20
24.

# UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES
"A C A T L A N"

## SELECCION DE HERRAMIENTAS CASE DE ACUERDO A LAS NECESIDADES DE UNA INSTITUCION

**T E S I S**

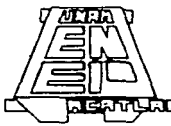
QUE PARA OBTENER EL GRADO DE:

**LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION**

P R E S E N T A :

**SERGIO FRAGOSO DE LA TORRE**

ASESOR: LIC. SARA CAMACHO CANCINO



ACATLAN, EDO. DE MEXICO. 1997

UNIVERSIDAD NACIONAL AVENMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES "ACATLÁN"

COORDINACIÓN DEL PROGRAMA DE MATEMÁTICAS APLICADAS Y COMPUTACIÓN

SR. SERGIO FRAGOSO DE LA TORRE
Alumno de la carrera de Matemáticas Aplicadas y Computación
P r e s e n t e.

De acuerdo a su solicitud presentada con fecha 14 de mayo de 1996, me complace informarle que esta Coordinación tuvo a bien asignarle el siguiente tema de tesis: "Selección de herramientas CASE de acuerdo a las necesidades de una institución", el cual desarrollará como sigue:

Introducción
I. Principales metodologías para el análisis y diseño de sistemas
II. ¿Qué son las herramientas CASE ?, y sus perspectivas
III. Beneficios de una herramienta CASE
IV. Parámetros para la selección de una herramienta CASE en función de las necesidades de una institución
Conclusiones

Asimismo fué designado como Asesora de la Tesis la Lic. Sara Camacho Cancino, profesora de esta Escuela.

Ruego a usted tomar nota que en cumplimiento de lo especificado en la Ley de Profesiones, deberán prestar servicio social durante un tiempo mínimo de seis meses como requisito básico para sustentar el examen profesional, así como de la disposición de la Coordinación de la Administración Escolar en el sentido de que se imprime en lugar visible de los ejemplares de la tesis el título del trabajo realizado. Esta comunicación deberá imprimirse en el interior de la tesis.

A t e n t a m e n t e
"POR MI RAZA HABLARÁ EL ESPÍRITU"
Acatlán, Edo. de Méx., a 6 de mayo de 1997.

LIC. BEATRIZ TRUEBA RÍOS
Jefe del Programa de M.A.C.

E.N.E.P. ACATLAN

JEFATURA DEL PROGRAMA DE MATEMATICAS APLICADAS Y COMPUTACION

# SELECCIÓN DE UNA HERRAMIENTA CASE PARA UNA EMPRESA

*A mis padres:*
*gracias por su comprensión y apoyo,*
*sin los cuales no hubiera sido posible*
*la elaboración de este trabajo.*

SELECCION DE UNA HERRAMIENTA CASE PARA UNA EMPRESA

## CAPITULO II : DESCRIPCIÓN DE LA TECNOLOGÍA CASE

CAPITULO III : ADOPCIÓN DE UNA HERRAMIENTA CASE

CAPITULO IV : CATÁLOGO DE HERRAMIENTAS CASE

# INTRODUCCIÓN

El análisis y diseño de sistemas computarizados aplicado a las organizaciones es un campo que toma cada vez mayor importancia, ya que a medida que se incrementa el uso de computadoras dentro de las empresas, surgen muchas inquietudes acerca de cómo utilizarlas para mejorar la productividad y lograr satisfacer exitosamente los objetivos de la organización.

Sin embargo para la gente encargada del desarrollo de sistemas, la complejidad que causa el tamaño, tiempo o costo de éstos hace necesario que se provea de herramientas de ayuda que les permitan desempeñar eficientemente su función. Es por esto que se buscan alternativas para agilizar la producción de software de alta calidad en las empresas, una de estas nuevas opciones es la ingeniería de software asistida por computadora CASE (por sus siglas en inglés).

Este trabajo no sólo pretende mostrar los aspectos más importantes de la tecnología CASE como lo son; sus características, clasificación, evolución, funciones, o metodologías más apropiadas para su intervención, si no proporcionar una serie de parámetros para la adopción correcta y la implantación exitosa de una herramienta dentro de cualquier empresa.

El estudio se encuentra estructurado en cuatro capítulos cuyos objetivos son:
Capítulo I, introducir al análisis y diseño de sistemas, describiendo las actividades de tres metodologías de desarrollo.
Capítulo II, describir las herramientas CASE, sus características y perspectivas dentro de la ingeniería de software.
Capítulo III, mostrar los beneficios que ofrece el uso de herramientas CASE, además de proporcionar algunos parámetros que deben considerarse en la selección y evaluación de una herramienta CASE.
Capítulo IV, aportar un catálogo donde se muestren diferentes productos CASE existentes en el mercado, con el fin de informar a los interesados sobre herramientas, características y distribuidores.

Durante la elaboración del trabajo se presentaron algunos obstáculos (como es normal en el desarrollo de una tesis) estos fueron; el encontrar la persona indicada para el apoyo de la investigación, la delimitación del tema pero, sin duda el más importante fue reunir el material de información que se requería, ya que en México no se ha difundido lo suficiente el desempeño de la tecnología CASE, por este motivo se tuvo que recurrir en parte a bibliografía procedente de E.U. y España para solucionar el problema.

Finalmente deseamos que el siguiente estudio sirva para animar a la gente involucrada en el desarrollo de sistemas de información así cómo a la que se encuentra relacionada con el ámbito de la computación para que, investigue, se relacione y adopte la tecnología CASE.

# CAPITULO I:

## Metodologías de desarrollo la base para la tecnología CASE

Para tener una mayor visión del campo que las herramientas CASE pretenden mejorar es necesario realizar un estudio de diferentes metodologías de desarrollo de sistemas como lo son; el ciclo de vida para el desarrollo de sistemas, el análisis estructurado y la estrategia por prototipos. Dentro del estudio de las metodologías se analizan sus fases, elementos, objetivos y lineamientos.

## ¿ QUÉ ES EL ANÁLISIS Y DISEÑO DE SISTEMAS ?

El análisis de sistemas puede llegar a ser una de las más complejas actividades humanas, lo puede ser intelectualmente (en el orden de entender un sistema y organizarlo conceptualmente), pero también lo puede ser en la práctica (en el orden para coordinar y organizar los esfuerzos e interacciones de los participantes en las actividades del análisis). Cuando analizamos sistemas en general y sistemas de información computacional en particular, necesitamos competir con la complejidad inherente en nuestro mundo no obstante nuestras limitaciones humanas.

***Dentro de las organizaciones el análisis y diseño de sistemas se refiere al proceso de examinar la situación de una institución con el propósito de mejorarla con métodos y procedimientos más adecuados.*** [1]

El desarrollo de sistemas puede considerarse, en general formado por dos grandes componentes; el análisis de sistemas y el diseño de sistemas. El diseño de sistemas es el proceso de planificar, reemplazar o completar un sistema organizacional existente. Pero antes de llevar a cabo esta planeación es necesario comprender, en su totalidad, el viejo sistema y determinar la mejor forma en que se pueden si es posible, utilizar las computadoras para hacer la operación más eficiente. El análisis de sistemas, por consiguiente, es el proceso de clasificación e interpretación de hechos, diagnóstico de problemas y empleo de la información para recomendar mejoras al sistema.

***El análisis especifica qué es lo que el sistema debe hacer. El diseño establece como alcanzar el objetivo.*** [2]

## ¿ QUÉ ES UN SISTEMA ?

**En el sentido más amplio, un sistema es un conjunto de componentes que interaccionan entre sí para lograr un objetivo común.** Nuestra sociedad está formada por sistemas.

---

1 El análisis y el diseño son las dos actividades que soportan todo el peso del desarrollo de sistemas de información

2 TEAGUE y PIDGEON Structured analysis methods for computer information systems.

Una organización es un sistema. Sus componentes (mercadotecnia, manufactura, ventas, investigación, embarques, contabilidad y personal) trabajan juntos para crear utilidades que beneficien tanto a los empleados como a los accionistas de la compañía.

Todo sistema organizacional depende en mayor o menor medida, de una entidad abstracta llamada sistema de información. Este sistema es el medio por el cual los datos fluyen de una persona o entidad hacia otras y pueden ser cualquier cosa.

Los sistemas de información proporcionan servicios a todos los demás sistemas de una organización y enlazan todos sus componentes en forma tal que trabajen con eficiencia para alcanzar el mismo objetivo. Para lograr sus objetivos, los sistemas interactúan con su medio ambiente, que está formado por todos los objetos que se encuentran fuera de las fronteras de los sistemas.

Todos los sistemas tienen niveles aceptables de desempeño, denominados estándares y contra los que se comparan los niveles de desempeño actuales.

La información proporcionada al comparar los resultados con los estándares junto con el proceso de reportar las diferencias a los elementos de control recibe el nombre de retroalimentación.

## MODELADO Y REPRESENTACION

El resultado de la descripción del problema es usualmente un modelo, manipulado a través de una o más representaciones. **El propósito básico de un modelo es poder entender el sistema.** Los buenos modelos soportan el proceso de análisis y diseño de sistemas, además deben ser fáciles de manipular y cambiar siendo transparentes hacia los usuarios.[1]

La representación de un modelo es de forma gráfica o física para demostrar los componentes y relaciones en el modelo, algunas veces el modelo puede ser presentado de una o más formas.

---

1 La elaboración de modelos puede resultar una tarea difícil, sin embargo, una vez que el modelo es terminado, se reducen considerablemente los tiempos de generación del sistema

Los sistemas emplean un modelo de control básico consistente en:

1 .- Un estándar para lograr un desempeño aceptable.

2 .- Un método para medir el desempeño actual.

3 .- Un medio para comparar el desempeño actual con el estándar.

4 .- Un método de retroalimentación.

***Los sistemas que pueden ajustar sus actividades para mantener niveles aceptables continúan funcionando. Aquellos que no lo hacen tarde o temprano dejan de trabajar.***[4]

## ESTRATEGIAS PARA EL DESARROLLO DE SISTEMAS.

Los sistemas de información basados en computadora sirven para diversas finalidades que van desde el procesamiento de las transacciones de una empresa, hasta el proveer información necesaria para decidir sobre asuntos que se presentan con frecuencia, asistencia a los altos funcionarios con la formulación de estrategias difíciles y la vinculación entre la información de las oficinas y los datos de toda la corporación. En algunos casos los factores que deben considerarse en un proyecto de sistemas de información, tales como el aspecto más apropiado de la computadora o la tecnología de comunicaciones que se va a utilizar, el impacto del nuevo sistema sobre los empleados de la empresa y las características específicas que el sistema debe tener, se pueden determinar de una manera secuencial. En otros casos, debe ganarse experiencia por medio de la experimentación conforme el sistema evoluciona por etapas.

A medida que las computadoras se emplean cada vez más por personas que no son especialistas en computación, el rostro del desarrollo de sistemas de información adquiere una nueva magnitud. Los propios usuarios emprenden ya el desarrollo de algunos de los sistemas que ellos emplean, esto ocasiona el buscar métodos eficientes para su desarrollo.

[4] TEAGUE y PIDGEON Structured analysis methods for computer information systems.

Antes de iniciar el estudio de distintos enfoques para el desarrollo de sistemas de información, se definirán los términos de técnica y metodología para elegir cual de ellos se estudiará.

**Metodología**.- Conjunto de reglas que deben seguirse para el estudio de una ciencia o problema.

**Técnica** .- Conjunto de procedimientos que se han sistematizado racionalmente para la ayuda en la resolución de problemas.

Con esto observamos que la metodología ofrece un marco más amplio de estudio, ya que se vale de reglas bien establecidas para atacar la solución de problemas, mientras que la técnica provee procedimientos que apoyan a estas reglas en la resolución de los mismos. Tomando en cuenta lo anterior se optó por enfocar el capítulo en la descripción de tres de las metodologías más mencionadas en el estudio del desarrollo de sistemas,[5] éstas son:

1.- Metodología del ciclo de vida para el desarrollo de sistemas.

2.- Metodología de desarrollo de análisis estructurado.

3.- Metodología de desarrollo por prototipos de sistemas.

[5] Esto se investigó en varios libros de análisis y diseño de sistemas

## 1.- CICLO DE VIDA DE DESARROLLO PARA SISTEMAS DE INFORMACIÓN

Es el conjunto de actividades que los analistas, diseñadores y usuarios realizan para desarrollar e implantar un sistema de información. **Este provee una estructura conceptual para presentar y entender las actividades envueltas durante el proceso de desarrollo del sistema figura (1.1).**[6]

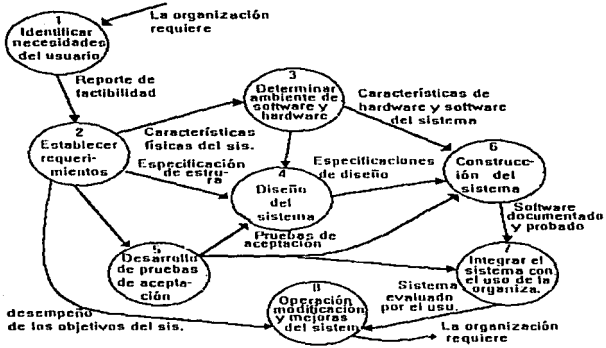



**Figura 1.1. El ciclo de vida de los sistemas de información**

Dentro de una empresa todas las actividades están muy relacionadas, en general son inseparables, y quizá sea difícil determinar el orden de los pasos que se siguen para efectuarlas. Las diversas partes del proyecto pueden encontrarse al mismo tiempo en distintas fases de desarrollo; algunos componentes en la fase de análisis mientras que otros en etapas avanzadas de diseño.

6 Otra característica importante del ciclo de vida es aportar un esquema para el manejo y coordinación del desarrollo del sistema, además de monitorear su progreso

Los expertos no están muy de acuerdo respecto al número exacto de las etapas que conforman el ciclo de desarrollo de los sistemas; sin embargo por lo general, se reconoce la importancia de su enfoque sistemático. ***Aquí dividimos el ciclo en ocho etapas.***[7]

1. - Identificar las necesidades del usuario.
2. - Establecer los requerimientos del usuario
3. - Determinar el ambiente de Hardware y Software del sistema.
4. - Diseño del sistema.
5. - Desarrollo de las pruebas de aceptación del sistema.
6. - Desarrollo del sistema.
7. - Integración del sistema con el uso en la organización.
8. - Operación, modificación y mejoramiento del sistema.

Aunque cada fase se presenta de manera discreta, nunca se lleva a cabo como un elemento independiente. En lugar de ello, se realizan al mismo tiempo diversas actividades, y éstas llegan a repetirse. Por ello es de mayor utilidad suponer que el ciclo de desarrollo transcurre en etapas (con actividades en plena acción que luego cesan poco a poco) y no como elementos separados.

**1.- Identificar las necesidades del usuario**

En esta primera etapa del ciclo de desarrollo de los sistemas, **el analista se involucra en la investigación de los problemas, de las oportunidades y de los objetivos y, toma el requerimiento de un usuario para un nuevo sistema.** Esta fase es crucial para el éxito del resto del proyecto, pues nadie estará dispuesto a desperdiciar su tiempo dedicándolo al problema equivocado.

La primera etapa requiere que el analista observe de forma objetiva lo que ocurre en una empresa. Luego, en conjunto con los otros miembros de la organización hará notar los problemas. La identificación de objetivos también es un componente importante de la primera fase. En primera instancia, el analista deberá descubrir lo que la empresa intenta realizar. Y luego, estará en posibilidad de determinar si el uso de los sistemas de

7 TEAGUE y PIDGEON Structured analysis methods for computer information systems.

información apoyaría a la empresa para alcanzar sus metas, el encaminarla a problemas u oportunidades específicas, preparando un informe explícito de los objetivos del sistema, alcances y funciones.[8]

Esta etapa tiene tres partes: aclaración de la solicitud, estudio de factibilidad y aprobación de la solicitud.[9]

Aclaración de la solicitud. Muchas solicitudes que provienen de empleados y usuarios no están formuladas de manera clara. Por consiguiente, antes de considerar cualquier investigación de sistemas, la solicitud del proyecto debe examinarse para determinar con precisión lo que el solicitante desea, por consiguiente, antes de seguir adelante, la solicitud de proyecto debe estar claramente planteada.

Estudio de factibilidad. Un resultado importante de la investigación preliminar es la determinación de que el sistema solicitado sea factible. El estudio de factibilidad lo lleva a cabo un pequeño equipo de personas que está familiarizado con técnicas de sistemas de información; dicho equipo comprende la parte de la empresa u organización que participará o se verá afectada por el proyecto, y es gente experta en los procesos de análisis y diseños de sistemas.

Aprobación de la solicitud. No todos los proyectos solicitados son deseables o factibles. Algunas organizaciones reciben tantas solicitudes de sus empleados que sólo es posible atender unas cuantas. Sin embargo, aquellos proyectos que son deseables y factibles deben incorporarse en los planes. Cuando esto ocurre, la administración decide qué proyectos son los más importantes y deciden el orden en que se llevarán a cabo. Muchas organizaciones desarrollan sus planes para sistemas de información con el mismo cuidado con el que planifican nuevos productos y programas de fabricación o la expansión de sus instalaciones. Después de aprobar la solicitud de un proyecto se estima su costo, el tiempo necesario para terminarlo y las necesidades de personal; con esta información se determina dónde ubicarlo dentro de la lista existente de proyectos.

