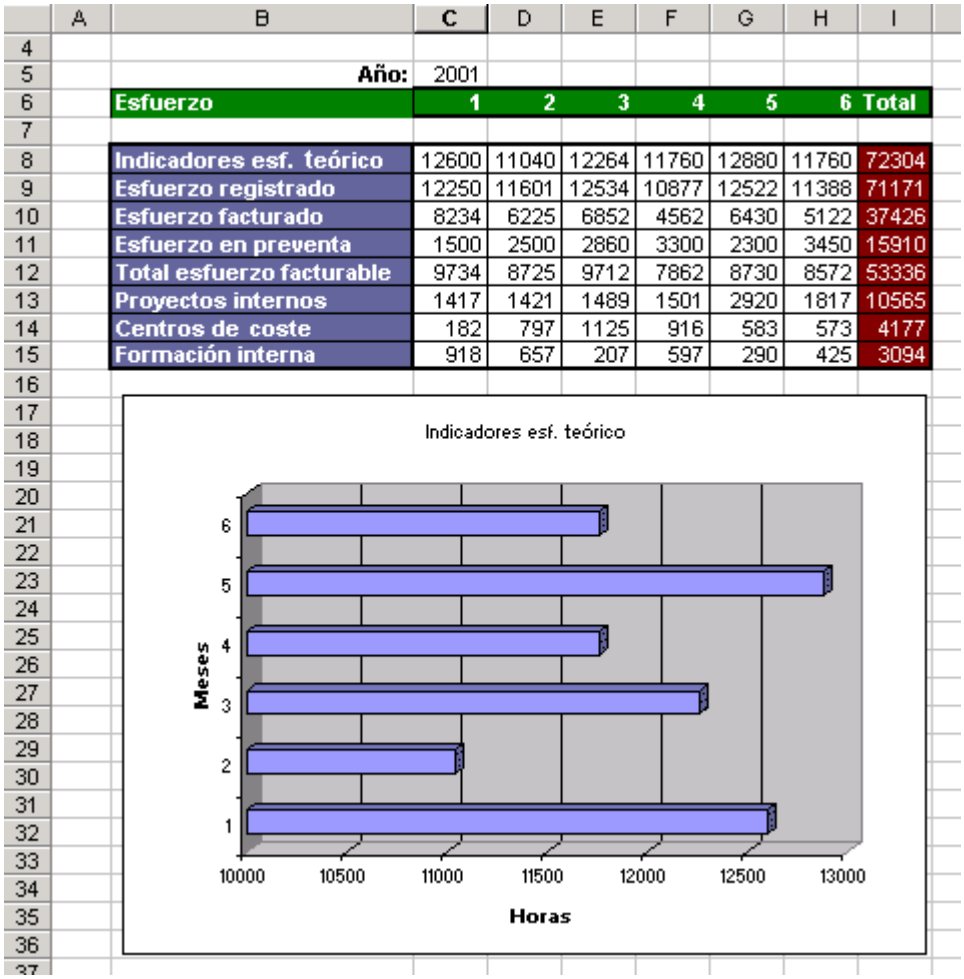
- **Impresión de documentos**. Para imprimir un documento, el *software* del procesador de textos utiliza un ***driver*** (dispositivo) de impresión. Este *driver* convierte los códigos del texto en instrucciones que la impresora pueda entender para producir el efecto deseado. Cada impresora requiere un *driver* diferente.

Podéis consultar el núcleo de conocimiento "La unidad de entrada/salida" del apartado "Arquitectura de von Neumann" del módulo "El ordenador, máquina informacional".

Hojas de cálculo

A continuación detallamos algunas de las funciones habituales de un programa de gestión de hojas de cálculo:

- **Manipulación de celdas**. La función principal de la hoja de cálculo consiste en permitir la gestión de los datos numéricos por medio de cálculos que van desde una simple suma hasta la inclusión de centenares de fórmulas disponibles como senos, cosenos, máximo, media, etc. Cada hoja de cálculo está compuesta por una cuadrícula formada por celdas, que son unos contenedores en los que se insertan y procesan datos y en los que se muestra el resultado. Los datos que se introducen pueden tener varios formatos y estar vinculados al contenido de otra celda.



Ejemplo de hoja de cálculo

- **Vinculación** con otras celdas. A menudo es necesario utilizar datos que han sido generados en otra hoja de cálculo del mismo documento o que proceden de otros documentos. En lugar de generar de nuevo esos datos, se remiten los originales a la nueva hoja de cálculo, de manera que si los datos de referencia cambian, estos cambios también se verán reflejados en la nueva hoja de cálculo.

C18 = =(C9/C8) *100

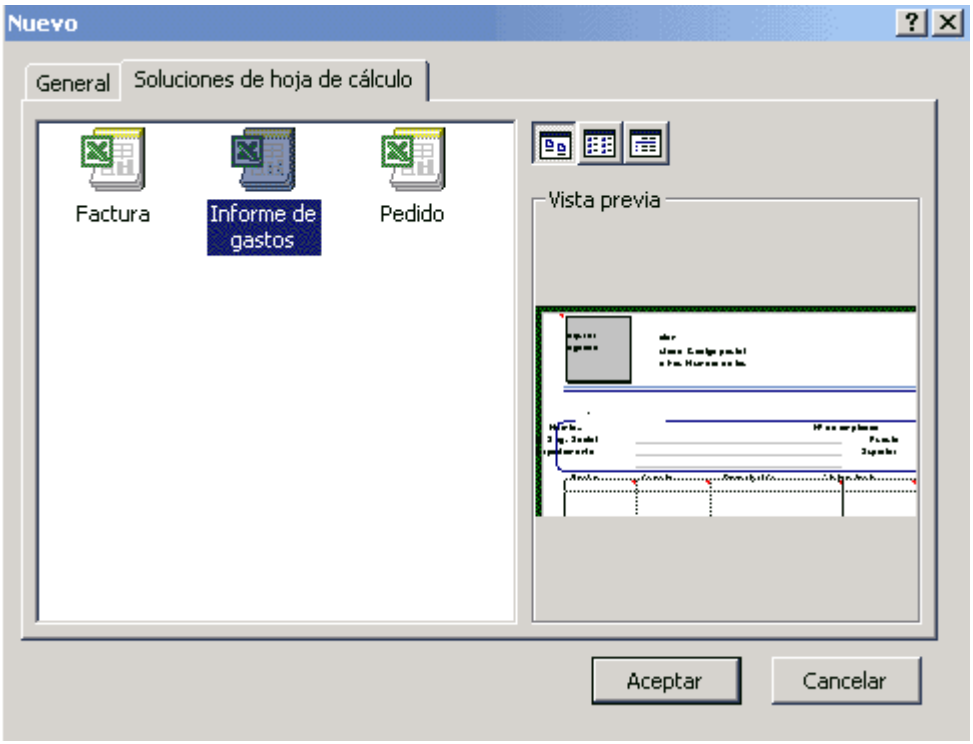
	A	B	C	D
4				
5		**Año:**	2001	
6		**Esfuerzo**	**1**	**2**
7				
8		**Indicadores esf. teórico**	12600	11040
9		**Esfuerzo registrado**	12250	11601
10		**Esfuerzo facturado**	8234	6225
11		**Esfuerzo en preventa**	1500	2500
12		**Total esfuerzo facturable**	9734	8725
13		**Proyectos internos**	1417	1421
14		**Centros de coste**	182	797
15		**Formación interna**	918	657
16				
17				
18		**Preventa vs teórico**	97,22	105,1
19				
20				
21				

Ejemplo de vinculación de datos de diferentes celdas de una hoja de cálculo

- **Formato de las celdas.** El fondo de las celdas y el texto pueden tener una amplia gama de colores y tramas para que el usuario pueda comprender mejor los datos que contiene la hoja de cálculo. También es posible realizar diferentes diseños para obtener una gran variedad de informes de uso frecuente. Esto incluye elementos como, por ejemplo, facturas, formularios de pedidos, etc.

- **Configuración.** Igual que los documentos creados con un procesador de textos, las hojas de cálculo se pueden configurar de muchas maneras, ya sea en su conjunto o en una o varias celdas. Disponen desde las funcionalidades de formatear los números hasta la cantidad de decimales que se emplean en los cálculos y/o en la pantalla, el tamaño y el estilo del texto, el símbolo de la moneda, el signo negativo o color en casos de valores negativos, etc.

- **Plantillas**. Una funcionalidad que también poseen los procesadores de texto es el uso de plantillas para disponer de modelos de hojas de cálculo que hay que utilizar de manera hahitual.



Selección de una plantilla de hojas de cálculo

- **Gráficos.** Ésta es una de las prestaciones más utilizadas en las hojas de cálculo, puesto que hace que el usuario pueda generar una gran variedad de gráficos para representar información y datos de las hojas.

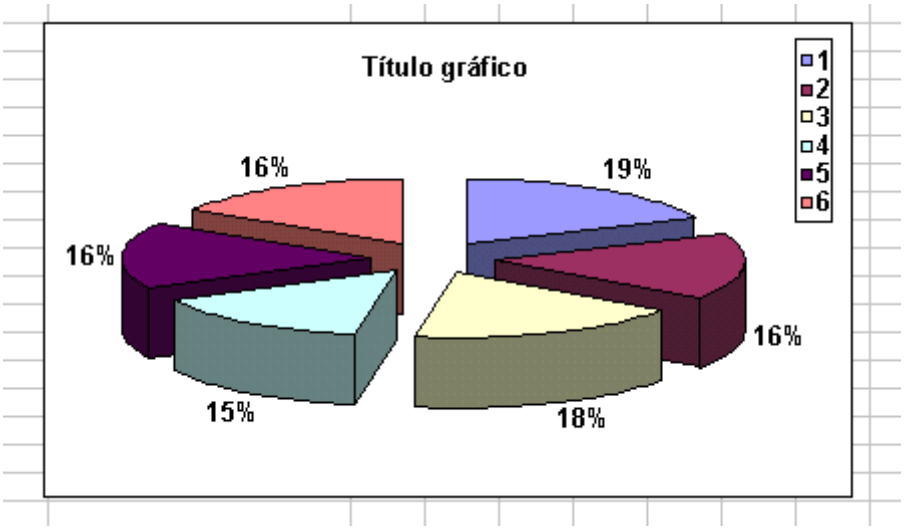


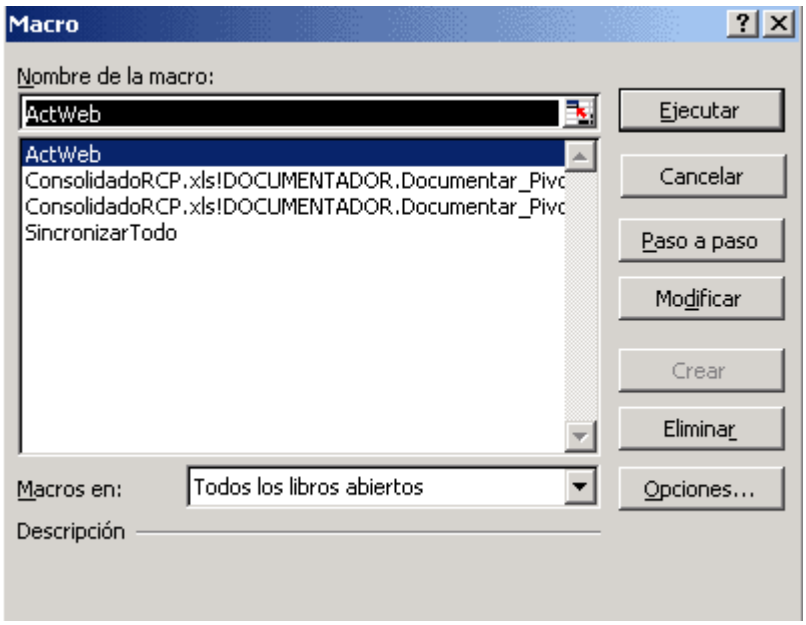
Gráfico generado en una hoja de cálculo a partir de los datos contenidos en la misma

- **Inserción de objetos.** Como en la mayoría de las aplicaciones de productividad personal (hojas de cálculo, gestores de presentación, gestores de proyectos, etc.), el usuario de éstas dispone de la posibilidad de incluir toda clase de objetos en estos documentos. De esta manera, se puede incluir una porción de una hoja de cálculo en un documento de texto o al revés.