---

8 Es aquí donde la habilidad del analista para identificar los requerimientos de información juega un papel sumamente importante

9 SEEN, James A. Análisis y diseño de sistemas de información.

**2.- Establecer los requerimientos del usuario**

Para identificar los requerimientos del usuario dentro de la empresa, pueden utilizarse diversos instrumentos, los cuales incluyen: el muestreo, el estudio de datos y formas usadas por la organización, la entrevista, los cuestionarios la observación de la conducta de quien toma las decisiones, así como de su ambiente; y también el desarrollo de prototipos.

En esta etapa **el analista hace todo lo posible por identificar qué información requiere el usuario para desempeñar sus tareas.** Puede ver cómo varios de los métodos para establecer las necesidades de información, lo obligan a relacionarse directamente con los usuarios. Esta etapa sirve para elaborar la imagen que el analista tiene de la organización y de sus objetivos

Conforme se reúnen los detalles, los analistas estudian los datos sobre requerimientos con la finalidad de identificar las características que debe tener el nuevo sistema, incluyendo la información que deben producir los sistemas junto con características operacionales tales como controles de procesamiento, tiempos de respuesta y métodos de entrada y salida.[10]

**3.- Determinar el ambiente de hardware y software del sistema**

**El ambiente de software del sistema puede incluir más que el sistema operativo,** este puede también incluir software para el manejo de la base de datos, reportes de la misma, creación y soporte de gráficas y otros aspectos. La selección de hardware puede estar con relación a la equivalencia en la configuración de los fabricantes.[11]

10 Como se verá posteriormente es en esta etapa donde las herramientas CASE ponen mayor énfasis
11 Frecuentemente se pasa por alto la determinación del ambiente del sistema, olvidando su importancia debido a la gran variedad de productos existentes en el mercado

**4.- Diseño del sistema**

Los requerimientos presentados en la especificación de la estructura son la base para el diseño del sistema. **Esto incluye diseñar la aplicación del software para un sistema que puede operar dentro del ambiente de hardware.**

Los analistas de sistemas comienzan el proceso de diseño identificando los reportes y demás salidas que debe producir el sistema. Hecho lo anterior se determinan con toda precisión los datos específicos para cada reporte y salida. Una parte es el diseño de la interfaz con el usuario. La interfaz conecta al usuario con el sistema, y evidentemente, es de suma importancia. Es común que los diseñadores hagan un bosquejo del formato o pantalla que esperan que aparezca cuando el sistema este terminado. Lo anterior se efectúa en papel o en la pantalla de una terminal utilizando para ello algunas de las herramientas automatizadas disponibles para el desarrollo de sistemas.

El diseño de un sistema también indica los datos de entrada, aquellos que serán calculados y los que deben ser almacenados, esto incluye el diseño de los archivos o la base de datos que almacenará aquellos datos requeridos por quien toma las decisiones en la organización. Una base de datos bien organizada es fundamental para cualquier sistema de información.[12]

Los documentos que contienen las especificaciones de diseño representan a éste de muchas manera (diagramas, tablas y símbolos especiales). La información detallada del diseño se proporciona al equipo de programación para comenzar la fase de desarrollo de software.

**5.- Desarrollo de pruebas de aceptación del problema**

Los objetivos establecidos en la especificación de la estructura son usados para desarrollar detalladas y específicas pruebas de un desarrollo aceptable del sistema. **Estas pruebas determinan qué tanto el sistema está constituido por objetivos que satisfagan los requerimientos del usuario.** Los diseñadores son los responsables de dar a los programadores las especificaciones de software completas y claramente delineadas.

12 Para el diseño de Bases de Datos, comenzaron a surgir Sistemas Generadores, estos mostraron las ventajas del uso de herramientas para el desarrollo de software

Una vez comenzada la fase de programación, los diseñadores contestan preguntas, aclaran dudas y manejan los problemas que enfrentan los programadores al utilizar las especificaciones de diseño.

En muchas organizaciones, las pruebas son conducidas por personas ajenas al grupo que investigó los requerimientos; con esto se persigue asegurar, por una parte que las pruebas sean completas e imparciales y, por otra que el software sea más confiable.

### 6.- Construcción del sistema

***Construir un sistema de información de las especificaciones del diseño, envuelve; código (usando programación estructurada), depuración y pruebas de cada uno de los módulos.***[13] Esto significa la elección final de un lenguaje de programación y decisiones detalladas de como los módulos del sistema deben trabajar con el sistema operativo y el ambiente de la base de datos. Los encargados de desarrollar software pueden instalar software comprado a terceros o escribir programas diseñados a la medida del solicitante. La elección depende del costo de cada alternativa, del tiempo disponible para escribir el software y de la disponibilidad de los programadores.

Dentro de las técnicas estructuradas para el diseño y documentación de software se tienen: el método HIPO, los diagramas de flujo, los diagramas Nassi-Schneiderman, los diagramas Warnier-Orr y el pseudocódigo. Aquí es donde, el analista de sistemas transmite al programador los requerimientos de programación. En el final de esta fase se realiza una prueba para determinar si el desempeño de los requerimientos definidos en las especificaciones de la estructura ha sido el correcto. Esta prueba la realizan usuarios y gente que no ha programado el sistema.

Durante esta fase, el analista también colabora con los usuarios para desarrollar la documentación indispensable del software, incluyendo los manuales de procedimientos. La documentación le dirá al usuario cómo operar el software, y así también, qué hacer en caso de presentarse algún problema.

13 Una de las principales preocupaciones por parte de los fabricantes de CASE es la generación automática de código

**7.- Integrar el sistema con el uso de la organización**

**Después de que la documentación y el software probado han sido terminados por los diseñadores y aceptado por los usuarios, este deberá ser integrado dentro de la organización.** Esto requiere de entrenamiento para los usuarios, propuesta e instalación de cualquier equipo adicional y la conversión o creación de archivos o bases de datos para el sistema [14]

Dependiendo del tamaño de la organización que empleará la aplicación y el riesgo asociado con su uso, puede elegirse comenzar la operación del sistema sólo en un área de la empresa (prueba piloto). Algunas veces se deja que los dos sistemas el viejo y el nuevo trabajen en forma paralela con la finalidad de comparar los resultados. Cada estrategia de implantación tiene sus méritos de acuerdo con la situación que se considere dentro de la empresa. Sin importar cuál sea la estrategia utilizada, los encargados de desarrollar el sistema procuran que el uso inicial del sistema se encuentre libre del problema

**8.- Operación, modificación y mejoramiento del sistema**

En esta fase, el diseño y la instalación del sistema han sido completadas. El periodo de inicio y aprendizaje ha terminado y los usuarios se han familiarizado con el sistema.

**Si la revisión de post-implantación encuentra una discrepancia entre el desempeño del sistema y las especificaciones de los requerimientos, necesariamente se tendrán que corregir las deficiencias.** Pero, en muchos casos, la evaluación es satisfactoria y el sistema opera eficientemente. Aún así, se necesitaran menores cambios ya sea por acomodo en hardware o sistema operativo, otras modificaciones serán necesarias como resultado de cambios en el ambiente de negocios, nuevas regulaciones gubernamentales, incremento en las expectaciones de clientes, o acciones de la competencia. Estos sucesos usualmente crean razones para mejoramiento y aumento en los alcances de un sistema.

[14] Un sistema desarrollado bajo la tecnología CASE requiere de mayor cuidado para su integración, como se verá más adelante

## LIMITACIONES Y FALLAS DEL CICLO DE VIDA

El entender los conceptos, herramientas y técnicas del desarrollo de sistemas de información, es siempre importante para estar consciente de sus limitaciones.

La idea del ciclo de vida del sistema está basada en una analogía - una similitud entre el desarrollo de un sistema de información y el de un organismo. Los dos tienen un tiempo de inicio; su desarrollo pasa por una secuencia de etapas, ambos deben ajustarse a su ambiente y eventualmente su existencia termina.

Pero se debe ser cuidadoso para no llevar la analogía muy lejos, **aunque el proceso de información del ciclo de vida del sistema es una valiosa abstracción, esta no corresponde exactamente a las realidades del desarrollo y uso del sistema.** En la práctica, las fases y actividades dentro del ciclo están frecuentemente traslapadas y repetidas. Este modelo idealizado debe ser a la medida de las necesidades específicas de cada proyecto de desarrollo de un sistema de información

## 2. DESARROLLO POR ANÁLISIS ESTRUCTURADO

Muchos especialistas en sistemas de información reconocen la dificultad de comprender de manera completa sistemas grandes y complejos. El método de desarrollo del análisis estructurado tiene como finalidad superar esta dificultad por medio de:
1) la división del sistema en componentes
2) la construcción de un modelo del sistema.
El método incorpora elementos tanto de análisis como de diseño.

**El análisis estructurado se concentra en especificar lo que se requiere que haga el sistema o la aplicación.** No se establece cómo se cumplirán los requerimientos o la forma en que implantará la aplicación. Más bien permite que las personas observen los elementos lógicos (lo que hará el sistema) separados de los componentes físicos

(computadoras, terminales, sistemas de almacenamiento, etc.). Después de esto se puede desarrollar un diseño físico eficiente para la situación donde será utilizado.[15]

## OBJETIVOS DEL ANÁLISIS ESTRUCTURADO

**Los principales objetivos del análisis estructurado incluyen:**

1.- Primero y fundamental: **Mantener los requerimientos de los usuarios para un nuevo sistema de procesamiento de información.**

Esto es fundamental, por que el objetivo es común en todos los métodos de análisis de sistemas de información.
Una instrucción precisa de los requerimientos de usuarios tiene las siguientes características.

a. **Explícita.** Está documentada y abierta a todos los participantes en el desarrollo del sistema.

b. **Completa.** En todo lo esencial que concierne al usuario, toda la información que entra, se archiva, es transformada y la que sale se especifica.

c. **No ambigua.** No se deja nada vago, o sujeto a más de una interpretación.

d. **Consistente.** No hay conflictos e incompatibilidad dentro de la especificación de los requerimientos.

e. **Precisa y específica.** Los requerimientos son presentados claros y definidos. Son suficientemente detallados para que nada crucial tenga que ser solicitado más adelante.

---

15 Varias herramientas CASE basan su funcionamiento en técnicas del análisis estructurado, como son: Warnier-Orr, Gane/Sarson, Yourdon, Diagramas Entidad-Relación, etc.

**2.- Entender los requerimientos de los usuarios.**

Una especificación precisa es imposible si los requerimientos no fueron entendidos. El análisis de sistemas es un proceso en el cuál los usuarios y los analistas se ayudan unos a otros para entender los requerimientos del sistema. Esto usualmente requiere un conocimiento de cómo se piensa que esta operando el sistema antes de decidir cómo podría ser cambiado.

**3.- Comunicar el entendimiento actual del sistema propuesto.**

Llegar a una mutua comprensión de los requerimientos del usuario se logra a través de la interacción y comunicación. En la práctica, la comunicación es un pre-requisito para un desarrollo completo del sistema. Esta debe ocurrir entre usuarios, entre analistas, y entre otros envueltos en el desarrollo del sistema.

**4.- Prevenir errores costosos.**

La perfecta comprensión de los requerimientos es un objetivo difícil de lograr. Los métodos de análisis de sistemas intentan minimizar su impacto en el desarrollo del sistema antes de reducir el numero de omisiones, inconsistencias y errores indetectables. No hay garantías de éxito, pero las faltas pueden ser reducidas si se previenen las equivocaciones, detectando y corrigiendo errores rápidamente.

**5.- Mantener un diseño del problema.**

Los requerimientos del usuario son la base para diseñar un sistema de proceso de información que los satisfaga. El resultado del análisis de sistemas debe comunicar a los requerimientos con los diseñadores de sistemas, dando la información necesaria para producir la descripción de un sistema ejecutable de programas de computación.

**6.- Mantener las condiciones de aceptación del sistema.**

Es necesaria una especificación precisa de requerimientos. Pero aquí debe existir algún desempeño explícito de los requerimientos del nuevo sistema. Este desempeño será usado para probar el sistema completo y determinar si es, o no aceptable.

## LAS ACTIVIDADES DEL ANÁLISIS ESTRUCTURADO

Son numerosas las propuestas de las actividades necesarias para establecer los requerimientos de los usuarios en el desarrollo de un sistema de información. La figura (1.2) muestra una de estas propuestas.[16]

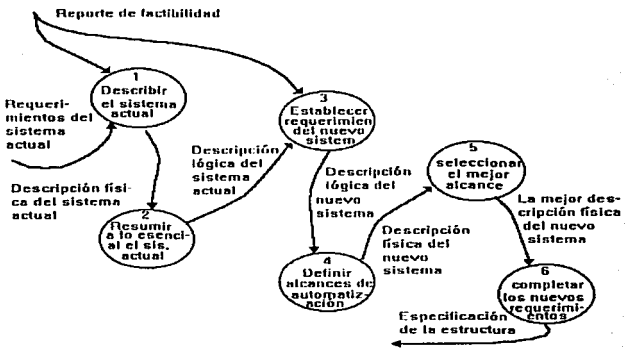



**Figura 1.2. Actividades para establecer requerimientos**

16 Como en todas las metodologías existentes para el desarrollo de sistemas de información, en el análisis estructurado, la determinación de los requerimientos es lo primordial

**1.- Describir el sistema actual.**

En muchos casos, el estudio del sistema actual permite prepararse para establecer los requerimientos del usuario para un nuevo sistema. Esta actividad debe identificar la información usada actualmente, archivada y transformada por una organización. El resultado es un modelo inicial de flujo de información en el sistema actual

**2.- Reducir la descripción del sistema actual hacia su esencia.**

En esta actividad, la descripción del sistema actual es reducida a su esencia. La redundancia es eliminada, y los detalles de cómo se transforma la información se eliminan, hasta que la descripción del sistema es mínima. Un considerable esfuerzo debe ser invertido para describir la base de datos del sistema en una forma comprensible, unificada, mínima y lógica.

**3.- Establecer los requerimientos del nuevo sistema.**

La descripción lógica del sistema actual es reemplazada por una descripción lógica del nuevo sistema. El énfasis esta en especificar correctamente los requerimientos de flujo y transformación de la información. La nueva descripción del sistema sirve para identificar las porciones a ser cambiadas; las cuales son generadas como respuesta a los objetivos del nuevo sistema. Los nuevos requerimientos deben satisfacer las constantes definidas en el estudio de factibilidad.

**4.- Definir alcances alternativos de automatización.**

Esta actividad designa que porción del nuevo sistema será automatizada y las transformaciones que se realizarán por computadora. La decisión para automatizar una transformación implica un medio electrónico de información, éste tiene consecuencias de implementación en términos de hardware y software. Así el resultado, aunque de gran nivel y abstracto es una nueva descripción física del sistema.

**5.- Seleccionar el mejor alcance de automatización.**

Se evalúan todas las alternativas de la descripción del nuevo sistema para determinar cuál es la más deseable. El procedimiento se conoce como análisis de costo/beneficio pero este no se debe limitar al costo mesurable en dinero. El primer paso es definir cuantitativa y cualitativamente medidas esperadas de desempeño del sistema. El accionar completo de cada alternativa debe ser estimado; para compararlas después y seleccionar la mejor alternativa para la organización.

**6.- Completar y organizar los nuevos requerimientos del sistema.**

El proceso de información del modelo del sistema descrito en el diagrama de flujo de datos, el diccionario del sistema y la descripción de la base de datos se complementan con algunos otros requerimientos esenciales del usuario y con objetivos explícitos para el desempeño del sistema. Estos se combinan con las especificaciones de la estructura del sistema, las bases para el diseño del sistema y el curso de las pruebas de aceptación del sistema.

## ELEMENTOS DEL ANÁLISIS ESTRUCTURADO

***Los elementos esenciales del análisis estructurado son los símbolos gráficos, diagramas de flujo de datos y el diccionario centralizado de datos.***[17]

**Descripción gráfica** Una de las formas de describir un sistema es preparar un bosquejo que señale las características, identifique la función para la que sirve e indique cómo interactúa con otros elementos, entre otras cosas. Sin embargo, describir de esta manera un sistema grande es un proceso tedioso y propenso a errores ya que es fácil omitir algún detalle o dar una explicación que quizá los demás no entiendan.

En lugar de las palabras el análisis estructurado utiliza símbolos, o iconos, para crear un modelo gráfico del sistema. Los modelos de este tipo muestran los detalles del

17 SEEN, James A. Análisis y diseño de sistemas de información.

sistema pero sin introducir procesos manuales o computarizados, archivos en cinta o disco magnético, o procedimientos operativos y de programas. Si se seleccionan los símbolos y notación correctos entonces casi cualquier persona puede comprender la forma en que los componentes se acomodarán entre sí para formar el sistema.

El diagrama lógico de flujo de datos muestra las fuentes y destinos de los datos, identifica y da nombre a los procesos que se llevan a cabo, a los grupos de datos que relacionan una función con otra y señala los almacenes de datos a los que se tiene acceso.