- **Seguridad y protección de los datos.** La mayoría de los gestores de hojas de cálculo permiten bloquear el acceso a una o más celdas u ocultan el contenido de ciertas columnas o filas. También disponen de la prestación de proteger todo el documento, de manera que sólo los usuarios que dispongan de la clave de acceso podrán modificar los datos evitando cambios accidentales.

- **Auditoría.** Ciertos programas de gestión de hojas de cálculo, como Microsoft Excel, disponen de una función de auditoría que crea un mapa de las interdependencias existentes en toda la hoja de cálculo. Y eso se realiza uniendo con fechas las celdas relacionadas entre sí y mostrando las que tienen vínculos con otras anteriores, lo cual facilita la comprensión y actualización de las hojas de cálculo complejas y con gran cantidad de fórmulas, datos y vínculos.

- **Macros.** Todas las aplicaciones ofimáticas ofrecen esta prestación. Las macros son muy útiles cuando se trabaja con hojas de cálculo, sobre todo cuando se tienen que realizar informes con bastante frecuencia. De esta manera, se puede asignar cierto conjunto de instrucciones o acciones a una combinación de teclas, o se puede ampliar el conjunto de funciones del programa utilizando prestaciones de programación con los lenguajes de programación propios de la aplicación (como por ejemplo el **lenguaje de macros**, **Visual Basic for Applications**, existente en todas las aplicaciones de Ms Office).



Selección de una macro de las disponibilidades en una hoja de cálculo concreta

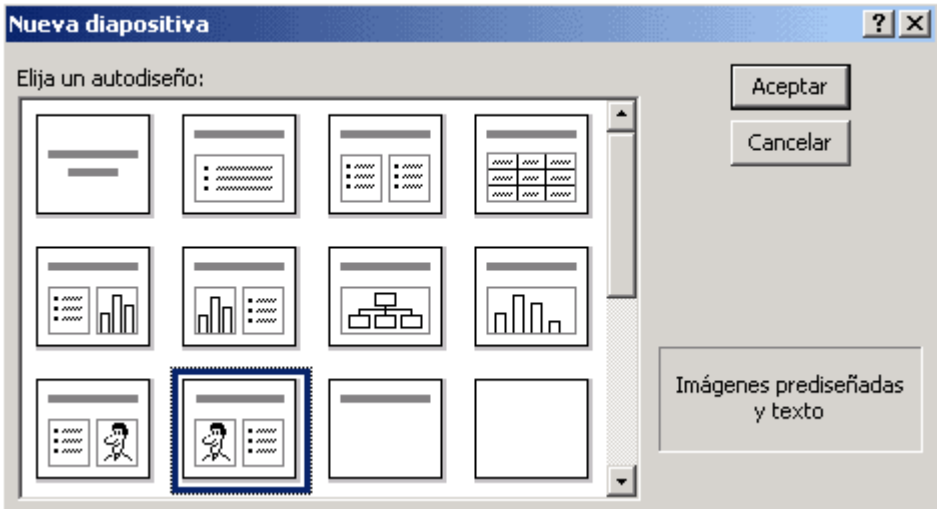
- **Vinculación con bases de datos.** Como ya tuvimos ocasión de comentar en el módulo 2, las aplicaciones naturales para almacenar datos e información son las **bases de datos**. Las BB.DD. contienen grandes cantidades de información de la empresa que puede ser necesario analizar por medio de una hoja de cálculo. La mayoría de estos programas suelen disponer de un dispositivo que permite crear, guardar y utilizar consultas, tanto de Internet a una base de datos como desde una base de datos de una red de trabajo interna.

Edición de presentaciones

Las presentaciones clásicas utilizando transparencias o diapositivas están siendo reemplazadas a un ritmo vertiginoso por las presentaciones electrónicas. Los beneficios que aportan este tipo de presentaciones son muchos frente a las primeras: prestaciones de animación, vinculación de documentos de texto, hojas de cálculo, facilidad de cambio, posibilidad de transmisión electrónica, etc.

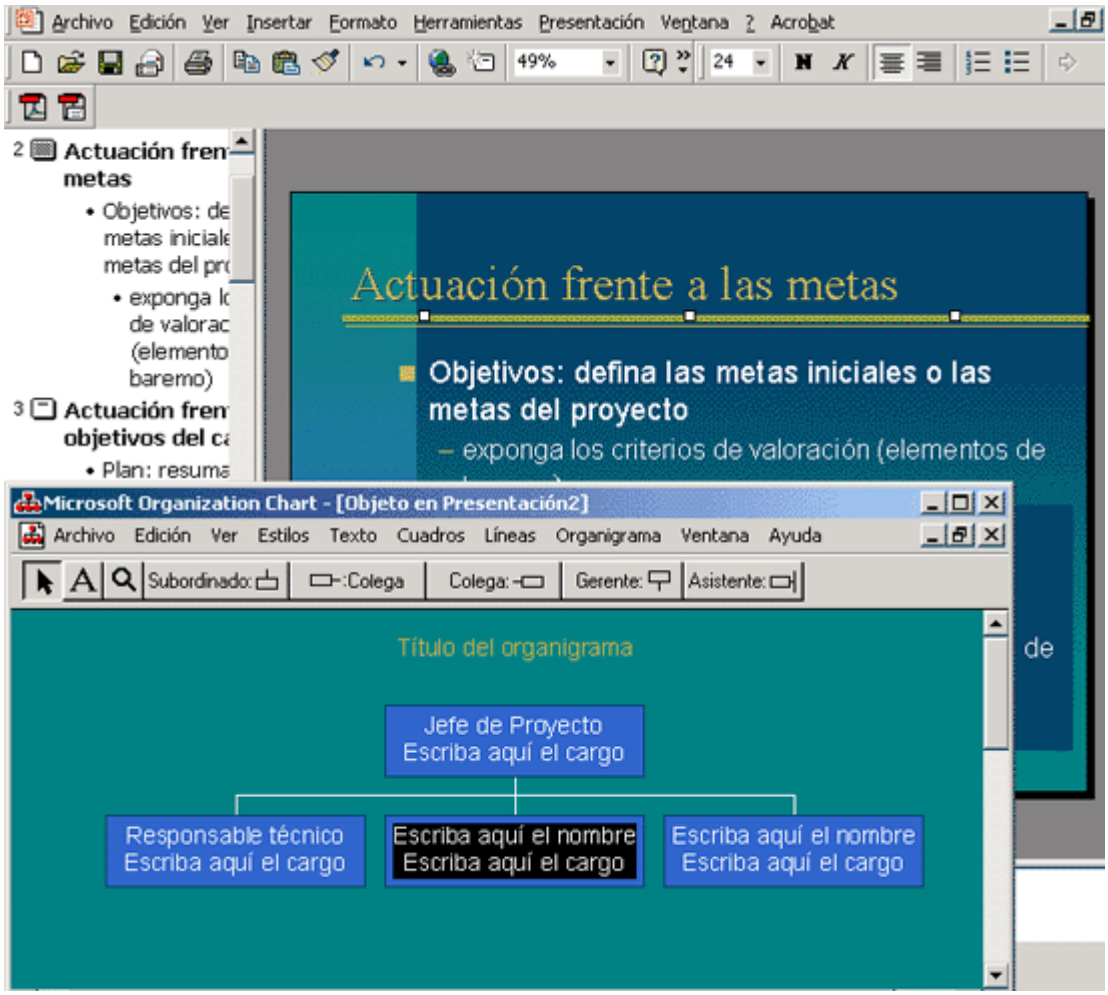
En la actualidad, uno de los programas de gestión de presentaciones es **Ms PowerPoint.** A continuación pasamos a detallar algunas de las principales funcionalidades de estas aplicaciones:

- **Generación de diapositivas.** A partir de diferentes plantillas ya predefinidas o partiendo de diapositivas en blanco, el usuario puede diseñar las diferentes "hojas" de su presentación incluyendo texto, símbolos, objetos, etc.



Modelos de diapositivas preestablecidos para presentaciones

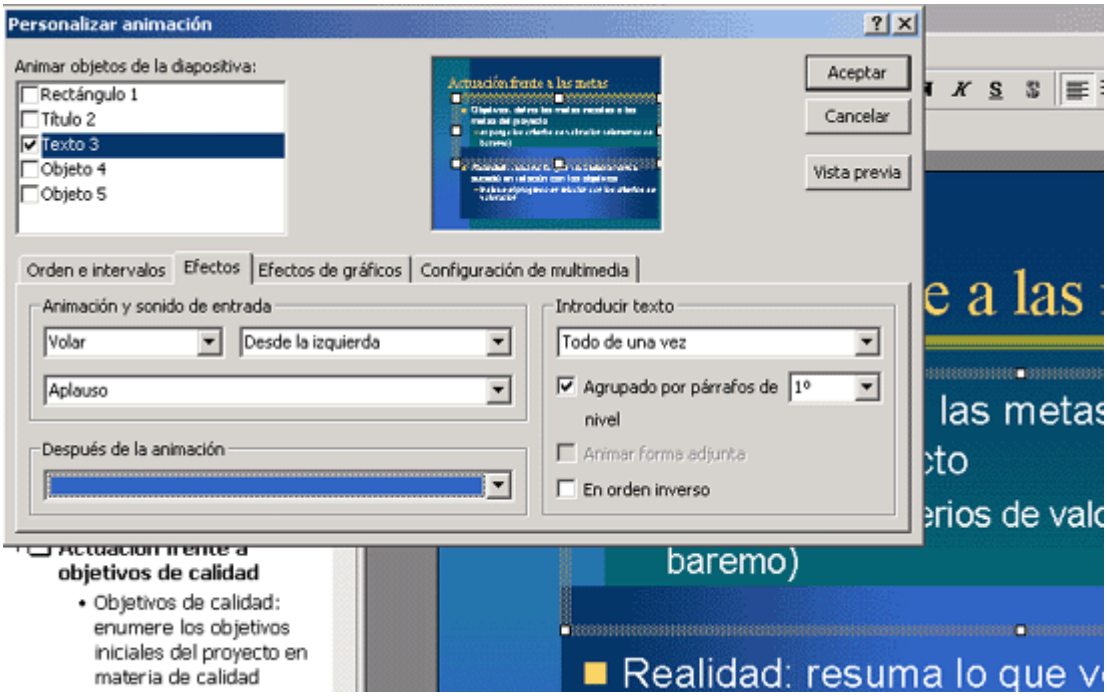
- **Inclusión de elementos.** Los elementos que hay que incluir en una diapositiva van desde texto hasta símbolos, dibujos, ficheros externos, datos provenientes de otras aplicaciones e incluso sonidos o vídeos que se ejecutarán durante la presentación.



Herramienta de Ms Powerpoint para insertar un organigrama en una presentación

- **Formatos**. Una de las principales prestaciones que ofrece este tipo de aplicaciones es la posibilidad de dotar de formato y estilos homogéneos a todas las diapositivas de la presentación.

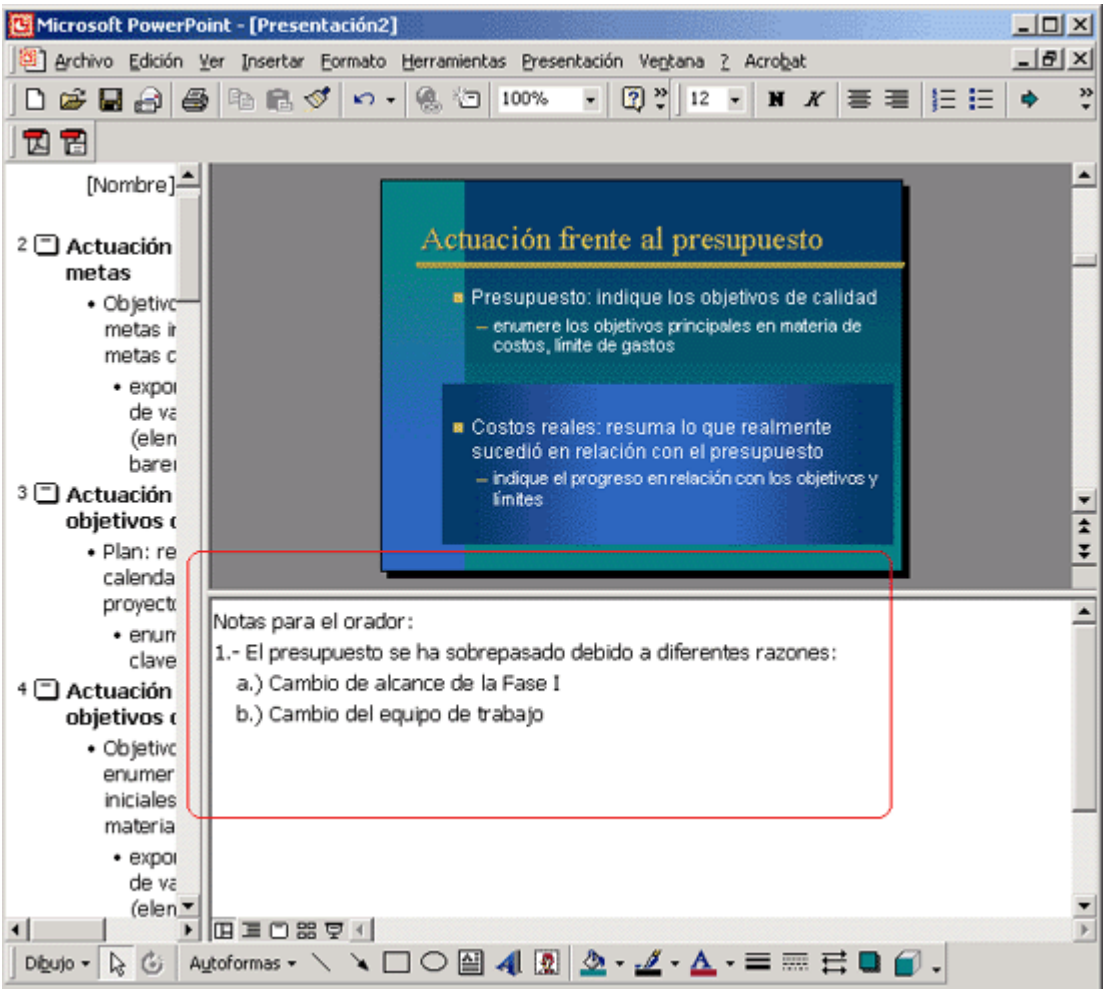
- **Animaciones y transiciones**. Estas aplicaciones permiten definir cómo se muestran los diferentes elementos de las diapositivas durante la presentación (aparecen de repente, desde una parte de la diapositiva, con efectos especiales, etc.), así como definir la transición entre una diapositiva y la siguiente. También permiten configurar si dichas animaciones son automáticas cada cierto tiempo o se activan al hacer un clic con el ratón.



Configuración de la animación de los elementos de una diapositiva de una presentación

- **Presentación de los datos**. Una vez generada la presentación, ésta se puede mostrar en diferentes formatos y modalidades: por pantalla, por medio de impresión en papel -en blanco y negro o en color-, en diapositivas de 35 mm, etc.

- **Generación de notas y documentos**. El usuario puede generar para cada diapositiva de su presentación un conjunto de notas asociadas a ésta. Asimismo, las diferentes posibilidades de impresión de las citadas aplicaciones permiten la generación de "documentos" formateados para la audiencia de la presentación.



Redacción de notas para una diapositiva en una presentación

- **Uso de plantillas y lenguajes de macros**. Tal y como ya comentamos en las aplicaciones que mencionamos antes (procesadores de texto, hojas de cálculo, etc.), los programas de edición de presentaciones también disponen de la funcionalidad de generar y usar plantillas tanto para una diapositiva en concreto como para toda la presentación. Asimismo, disponen de un lenguaje de macros que simplifica la creación de conjuntos de instrucciones y permite ampliar las funciones existentes en la herramienta.

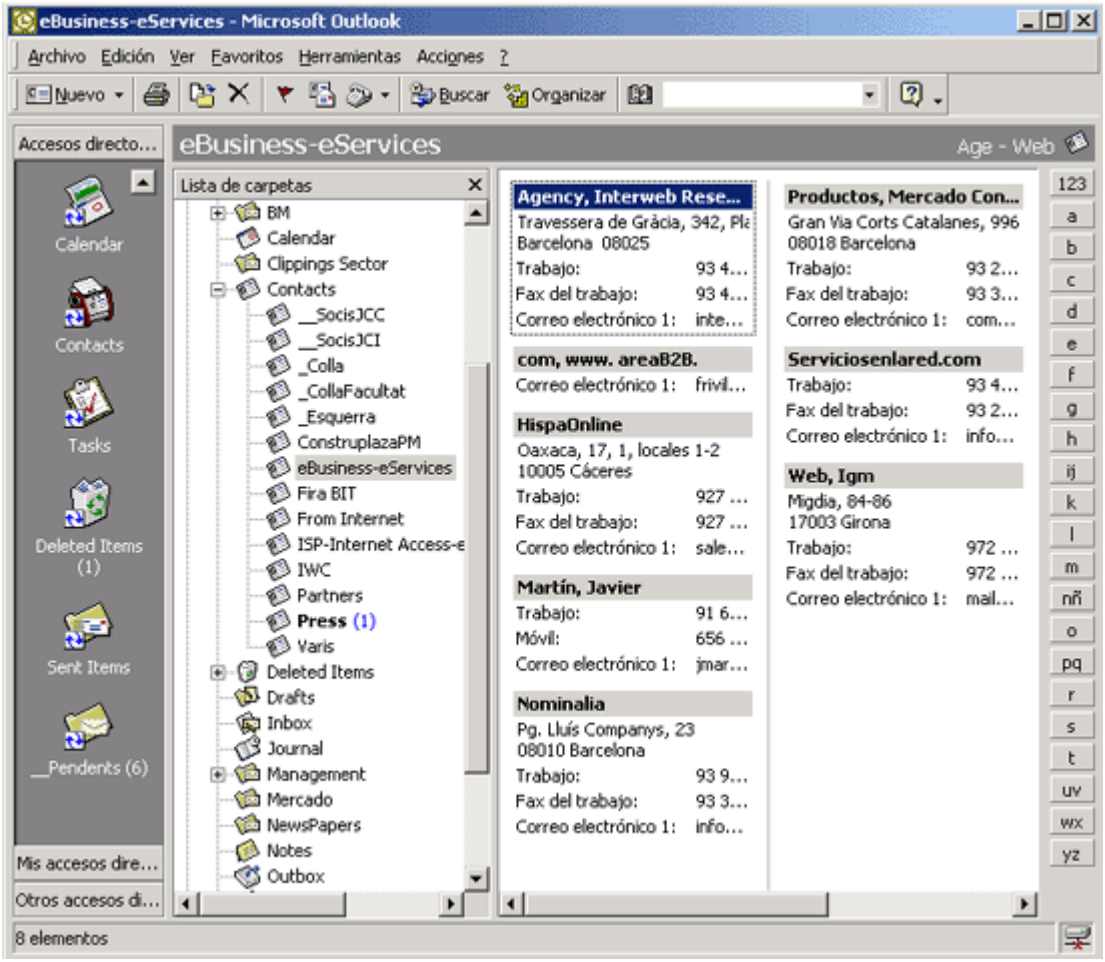
Organizadores personales

Uno de los programas más usados como apoyo al trabajo personal de los miembros de una organización son las llamadas **agendas electrónicas o asistentes personales**.

Hoy día, las comunicaciones permiten que un usuario **intercambie y actualice información entre su PC y su agenda electrónica**. Hay en el mercado una gran variedad de aplicaciones de este tipo. Uno de los organizadores personales más conocido y usado es **Microsoft Outlook**.

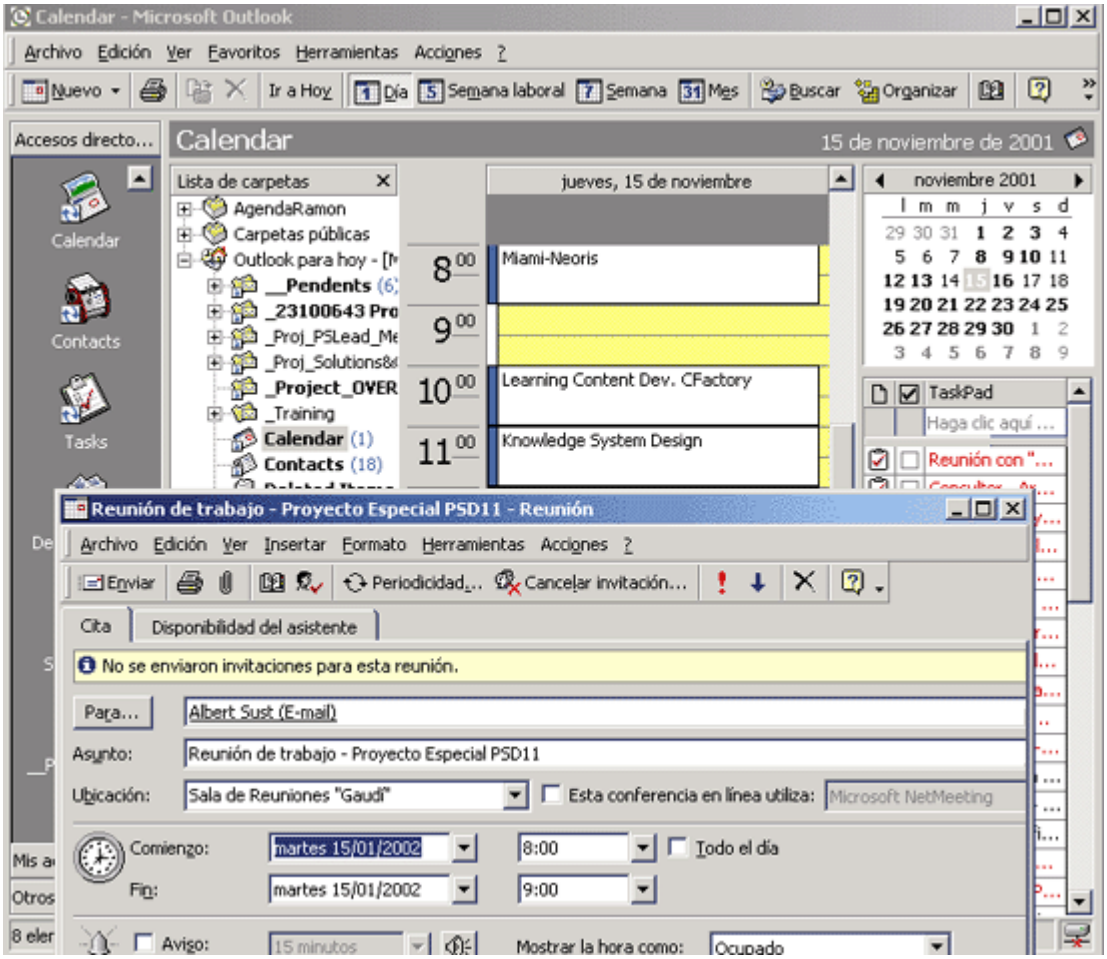
Algunas de las principales funcionalidades de dichos programas son las siguientes:

- **Gestión de contactos**. Permiten disponer de una agenda de contactos, con una gran variedad de datos (nombre, dirección, teléfono personal, profesional, correo, etc.). La lista de los contactos se puede agrupar bajo casi cualquier categoría y, si no existe la clasificación que necesita, el usuario la puede crear. Una de las mayores prestaciones de estas herramientas consiste en la posibilidad de mantener un histórico de actividades llevadas a cabo con el contacto: reuniones, llamadas de teléfono, envíos, etc.