**Diagramas de flujo de datos** El modelo del sistema recibe el nombre de diagrama de flujo de datos (DFD). La descripción completa de un sistema está formada por un conjunto de diagramas de flujo de datos.

Para desarrollar una descripción del sistema por el método de análisis estructurado se sigue un proceso descendente (top-down). El modelo original se detalla en diagramas de bajo nivel que muestran características adicionales del sistema. Cada proceso puede desglosarse en diagramas de flujo de datos cada vez más detallados. Esta secuencia se repite hasta que se obtiene el detalle suficiente que permita al analista comprender en su totalidad la parte del sistema que se encuentra bajo investigación.

**Diccionario de datos** Todas las definiciones de los elementos en el sistema -flujo de datos, procesos y almacenes de datos- están descritos en forma detallada en el diccionario de datos. Si algún miembro del equipo encargado del proyecto desea saber alguna definición del nombre de un dato o el contenido particular de un flujo de datos, esta información debe encontrarse disponible en el diccionario de datos.

## EL DISEÑO ESTRUCTURADO

El diseño estructurado, otro elemento del análisis estructurado que emplea la descripción gráfica, se enfoca en el desarrollo de especificaciones del software. **La meta del diseño estructurado es crear programas formados por módulos independientes desde el punto de vista funcional.** Este enfoque debe conducir hacia mejores programas y facilitar su mantenimiento.

El diseño estructurado es una técnica para la elaboración de código y no un método de diseño de comprensión. Es decir, no indica nada relacionado con el diseño de archivos o bases de datos, la presentación de entradas o salidas, la secuencia de procesamiento o el hardware que dará soporte a la aplicación. Esta técnica conduce a la especificación de módulos de programas que son funcionalmente independientes.

La herramienta fundamental del diseño estructurado es el diagrama estructurado figura(1.3). Al igual que los diagramas de flujo de datos, los diagramas estructurados son de naturaleza gráfica y evitan cualquier referencia relacionada con el hardware o detalles físicos.[18] Su finalidad no es mostrar la lógica de los programas. Los diagramas estructurados describen la interacción entre módulos independientes junto con los datos que un módulo pasa a otro cuando interacciona con él. Estas especificaciones funcionales para los módulos se proporcionan a los programadores antes que dé comienzo la fase de escritura de código.

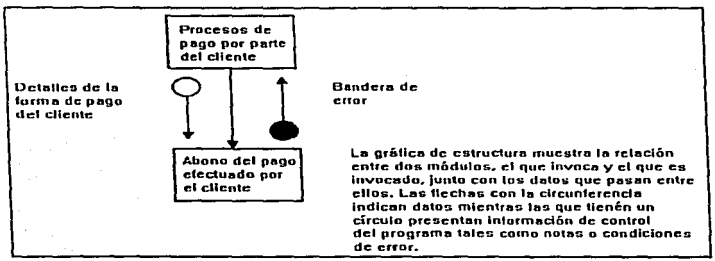



**Figura 1.3**

18 Las herramientas CASE son de gran ayuda para el diseño estructurado, debido a su naturaleza de desarrollo en ambiente gráfico

## 3. DESARROLLO POR PROTOTIPO DE SISTEMAS.

**Este método hace que el usuario participe de manera más directa en la experiencia de análisis y diseño que cualquiera de los ya presentados.**[19] La construcción de prototipos es eficaz bajo las circunstancias correctas. Sin embargo, al igual que los otros métodos, el método es útil sólo si se emplea en el momento adecuado y en la forma apropiada.

### ¿ QUE ES UN PROTOTIPO ?

**El prototipo es un sistema desarrollado con la finalidad de probar ideas y suposiciones relacionadas con el nuevo sistema, por lo tanto funciona.** Al igual que cualquier sistema basado en computadora, está constituido por software que acepta entradas, realiza cálculos, produce información ya sea impresa o en una pantalla, o que lleva a cabo otras actividades significativas. Es la primera versión, o iteración, de un sistema de información; es el modelo original.

Los usuarios evalúan el diseño y la información generada por el sistema. Lo anterior sólo puede hacerse con efectividad si los datos utilizados, al igual que las situaciones, son reales. Por otra parte, deben esperarse cambios a medida que el sistema es utilizado.

Las innovaciones del prototipo (se evalúan, y si son valiosas se integrarán al sistema definitivo) forman parte de la información que se encuentra indagando el equipo de analistas de sistemas. Los prototipos prevén el sistema futuro. Los planes de revisión permiten identificar las propiedades que deberán considerarse próximamente para el desarrollo del prototipo.

---

19 Debido a la retroalimentación que existe entre el usuario y el diseñador en esta metodología, se requiere de rapidez para efectuar los cambios al sistema, es aquí donde las herramientas facilitan el trabajo del analista

## RAZONES PARA DESARROLLAR PROTOTIPOS DE SISTEMAS

◆ **Los requerimientos de información no siempre están bien definidos.** Es probable que los usuarios conozcan sólo ciertas áreas de la empresa donde se necesiten mejoras o cambios en los procedimientos actuales. También es posible que se reconozca la necesidad de tener mejor información para administrar ciertas actividades pero que no estén seguros cuál de esta información será la adecuada. Los requerimientos del usuario pueden ser vagos aún al formular el diseño. En otros casos, es probable que una investigación de sistemas bien llevada dé como resultado un conjunto muy amplio de requerimientos de sistemas, pero construir un sistema que satisfaga a todos ellos quizá necesite del desarrollo de nueva tecnología. Esta podría ser la reingeniería de software[20], que aprovecha elementos ya construidos anteriormente, mejorando o adoptándolos para aprovecharlos en aplicaciones actuales.

◆ **Los prototipos permiten evaluar situaciones extraordinarias** donde los encargados de diseñar e implantar sistemas no tienen información ni experiencia, o también donde existen situaciones de riesgo y costo elevados, y aquellas donde el diseño propuesto es novedoso y aún no ha sido probado.

◆ **Aunque el prototipo es un sistema que funciona, está diseñado para ser modificado con facilidad.** La información obtenida con su uso se aplica en un nuevo diseño que se emplea, otra vez, como prototipo y que revela más información valiosa sobre el diseño. El proceso se repite las veces que sea necesario para revelar los requerimientos esenciales del diseño.

En general, los analistas de sistemas encuentran que los prototipos tienen mayor utilidad bajo las siguientes condiciones:

- Los encargados de diseñar e implantar sistemas nunca han desarrollado uno con las características del sistema propuesto.

- Se conoce sólo una parte de las características esenciales del sistema; las demás no son identificables a pesar de un cuidadoso análisis de requerimientos.

---

20 Mayor información sobre reingeniería en: CASE Using software development tools, FISHER, Alan S.

- La experiencia con el uso del sistema añadirá una lista significativa de requerimientos que el sistema debe satisfacer (más que la que puede obtener con cualquier otro método de desarrollo).

- Las diferentes versiones del sistema evolucionan con la experiencia al igual que el desarrollo adicional y el refinamiento de sus características.

- Los usuarios del sistema participan en el proceso de desarrollo.

***El principio fundamental del desarrollo de prototipos es el siguiente:***[21]

Los usuarios pueden señalar las características que les agradaría o no tener, junto con los problemas que presenta un sistema que existe y funciona, con mayor facilidad que si se les pidiese que las describieran en forma teórica o por escrito. El uso y la experiencia producen comentarios más significativos que el análisis de diagramas y la propuesta por escrito.

**El desarrollo de prototipos de sistemas es un proceso interactivo. Comienza con unas cuantas funciones y crece al incluir otras que son identificadas posteriormente.** También puede comenzar con un conjunto de funciones que tanto el analista como los usuarios consideran completo y que pueden aumentar o disminuir con el uso y la experiencia. ***En general, los pasos a seguir en el proceso de desarrollo de prototipos son los siguientes:***[22]

1.- Identificar los requerimientos de información que el usuario conoce junto con las características necesarias del sistema
2.- Desarrollar un prototipo que funcione.
3.- Utilizar el prototipo anotando las necesidades de cambios y mejoras. Esto expande la lista de los requerimientos de sistemas conocidos.
4.- Revisar el prototipo con base en la información obtenida a través de la experiencia del usuario.
5.- Repetir los pasos anteriores las veces que sea necesario, hasta obtener un sistema satisfactorio.

21 SEEN, James A. Análisis y diseño de sistemas de información.
22 SEEN, James A. Análisis y diseño de sistemas de información.

Tal como sugieren los pasos anteriores, **la construcción de prototipos no es un proceso de desarrollo por prueba y error.** Antes que dé inicio cualquier actividad de diseño o programación, el analista se reúne con los usuarios una o dos veces con la finalidad de identificar los requerimientos. El resultado de estas reuniones forma la base para la construcción de prototipo

Que funcione el desarrollo de un prototipo es responsabilidad del analista de sistemas. El diálogo de interfase permite a los usuarios actuar recíprocamente con el sistema, las rutinas de procesamiento y las salidas deben ser adecuadas (aunque no necesariamente completas) para que las personas puedan comprender cómo utilizar el sistema para realizar estas funciones. Los mensajes y pantallas no incluidos en el prototipo se añaden más tarde, cuando se conoce un conjunto más completo de requerimientos.

Cuando el analista y el usuario deciden que cuentan ya con la suficiente información proveniente del proceso de construcción del prototipo, determinan cómo satisfacer los requerimientos ya identificados. En general, se opta por una de las siguientes cuatro opciones:

1 - Volver a desarrollar el prototipo. Esta alternativa quizá signifique volver a programar por completo

2.- Implantar el prototipo como sistema terminado. La eficiencia en el funcionamiento junto con los métodos para interactuar con el usuario son suficientes.

3.- Abandonar el proyecto. En este caso el prototipo ha proporcionado información suficiente para demostrar que no es posible desarrollar el sistema para satisfacer los objetivos deseados dentro del marco de la tecnología existente o de lineamientos económicos u operacionales.

4.- Iniciar otra serie de construcción de prototipos. La información ganada con la experiencia sugiere un enfoque totalmente distinto.

## LOS PROTOTIPOS COMO ALTERNATIVAS EN EL CICLO DE VIDA DEL DESARROLLO DE SISTEMAS (CVDS)

Las reservas que se tienen respecto al ciclo de desarrollo de sistemas se centran en dos preocupaciones relacionadas entre si. La primera se debe al extenso tiempo que transcurre a lo largo del CVDS. Conforme al analista invierte más tiempo, el costo del sistema desarrollado se incrementa proporcionalmente. La segunda preocupación respecto al ciclo de desarrollo de sistemas, es que los requerimientos del usuario se van modificando a lo largo del tiempo

Con el fin de resolver estos inconvenientes, algunos analistas proponen que el desarrollo de prototipos sea una alternativa al CVDS. **Cuando los prototipos se desarrollan de esta manera, el analista reduce efectivamente el periodo que transcurre entre la indagación de los requerimientos de información y la entrega de un sistema funcional.** Además al diseñar prototipos, en lugar de apegarse al ciclo formal de desarrollo, pueden evitarse ciertos problemas concernientes a la identificación precisa de los requerimientos de información del usuario.

Con un prototipo, el usuario puede ver en realidad lo que llegará a ser posible; y asimismo, cómo se traducen sus necesidades en hardware y software.

Entre los inconvenientes que implica suplantar el CVDS con el desarrollo de prototipos se tiene que el sistema de información puede conformarse de manera prematura, aun antes de entender cabalmente el problema o la oportunidad que se aborda.

## DESARROLLO DE PROTOTIPOS

Para decidir si el prototipo debe incluirse o no en el ciclo de desarrollo de sistemas, el analista debe considerar qué tipo de problema va a solucionarse y analizar de qué manera un sistema ofrecería una solución figura (1.4). Un programa de nómina, tal cual o un sistema de inventarios, los cuales resuelven problemas muy estructurados, desde un punto de vista tradicional, no serían buenos candidatos para incluir prototipos durante su desarrollo. Esto es debido a que las salidas de estos sistemas son predecibles y se encuentran bien definidas.

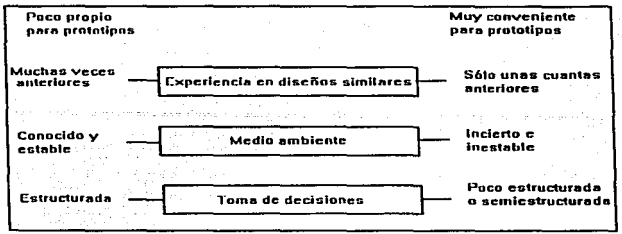



**Figura 1.4 Ciertos factores determinan si un sistema es más o menos conveniente para desarrollarse a partir de prototipos.**

Más bien, consideremos aquellos problemas novedosos y complejos, que requieran soluciones novedosas y complejas. Un sistema que se oriente a problemas poco o nada estructurados, sería adecuado para desarrollarse a partir de un prototipo.

El analista de sistemas debe evaluar también el contexto del sistema para decidir si usa prototipos. Si el sistema existirá durante largo tiempo en un ambiente estable, quizá sería innecesario el uso de prototipos. Sin embargo, si el ambiente para el sistema es sumamente variable, debe considerarse con seriedad el uso de prototipos. Un prototipo es evolutivo por naturaleza, y en consecuencia, estará sujeto a numerosas revisiones.

El sistema prototipo es sólo una parte del sistema que eventualmente se instalará. No es un sistema completo, ya que al desarrollarlo con rapidez, puede quedar limitado, contando con sólo ciertas funciones elementales. Sin embargo, es importante imaginar y luego construir el prototipo, como parte de un sistema actual, con el cual interactuará el usuario. Debe incorporar suficientes funciones representativas para que el usuario acepte que interactúa con un sistema real. El uso de prototipos permite reducir retroalimentación sobre el sistema propuesto, y además qué tan bien satisface las necesidades de información del usuario. Uno de los primeros pasos en el desarrollo de un prototipo es la estimación de los costos involucrados para su construcción, considerándolo como uno de los módulos

del sistema. Si tanto el costo del tiempo de los programadores y de los analistas, como el costo del equipo se encuentran dentro del monto presupuestado, entonces debe procederse al desarrollo del prototipo. ***Los prototipos son un excelente instrumento para integrar el sistema de información dentro del gran sistema de la organización.*** [23]

## MÉTODOS PARA EL DESARROLLO DE PROTOTIPOS

Con los prototipos la velocidad de desarrollo es más importante que la eficiencia en el procesamiento. Un sistema prototipo se construye con rapidez, frecuentemente en días o semanas. Por otro, lado el costo asociado con esta tarea es mucho menor comparado con el de un sistema convencional, aún a pesar de no ser tan eficiente como los sistemas desarrollados sobre periodos de meses.

Los sistemas prototipo pueden desarrollarse con métodos y lenguajes de programación convencionales, aunque no contengan todas las características y toques finales que normalmente se incluyen en un sistema terminado. La organización de los archivos puede ser temporal y las estructuras de registros pueden dejarse incompletas. Quizá falten los controles de entrada y procesamiento y, en general, la documentación del sistema es un punto que suele evitarse. Lo importante es ensayar ideas y generar hipótesis relacionadas con los requerimientos y no la eficiencia y perfección alcanzadas.

**En algunos casos se toman segmentos de programas que forman parte de otros sistemas o se utilizan librerías de código reutilizable.** Por ejemplo, todos los sistemas en línea tienen rutinas de entrada de edición que son muy similares en sus estructuras de procesamiento, aunque los detalles de las aplicaciones sean diferentes. Durante la construcción de prototipos los analistas enlazan partes de código reutilizable con código que ellos mismos escriben con la finalidad de tener listo el sistema para su evaluación y operación.

La industria de la computación busca continuamente generadores de aplicaciones, programas que sirven para generar otros programas, para apoyar los esfuerzos de la construcción de prototipos. Estas herramientas automatizan la construcción de sistemas de

---

23 KENDALL y KENDALL, Análisis y diseño de sistemas.

información, lo que permite a los analistas definir la estructura visual de las pantallas, los registros de entrada y el formato de los reportes; estas especificaciones son procesadas por los generadores de aplicaciones para producir con rapidez, usualmente en cuestión de horas, programas que trabajan.

En algunos casos aquellos donde el sistema será utilizado con poca frecuencia, el prototipo puede, de hecho, convertirse en el sistema terminado. Una vez que existe acuerdo en los requerimientos o diseños formulados, el sistema puede ser reprogramado para alcanzar mayor rapidez en su ejecución o para tener todas las características deseadas que fueron ignoradas al inicio del proyecto.

## LINEAMIENTOS PARA EL DESARROLLO DE PROTOTIPOS

Una vez que se ha tomado la decisión, deben observarse cuatro lineamientos básicos al construir el prototipo.[24] Tal como se plantea en la figura (1.5) son:

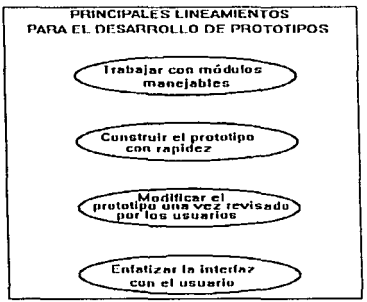



**Figura 1.5 Cuatro lineamientos principales para el desarrollo de prototipos.**

[24] KENDALL y KENDALL, Análisis y diseño de sistemas.

Estos lineamientos sugieren la manera de proceder con el prototipo y que necesariamente se encuentran relacionados. Cada lineamiento se analiza a continuación.

◆ **Trabajar con módulos manipulables**

Desarrollar el prototipo de ciertas características del sistema dentro de un modelo funcional, es imperativo que los módulos sean manipulables. Una ventaja distintiva de los prototipos es que no necesariamente se construye un sistema funcional entero como prototipo, hay que recordar que un módulo manipulable es aquél que nos permite relacionarnos con sus características, y además su construcción es independiente de otros módulos del sistema. Aquellas características que se consideren poco importantes quedarán fuera del prototipo.

◆ **Construir el prototipo con rapidez**

La esencia del desarrollo del prototipo de un sistema de información con éxito es la rapidez de su realización. Una vez realizado un breve análisis sobre los requerimientos de información por medio de métodos tradicionales, tales como la entrevista, la observación y la investigación de datos de archivo, se construyen los módulos funcionales de prototipos. El prototipo debe estructurarse en menos de una semana, de preferencia y si es posible en dos o tres días. **Recuerde que con el fin de construir un prototipo así de rápido, se debe contar con instrumentos especiales, tales como un sistema administrador de base de datos y aquel software que le permita generalizar las entradas y las salidas, los sistemas interactivos, etc.** Todas estas herramientas de programación permiten acelerar la construcción de un sistema que sería imposible realizar por medio de técnicas convencionales de programación.

Por medio del prototipo, el analista establece una retroalimentación con el usuario, con el fin de obtener una mejor visión de las necesidades de información. Al desarrollar el prototipo con rapidez, el analista contará con una excelente visión sobre la manera en que deberá desarrollar el resto del proyecto.

- **Modificaciones en el prototipo**

Un tercer lineamiento en el desarrollo de prototipos es que éste debe tolerar modificaciones. Para ello el prototipo requiere contar con módulos que tengan entre si una baja dependencia. Al observar este lineamiento, será más fácil modificar el prototipo, si llegara a ser necesario.

Un prototipo se modifica en repetidas ocasiones por medio de varias interacciones. Tales cambios deben acercar el prototipo al sistema que el usuario considera como relevante. Cada modificación requiere de una nueva evaluación por parte de los usuarios.

Así como ocurrió con el desarrollo inicial, las modificaciones al prototipo deben realizarse de inmediato, comúnmente en uno o dos días, con el fin de conservar el momentum del proyecto. Sin embargo, la programación exacta de las modificaciones dependerá de la dedicación de los usuarios para interactuar con el prototipo modificado. El analista de sistemas debe motivar al usuario para que realice su parte, y que consiste en evaluar con rapidez tales cambios.

- **Enfatizar la interfaz con el usuario**

La interfaz del usuario con el prototipo, y eventualmente con el sistema final es fundamental. Puesto que lo que se trata de obtener en realidad, es que los usuarios planteen más allá sus requerimientos de información. Y para ello deben ser capaces de interactuar, sin complicaciones con el prototipo del sistema. Para muchos usuarios la interfaz, se contempla como el sistema y no debería ser un obstáculo.

En esta etapa, la meta del analista es diseñar una interfaz tal, que permita al usuario interactuar con el mínimo de adiestramiento, y además contar con el máximo control sobre las funciones presentadas. Aunque muchos aspectos del sistema permanecen sin desarrollo en el prototipo, la interfaz con el usuario debe quedar muy bien desarrollada para que el usuario se involucre en el sistema con rapidez y no se desaliente.

Durante el desarrollo de prototipos debe hacerse a un lado el embrollo de las interfases, o simplemente ignorarlo. Sin embargo, si la interfaz del prototipo no es lo que los usuarios requieren o buscan, o si los analistas de sistemas se percatan de que no hay un acceso adecuado al sistema, en ambos casos debe modificarse.

## DESVENTAJAS DE LOS PROTOTIPOS

Como cualquier otra técnica de recopilación de información, los prototipos cuentan con varias desventajas. La primera es que su administración llega a ser difícil como un proyecto de desarrollo de prototipos, dentro de un gran esfuerzo de sistemas. Otra desventaja, es que tanto los usuarios como los analistas pueden considerar el prototipo como un sistema concluido, cuando de hecho no lo es, y nunca se planteó como un sistema final.

## VENTAJAS DE LOS PROTOTIPOS

Como se ha visto, no en todos los proyectos de sistemas, los prototipos llegan a ser necesarios o apropiados. Sin embargo, las ventajas de su uso deben plantearse para tomar una decisión. Existen tres ventajas relevantes en el uso de prototipos, las cuales son:

1.- La pronta modificación del sistema en su desarrollo.
2.- La oportunidad de detener un sistema que no sirve.
3.- La posibilidad de desarrollar otro sistema que se ajuste mejor a las necesidades y a las expectativas del usuario.

Estas tres ventajas se encuentran relacionadas entre si.

## PROBLEMÁTICA ACTUAL EN EL DESARROLLO DE SISTEMAS, E INTRODUCCIÓN A UNA NUEVA TECNOLOGÍA

**La ingeniería de software asistida por computadora (CASE) está revolucionando el mundo del desarrollo de sistemas de información** y este capítulo se encargará de adentrarnos en este tema.

Solo 17 años atrás una computadora con 64,000 bytes (64K) de memoria era considerada de gran capacidad. Programas "extensos" podían ser almacenados en 64K. Tan recientemente como lo es 1978, con la introducción de las computadoras personales,

64K eran considerados como el límite superior de memoria que un programa grande debería consumir.

Casi todos los programas de aplicación eran elaborados en lenguaje ensamblador, el cuál proveía simultáneamente lo mejor en desempeño de ejecución y respetaba los límites de memoria de un programa, ya que representaba una dificultad construir programas muy grandes en ensamblador.

En septiembre de 1981, IBM introduce su IBM-PC computadora personal con mayor capacidad de memoria 640,000 bytes, diez veces más grande que la generación previa de computadoras personales sólo tres años atrás. Muchos programas de aplicación crecieron en tamaño para retar las bondades de esta vasta cantidad de memoria. Estos programas, ahora escritos en lenguajes de alto nivel, algunos como C, y Pascal, se volvieron más complejos y sofisticados que los anteriores.

Los diseñadores de software en todas las plataformas, ya sean computadoras personales o mainframes, se esfuerzan cada vez más para colocar sus trabajos finales con incremento en inteligencia y poder en las aplicaciones de software. Continuamente, estas traducciones a más sofisticadas interfaces de usuario final, permitían expandir la interacción entre él y la máquina.

Sin embargo como los proyectos de software se hacen cada vez más grandes y complejos, están propensos a tomar una conducta impredecible además de presentar fallas. Estos problemas se deben a las grandes cantidades de software que se escribe para diferentes aplicaciones. Literalmente cientos de miles de líneas de código para proyectos son escritas para diferentes aplicaciones.

En programas de esta magnitud algunas veces es imposible lograr el cien por ciento de confianza. Mientras que los viejos usuarios de mainframes estaban dispuestos a tolerar algunas "irregularidades", ahora millones de usuarios de computadoras personales no lo están.

Desafortunadamente, muchas de estas grandes aplicaciones de software son críticas para el éxito y correcta operación de empresas comerciales, departamentos de gobierno, etc. Sin embargo al desarrollar sistemas tomando como base alguna de las metodologías revisadas anteriormente es inevitable generar errores en el software, provocados ya sea por un mal análisis o por distracciones en el diseño, estas fallas

generalmente no son detectadas sino hasta que se llega a las fases de desarrollo. Un estudio conducido por IBM estima que el 64 por ciento de los errores en el software provienen de las fases de análisis y diseño y sólo el 30 por ciento de estos errores son detectados previamente a que el software este listo para probarse, ver figura (2.1). Esto resalta la necesidad antes que nada de un diseño correcto.

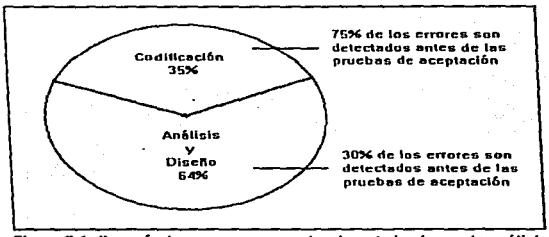



**Figura 2.1** Son más los errores generados durante las fases de análisis y diseño, desafortunadamente, sólo el 30% de estos errores son detectados antes de que las pruebas de aceptación inicien.

En contraste, sólo el 36 por ciento de los errores de programación son originados en la fase de implementación y de éstos el 75 por ciento de los errores de codificación son encontrados antes de que las pruebas de aceptación comiencen.[25]

Estos errores en el software generalmente provocan retardos en los tiempos determinados inicialmente, desafortunadamente para cualquier empresa este retraso de tiempo regularmente genera perdidas económicas. Por esta razón, continuamente se están desarrollando nuevos instrumentos que sirvan de ayuda a los analistas, diseñadores y programadores, aligerando las tareas tediosas y repetitivas, pero sobre todo apoyándolos durante las primeras fases del ciclo de vida, con el fin de producir aplicaciones de mayor calidad y confianza. Uno de estos relativamente nuevos instrumentos de desarrollo son lo que se conoce como herramientas CASE.

[25] FISHER, Alan S., Using software development tools.

# CAPITULO II:

## Descripción de la tecnología CASE

La necesidad que existe en la actualidad de proporcionar instrumentos que apoyen a la gente encargada del desarrollo de sistemas para facilitar sus funciones, provoca el surgimiento de tecnologías especializadas como lo es CASE.

Por ello es necesario iniciar con la definición de CASE, mencionando las categorías en las que se clasifican y los elementos que las componen, la evolución de la ingeniería de software aplicada a la computación, además de enumerar las características de CASE. También se plantean algunas preguntas como; ¿ Dónde puede ayudar la tecnología CASE ? y si ésta sofoca la creatividad de la gente que la utiliza, dando sus respectivas respuestas.

## LOS RECURSOS INFORMÁTICOS

**Cuando comienza un proyecto, uno de los factores que hay que tener en cuenta es el de los recursos informáticos necesarios para llevarlo a cabo:**[1] los recursos de hardware, los recursos de software y los recursos humanos. Para cuantificar y seleccionar los recursos de hardware son necesarios:

* La computadora y máquinas donde se realice el desarrollo del nuevo sistema.
* La computadora y máquinas destino (donde se vaya a ejecutar el nuevo sistema).

Los recursos humanos se estiman empleando diversos métodos de evaluación del esfuerzo de desarrollo para obtener el número de personas-mes, o personas-año.

Los recursos del software, que a menudo se descuidan durante la planificación, suelen ganar importancia durante las fases siguientes, aunque es recomendable especificar cuanto antes los requerimientos de recursos del software, de este modo se puede realizar una selección oportuna.[2]

***Existen tres tipos de herramientas para el software:***[3] herramientas orientadas al código, herramientas de cuarta generación y herramientas CASE.

### Herramientas orientadas al código.

Las herramientas de esta categoría son las más tradicionales, y prácticamente las únicas que se utilizan para el desarrollo de software. Incluyen:

* Compiladores.
* Utilidades específicas que acompañan a los compiladores.
* Editores.
* Enlazadores.

[1] DAVIS, Williams S., Herramientas CASE.
[2] Es de gran importancia para el desarrollo de sistemas, el determinar previamente los recursos de software que este necesita. Sin olvidar que la determinación del ambiente software y hardware es una de las etapas del ciclo de vida
[3] DAVIS, Williams S., Herramientas CASE.

**Herramientas de cuarta generación.**

Las herramientas de cuarta generación permiten especificar las características de una aplicación a muy alto nivel, y realizan la generación de código automático basándose en esas especificaciones. Se emplean lenguajes de cuarta generación (4GLs) no procedurales para la creación de informes, definición de pantallas, de menús, etc. También incorporan un lenguaje de consulta y manipulación de datos (SQL). Las herramientas de esta categoría dan soporte a las distintas fases del ciclo de vida de un proyecto.

**Las herramientas CASE.**

Debido a la exigencia creciente de aplicaciones más innovadoras y la necesidad de mantener los sistemas antiguos, las empresas de software están intentando romper con el pasado, pasando de una labor artesanal a una disciplina de ingeniería, mucho más rigurosa y previsible. **El primer paso se dio en los años sesenta, con la programación estructurada**, que, junto con otras disciplinas de la incipiente ingeniería de software intentaban aplicar la primera crisis del software. Posteriormente aparecieron nuevas técnicas y metodologías que comenzaron a usar en los años setenta. **Actualmente, las empresas de software han comenzado a implantar nuevas generaciones de métodos, técnicas y herramientas de desarrollo:**

* Nuevas metodologías orientadas a datos.
* Herramientas de análisis y diseño basadas en gráficos.
* Lenguajes de cuarta generación.
* Diseño y prueba de prototipos.
* Sistemas expertos.
* Técnicas de análisis, diseño y programación estructurada.
* Técnicas de análisis, diseño y programación orientada a objetos.

**que, integradas a través de un entorno de ingeniería de software asistida por computadora (CASE),** permitan al ingeniero de software mejorar su entorno de trabajo e incrementar su productividad.

El objetivo fundamental de la tecnología CASE es separar el diseño de la implementación. Generalmente lo más olvidado del proceso de diseño es la generación actualizada de código. Muchos han reconocido este principio básico de planeación y estructuración, en el curso de estos últimos quince años, varias metodologías estructuradas han sido desarrolladas e introducidas a un gran número de programadores. Estas metodologías estructuradas proveen un esquema en el diseño así como un conjunto de prácticas y formalismos los cuales se han convertido en la base para el desarrollo de software.

En muchos aspectos, **las herramientas CASE son una evolución directa de antiguas técnicas estructuradas basadas en papel.** Ahora muchas de estas mismas metodologías y técnicas organizacionales son implementadas como programas de software por si mismas.

Una definición de CASE es el uso de herramientas que provean un apoyo en cualquier punto del ciclo de desarrollo de software. Esta definición podría incluir más herramientas de las que los diseñadores de software conocen hasta hoy, incluyendo compiladores, depuradores y sistemas de control de código fuente.

***Una definición más restrictiva pero operacionalmente mejor para CASE, es el uso de herramientas que provean una ayuda en las fases del análisis de requerimientos de software y especificaciones de diseño, así como la generación de código automáticamente de las especificaciones de diseño.***[4] Esta definición será usada con objeto de enfocar el gran apoyo de las herramientas en el diseño y generación de código.

La interacción usuario-analista es fundamental a la hora de concebir, analizar y diseñar el nuevo sistema. Para ello, se utilizan técnicas y metodologías basadas en gráficos, que permiten modelar los componentes de software de cada nivel: datos, procesos y, según la herramienta, incluso operaciones comerciales.

Las herramientas CASE, que soportan estas metodologías, disponen de un diccionario, para almacenar toda la información necesaria, este es la base de la herramienta CASE, pues en él se almacena toda la información necesaria para poder crear, modificar y mantener al sistema.

[4] FISHER, Alan S., Using software development tools.

Al automatizar el ciclo de vida, se espera que mejoren tanto la productividad como la calidad de los productos de software. Algunas herramientas incluyen generadores de código, que eliminan la necesidad de codificar, además de facilitar el mantenimiento. El objetivo final consiste en mantener los programas estrictamente en los niveles de especificación, análisis y diseño.

Si se utiliza una metodología de desarrollo que se adecue al problema que se quiera resolver, y se emplea correctamente un entorno de desarrollo CASE, se pueden llegar a obtener las siguientes ventajas:

* Menor tiempo de desarrollo y mantenimiento.
* Mayor independencia entre análisis, diseño y codificación.
* Trabajar con tareas de mayor nivel que la codificación pura.
* Mejora de la calidad del proyecto de software.
* Aplicaciones más productivas para la empresa.

En el siguiente capítulo se describen estas ventajas con mayor detalle.

Dado que en los departamentos de software existen diferentes formas de abordar los problemas, y que, a su vez, existen diferentes tipos de problemas que hay que resolver, se buscan herramientas que permitan una diversidad de enfoques, para acomodarse a la variedad de situaciones que se dan en la vida real.

Pese a todas las ventajas descritas anteriormente, muchas veces la herramientas CASE no cumplen con los objetivos esperados, o simplemente no son atractivas para las empresas. Las razones pueden diferir significativamente:

* Dificultades para adaptarse al cambio, pues es un cambio cultural, no sólo técnico, el que se debe asimilar.

* Muchas de las herramientas necesarias para automatizar el proceso son inmaduras o no están disponibles. La inmensa mayoría presenta insuficiencias a la hora de integrar las actividades de un proyecto.

* Es difícil pasar de un análisis realizado casi en solitario, a la realización del análisis en colaboración con los usuarios.

* Muchas empresas no tienen o no cumplen con las prácticas de gestión de software apropiadas, sin las cuales la automatización del proceso suele resultar ineficaz o imposible.

* A veces se tiene la creencia de que las nuevas herramientas son soluciones mágicas, cuando en realidad el resultado se ve a medio y largo plazo.

* Desengaño debido a la ficción de los anunciantes.