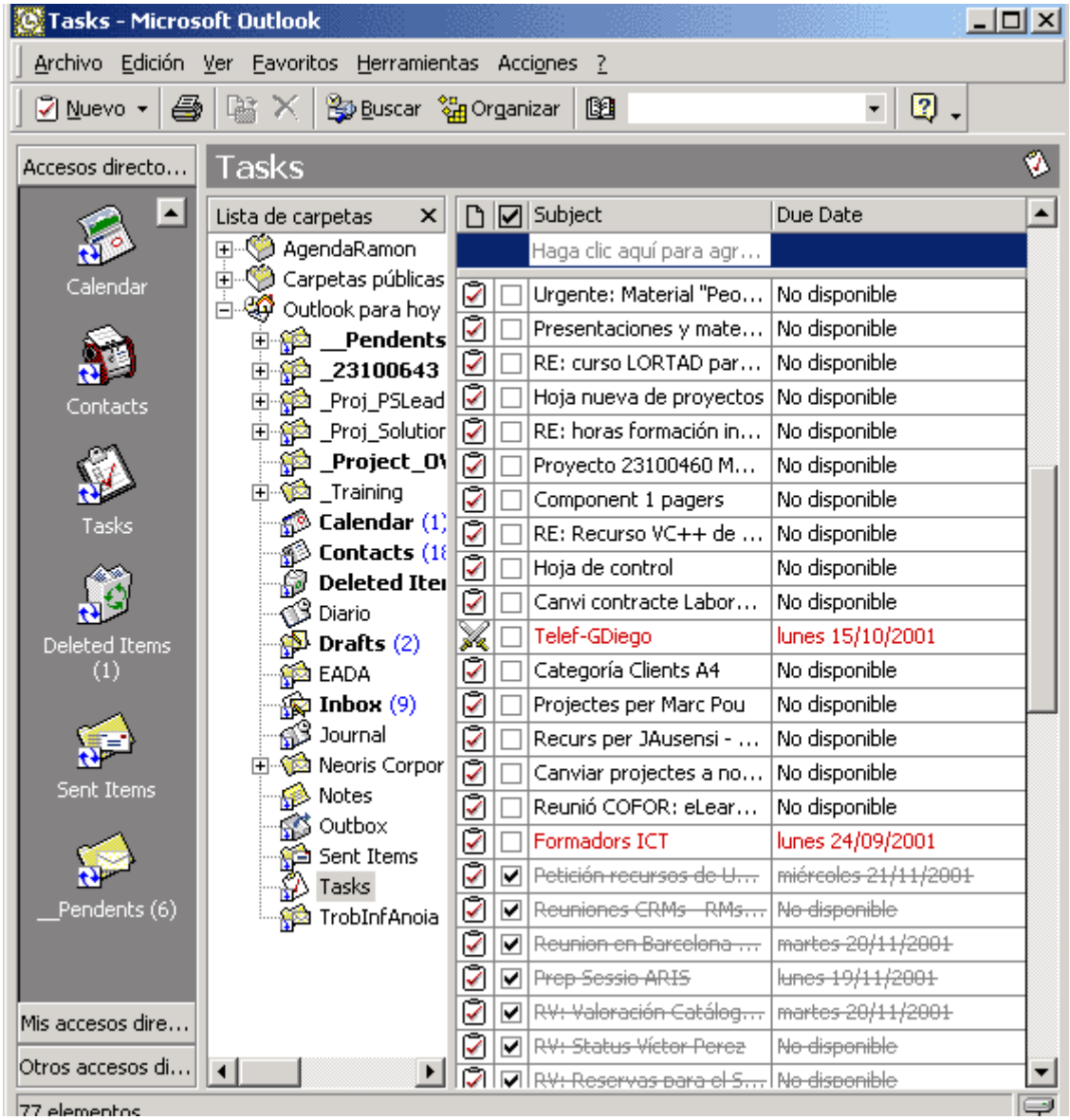
Ejemplo de agenda de contactos

- **Agenda/calendario**. Agenda personal en la que se pueden apuntar citas y compromisos; disponen de alarmas y sistemas de aviso. La mayoría de éstas se encuentran **vinculadas a sistemas de correo electrónico** internos y externos, con lo que los usuarios tienen la posibilidad de **organizar reuniones** o convocar a los participantes por vía electrónica.



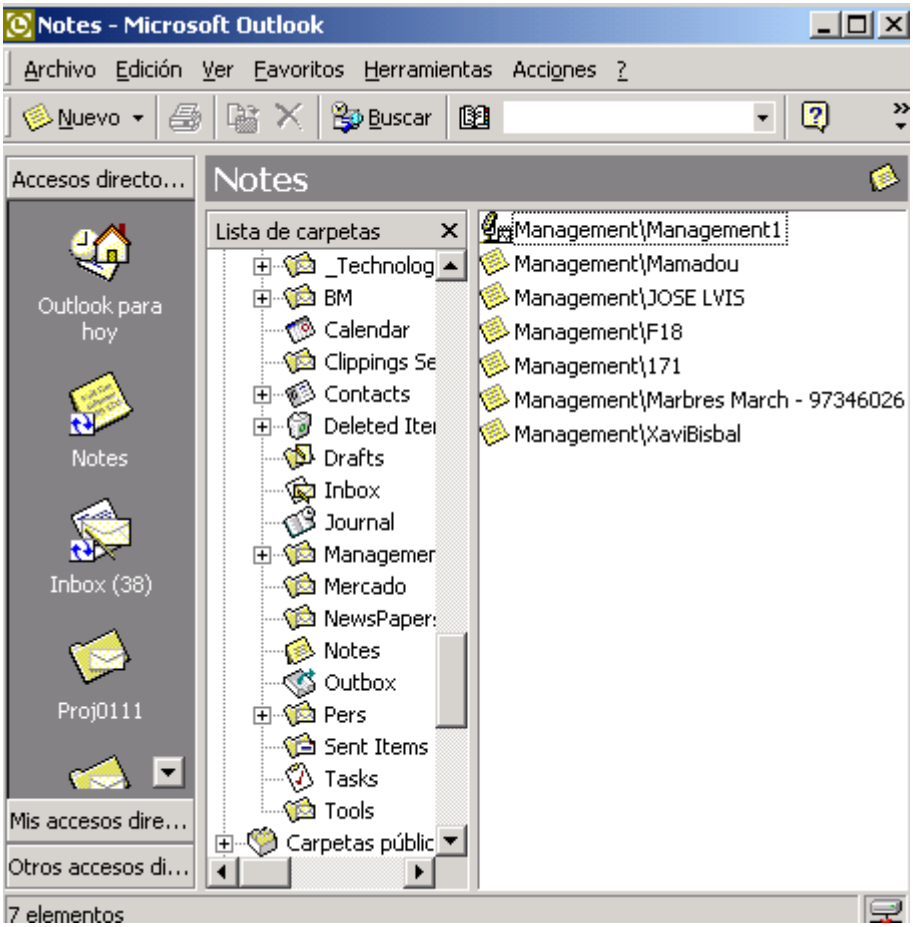
Ejemplo de convocatoria de reunión

- **Lista de tareas**. Esta funcionalidad permite organizar los asuntos según su prioridad y se avisa al usuario en el día previsto. También disponen de la posibilidad de indicar el porcentaje de resolución de cada una de las tareas.



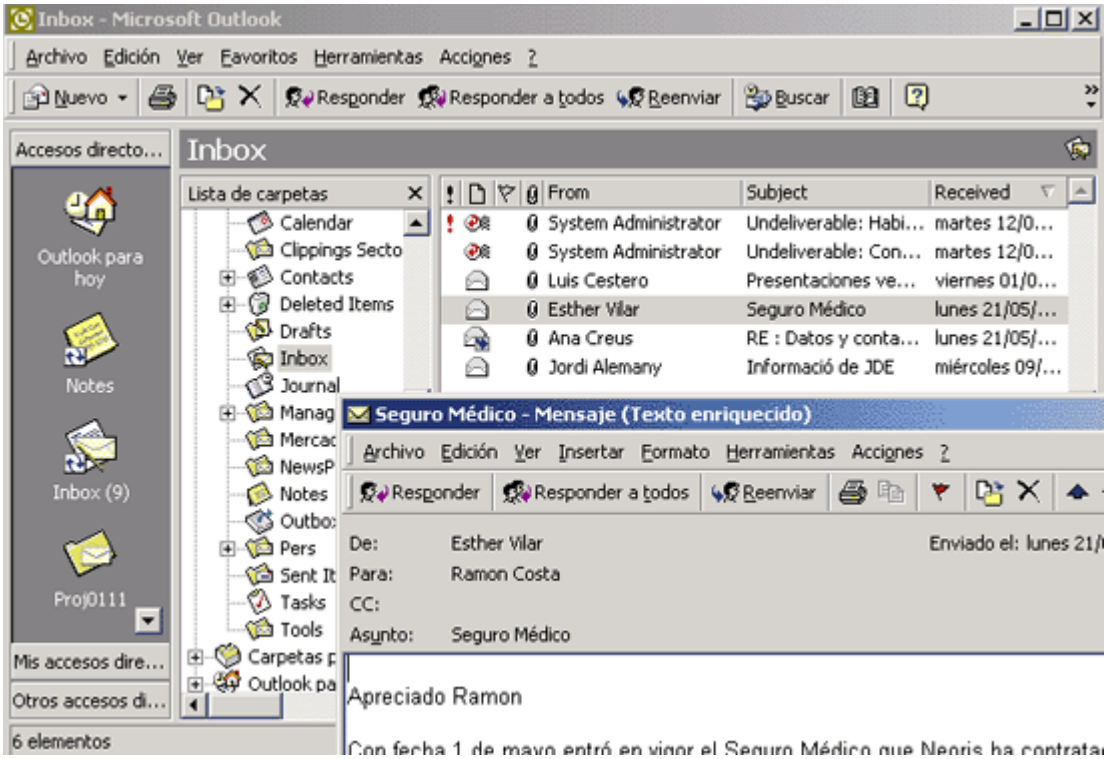
Lista de tareas activas, con y sin fecha de finalización, y finalizadas

- **Notas**. No son más que la versión electrónica de los *post it* e incluyen, además del texto o aviso en sí, la fecha y la hora en que se crearon.