Es fácil dejarse llevar por la verborrea de los anunciantes, que inducen a pensar que los productos CASE son la respuesta definitiva a todos los problemas. Los fabricantes tienden a prometer sueños antes que realidades. La reutilización de software es un buen ejemplo de ello, pues dentro de la ingeniería de software existen muy pocas técnicas sistemáticas para realizar adiciones a una biblioteca, y aún no están estandarizadas las interfases para que el software sea totalmente reutilizable.

Un elemento de confusión es el hecho de que no existe una definición universal establecida sobre las funciones de una herramienta CASE. **La realidad es que el ciclo de desarrollo de software es muy amplio y esto permite que en el mercado existan productos CASE orientados a múltiples plataformas y con diferentes objetivos reales.**[5] Debido al hecho de que la tecnología CASE es relativamente reciente, buena parte del trabajo en CASE es experimental y depende de concepciones metodológicas o funcionales dependientes del fabricante. Seguramente, el mayor nivel de productividad y calidad se consigue con una herramienta CASE que permita pasar del análisis y diseño a la generación de código para un hardware específico.

Algunos productos CASE permiten conexiones con otros productos, de forma que se facilita la migración de información entre las diferentes herramientas. Sin embargo, la conexión todavía es muy limitada.

[5] Básicamente a este problema se debe la idea de realizar este trabajo, con el propósito de ofrecer un marco para la elección de la herramienta correcta

La tecnología CASE presenta hoy en día una amplia gama de productos eficaces siempre que no se pretenda solucionar sin ningún esfuerzo todas las necesidades del desarrollo de software. Dentro de pocos años, los sistemas expertos producirán un avance en la capacidad de desarrollo de software, pero probablemente será con computadoras y metodologías diferentes a las actuales. Actualmente, la creación de software todavía se basa en la escritura manual de líneas de código, y la implantación de productos CASE aún es minoritaria.

## CLASIFICACION DE HERRAMIENTAS CASE

***Por lo general, las herramientas automatizadas se agrupan en tres categorías:***[6] front-end, back-end e integrales. Esta clasificación recalca las actividades del proceso de desarrollo donde las herramientas tienen su mayor papel figura (2.2). Cada categoría es de utilidad y ninguna es más valiosa que otra.

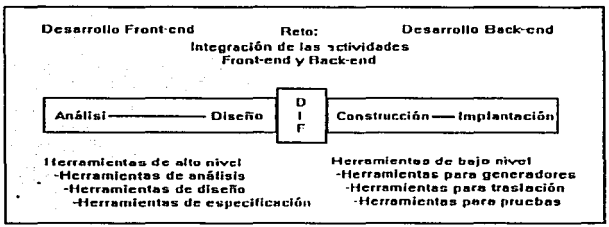



**Figura 2.2 Actividades de desarrollo de tipo front-end y back-end.**

[6] SEEN, James A., Análisis y diseño de sistemas de información

### Herramientas de tipo front-end

**Las herramientas de tipo front-end, automatizan las primeras actividades (análisis de requerimientos y diseño de especificaciones) del proceso de desarrollo de sistemas.** Entre los muchos aspectos que se toman en cuenta al desarrollar herramientas para esta fase, se hallan las técnicas de soporte para ayudar al analista a preparar especificaciones formales que carezcan de ambigüedades, a validar las descripciones del sistema con el objeto de determinar su consistencia y completes, y a seguir la evolución de los requerimientos de la aplicación en características que formen parte del sistema que finalmente será implantado. Hasta donde sea posible, esta ayuda debe ser automatizada (por ejemplo, la computadora valida automáticamente las descripciones del sistema).

A menudo, las herramientas de tipo front-end proporcionan soporte para el desarrollo de modelos gráficos de sistemas y procesos. Los diagramas de flujo de datos son representativos de este tipo de herramientas.

### Herramientas de tipo back-end

Las herramientas de tipo back-end tienen como finalidad ayudar al analista a desarrollar las especificaciones del programa, los algoritmos de procesamiento y la descripción física de datos, también ayudan a la interacción con los dispositivos (para entrada y salida), etc. **Estas actividades convierten los diseños lógicos de software en un código de programación que es el que finalmente da existencia a la aplicación.** Dentro de estas herramientas se encuentran los generadores de aplicaciones, de programas y de pantallas.

### Herramientas integrales

Las actividades del análisis abordan los detalles de alto nivel mientras que las actividades de desarrollo dan mayor importancia a los detalles de bajo nivel. El nivel de detalle es una forma familiar para caracterizar el tipo de información que se está reuniendo. Las especificaciones de alto nivel describen requerimientos del usuario, como entradas, salidas y expectativas de funcionamiento. Las especificaciones de bajo nivel

indican la forma en cómo serán satisfechos estos requerimientos por medio de detalles que son específicos de la computadora.

En algún momento, los requerimientos y diseño deben transladarse en especificaciones que tengan la forma de código ejecutable (o fuente). En la actualidad, es aquí donde existe un hueco. En general, las herramientas front-end y back-end no están integradas, a tal grado que las especificaciones generadas por una puedan ser procesadas sin problemas por la otra. Por ejemplo, no es posible trasladar con facilidad diagramas de flujo de datos a código fuente, y lo mismo ocurre con las estructuras de datos. A pesar de lo anterior, la transición de una herramienta front-end hacia una back-end puede ahorrar tiempo y aumentar la velocidad de implantación.

Cuando las herramientas front-end y back-end están separadas el analista debe hacerse cargo del proceso de transición entre estas herramientas figura (2.3). Los responsables de desarrollar sistemas de información junto con los investigadores, buscan formas para integrar las tareas de análisis y desarrollo (desde la determinación de los requerimientos hasta la implantación de la aplicación). Sin embargo, alcanzar este grado de integración es un reto difícil.

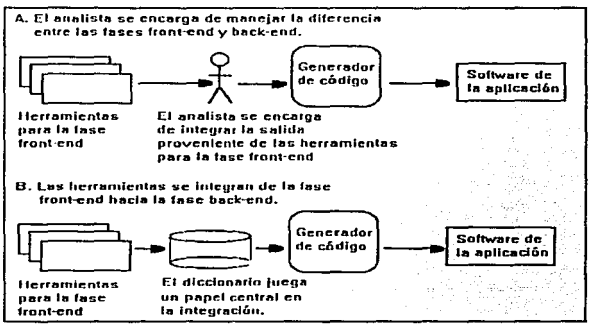



figura 2.3 Manejo de la diferencia que existe entre los desarrollos front-end y back-end.

**Las herramientas integrales proporcionan un ambiente que automatiza tareas clave a lo largo de todo el proceso de desarrollo.**[7] Estas herramientas abarcan todo el ciclo de vida de la aplicación, no sólo el proceso de desarrollo. Si bien estas herramientas incluyen facilidades para manejar aspectos de análisis y desarrollo, también facilitan el diseño, administración y mantenimiento del código. Asimismo, brindan un ambiente eficiente para crear, almacenar, manipular, administrar y documentar sistemas.

Algunas herramientas están vinculadas con metodologías específicas de desarrollo (por ejemplo, análisis estructurado). Otras soportan sólo lenguajes específicos (como COBOL) o a determinado fabricante de hardware (quizá IBM o Digital). De acuerdo con las necesidades de la organización, estas características tal vez limiten la utilidad de ciertas herramientas.

## COMPONENTES DE CASE

***En general, las herramientas de tipo CASE incluyen alguno de los siguientes ocho componentes:***[8] herramientas para diagramación,herramientas para modelado de datos, Herramientas para análisis y diseño, un depósito de información, generadores de interfases, Lenguajes de bases de datos de cuarta generación, generadores de código y herramientas de administración. Las actividades de alto nivel reciben la mayor importancia, aunque ya están apareciendo generadores de código de bajo nivel.

### Herramientas para diagramación

**Estas dan soporte al análisis y documentación de los requerimientos de una aplicación.** Por lo general, incluyen facilidades para producir diagramas de flujo de datos. Las herramientas CASE incorporan, de manera extensa, métodos propios del análisis estructurado.

[7] La tendencia actual por parte de los fabricantes de CASE es construir herramientas que soporten todo el ciclo de vida es decir herramientas integrales
[8] SEEN, James A., Análisis y diseño de sistemas de información.

Estas herramientas ofrecen la capacidad de dibujar diagramas y cartas, además de guardar los detalles en forma interna. Cuando es necesario realizar cambios, la naturaleza de éstos se describe en el sistema, el cual puede entonces volver a dibujar todo el diagrama de manera automática. La capacidad para cambiar y volver a dibujar elimina una actividad que los analistas encuentran tediosa y poco deseable.

**Herramientas de modelado de datos.**

En aplicaciones de bases de datos, una sola es frecuentemente dividida para diferentes aplicaciones cada una de las cuales introduce o extrae información. Antes de establecer una base de datos para múltiples aplicaciones y llenarla con datos, una prudente consideración debe de darse para cuidar su estructura y contenido. Porque las aplicaciones de bases de datos constituyen una gran proporción de aplicaciones comerciales de software, las técnicas de modelado de datos fueron desarrolladas para auxiliar a los diseñadores a construir bases de datos para que sean tan versátiles y eficientes como sea posible.

Las herramientas de modelado CASE ayudan a los diseñadores a modelar el flujo de información por toda la empresa y construir planos adecuados para los varios requerimientos de acceso en la organización. Las herramientas de modelado de datos ayudan en la semántica de la base volviéndola explícita para que todo mundo interprete las relaciones de los datos de la misma forma. **Las herramientas de modelado, asisten al diseñador a modelar el flujo de información de la organización.**

**Herramientas para análisis y diseño.**

Las herramientas de diseño generalmente usadas en el análisis y diseño estructurado desarrolladas por Tom DeMarco y Edward Yourdon. Estas especificaciones de propósito general y herramientas de diseño, pueden usarse para especificar y diseñar cualquier pieza de software. Las herramientas de análisis y diseño **usualmente implementan diagramas de flujo de datos, técnicas de estructuración y son excelentes descriptoras de flujo de información entre procesos.**

Varias herramientas de diseño tienen extensiones para especificar interacciones temporales efectuadas en sistemas de control de tiempo real. Otras herramientas también asisten en la elaboración de documentación actividad crítica en proyectos de software.

Las herramientas de especificación de diseño son un excelente apoyo para las partes de una aplicación donde una herramienta enfocada no lo sería. El diseñador debe, juzgar cuando una herramienta más enfocada, tal como un diseñador de interfases, es utilizado o cuando es apropiado el uso de una herramienta de propósito más general.

**Depósito centralizado de información**

La captura análisis, procesamiento y distribución de todos los sistemas de información es asistida por un depósito de información centralizado o diccionario de datos. **El diccionario contiene detalles sobre los componentes del sistema,** tales como datos, flujo de datos y procesos; asimismo, también incluye información que describe el volumen y frecuencia de cada una de las actividades.

Aunque los diccionarios son diseñados para que el acceso a la información sea sencillo, también incluyen controles y medidas de protección que preservan la exactitud y consistencia de los detalles del sistema. El uso de 1)niveles de autorización, 2)validación de procesos y 3)procedimientos para verificar la consistencia de las descripciones, asegura que el acceso a las definiciones y las revisiones hechas a ellas en el depósito de información, ocurran en forma apropiada y acorde con procedimientos ya establecidos.

**Generador de interfases**

Las interfases con el sistema son los medios que permiten a los usuarios interactuar con una aplicación, ya sea para dar entrada a información y datos o para recibir información. **Los generadores de interfases ofrecen la capacidad para preparar imitaciones y prototipos para las interfases con los usuarios.** Por lo general, soportan la rápida creación de menús de demostración para el sistema, de pantallas de presentación y del formato de los informes.

Los generadores de interfases son un elemento importante para el desarrollo de prototipos de aplicación, aunque también son de utilidad para los demás métodos de desarrollo.

Para muchas aplicaciones comerciales las interfases de usuario son sólo un gran componente del programa de aplicación. La interfase del usuario varia mucho en estilo y contenido. Algunas, como las máquinas registradoras, son diseñadas para un fácil uso y un operador común, mientras otras interfases, como los procesadores de palabra son construidos para un gran volumen de procesamiento.

**No importa la aplicación de la interfase del usuario esta debe recibir especial atención.** Todos hemos visto ejemplos de interfases de usuario que han fallado en su objetivo. Frecuentemente, la diferencia entre una interfase cuidada y otra no tanto es el éxito. ¿Qué diferencia existe entre una interfase ganadora y una perdedora? A menudo es el tiempo que toma la interfase en incorporarse a la comunidad de usuarios lo que distingue su calidad.

Los prototipos pueden ser un proceso largo y laborioso y siempre existe la tentación de usarlo como la implementación final, antes que rediseñarlo basado en los requerimientos del usuario. Afortunadamente, las herramientas CASE para el diseño de interfases de usuarios están disponibles en locales comerciales.

Este tipo de prototipos rápidos ofrecen ventajas de gran productividad y exitosos proyectos de software. Más aún, sólo con ser capaces de desplegar pantallas temporales de secuencias es suficiente para abrir la comunicación entre usuarios finales y diseñadores. Si el programador tiene la capacidad de implementar interfases rápidamente, posee un vehículo valioso para la retroalimentación con la comunidad usuaria.

Las herramientas de interfases están orientadas primordialmente hacia plataformas específicas de hardware como la PC de IBM y las populares minicomputadoras. Esta orientación se debe a la necesidad, las plataformas de hardware varian mucho en términos de despliegue de tecnología, haciendo difícil el campo del diseño de interfases en varias plataformas radicalmente diferentes. Una interfase de usuario aceptada en una IBM-PC es algo diferente a la de una terminal de Mainfraime.

Por varios años, paquetes para generación de formas, tales como ISPF para equipos mainframes de IBM y Panel para las IBM-PC, ayudaban a los profesionales de diseño de software a crear una gran variedad de interfases para aplicaciones. En sus inicios, muchos de estos paquetes, se convirtieron en subrutinas y no en herramientas CASE. En todo caso estas evolucionaron hacia CASE cuando los vendedores adicionaron editores orientados a gráficos, los cuales permitían a los diseñadores visualizar sus pantallas y generarlas automáticamente.

**Lenguajes de bases de datos de cuarta generación.**

Los lenguajes de cuarta generación son lenguajes de alto nivel, los cuales proveen facilidades para el acceso a bases de datos. Son mucho más fácil de manejar que los lenguajes tradicionales, como COBOL y C. El objetivo es remover la carga del tedioso código para acceso a bases de datos y reemplazarlo con una mucha mejor cantidad de código escrito en un 4GL de alto nivel especialmente diseñado para esto. Muchos 4GL's proveen formas eficientes de planes y diseño usando editores de texto comunes además de que **fueron la primera tecnología que demostró la ventaja de las herramientas enfocadas a la reducción de código.**

Con un 4GL, el programa de aplicación puede declarar las formas de pantalla para entradas y salidas que tendrá el programa de aplicación a los usuarios finales. Estas formas son declaradas como conjuntos de campos con propiedades bien definidas, algunas como revisión de entradas y desplegado de campos protegidos. El programa de aplicación también especifica las consultas para el acceso a la base en 4GL, permitiendo recuperar datos para ser desplegados de la forma deseada. Los 4GL's fueron los predecesores del movimiento CASE, habiendo aparecido algunos años antes que la primera herramienta creada para ser CASE. Estos, en todo caso no niegan su importancia como una herramienta de desarrollo de software. Los 4GL's de base de datos son una tecnología muy valiosa y ahorradora de tiempo.

**Generadores de código**

Este es el último objetivo de la mayoría de los vendedores de herramientas y ciertamente de todos los usuarios de CASE. **La generación de código es la habilidad**

**para generar automáticamente software compilable directamente de las especificaciones del diseño.** Finalmente, el tiempo de diseño es mucho mejor invertido especificando y diseñando, antes que codificando y compilando. Desafortunadamente, la verdadera generación de código no esta disponible actualmente en ninguna de las herramientas de propósito general o los llamados productos generadores de aplicaciones. Pero si en una variedad de herramientas enfocadas, especialmente en herramientas diseñadoras de interfases para usuarios.

En el pasado la tecnología CASE se enfocó a requerimientos de propósito general y especificaciones de diseño. Los nuevos programadores en tecnología CASE enfatizan en herramientas especializadas en desarrollo. Herramientas especiales de desarrollo enfocadas en un tipo particular de software. Las herramientas del futuro atacarán problemas más generales como el de generación automática de código. La programación automática es un problema difícil y se sigue considerando como un tópico de investigación. Cada nueva herramienta aporta pequeñas innovaciones en esta área y eventualmente la generación será un tema común.

Los mayores beneficios se obtienen cuando los generadores de código se encuentran integrados con un depósito central de información. Esta combinación alcanza el objetivo de crear un código que pueda volverse a emplear. Cuando las especificaciones cambian, se puede volver a generar el código al alimentar los detalles del diccionario de datos a través del generador de código. El contenido del diccionario puede emplearse de nuevo para preparar el código ejecutable.