Sección "Gestión de Notas"

- **Gestión del correo electrónico**. La mayoría de estas aplicaciones están relacionadas con el correo electrónico de la compañía, accediendo a los servidores de correo corporativos o a cuentas de correo electrónico personales del usuario. Permiten toda la gestión de los correos enviados y recibidos, así como de los archivos adjuntados.



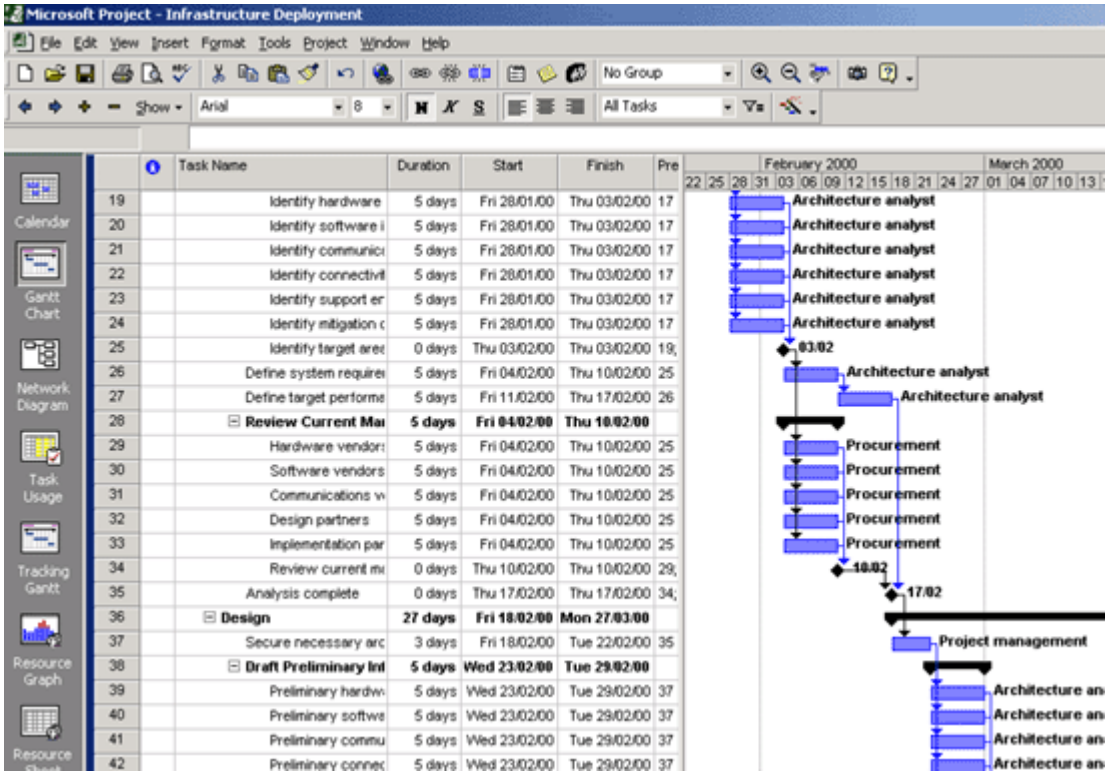
Ejemplo de carpeta de correos electrónicos

Sistemas gestores de proyectos

Una de las aplicaciones de escritorio más difundida en las compañías de *software*, consultoras y empresas de ingeniería, arquitectura y todas las que necesitan diseñar y controlar proyectos es los **sistemas gestores de proyectos**.

Estos programas facilitan la tarea tanto al jefe de proyectos como al equipo de trabajo del mismo. Las principales funcionalidades de este tipo de programas son las siguientes:

- **Planificación del proyecto.** Los sistemas gestores de proyectos permiten definir el conjunto de tareas que formarán parte del proyecto, los recursos que participarán en éste y su asignación a las diferentes tareas. También permiten definir los costes asociados a las distintas tareas y trabajos.



Ejemplo de proyecto (tareas y diagrama Gantt) en un sistema de gestión de proyectos

- **Seguimiento y actualización.** Una vez definido el proyecto y puesto en marcha, disponen de funcionalidades para poder llevar a cabo su seguimiento, indicando el trabajo realizado por los diferentes recursos en cada tarea, así como los cambios que se producen con respecto a la planificación inicial. Ofrecen todo un conjunto de herramientas (filtros, informes, etc.) para analizar las desviaciones del proyecto tanto en tiempo como en esfuerzo y coste.

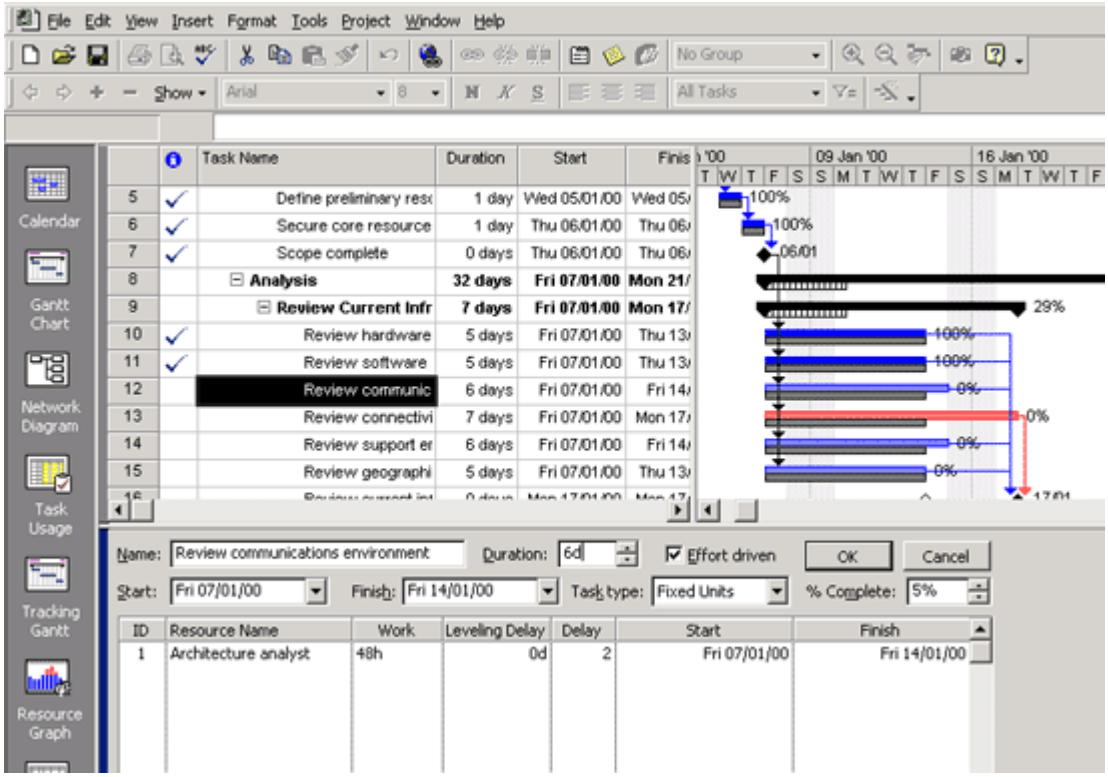
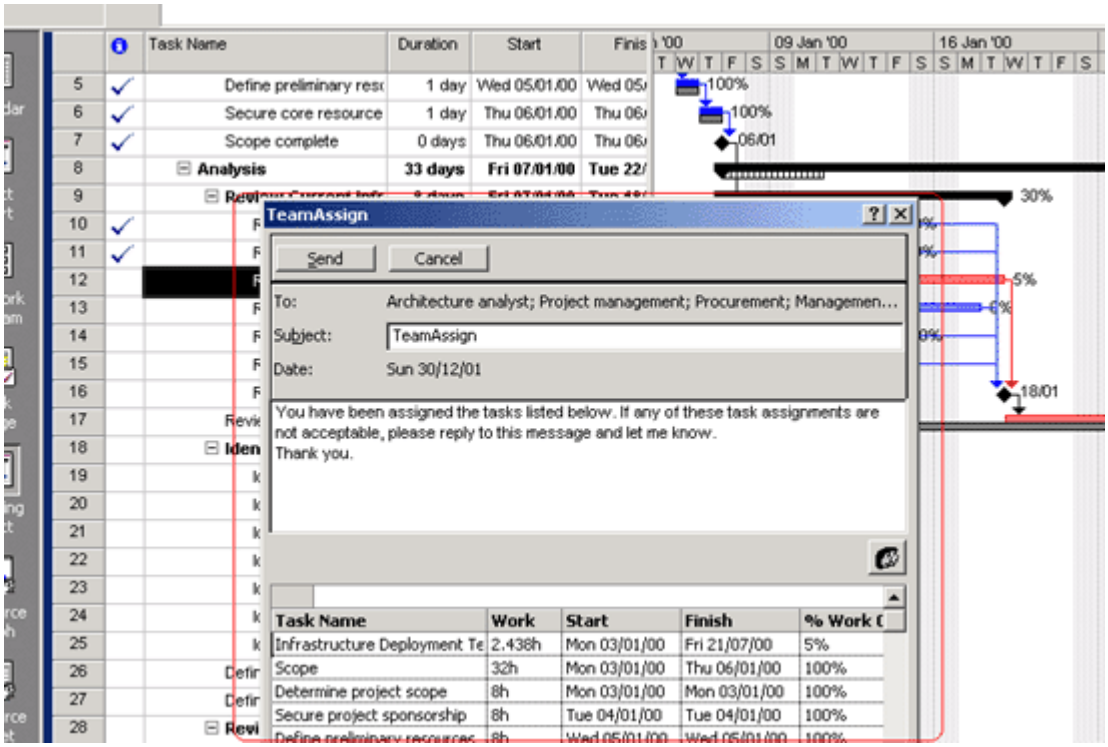


Diagrama Gantt de seguimiento y formulario de actualización de datos de una tarea

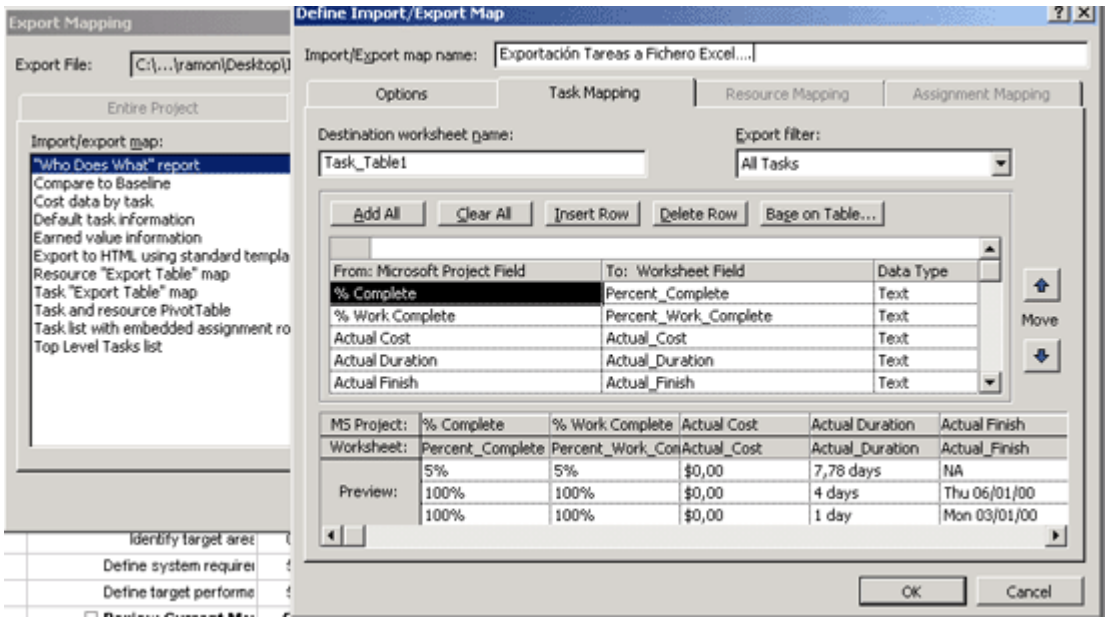
- **Trabajo y comunicación en equipo.** La mayoría de estos productos facilitan herramientas de comunicación, tanto por medio de la integración con los sistemas de mensajería de la compañía como de tecnologías Internet entre los miembros del equipo. De esta manera, la asignación de tareas a recursos, los cambios que se producen y el reporte individual de cada miembro al jefe de proyecto se pueden automatizar.



Envío de mensajes de asignación de tareas a recursos en un proyecto

- **Trabajo con múltiples proyectos.** El hecho de poder trabajar de forma simultánea con varios proyectos de la compañía permite analizar las cargas de trabajo de los recursos entre proyectos, así como definir posibles dependencias entre tareas de distintos proyectos.

- **Intercambio con otras aplicaciones.** Estas herramientas ofrecen distintas funcionalidades para importar y exportar información de los proyectos hacia otras aplicaciones y desde las mismas, como hojas de cálculo y bases de datos.



Configuración de generación de un fichero Excel a partir de un proyecto

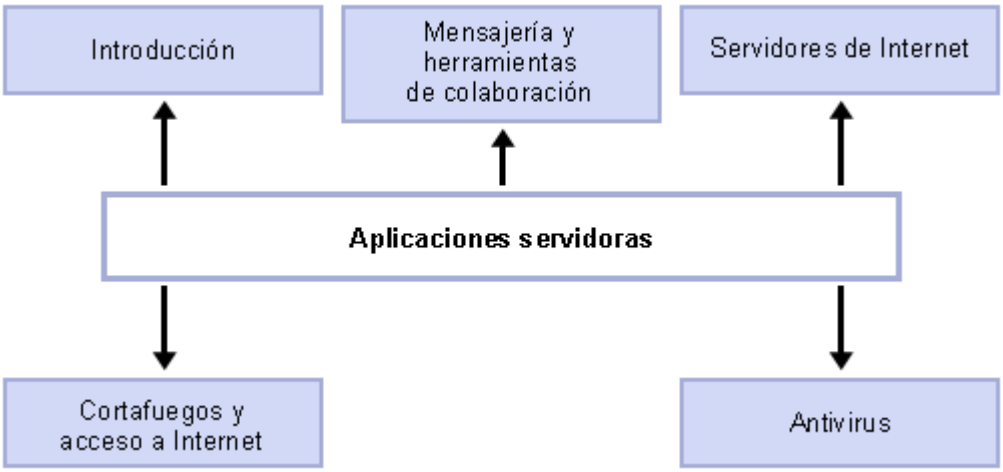
Un ejemplo de sistema gestor de proyectos es **Ms Project**.

Otras aplicaciones

Otro tipo de aplicaciones o herramientas de productividad pueden ser las siguientes:

- **Diseño gráfico**. Programas que hacen que los usuarios puedan editar e ilustrar sus documentos y páginas web con gráficos y fotografías. Los usuarios pueden crear gráficos personalizados para incorporarlos en otras herramientas como procesadores de texto u hojas de cálculo, en los que se añaden gráficos, ilustraciones, fotos retocadas y con efectos especiales. Ejemplos concretos de ello son Adobe Photoshop, 3D Studio, Macromedia Director.
- **Gestores de *web sites***. Este tipo de aplicaciones ofrecen a los usuarios las funcionalidades de creación de sitios web permitiendo definir su estructura y crear las páginas e incorporarlas a éste, añadiendo el contenido y elementos necesarios (texto, imágenes, vídeos, hipervínculos, etc.). Unas de las herramientas más difundidas en el mercado son Ms FrontPage y Macromedia Dreamweaver.
- **Navegadores**. Para acceder a las páginas de hipertexto que se encuentran en la web, es necesario el uso de unos programas especiales: los navegadores. Los dos productos destacados en el mercado, en la actualidad, son Ms Explorer y Netscape Comunicator.
- **Utilidades**. Otro grupo de herramientas *software* de amplia difusión son las diferentes utilidades como las aplicaciones de compresión (Winzip y WinRAR), los programas de copias de seguridad (BackupXpress) y recuperación de información (Recover4all, Winrescue98), mantenimiento de disco (WashAndGo), mantenimiento de RAM (WimRAM-Booster 2000), programas para optimizar el acceso a Internet (Internet Watcher 2000, Acelérate 2000, Intelli-Dial Up, MODEM Booster) o herramientas de comunicación por medio de Internet (Ms NetMeeting, Netscape Conference, CoolTalk).

Aplicaciones servidoras



Introducción

En el apartado anterior hemos descrito un conjunto de *software* que se podría considerar de uso personal para los empleados de la compañías. Dichos programas facilitan el trabajo diario de cada uno de los diferentes miembros de la organización.

Existe otro conjunto de *software* llamado **aplicaciones servidoras** que, mediante el servidor de aplicaciones, permite el acceso y trabajo común de todos los empleados a bases de datos, Internet, correo electrónico, etc.

> **Las aplicaciones servidoras se instalan en uno o más ordenadores "centrales" denominados servidores, a cuyas prestaciones y funcionalidades acceden el resto de los ordenadores por medio de un *software* cliente.**

Un ejemplo de este tipo de servicios para **compartir** son los **ficheros** y las **impresoras**, servicios que los sistemas operativos ofrecen, así como las **copias de seguridad y recuperación** de información.

Podéis consultar el apartado "Sistemas operativos" del módulo "El ordenador, máquina informacional".

En este apartado describiremos tres tipos de aplicaciones servidoras: los **sistemas de mensajería y herramientas de colaboración**, los **servidores de Internet**, y los **cortafuegos** y el acceso a Internet. Un cuarto tipo de servidores son los **gestores de bases de datos**.

Podéis consultar el subapartado "Sistemas gestores de bases de datos" del apartado "Bases de datos" que se incluye en el módulo "El ordenador, máquina informacional".

Y ya por último este apartado presenta un tipo concreto de *software*, los **antivirus**, por el imprescindible e importante papel que juegan en las organizaciones.

Mensajería y herramientas de colaboración

Actualmente, otro de los servidores imprescindibles en cualquier compañía son los servidores de correo y herramientas de colaboración. Por una parte ofrecen las prestaciones de poder implementar en la organización un **sistema de correo interno** entre los empleados, **vincularlo a correos electrónicos** externos, así como permitir la **construcción de soluciones de colaboración**.

Las soluciones de colaboración mencionadas arriba son aplicaciones que permiten el **flujo de datos e información** entre los miembros de la empresa por medio del diseño y generación de **formularios**, la planificación de grupos, los **grupos de discusión**, la gestión de tareas, la **gestión documental** y las **conferencias en tiempo real**. Permiten generar desde *workflows* departamentales hasta aplicaciones de negocio corporativas.

Todos estos servidores posibilitan la interconexión de diferentes sistemas, lo que facilita la comunicación entre empresas de un mismo grupo con distintos sistemas de mensajería o entre sistemas heterogéneos dentro de una

misma organización.

Dos de los servidores de mayor implantación en las empresas son Ms Exchange Server y Lotus Notes.

Ejemplos del servidor de mensajería y de su correspondiente aplicación cliente

Servidores de Internet

Existe un conjunto de servidores Internet/intranet para las organizaciones que posibilita el **diseño y creación de un sitio web**, lo cual permite la fácil y rápida publicación y búsqueda de información.

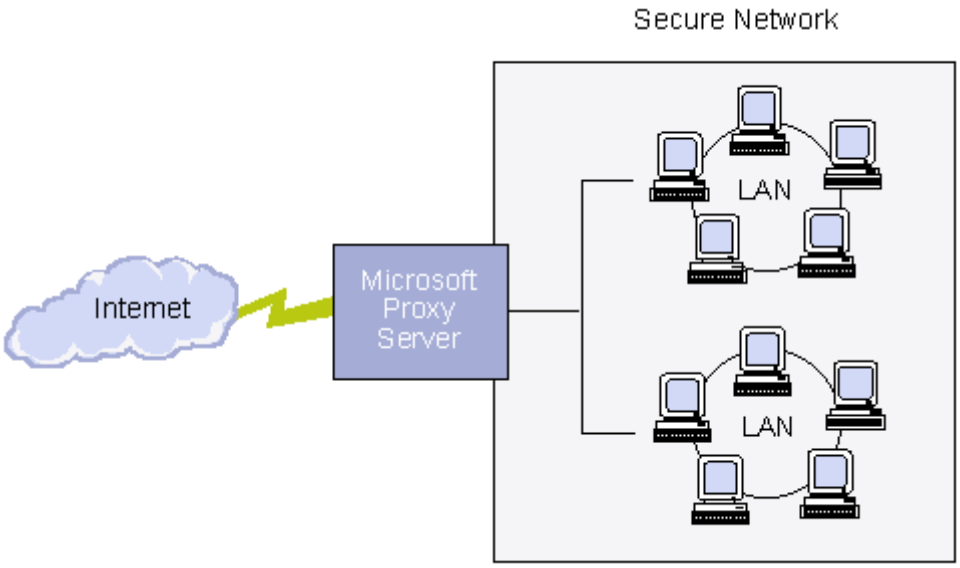
Muchos de estos servidores incorporan funcionalidades específicas que permiten a las organizaciones **captar clientes** y **realizar transacciones comerciales *on-line*** con clientes y proveedores. Algunas de las principales funcionalidades de estos servidores son las siguientes:

- **Gestión y publicación de contenido**. Permite la publicación de información, hecho que proporciona a los autores un conjunto de procesos de presentación y publicación y la posterior aprobación del contenido por parte de los editores del sitio web.
- **Búsqueda de información**. Muchos de estos servidores incorporan prestaciones que permiten buscar información entre los sitios web en Internet y en intranets, servidores de archivos, bases de datos o servidores de correo.
- **Personalización**. Una de las principales prestaciones de este tipo de servidores es la capacidad de personalizar las webs en función de los navegantes, con lo que se muestra diferente información según el visitante.
- **Análisis de la web**. Una de las características más importantes que debe tener un servidor Internet/intranet es la capacidad de analizar la información sobre la actividad de los sitios, permitiendo, así, que los administradores de la web lleven a cabo el diseño y generación de informes acerca de las páginas visitadas en una web, los caminos que siguen los navegantes, el origen de éstos, fechas y horas de acceso, etc.

En el mercado hay una gran variedad de servidores de Internet, entre los cuales destacan Ms Internet Information Server y Ms Site Server.

Cortafuegos y acceso a Internet

Uno de los servidores de mayor necesidad en toda organización para realizar las conexiones a Internet son los llamados **cortafuegos**. Este tipo de servidores proporciona la **seguridad necesaria en la conexión de la compañía a Internet, ya que bloquean los accesos hacia los sitios indexados y desde éstos**.