### Herramientas de administración

Los sistemas CASE también ayudan a los generadores de proyecto a mantener la efectividad y eficiencia de todo el proceso de desarrollo de una aplicación. **Este componente de CASE ayuda a los gerentes de desarrollo a calendarizar las actividades** de análisis y diseño así como la asignación de recursos a las diferentes actividades del proyecto. Por ejemplo, algunos sistemas CASE soportan el seguimiento de los tiempos de desarrollo de un proyecto y los comparan con los ya planificados; también realizan la misma labor con la asignación de tareas específicas al personal. Los calendarios e informes pueden prepararse utilizando para ellos los detalles contenidos en el diccionario de datos.

Algunas herramientas CASE para la administración permiten que los gerentes de proyecto especifiquen elementos de su propia elección. Por ejemplo, ellos pueden seleccionar los símbolos gráficos que desean para describir procesos, personas, departamentos, etc. Otros permiten definir metodologías de desarrollo propias, incluyendo las reglas de validación y los estándares para datos y nombres de procedimientos. Sin embargo, la mayor parte de los sistemas CASE dependen en gran medida de la notación, principios y prácticas del método del análisis estructurado.

## INTEGRACIÓN DE HERRAMIENTAS EN CASE

CASE incorpora varias herramientas que pueden considerarse por separado, como elementos discretos, o como parte de un sistema -un grupo de herramientas-. Por lo general, se prefiere esto último. **La integración de las herramientas permite que la información obtenida con una de ellas sea utilizada por otra dentro del mismo proyecto.**[9]

La integración de herramientas ocurre en tres formas:[10]

* Creación de una interfase para desarrollo uniforme o adaptable
* Proporcionar la facilidad para transferir datos entre las herramientas
* Unir las actividades de desarrollo

### Interfase uniforme

**Una interfase uniforme significa que todas las herramientas en el sistema CASE son activadas de la misma manera y desde un lugar común en el sistema.**

A menudo la interfase determina la comodidad que experimentan los analistas al utilizar un sistema CASE. La interfase debe adaptarse a los usuarios expertos y novatos así como a la tarea que se está realizando. Los resultados, mensajes e instrucciones deben

[9] En el trancurso de los últimos cinco años han salido al mercado herramientas con esta característica, en el cuarto capítulo se mencionan algunas de ellas
[10] SEEN, James A., Análisis y diseño de sistemas de información.

mostrarse en un lugar y formato consistentes. También son importantes los mensajes interactivos y los buenos diagnósticos. Sin embargo, las herramientas deben proporcionar soporte directo para los procedimientos con los que trabajan los encargados de un desarrollo, es decir, el usuario no tiene que estar forzado a utilizar métodos y técnicas que no se ajusten a los procedimientos de trabajo existentes.

**Facilidad para la transferencia de datos**

**Significa que los detalles desarrollados con una herramienta pueden estar disponibles para otras.** Por ejemplo los generadores de código y los de interfases pueden utilizar las descripciones preparadas por medio de la creación de diagramas de flujo de datos. El diccionario de datos es el elemento crítico que hace posible la transferencia de datos entre herramientas distintas. Es de este modo como todas las herramientas interactúan con el diccionario de datos para utilizar las definiciones y descripciones contenidas en él figura (2.4)

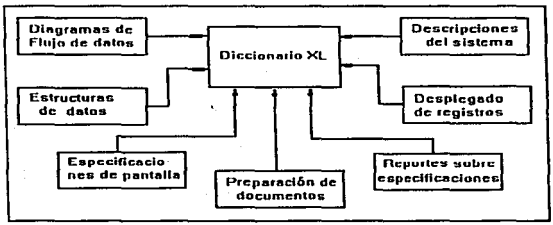



**Figura 2.4 Un diccionario central permite la transferencia de datos entre distintos componentes.**

**Unir las actividades de desarrollo**

La facilidad para transmitir datos y la unión de las fases de desarrollo se encuentran relacionadas, ya que se pueden utilizar una y otra vez los datos transferidos entre herramientas a través de todo el proceso de desarrollo. Los enlaces se pueden crear en forma manual, con una participación extensa del analista, o en forma automatizada, donde el analista no participa directamente en la interacción.

**La herramienta ideal (aunque todavía no existe) debe tener la capacidad para volver a conformar la salida de una actividad en una entrada para la siguiente actividad.** Por ejemplo, los diagramas de flujo de datos, las descripciones de procesos, los almacenes y flujo de datos definidos en la fase de análisis, deberían transformarse de manera automatizada en diagramas estructurados, funciones y módulos para el proceso de diseño.

## ¿ DÓNDE PUEDE AYUDAR CASE ?

Si la causa de fallas en la implementación es el insuficiente o impropio análisis de requerimientos y mal diseño ¿ Cuál es el remedio ?. Durante los 60's y 70's varias metodologías estructuradas imponían reglas en las fases de análisis de requerimientos y especificaciones de diseño del ciclo de desarrollo de software.

Estas metodologías presentaban un franco aprovechamiento de la disciplina en el desarrollo de software el cual, siguiendo la metodología, podría reducir en gran medida el riesgo causado por errores en los requerimientos y diseños.

Varias de estas metodologías son Yourdon/DeMarco análisis estructurado (diagramas de flujo de datos) para análisis de requerimientos, Hierarchical Plus and Output (HIPO) para estructuración de módulos y diagramas Warnier-Orr y entidad-relación para modelado de datos. Estas metodologías están dirigidas a diferentes partes del proceso del análisis y diseño. Nuevas tecnologías como el diseño orientado a objetos y reingeniería son guardadas en el arsenal tecnológico de manejadores de software inteligentes que saben cuando y como emplearlas. Ellos se encargan de que diferentes metodologías se usen frecuentemente en algunos proyectos, cada una en su propia especialidad.

***Las metodologías estructuradas formales son la columna vertebral de CASE. Ellas proveen la rigurosa estructura necesaria para una especificación completa y un diseño de la aplicación aceptable.*** [1]

## EVOLUCIÓN DE LA DE LA TECNOLOGÍA CASE

La tecnología CASE surge a mediados de los años setenta, cuando empiezan a aparecer las primeras metodologías estructuradas y se inician las investigaciones sobre entornos de desarrollo.

Se puede afirmar que, salvando las diferencias, los entornos Unix e InterLips constituyen un conjunto de herramientas que, junto a los lenguajes de cuarta generación (4GL), surgidos a finales de los setenta, han influido en la concepción de las herramientas CASE.

A mediados de los años ochenta, el CASE se populariza y surgen las primeras herramientas de documentación y diagramación automática. Es una época en la que explosionan el número de seminarios, cursos, revistas, libros y congresos dedicados al tema. También tiene un papel importante en este desarrollo la aparición de las estaciones de trabajo, que aportan una buena interfaz gráfica asociada a una gran capacidad de proceso y que constituyen la plataforma original de muchos CASE.

A finales de los años ochenta se produce un considerable aumento en la venta de estos productos y empieza la etapa de asimilación de la tecnología, que fracasa debido, fundamentalmente a tres factores:

- Limitaciones de la primera generación de productos.
- Falsas expectativas sobre sus posibilidades.
- Implantación incorrecta.

[1] FISHER, Alan S., Using software development tools.

En definitiva, la tecnología CASE ha experimentado la clásica evolución que sufren aquellos paradigmas que aparecen como panacea universal capaz de resolver todos los problemas del desarrollo de sistemas de información: técnicas estructuradas, inteligencia artificial, lenguajes de cuarta generación y, en estos momentos la orientación al objeto.

En la figura (2.5) se representan en un trazo discontinuo las prestaciones reales de las herramientas, mientras que en trazo continuo se representan las expectativas que poseen los usuarios. En un primer momento se desconoce la tecnología, por lo que las expectativas son nulas, pero en cuanto se empieza a tener noticia de los primeros casos exitosos de su implantación y se empieza a divulgar, sobre todo por parte de vendedores y fabricantes, las expectativas se tornan tan altas que se crea un vacío tecnológico, que suele durar un tiempo considerable, en el que la implantación de la tecnología está destinada a un seguro fracaso, al no responder estas expectativas a la realidad, lo que provoca que la tecnología sea aparcada y descalificada.

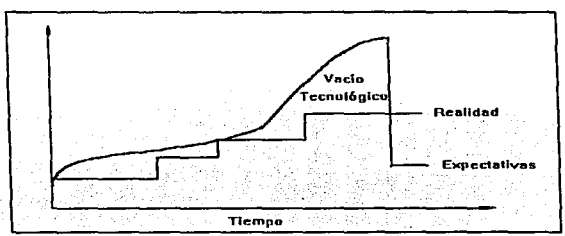



**Figura 2.5 Evolución de la tecnología CASE y sus expectativas**

**Afortunadamente, a mediados de los años noventa, se está entrando en una fase de madurez,** en la que empieza a surgir una "segunda generación" de herramientas (algunas de las cuales ya no aparecen con el nombre de CASE, para no rememorar el

fracaso anterior) que superan gran parte de las limitaciones existentes. Además, **los propios usuarios de las herramientas conocen mejor sus posibilidades y han aprendido a poner unas expectativas más justas sobre éstas, mejorando también los procesos de adopción de metodologías y herramientas.** Es por eso que en la actualidad la tecnología CASE renace de sus cenizas, resultando la mayor parte de sus implantaciones todo un éxito, aunque se requieran para ello una buena administración y tiempo considerable.

Esta nueva generación de herramientas CASE suele aparecer bajo la clasificación de herramientas de desarrollo de aplicaciones, como término más general al que pueden abrirse tanto herramientas sencillas como complejas; las que se dedican a ayudar en la creación de aplicaciones con un aspecto gráfico; herramientas de programación orientada a eventos; de programación orientada a objetos, para desarrollo en arquitecturas cliente/servidor, etc.

De cara al final del siglo, podemos ser optimistas en cuanto al futuro de esta tecnología, que se caracterizará, sin lugar a dudas, por una mejor integración entre herramientas (más abiertas), adecuadas a estándares[12], con una mayor inteligencia y adaptabilidad, incorporando nuevas formas de desarrollo de software que permitan una reutilización efectiva no sólo a nivel de código, sino también de análisis y diseño, con afán de mejorar la calidad y productividad de los sistemas informáticos.

## TAXONOMIA DE UNA HERRAMIENTA CASE

**El despunte tecnológico de las herramientas de ingeniería de software no es nada nuevo.** Más herramientas CASE son usadas en metodologías desarrolladas durante los 60's y 70's. Estas metodologías fueron popularizadas en muchas tiendas comerciales de procesamiento de datos como una forma de manejar sus aplicaciones para reducir el riesgo en el desarrollo técnico.

[12] En algunos países de Europa se han adoptado metodologías estandar para el desarrollo de sistemas de información, un ejemplo es el gobierno Francés que pide como requisito sistemas desarrollados bajo la metodología MERISE

Durante los 80's, estas metodologías de diseño y análisis migraron hacia herramientas CASE tanto como las estaciones gráficas de trabajo y computadoras personales lo permitían. Esta migración se acelera con la complejidad del crecimiento de las tareas del diseño, más allá de la seguridad que podría brindar el uso de lápiz y papel. Por lo tanto, la tecnología CASE esta evolucionando y no revolucionando.

Una sola herramienta o metodología no puede realizar el trabajo entero de diseño y especificación. Ciertamente varias metodologías complementarias son requeridas para apoyar todas las fases del desarrollo de software, desde el diseño de la estructura de datos hasta la especificación de interfases con el usuario. Aunque existe la tendencia a combinar metodologías complementarias dentro de ambientes integrados de herramientas, la herramienta universal permanece viva en el futuro.

## CARACTERÍSTICAS DE UNA BUENA HERRAMIENTA CASE

Más herramientas de propósito general alcanzan varias fases de desarrollo, usualmente el análisis de requerimientos y especificaciones de diseño. Estas herramientas son implementadas típicamente en metodologías de análisis estructurado y son capaces de transformar diagramas de flujo de datos a miniespecificaciones. Con todas estas tecnologías, cierto tipo de herramientas están avanzando más que otras. Esto dificulta el evaluar el estado de avance de CASE. De todas formas, ***existe una tendencia general de las herramientas a cubrir el ciclo entero de desarrollo,***[13] incluyendo la generación automática de código figura (2.6).

Muchos de los vendedores de ahora, están reconociendo que sus herramientas no pueden enfocarse solamente a una o dos fases del proceso de desarrollo.[14] En el futuro, las herramientas exitosas deberán tratar con todas las fases del ciclo de desarrollo, y resolver la dificultad fundamental de generar software libre de errores directamente de las especificaciones del diseño

[13] FISHER, Alan S., Using software development tools.
[14] Es por esto que ahora la mayoría de los fabricantes de CASE se enfoque en construir herramientas que a través de módulos de desarrollo abarquen todo el ciclo de vida

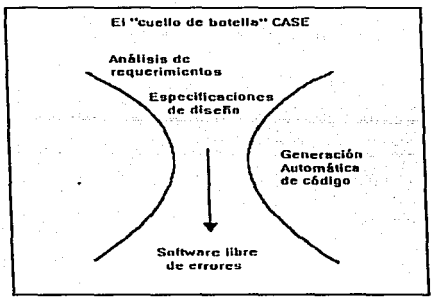



**Figura 2.6 El principal "cuello de botella" en la tecnología CASE es la abilidad para generar software directamente de las especificaciones de diseño**

Fundamentalmente, las herramientas CASE deben satisfacer algunos criterios para ser adoptadas como parte de un equipo de herramientas para desarrollo de software. Cumplir estos criterios es esencial para la adaptación dentro de las prácticas normales de una organización. Las herramientas deben:

**Simplificar.** El mayor objetivo de la tecnología CASE es descomponer requerimientos y diseños en componentes manejables. Su función es simplificar, explicar y reducir.

**Servir a varias audiencias.** Las herramientas CASE para las fases de requerimientos y diseño del ciclo de desarrollo sirven a varios propósitos. Sus resultados deben ser entendibles para los usuarios finales y los responsables del desarrollo de software en la organización. En suma las herramientas deben proveer real valor al diseño.

**Ahorrar tiempo y dinero.** Usar CASE deberá ser barato y más eficiente a largo plazo que construir el sistema usando métodos tradicionales. Las herramientas reducirán substancialmente los esfuerzos de implementación y mantenimiento para producir especificaciones y diseño de gran calidad.

**Producir diseños confiables.** Las especificaciones y diseño generados por herramientas CASE deben articular exacta y consistentemente las características de software y los componentes a ser construidos. Cada requerimiento en la implementación de software debe ser verificado y rastreado en la documentación de los requerimientos.

**Soportar cambios.** Las especificaciones y diseño producidos por una herramienta CASE deben adaptarse conforme cambian los objetivos del proyecto. Un diseño que cae fuera de sincronización fomenta el abandono del código y causa pérdida de tiempo a los diseñadores en futuras mejoras al software.

**Mostrar, no describir.** Las buenas herramientas CASE presentan visualmente especificaciones y diseños de información. Para los usuarios finales y diseñadores, es mucho más fácil el comprender una ilustración gráfica que leer varias páginas de descripción en un texto.

## ¿ HACE CASE SOFOCAR LA CREATIVIDAD ?

Como en toda empresa de software, existen factores humanos que considerar. Más profesionales de software que se consideran a si mismos como artesanos -expertos que han resistido al cuidadoso rigor de aprender el arte y práctica del desarrollo de software-. Sin embargo **la tecnología Case no reemplaza el verdadero talento e ingenio.**

Superficialmente la tecnología CASE parece reducir el aura de artesanos por dar herramientas que estimulan el seguir un régimen estricto de desarrollo de software. ¿Quizá las herramientas CASE restringen la creatividad de programación? En grandes organizaciones que desarrollan software donde el uso de herramientas particulares es frecuentemente obligatorio, estos productos se convierten en importantes.

Más profesionales de software creen que la creatividad en la ingeniería de software es la esencia de la arquitectura del programa y del diseño de una buena solución de software que un grupo de usuarios finales necesita. En efecto muchos diseñadores de software deberán estar de acuerdo con que **lo primero y más importante, en una aplicación de software es satisfacer las necesidades del usuario final.** Sólo cuando las

necesidades funcionales son realizadas, el programador se ve y se siente como un verdadero artesano del software.

La verdadera artesanía reside en adaptar la interfases con el usuario, en la organización de las estructuras internas de datos y algoritmos. Aunque la construcción física del software -el código- provee al programador la oportunidad de demostrar su habilidad individual, el verdadero genio de un programador es reflejado en cómo el usuario final es servido.

CASE proporciona herramientas poderosas para complementar nuestros implementos manuales. Las herramientas CASE mejoran nuestra creatividad permitiéndonos construir diseños adecuados en una menor cantidad de tiempo. Una herramienta para diagramas de flujo de datos elimina la necesidad del dibujo a mano, permitiendo al analista de software más libertad para probar alternativas de arquitectura.

Un equipo de programación de interfases de usuario permite al ingeniero de software enfocarse en factores humanos del diseño de la interfase, permitiendo a la herramienta generar un nuevo trabajo de implementación basado en la reutilización de código. Sin embargo, los programadores deben seguir utilizando la habilidad en la "artesanía" de utilizar módulos en muchas situaciones y estos no deben abandonar sus herramientas manuales, ya que la tecnología CASE no está cerca de la fase donde el 100% del código de la aplicación pueda ser generado automáticamente. Pero la tecnología puede proveer una ayuda en muchas situaciones, convirtiéndola en un poderoso amplificador de la creatividad humana.

# CAPITULO III:

## Adopción de una herramienta CASE

Una vez comprendido el concepto de la tecnologia CASE, sus caracteristicas, clasificación y evolución, es necesario abordarla desde el punto de vista de la organización interesada en adquirirla. Para ello es preciso conocer los beneficios que se pueden obtener de la compra de una herramienta, así cómo sus debilidades y qué hay con respecto a la calidad del software generado mediante CASE. Esto en un momento dado puede dar la impresión de que es un proceso sencillo, pero no lo es, ya que si se desea el éxito se requiere de una metodologia que proporcione el proceso de asimilación que debe seguirse al adquirirla y el costo que esto lleva consigo. Básicamente este es el punto fundamental del capitulo.

## BENEFICIOS QUE SE PUEDEN OBTENER DE LAS HERRAMIENTAS CASE

***Con la implantación progresiva, sistemática, y correctamente evaluada de las herramientas CASE en una empresa, se puede llegar a obtener una serie de ventajas que dependerán de cada caso en particular***[1] figura (3.1). El primer requisito para llegar a alcanzar los objetivos deseados comprende la utilización de una metodologia, pues no sirve de mucho que la labor de análisis y diseño se realice con una computadora, si no se tiene de antemano una base metodológica adecuada. El método debe ser lo primordial. Una vez superada esta etapa, se podrán considerar las siguientes ventajas:

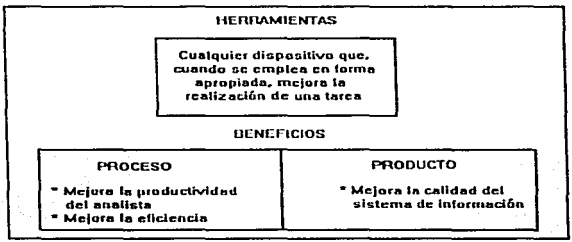



**Figura 3.1 Beneficios obtenidos con el uso de herramientas en el desarrollo de sistemas de información.**

· **Comprensión.** Relaciones más estrechas entre analistas y usuarios. Mejor comprensión por parte de estos últimos de los tipos de datos y procesos disponibles para las aplicaciones.

· **Homogeneización.** Se consigue al utilizar una técnica estándar para todos los proyectos de la empresa, o entre los diferentes grupos que participan en un proyecto. Como resultado, se conseguirá un aumento importante de la transparencia y compresión de las especificaciones técnicas, y una mayor precisión y eficacia de ésta; además de un fuerte incremento en la reutilización de las especificaciones. Unas herramientas

1 DAVIS, William S., Herramientas CASE.

homogéneas y de alto nivel permiten construir las aplicaciones independientes a la máquina en la que se desarrolla, o la máquina destino.

· **Continuidad y coherencia.** Todas las faces se realizan en el mismo soporte, o en soportes compatibles; y todos los elementos son coherentes entre una fase y otra. A través del diccionario de datos, las herramientas controlan automáticamente la definición de los datos, la utilización de sinónimos, etc. La utilización de un diccionario sirve de vínculo común entre todos los sistemas y aplicaciones, asegurando que tengan las mismas informaciones y las mismas características.

· **Elaboración de juegos de ensayo.** En estos casos la herramienta facilita, para cada fase, la definición de las pruebas y una ayuda para valorar su complejidad, puede proporcionar por ejemplo, el porcentaje de código probado.

· **Cálculo de tiempos y cargas.** Simulación. En muchos sistemas (sistemas de control de tiempo real, de teleproceso, etc.) es necesario realizar un estudio de los tiempos de respuesta, volúmenes de ocupación, tiempos de transmisión, etc. La posibilidad de plasmar estos requerimientos en el diseño, y de realizar pruebas simulando su comportamiento, suministra una información importante para conocer cómo se comportará el sistema, y para realizar las prevenciones necesarias para mejorar su comportamiento.

· **Mayor velocidad de programación** o generación automática de código, si procede.

· **Mayor calidad en el código.**

· **Aumento del nivel de independencia de los programas respecto a sus autores.**

· **Documentación gráfica.** Consiste en un conjunto de gráficos (DFDs, diagramas HIPO, ERDs, etc.) que forman el documento técnico (de análisis, diseño, etc.).

· **Documentación de referencias cruzadas.** Suministran información acerca de las relaciones existentes entre los distintos componentes de una aplicación. De esta forma es posible conocer qué ficheros o rutinas se utilizan en los programas; qué pantallas o registros aparecen en un determinado programa, etc. Esta información es básica para poder evaluar el impacto de los cambios en el mantenimiento.

- **Documentación técnica.** Las herramientas CASE permiten la creación automática de manuales de documentación de las aplicaciones. Algunas herramientas generan la documentación con formatos estándar.

- **Mantenimiento efectivo.** En la etapa de mantenimiento es necesario realizar el control de los cambios, la obtención de las versiones de los programas, el control de las nuevas versiones, etc. La utilización de las herramientas de ingeniería de software en las fases anteriores al mantenimiento aporta la documentación y orden suficiente para realizar el mantenimiento efectivo y fiable.

- **Mejoras en la explotación de los sistemas.** Desde hace ya algunos años, y en ciertos entornos, existen herramientas que permiten optimizar los diferentes parámetros de rendimiento de un sistema: por ejemplo de los discos, impresoras, comunicaciones, cintas, backups, etc., pero al ser herramientas casi siempre específicas para cada entorno, no tienen una aplicación general.

- **Ventajas para la reutilización del software.** Con la ayuda de las herramientas adecuadas, se puede considerar la reusabilidad del software, es decir, la creación y reutilización de los componentes que forman el software. La reutilización permite el abaratamiento de costos, el aumento de la productividad y la mejora de la calidad de los productos. Si se reutilizan todos los elementos que intervienen en el desarrollo de una aplicación, desde su definición hasta la explotación, se evitará la repetición de esfuerzos, y se reducirán los recursos y tiempos necesarios. Los componentes del software se pueden catalogar para su fácil referencia, estandarizar para poder utilizarlos en cualquier aplicación.

- **Facilidad para la revisión de aplicaciones.** La experiencia muestra que una vez que las aplicaciones se implantan, se emplean por mucho tiempo. Las herramientas CASE proporcionan un beneficio substancial para las organizaciones al facilitar la revisión de las aplicaciones. Contar con un depósito central, agiliza el proceso de revisión ya que éste proporciona bases para las definiciones y estándares para los datos. Las capacidades de generación interna, si se encuentran presentes, contribuyen a modificar el sistema por medio de cambios en las especificaciones más que por ajustes al código fuente.

· **Soporte para el desarrollo de prototipos de sistemas.** En general, el desarrollo de prototipos de aplicaciones toma varias formas. En ocasiones se desarrollan diseños para pantallas y reportes con la finalidad de mostrar la organización y composición de datos, encabezados y mensajes. Los ajustes necesarios al diseño se hacen con rapidez para alterar la presentación y las características de la interfase. Sin embargo, no se prepara el código fuente, de naturaleza orientada hacia procedimientos, como una parte del prototipo. Muchas herramientas CASE soportan las primeras etapas del desarrollo de un prototipo. Muy pocas brindan apoyo durante todo el proceso de desarrollo de este.

· **Mejora la habilidad para satisfacer los requerimientos del usuario.** Es bien conocida la importancia de satisfacer los requerimientos del usuario, ya que esto guarda relación con el éxito del sistema. De manera similar, tener los requerimientos correctos mejora la calidad de las prácticas de desarrollo. Parece ser que las herramientas CASE disminuyen el tiempo de desarrollo, una característica que es importante para los usuarios. Las descripciones gráficas y los diagramas, así como los prototipos de reportes y la composición de las pantallas, contribuyen a un intercambio de ideas más efectivo[2].

## DEBILIDADES DE CASE

***Las herramientas CASE tienen puntos débiles significativos, que van desde la confiabilidad en los métodos estructurados hasta su alcance limitado,***[3] los cuales amenazan con minar los beneficios potenciales descritos con anterioridad.

### Confiabilidad en los métodos estructurados

Muchas herramientas CASE están construidas teniendo como base las metodologías del análisis estructurado y del ciclo de vida del desarrollo de sistemas. Por sí sola, esta característica puede convertirse en la principal limitante ya que no todas las organizaciones emplean métodos de análisis estructurado.

---

2 Generalmente los beneficios de utilizar la tecnología CASE se notan después de la implantación del sistema: en su éxito, en la facilidad para realizar modificaciones, en la buena documentación generada, en su fácil mantenimiento o en la reutilización de código

3 SEEN, James A. Análisis y diseño de sistemas de información.

Los métodos estructurados, introducidos en la década de los sesentas, fueron muy elogiados por su habilidad para mejorar la exactitud de los requerimientos específicos de las aplicaciones. El nivel de conocimiento de los métodos estructurados es alto entre los profesionales de sistemas -de acuerdo con algunas estimaciones (Yourdon), casi el 90% de todos los analistas está familiarizado con estos métodos- Aproximadamente la mitad de todas las organizaciones en Estados Unidos han utilizado alguna vez estos métodos. A pesar de lo anterior, **si la organización o el analista no utilizan los métodos propios del análisis estructurado y tampoco desean considerar su uso, entonces el valor de CASE disminuye.** En algunos casos, los analistas evitan del todo emplear herramientas CASE.

**Falta de niveles estándar para el soporte de la metodología**

**Aún no aparece un conjunto "estándar" de herramientas CASE.** Por lo tanto, se debe tener precaución al seleccionar una herramienta de este tipo.[4]

Existen dos significados para las palabras "soporte de la metodología" Una herramienta puede:

1) dar soporte a los diagramas que emplea una metodología.
2) soportar e imponer la metodología, reglas y sus procesos.

Las herramientas CASE que existen en el presente, tienen una de las siguientes características:

- Son independientes de la metodología
- Permiten que los usuarios definan sus propias metodologías, reglas y estándares
- Soportan una metodología
- Soportan las metodologías más diseminadas

En todas ellas existen ciertos compromisos. Las herramientas que son independientes de la metodología, no pueden fomentar el uso de las reglas y estándares de la misma. Estas herramientas quizá proporcionen los componentes de una metodología, pero no el marco de referencia, reglas y procedimientos que en realidad constituyen el núcleo de ésta. Aunque se pueden llevar a cabo acciones básicas para la validación de diseños y diagramas

4 Hay que recordar, que una de las principales causas del fracaso de la primera generación de herramientas CASE, fué el adoptar cualquier herramienta suponiendo que resolvería los problemas de desarrollo.

para detectar componentes faltantes, éstas son sólo funciones mecánicas. Por otra parte, esta clase de herramientas no pueden proporcionar ayuda metodológica o pedir al usuario que realice tareas necesarias para la metodología que aún están sin terminar. Estas herramientas mejoran la productividad al efectuar tareas tediosas y de documentación, aunque ellas no pueden asegurar buenos resultados. Desde el punto de vista funcional, las capacidades que brindan para garantizar la calidad son mínimas.

Las herramientas que proporcionan un soporte limitado a una sola metodología pueden forzar el uso riguroso de reglas, procedimientos y estándares de ésta; además brindan ayuda sensible al contexto y bases de conocimiento que ofrecen asistencia experta. Sin embargo, entre más metodologías soporte una herramienta, existe la posibilidad cada vez mayor de que la seguridad y ayuda que ésta ofrece sea menor.

**Conflictos en el uso de diagramas**

**Las herramientas difieren en el uso que hacen de los diagramas.** Algunas son herramientas exclusivamente para gráficas, que se abocan al dibujo de diagramas para el análisis de entrada y salida de datos. Este tipo de herramientas pueden restringir ya sea el proceso de desarrollo normal seguido por una organización o el estilo particular de trabajo de los analistas.

Otros vendedores de herramientas consideran los diagramas como documentación y aceptan entradas por medio de formas o lenguajes de especificación y, en ocasiones, en forma gráfica por tanto se debe tener cuidado cuando se selecciona una herramienta para apoyar los métodos existentes dentro de una organización.

**Diagramas no utilizados**

En general los productos CASE emplean gráficas para modelar y generar informes sobre el análisis y desarrollo durante todo el proceso de desarrollo de sistemas. Una de las afirmaciones de los vendedores de herramientas es que las presentaciones gráficas y la documentación mejoran la comunicación entre los miembros del equipo de desarrollo, proporcionan una calidad mayor de la entrada proporcionada por el cliente y mejoran la productividad de desarrollo de software.

Sin embargo, los investigadores han encontrado que, **en algunos casos, las herramientas gráficas, automatizadas o manuales, no se emplean del todo.** O tal vez no se utilicen en la forma en que deberían emplearse. Por otra parte, algunos analistas

prefieren para algunas tareas un lenguaje estructurado o descriptivo. Muchos profesionales de los sistemas de información no hacen uso de herramientas gráficas en el desarrollo de software; más bien las emplean para automatizar la producción de informes y documentación del sistema, como los diagramas de flujo utilizados por los programadores para documentar un programa una vez terminado éste.

**Función limitada**

Aunque una herramienta puede apoyar varias fases del ciclo de vida de desarrollo de sistemas o adaptarse a diferentes metodologías de desarrollo, **por lo general su enfoque primario está dirigido hacia una fase o método específico.** Por ejemplo, los encargados de desarrollar un nuevo producto pueden afirmar que éste apoya todo el proceso de análisis y diseño. Sin embargo, las capacidades de comprobación y verificación de errores del producto quizá sean más rigurosas ya sea en el área de análisis o en la de diseño, pero no en ambas. Algunos productos están dirigidos hacia el diseño de bases de datos para la organización y al desarrollo de aplicaciones que giren entorno a las bases de datos, omitiendo el soporte para pantallas de presentación visual, los informes sobre requerimientos o las necesidades de seguridad. Algunos productos capaces de generar el código hacen mayor hincapié en el desarrollo de prototipos como el principal método de desarrollo de sistemas. Muchas herramientas para la fase de desarrollo recalcan el mantenimiento y la reestructuración del código, pero ofrecen un soporte débil durante la fase de análisis para la determinación y especificación de requerimientos.

**Alcance limitado**

Aunque muchas herramientas basadas en computadora incluyen la capacidad de verificar las especificaciones para determinar su completes o consistencia, virtualmente no llevan a cabo ningún análisis de los requerimientos de la aplicación. Por tanto, el alcance de las actividades de desarrollo asociado con las herramientas existentes es bastante limitado.

**La mayor parte de los productos CASE describe (documenta) pero no analiza.** De poca ayuda es proporcionar una regla de inclusión en los mejores enfoques y una regla de exclusión para los que son pocos satisfactorios. No ofrecen o evalúan soluciones potenciales para los problemas relacionados con sistemas. Y tampoco existe una garantía clara para que dos analistas que utilicen los mismos métodos aplicados a información idéntica, formulen recomendaciones igualmente aceptables.

**Las tareas humanas siguen siendo críticas**

La tecnología CASE ofrece herramientas que soportan las funciones de modelado, verificación, manejo de datos y de utileria que son necesarias para mejorar la productividad del desarrollo. Sin embargo, las herramientas deben estar en manos de personas con experiencia y deben "adaptarse" a la arquitectura de la información así como a las metodologías de desarrollo utilizadas por la organización. Por otra parte, las actividades críticas no son el desarrollo de gráficas que documenten al sistema existente sino que son aquellas tareas donde las personas interactúan entre si. determinación y verificación de requerimientos con el usuario. A medida que sean automatizadas las funciones de modelado y búsqueda de errores, la responsabilidad del éxito en un sistema caerá cada vez más sobre aquellos que especifican los requerimientos de información. Obtener y comprender los requerimientos son tareas realizadas por los seres humanos y lo más probable es que se continúe de tal forma **La importancia del elemento humano en el proceso de desarrollo es y será fundamental.**

## CALIDAD Y CASE

***La calidad de los productos y servicios de software es una necesidad creciente de todo tipo de usuario de los mismos,***[5] y por tanto es un factor de competitividad de las empresas que lo crean, ya que han de satisfacer las necesidades de sus clientes no sólo para continuar en el mercado, sino, además, para conseguir la superioridad y el liderazgo, como meta empresarial.

Sin embargo, no sólo los creadores de software han de satisfacer los factores de calidad que les demanda el mercado, también los usuarios del software, que crean, a su vez, nuevos productos y servicios, tanto para uso interno como externo a su organización, tienen la necesidad de cumplir los requerimientos de calidad que les son exigidos.[6] Hay pues una cadena de calidad que hay que vigilar, ya que, como toda cadena, siempre se rompe por el eslabón más débil.

5 PIATTINI Y DARYANANI, Elementos y herramientas en el desarrollo de sistemas de información.
6 Actualmente la demanda en el mercado exige que los productos de software satisfagan las necesidades de calidad de los usuarios y CASE aporta varios elementos para elevarla

Ganar la batalla de la calidad del software hoy, y en un próximo futuro, pasa por alcanzar la comprensión y el dominio de los procesos constructivos, por la implicación de los usuarios, por la gestión rigurosa de los procesos y por la tecnología de apoyo, donde se incluye genéricamente CASE.