Arquitectura de uso de un servidor cortafuegos

Dichos servidores filtran dinámica e inteligentemente el tráfico de información para permitir y/o denegar las entradas y salidas desde la red de la empresa con Internet, aumentando, de ese modo, su seguridad.

Ocultan las direcciones de la red interna para evitar que los intrusos puedan atacar aquello que no pueden identificar. Incorporan **alertas** para saber inmediatamente si la red está siendo atacada y emprender las acciones oportunas.

La combinación de estos cortafuegos con los servidores de Internet también permite la **publicación de información en Internet de manera segura**. Asimismo, estos servidores facilitan que los servidores de correo electrónico conecten con la red Internet con seguridad. Mediante el filtro de sitios y control de acceso de usuarios podemos determinar los servicios de Internet y sitios web a los que pueden acceder los usuarios.

La generación de ficheros *log* (de seguimiento de actividad) ofrecen a los administradores toda la información del uso de Internet e intranet de los usuarios de la empresa.

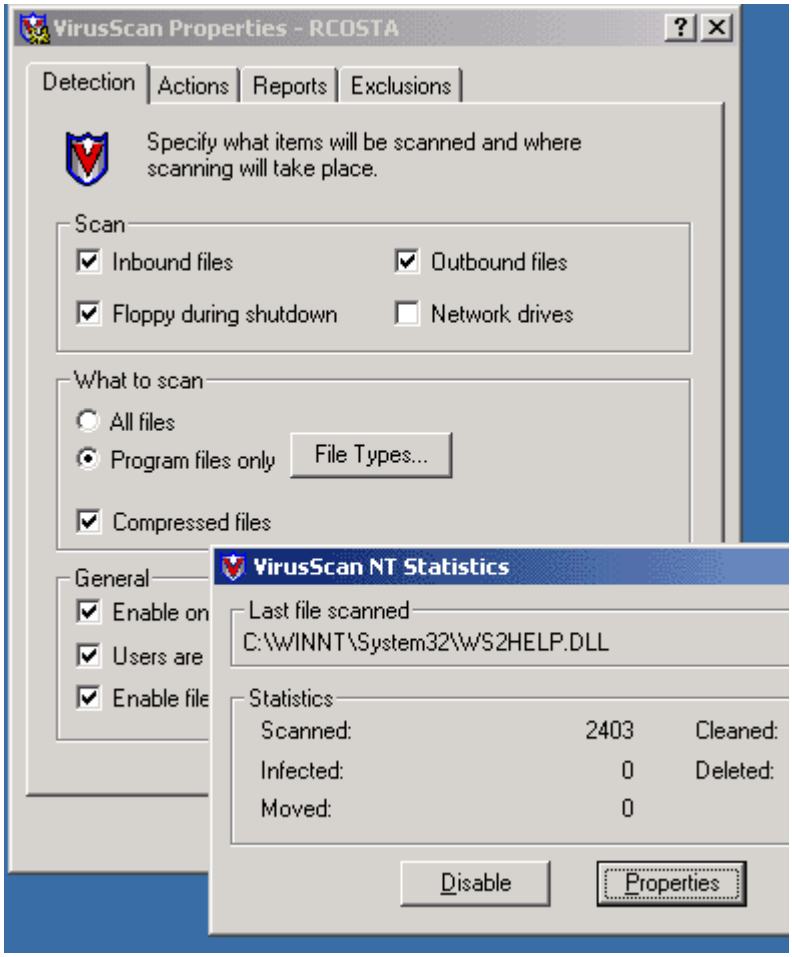
Por último, muchos de estos servidores disponen de funcionalidades para realizar **copias de seguridad y**

restauración en caso de fallo de los sistemas. Un ejemplo de servidor de cortafuegos es Ms Proxy Server.

Antivirus

Los antivirus son una utilidad que **busca** y **elimina los virus** que encuentra en los diferentes dispositivos de almacenamiento de la información: disquetes, discos duros, CD, etc.

Podéis consultar el núcleo de conocimiento "Dispositivos de almacenamiento de datos" del apartado "El almacenamiento externo de la información" del módulo "Gestión de datos".



Ejemplo de aplicación "Antivirus"

En la actualidad, la mayoría de los antivirus disponen de una propiedad que les permite actualizar esta información de forma automática por medio de Internet, ya que cuando el programa conecta con Internet, se dirige automáticamente a los servidores del fabricante para descargar la última versión. Esta versión se almacena en un servidor de la red interna, de manera que los diferentes ordenadores personales actualizan su versión de antivirus a partir de dicha nueva versión almacenada.

Tres de los antivirus más difundidos son McAfee Viruscan, Norton Antivirus y Panda Antivirus.

Leed el texto de Miquel Barceló: "Los virus informáticos".

Los "virus informáticos"

Pese a que, de hecho, podría parecer un fenómeno anecdótico, hay que mencionar, aunque sólo sea de paso, una curiosa "biologización" de ciertos hechos informáticos con los llamados "virus", tan difundidos hoy en día en la microinformática.

A partir de 1986 se ha dado una cierta proliferación y notoriedad de unos programas perturbadores del funcionamiento normal de un ordenador. Se trata de lo que genéricamente se denomina **virus**.

Cómo su homónimo biológico, un virus es un programa, a menudo pequeño, que se reproduce a sí mismo y es capaz de saltar de un sistema "infectado" a otro. La benignidad o malignidad de la infección depende de lo que haga el virus cuando se activa.

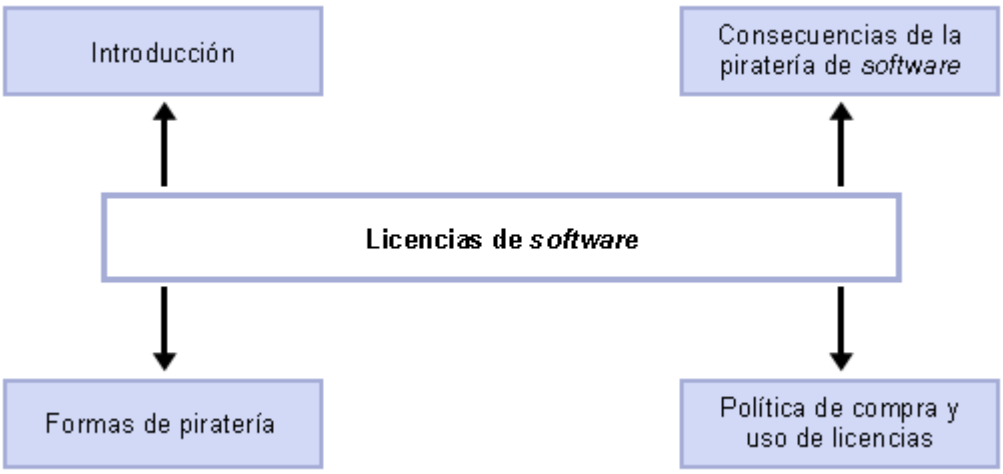
En ciertos casos pueden ser perturbaciones leves (bolas móviles en las pantallas, distorsión de textos e imágenes, mensajes especiales o propagandísticos, etc.), pero en otros casos puede tratarse de perturbaciones graves, como borrar una parte o toda la información almacenada en las memorias de masa del ordenador: discos y disquetes. La proliferación actual de los virus resulta posible, por una parte, por la abundancia de microordenadores personales con una gran difusión del *software*, a menudo incontrolada y "pirata", y, por otra parte, por la existencia de redes telemáticas con controles de seguridad insuficientes sobre el acceso a la red y a sus ordenadores.

Muchas veces se confunden los términos y se llama *virus* a lo que sólo es un **gusano** (*worm*). Los expertos separan claramente estos dos conceptos: un *worm* se ejecuta por sí mismo y se propaga con la simple duplicación de sus copias, mientras que un **virus** es un programa que no puede "vivir" solo y tiene que hacer de parásito incorporándose a otros programas, y, por tanto, sólo se activa cuando lo hace el programa donde se aloja.

Parece que todo empezó con el **Creeper** (‘enredadera’) que Bob Thomas, del BBN (*Bulletin Board Network*, red de tablones de anuncios), diseñó en 1970 como un programa de demostración. El programa no funcionó bien y se difundió y multiplicó por la red ARPANET. Este programa dejaba un mensaje allá donde iba que decía "I’m the Creeper, catch me if you can" (‘soy la enredadera, agárrame si puedes’) y no hacía nada más. Para eliminarlo de la red se utilizó un nuevo virus, el programa Reaper (un "segador" para eliminar la "enredadera"), que buscaba copias del *Creeper* en los ordenadores de la red y las borraba. De hecho, más que de virus se trataba de gusanos.

Otros virus posteriores no han sido tan pacíficos y se han difundido sobre todo en el mundo de los microordenadores personales. Son especialmente famosos el **pakistani brain** (1985) o el **Friday 13** (1987), que han llegado incluso a atemorizar al gran público. Pero los virus que se difunden por las redes de ordenadores siguen siendo los que pueden tener más consecuencias, debido al alcance que tiene la "infección". Algunos de los más recientes, y que el gran público ha conocido bastante, son el **Christmas**, difundido a finales de 1987 por un estudiante alemán en la red interna de IBM, y un programa escrito por **Robert Morris Jr.**, difundido el 2 de noviembre de 1988 por la red norteamericana Internet. Aunque es habitual llamarlos *virus*, ambos corresponden con más exactitud al tipo "gusano".

Licencia de software



Introducción

La **compra de una licencia** de un producto supone para el comprador el **derecho** real de **instalar y ejecutar un programa**.

Una licencia de *software* se aplica a cualquier tipo de *software* o programa: sistemas operativos, herramientas de productividad personal, antivirus, servidores corporativos, etc.

De manera simplificada, estas licencias se pueden clasificar en tres tipos:

- Personal
- Servidor
- Cliente

Una **licencia de *software* personal** supone el derecho de instalar una aplicación *software* en un ordenador personal; por ejemplo, la licencia de uso de un procesador de textos.

Podéis consultar el apartado "Herramientas de escritorio" de este mismo módulo.

Una **licencia de *software* servidor** corresponde al derecho de instalar y poder utilizar un *software* servidor de bases de datos, correo, Internet, etc. En este último caso, la licencia de *software* servidor se completa con la adquisición de **las licencias cliente**, una para cada ordenador que accederá y trabajará con el *software* servidor. Por ejemplo, una vez instalado el servidor de correo electrónico, la empresa debe adquirir una licencia cliente para cada usuario que dispondrá de acceso al mismo.

Podéis consultar el apartado "Aplicaciones servidoras" de este mismo módulo.

El **uso ilegal** de un programa *software* implica la comisión de un **delito** y está penalizado como tal. Existen varias organizaciones nacionales e internacionales encargadas de perseguir la piratería informática en las organizaciones y compañías.

Consecuencias de la piratería de *software*

Muchas personas creen que la piratería de *software* sólo afecta a la industria informática. Sin embargo, si tenemos en cuenta, por ejemplo, que el 57% del *software* utilizado en España ha sido copiado de forma ilegal, podremos apreciar el alcance real del problema.

En el año 2000, según Microsoft Ibérica, la piratería costó más de 37.000 millones de pesetas a la economía española en ingresos que las empresa de *software* dejaron de percibir, con la consiguiente pérdida de recaudación tributaria y la pérdida de más de 9.000 puestos de trabajo en el sector.

La utilización del *software* ilegal también comporta costes adicionales a las organizaciones que lo utilizan, como, por ejemplo, falta de soporte técnico, de todas las garantías posventa que ofrece el fabricante en una compra legal, el acceso a programas completos y libres de virus, acceso a las actualizaciones y adaptaciones, etc.

Asimismo, la violación del Código penal expone a las compañías a varias acciones judiciales y al desprestigio y pérdida de su imagen.

Formas de piratería

Hay diferentes formas de piratería, pero las **cuatro formas de piratería** más habituales son las siguientes:

- **Copias que realiza el usuario final**. Son simples copias sin licencia que hacen los usuarios individuales o empresas. También se pueden considerar, en el caso de la compra de licencias por volumen, una información incorrecta en torno al número de ordenadores en los que se ha instalado el *software*.
- **Carga en el disco duro**. Este tipo de piratería lo practican fabricantes deshonestos de sistemas de PC que comercializan los ordenadores con *software* ilegal preinstalado.

- **Falsificaciones**. Es la piratería de *software* a gran escala. Este tipo de piratería duplica ilegalmente el *software* y sus cajas, con frecuencia mediante redes de crimen organizado, que se distribuyen como productos supuestamente legales.
- **Distribución fraudulenta**. Corresponde al *software* distribuido como licencias con un descuento especial a clientes con un gran volumen, a fabricantes de ordenadores o a entidades académicas. Este *software* se redistribuye a otros usuarios que no disponen o no reúnen los requisitos necesarios para contar con tales licencias.

Política de compra y uso de licencias

Cada fabricante tiene una política de venta de licencias concreta. Estas políticas, además, evolucionan con el tiempo y se van adaptando a las nuevas características de las tecnologías de la información. Por ejemplo, la aparición de aplicaciones de Internet supuso la aparición de nuevas políticas de compra y uso de licencias.

Tras asumir la necesidad de comprar un programa, hay que seguir tres pasos básicos para adquirir las licencias de *software*:

- **determinar la licencia necesaria;**
- **decidir la mejor opción de compra o *leasing*;**
- **determinar dónde se realizará la compra.**

Una licencia de un producto de *software* se puede categorizar por cinco elementos: el tipo de producto, el producto, la versión, la edición y el tipo de licencia:

- **Tipo de producto**. Por ejemplo, los programas de *software* de Microsoft se clasifican en uno de los siguientes tres tipos:
  - **Aplicaciones**. Programas genéricos de productividad personal, utilidades o herramientas de desarrollo, como por ejemplo Ms Word, Ms Visio, Ms Project o Ms Visual Studio.
  - **Sistemas**. Corresponde a los sistemas operativos como, por ejemplo, Ms Windows NT, Ms Windows 2000 o Ms Windows XP.
  - **Servidores**. Engloba todas las aplicaciones corporativas o servidores corporativos tales como los servidores de bases de datos, servidores de Internet y de correo, entre otros. Ejemplos concretos serían Ms Exchange Server, Ms SQL Server, etc.
- **El producto**. El *software* en sí; por ejemplo: Ms Office, Ms Visual Studio o Ms SQL Server.
- **Versión**. Un mismo producto puede tener diferentes versiones, como en el caso de la *suite* de herramientas de productividad de Microsoft, Ms Office puede ser la versión 2000 o la versión XP (posterior a la primera).
- **Edición**. Especifica el conjunto de prestaciones y/o aplicaciones incluidas en el producto. Un producto puede tener, por ejemplo, las versiones estándar, profesional y experto. De esta forma, la versión estándar de Office 2000 incluye Ms Word, Ms Excel, Ms Outlook y Ms PowerPoint, mientras que la versión profesional incluye, además, Ms Publisher, Ms Small Business Tools y Ms Access.
- **Tipo de licencia**. En función del fabricante puede haber diferentes tipos de licencia: la **licencia de nuevo usuario**, que corresponde a la compra inicial del producto y supone el derecho a trabajar con una versión concreta del producto; la **actualización de versión**, correspondiente a la compra de una versión posterior del producto, y que posee la licencia de uso de una versión anterior del mismo producto; la **actualización competitiva**, que corresponde a la compra de una licencia a un coste menor, y que dispone de una licencia de un producto relacionado de otro fabricante.

Tal como mencionábamos antes, este tipo descrito de licencias es sólo un ejemplo concreto, puesto que **cada fabricante tiene su propia y específica política de compra y uso de licencias de *software***. Algunos fabricantes ofrecen la posibilidad de realizar un *leasing* del *software* en lugar de comprarlo, mientras que otros tienen una política de venta relacionada con el uso de la herramienta por parte del cliente en la que, por ejemplo, cobran una comisión por transacción realizada con el producto.

Este último tipo de compra de licencias puede corresponder a sistemas de comercio electrónico en los que el comprador abona un importe base por el producto y una comisión, por cada transacción realizada por medio del mismo.

Por último, hay que decidir la mejor opción en cuanto a **dónde comprar el *software***:

- ***Original Equipment Manufacturer* (OEM)**. Corresponde a aquel *software* que va incluido en la compra de un nuevo ordenador o equipo informático. Por ejemplo, al comprar un ordenador personal, el fabricante suele incluir una licencia de un sistema operativo y de un conjunto de aplicaciones de productividad: procesador de texto, hoja de cálculo, etc.
- **Compra del paquete**. Corresponde a la compra física de la caja con el producto en cualquier tienda o almacén. Incluye el producto en forma magnética, por ejemplo en CD, junto con la licencia correspondiente.
- **Licencia por volumen para organizaciones**. Este tipo de compra corresponde a las que realizan las compañías que necesitan una gran cantidad de licencias del mismo *software*. Por ejemplo, una organización de cien empleados quizá necesite adquirir sesenta licencias de un procesador de textos. En lugar de adquirir los sesenta CD del producto, se adquieren uno o dos ejemplares del mismo y las sesenta licencias (el derecho a instalar el producto en sesenta máquinas). De este modo se reduce el coste del producto y los costes de envío para el comprador. En estos casos, se aplican diferentes políticas de precio en función del número de licencias adquiridas o del tipo de organización. Las instituciones académicas, por ejemplo, cuentan con un tipo concreto de coste por compra de *software*.

Actividades

- **Actividad 1**

  Acceded a la web del fabricante Microsoft, consultad la información de las últimas versiones de los programas Ms Word y Ms PowerPoint y elaborad una lista de las diez mejoras que aporta con respecto a la última versión para cada uno de ellos.

- **Actividad 2**

  Buscad en Internet información sobre dos productos de gestión de proyectos que trabajen como soluciones Internet. Es decir, dos programas que tengan las prestaciones de poder publicar los proyectos en un *site* Internet y permitan a los usuarios y miembros del equipo de trabajo acceder a dicha información por medio de navegadores.

- **Actividad 3**

  Mediante la consulta de la web de Microsoft, elaborad un documento de dos páginas que describa las funcionalidades y manejo de trabajo del programa de edición de páginas web Ms FrontPage.

- **Actividad 4**

  Nuestra compañía ha adquirido recientemente el siguiente conjunto de programas *software* de utilidades: 3D Studio, Ms Internet Explorer, Ms FrontPage, BackupXpress, WinRAM-Booster 2000, Ms NetMeeting, Recover4all Internet y Watcher 2000.

  Se os pide que identifiquéis, de la siguiente lista de acciones, qué podrán hacer los miembros de la compañía con cada una de ellas:

  - Retocar imágenes y fotos.
  - Recuperar de espacio libre del disco.
  - Diseñar animaciones.
  - Establecer conversaciones virtuales (*chats*) por medio de Internet.
  - Editar vídeo.
  - Visitar páginas web.
  - Diseñar páginas web.
  - Definir la estructura de un *site*.
  - Acelerar la bajada de ficheros desde los *sites* de Internet, inclusión de imágenes y animaciones en una web.
  - Definir y crear copias de datos.

- Recuperar datos a partir de copias de seguridad.
- Recuperar archivos borrados en el disco duro.
- Borrar archivos temporales.
- Aumentar la capacidad de memoria que se asigna a las aplicaciones.
- Generar ficheros comprimidos.

- **Actividad 5**

  Consultad, mediante la web de Microsoft y de Lotus (IBM), información acerca de los dos servidores de mensajería y colaboración, Ms Exchange y Lotus Domino. Elaborad una lista de diez prestaciones y funcionalidades de cada uno.

- **Actividad 6**

  Nuestra organización ha decidido protegerse del posible ataque de virus, gusanos y caballos de Troya. Para ello, nos han pedido que analicemos posibles soluciones de antivirus del mercado. Consultad la información de dos de los tres productos de antivirus mencionados en el módulo y elaborad una lista de diez características de cada uno de ellos.

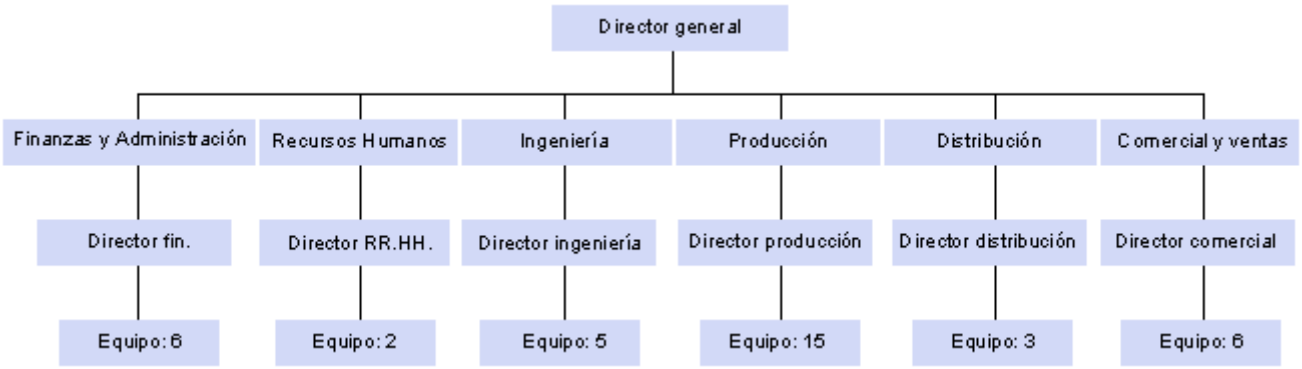
- **Actividad 7**

  Acceded al sitio web de la Alianza de *software* empresaria, www.bsa.org (www.bsa.org/es), y enumerad las principales actividades que lleva a cabo dicha organización en su lucha contra la piratería del *software*.

- **Actividad 8**

  Tras haber decidido todo el *software* necesario para nuestra empresa (el sistema operativo XOP, el procesador de textos PTX, la hoja de cálculo HJY, el servidor de mensajería SMZ, el gestor de proyecto PME y el programa de antivirus AVS), hay que proceder a comprarlo.

  Para cumplir con la legalidad y según la información trabajada en el módulo, identificad las licencias de servidor y cliente necesarias dada la siguiente estructura organizativa y las necesidades funcionales de la empresa.



  - Con el fin de optimizar el acceso a los ficheros e impresoras, se ha decidido disponer de un servidor para ficheros e impresoras y de otro, para el correo electrónico.
  - Todos los empleados de la compañía tendrán acceso al servidor de ficheros e impresoras.
  - Todos los miembros de la organización dispondrán de correo electrónico.
  - Sólo los miembros del equipo de ingeniería de la compañía y los directores de producción y distribución tienen que gestionar proyectos.
  - Todos los responsables de cada departamento y los miembros de los departamentos de finanzas y RR.HH. necesitan trabajar con el procesador de textos y el gestor de hojas de cálculo.