La complejidad de los sistemas de software es cada día mayor, y se está mostrando como un factor esencial de la calidad a las industrias y servicios que lo producen y operan. Dicho factor precisa dominarse a partir de métodos potentes de análisis y diseño, que han de verse apoyados por herramientas automáticas que facilitan su implementación. De otra parte, **existen identificados un conjunto de criterios de calidad aplicables a todo tipo de software: funcionabilidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad** que han sido aceptados por ISO (Organización Internacional de Normalización).

El papel de la administración de la calidad en relación a la tecnología CASE es, de una parte, negociar la relación y uso de las herramientas específicas para los procesos de desarrollo específicos.

La adecuación de las herramientas no sólo proporciona la consistencia, estructura y disciplina requeridas en los complejos programas actuales. También asegura un control de los procesos que generalmente se escapan de las manos cuando no se dispone de cierto grado de automatización. De otro lado, las actividades de control de la calidad deben utilizar también en parte la tecnología usada por la ingeniería para poder evaluar y medir los procesos y prestaciones de los productos de dicha ingeniería. Aspectos tales como el análisis de tendencias, determinación de métricas, aumento de la visibilidad en la administración, etc., son los elementos que para su ejecución debe verse apoyados por herramientas CASE.

La gestión de la calidad de software puede tener un papel fundamental en el uso de las herramientas disponibles actualmente y en promover una concurrencia sobre el uso de los avances de la tecnología CASE a fin de mantenerse en la vanguardia del mercado y seguir siendo competitivos.

**La tecnología CASE se revela como elementos imprescindibles para conseguir los niveles requeridos para cada uno de los criterios de calidad establecidos,** que son cada día más elevados.

El desarrollo de la tecnología CASE está facilitando la implantación sistemática de los planes de calidad de proyectos específicos de desarrollo y de programas de mejora continua de la calidad a través de modelos publicados y ampliamente aceptados para la evaluación de la capacidad. Por ejemplo, el modelo CMM (Capability Maturity Model), del instituto de ingeniería del software "SEI", de la Universidad Carnegie-Mellon, la mejora de procesos según el modelo SQPA (Software Quality and Productivity Analysis), de Hewlett Packard, o el modelo de aseguramiento de la calidad descrito en ISO 9001, que sirve con el fin de conseguir la confianza adecuada de los clientes donde la organización suministradora tiene las capacidades requeridas de diseño, desarrollo, instalación y servicio posventa. El CMM está sobre todo orientado desde el punto de vista de la ingeniería de software, y está siendo difundido internacionalmente junto a otros varios que tienen objetivos similares.

Las normas de la serie ISO 9000 están reconocidas por la mayor parte de los países de la comunidad internacional, sobre todo porque se han adaptado como normas europeas de sistemas de calidad, derogando las normas nacionales existentes. Las empresas certificadas frente a ISO 9001, de una parte pueden acceder a determinados mercados que exigen dicha certificación, y de otra, representa para ellas un valor añadido de carácter comercial en su presentación al mercado.

Esfuerzos específicos realizados en España, en este caso por el MAP ( Ministerio para las Administraciones Públicas), que están orientados a la mejora de la calidad del software adquirido por las Administraciones Públicas, son el documento titulado Plan General de Garantía de Calidad Aplicable al Desarrollo de Equipos Lógicos y la metodología Métrica. Hay que destacar que tales documentos están siendo ya pedidos como requisitos en las peticiones de ofertas por algunos organismos de las Administraciones Públicas.

Muchas de las herramientas utilizadas por el control de la calidad para llevar a cabo su programa de actividades son manuales, por ejemplo la lista de comprobación y las normas. **En general, las herramientas automáticas han sido orientadas hacia la productividad del software, pero la tendencia actual está dirigiéndose hacia la construcción de la calidad en el software.** Hay herramientas que permiten mejorar la calidad general de la documentación, otras facilitan la implantación de reglas de diseño, o facilitan el cumplimiento frente a ciertas normativas, etc.

Por ejemplo, hay herramientas gráficas de diseño que aseguran la adherencia a las normas de representación del diseño. Así pues, el uso de tales herramientas consiguen obtener de forma automática características de calidad asociadas a la normalización. La utilización de CASE, en cualquier caso, no es garantía de conseguir los requisitos de calidad mediante reglas de diseño, pero dan apoyo iterativo al proceso de diseño que facilita al diseñador la consecución de aquellos requisitos.

Dependiendo del tipo de industria, el uso de herramientas automáticas ha tenido, o está teniendo, un mayor grado de penetración. La criticidad del software respecto a la misión o la seguridad, por ejemplo, determina la inclusión, tanto en las actividades de ingeniería de software como en las de control de la calidad, del uso de herramientas más o menos complejas. Los analizadores lógicos, de base de datos, de complejidad, de normas, de consistencia, de interfaces, de métrica y de pruebas son ejemplos de un conjunto complementario de herramientas que sirven para realizar automáticamente verificaciones y evaluaciones del software desde una perspectiva de gestión de la calidad.

***Hay que indicar que las inversiones en el desarrollo, adquisición y explotación de estas herramientas es, en general, siempre rentable***[7].

## TENDENCIAS Y FUTURO DE CASE

Desde su aparición en los años sesenta, las herramientas CASE han experimentado una considerable evolución, dando respuestas a una buena parte de las necesidades del desarrollo de sistemas de software. Actualmente se soportan las tareas más rutinarias, especialmente la elaboración y el mantenimiento de la documentación, así como las transformaciones de una fase a la siguiente en el ciclo de la vida y la generación de código. Además CASE trae consigo un cambio en la mentalidad del ingeniero de software, y una nueva forma de trabajar figura (3.2).

7 PIATTINI Y DARYANANI. Elementos y herramientas en el desarrollo de sistemas de información.

| ANTES DEL CASE | DESPUES DEL CASE |
|---|---|
| * Entasis en la codificación y en las pruebas<br>* Trabajo sobre el papel<br>* Modelos sobre el papel<br>* Codificación manual<br>* Documentación manual<br>* Pruebas del código<br>* Mantenimiento del código | * Entasis en el análisis y diseño<br>* Trabajo sobre la computadora<br>* Creación y simulación de prototipos<br>* Generación automática de código<br>* Generación automática de documentación<br>* Pruebas de las especificaciones y del diseño<br>* Mantenimiento de las especificaciones y del diseño |

**Figura 3.2 Modos de trabajo del ingeniero de software**

Dependiendo del tipo de herramienta, también se ofrecen ciertas facilidades para la verificación y prevención de errores, comprobandose una serie de reglas, lo que contribuye a la calidad del sistema.

Quedan, sin embargo, muchos aspectos por resolver dentro de las actividades de mejora del software, muchas de las cuestiones planteadas surgen precisamente de la experiencia adquirida con la primera generación de herramientas, como señala NORMAN y FORTE (1992): **Cuanto más dependiente de CASE se vuelven los ingenieros de software y los profesionales de sistemas, mayores son las demandas sobre la tecnología.**

Existen dos caminos identificados por SCHWARTZ (1992) dentro del mundo de la tecnologia avanzada:

· La actitud innovadora, que consiste en ser pioneros a la hora de adoptar una nueva tecnologia, lo que entraña cierto riesgo, pero también ofrece la posibilidad de una recompensa mayor. Si tomamos este camino, trabajamos con algo nuevo que está empezando a surgir y podemos avanzar y madurar al mismo tiempo que la tecnologia, lo que nos puede proporcionar ventajas competitivas considerables.

- La actitud conservadora, en la que se prefiere seguir los pasos que ya han realizado otros con tecnología suficientemente madura, adoptando la nueva tecnología cuando ya no quede más remedio, por la propia inercia del mercado. Con esta actitud se corren menos riesgos, pero nos vemos obligados a ir unos pasos por detrás de los líderes.

El objetivo de esta parte es presentar las tendencias más importantes que se vislumbran dentro de la tecnología CASE, tendencias que nos ofrecen la posibilidad de adoptar una actitud innovadora en el desarrollo de sistemas de información.

En primer lugar, un desafío que tienen tanto los fabricantes como los organismos de estandarización es **conseguir una mejor integración de las distintas herramientas que deberán ser más abiertas, soportando una mayor variedad de técnicas y metodologías.**

Por otra parte, es necesaria una mayor investigación para soportar el desarrollo de sistemas distintos de los clásicos de gestión (para los que existen multitud de herramientas y metodologías), como pueden ser los sistemas en tiempo real.

También dentro de las tendencias cabe destacar la adaptación de las herramientas CASE, que surgieron en estaciones de trabajo (sobre todo en entorno UNIX), a los nuevos sistemas operativos que están apareciendo.

Una línea de investigación muy importante que empieza ya a aparecer en el mercado es **la integración de sistemas expertos en las herramientas CASE.** Estos sistemas expertos se encargan de guiar al usuario a lo largo de todo el ciclo de vida, aconsejando la utilización de la metodología y verificando la consistencia del desarrollo. Este nuevo entorno habitable, como señala McCLURE (1992), se caracteriza por ser:

- Orientado al usuario
- Reactivo
- Corrector
- Enseñador
- Conductor

También cabe resaltar el soporte que ofrecen las herramientas CASE para el diseño de bases de datos, aspecto que cada día cobra mayor importancia debido a la gran difusión experimentada por los SGBD.

Otro reto que se debe afrontar en el desarrollo de sistemas de información es comprender el propio proceso de desarrollo, para ser capaz de modelarlo y adaptarlo a las distintas empresas. Es quizás esta palabra, **adaptación, una de las características clave que ha de poseer el CASE del futuro.**

Todo ello llevará a herramientas CASE más dinámicas, NORMAN y FORTE (1992), que se completarán con: ejecución de especificaciones del sistema, analizadores de código dinámicos, entornos de prototipado rápido, simuladores, etc., lo que producirá una nueva generación de herramientas, que contribuirá a la mejora de calidad y productividad de los sistemas de software, transformando, con toda probabilidad, la manera en la que concebimos el proceso de desarrollo.

## IMPLANTACION DE CASE (Adopción de la tecnología CASE)

La tecnología CASE está experimentando considerables avances en distintas áreas, solventando bastantes de los defectos que presentaba la primera generación de productos. **Sin embargo, el principal desafío de esta tecnología sigue siendo su adopción por parte de los profesionales y las empresas.**

De hecho se calcula que en la práctica se abandona la tecnología CASE en un alto porcentaje: el 70% de las herramientas y técnicas se deja de utilizar un año después de su introducción, el 25% lo emplea sólo un grupo o una persona dentro de la empresa, mientras que el 5% restante si se emplea ampliamente, pero no al 100% de su capacidad.

No existe un método infalible para evitar esta situación y asegurar el éxito en la implantación de CASE; lo que no cabe duda es que una buena planificación, una cuidada gestión de expectativas, una sólida formación, una considerable inversión (tanto en los productos CASE como en la infraestructura hardware/software necesaria), junto a la participación de los directivos, aumenta la posibilidad de conseguir un resultado aceptable, que, de todas formas, no ha de plantearse a corto plazo.

Aquí se presentan, en forma general, las principales causas del fracaso en la implantación de CASE, proponiendo un plan para su adopción (plan que puede ser condición necesaria, pero no suficiente, para que esta adopción sea exitosa) y los costos asociados a la misma.

## CAUSAS DEL FRACASO EN LA ADOPCION DE CASE

Multitud de consultores han abordado las causas del fracaso en la adopción de la tecnología CASE; ***bajo nuestro punto de vista, las podemos agrupar en tres partes:***[8]

### Deficiencia de la propia tecnología

Un gran número de empresas que empezaron a utilizar herramientas CASE en los años ochenta, posteriormente las abandonaron debido a sus inconvenientes, entre los que se pueden destacar:

- Soporte parcial del ciclo de vida, lo que permite automatizar sólo parte de las actividades de desarrollo, mientras que otras se siguen realizando de forma tradicional.

- Incompatibilidad entre herramientas, incluso entre distintas versiones de la misma herramienta que no siempre se encuentran sincronizadas en todas las plataformas sobre las que actúan.

- Escasa integración entre herramientas y el resto del entorno: Sistemas Generadores de Bases de Datos, lenguajes de cuarta generación, generadores de informes, etc.

- Poca fiabilidad en el vendedor/distribuidor, ya que algunas empresas de CASE son relativamente pequeñas y corren el peligro de desaparecer o ser absorbidas.

- Escasa documentación generada por la herramienta.

8 PIATTINI Y DARYANANI, Elementos y herramientas en el desarrollo de sistemas de información.

- Gran abundancia de herramientas, señalada muchas veces como inconveniente, ya que produce una especie de bloqueo a la hora de adquirir una herramienta.

- Funcionamiento deficiente en entornos multiusuario, ya que muchas herramientas nacieron para computadoras personales y no han tenido una versión disponible para entornos UNIX o redes de área local hasta bastante tiempo después. Incluso aquellas que si soportan esta forma de trabajo, no siempre gestionan adecuadamente la concurrencia entre diferentes desarrolladores.

- Poca capacidad de adaptación.

- Un alto costo, no sólo en la herramienta si no en la plataforma que ésta conlleva.

**Todas estas deficiencias pueden ser superadas actualmente**[9], en mayor o menor medida, **evaluando varias herramientas**, considerando, si fuera posible, un cambio en el hardware/software utilizado, intentando cuantificar el costo de la no adopción (con especial énfasis en el mantenimiento) y valorando los beneficios que CASE puede aportar como tecnología estratégica.

**Deficiencias en la aplicación de la tecnología a los problemas**

Otra causa del fiasco se debe a la utilización de herramientas CASE en problemas para los que no están preparadas, debido a que:

- Soportar una sola metodología y pretender emplearla para construir sistemas en otra diferente.

- No soportan la técnica más adecuada

- Metodologías y herramientas que funcionan relativamente bien en proyectos pequeños o medianos, pueden fracasar en proyectos grandes.

9 En México no tuvo tanta difución la primera generación de herramientas y por esta causa no se adquirió la experiencia suficiente en los problemas de adopción, sin embargo se deben tomar los errores cometidos por otros para evitarlos

· La selección se centra solo en factores técnicos, por lo que la herramienta resulta insuficiente para los aspectos relativos a la gestión que todo desarrollo lleva consigo.

**Las medidas más eficientes para afrontar estos problemas pueden ser: comprender y analizar los distintos tipos de metodologías y herramientas existentes** (junto a su escalabilidad), utilizando las herramientas adecuadas a cada problema, lo que supone un gran esfuerzo en formación e inversión en consultoría.

**Deficiencias en la propia organización**

A pesar de las deficiencias citadas anteriormente, la mayor parte de los fracasos en la adopción de herramientas CASE son debido a deficiencias de la propia organización. En definitiva, la adopción de la filosofía CASE es, como la transferencia de cualquier otra tecnología, un problema más cultural que tecnológico.

Las causas de fracaso más notables en esta área son:

· Actitud por parte de los directivos, que pretenden introducir la tecnología CASE como la panacea o salvación de todos los males del desarrollo, sin contar con una base metodológica.

· Infravalorar el esfuerzo requerido, no sólo el económico, sino también el de formación y aceptación por parte del personal.

· Incapacidad para encontrar las metodologías y herramientas adecuadas al nivel de madurez de la organización.

· Inadecuada formación, que a veces no existe o se limita a que el primer estudiante forme a los demás.

· No medir la productividad ni la rentabilidad de la tecnología.

**Estas deficiencias se pueden superar con una visión adecuada de las expectativas**, siendo realista (conociendo la cultura de la empresa y su historia frente a los cambios tecnológicos) y con una buena administración.

## PLAN PARA LA ADOPCION DE CASE

**La adopción de CASE debe abordarse como uno de los proyectos más importantes de la empresa, estableciendo un plan detallado de implantación.**[10]

La tecnología CASE como el desarrollo de sistemas tiene un ciclo de vida[11] figura (3.2); a continuación comentaremos las principales actividades a realizar en cada una de sus fases.

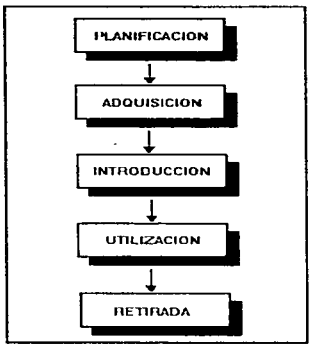



**Figura 3.2 Ciclo de vida de la tecnología CASE**

### Planificación

En primer lugar, **se debe elegir un grupo de personas capaces de llevar a cabo la implantación**, que será tambien responsable de vender la tecnología al resto de la empresa.

Este grupo de personas deberá establecer los estándares y procedimientos a seguir durante todo el proyecto, el calendario previsto y el personal que participará en todas las actividades. Además el grupo CASE deberá estudiar los sistemas de información que se

10 El adquirir una herramienta CASE, incluye un estudio partiendo de la elección, pasando por un periodo de asimilación y entrenamiento, para una optima explotación y finalizando con el reemplazo
11 PIATTINI Y DARYANANI, Elementos y herramientas en el desarrollo de sistemas de información.

desean desarrollar a mediano plazo, más adelante se hablará de la forma de como se reúne un equipo de trabajo.

En esta fase también **se debe valorar el nivel de madurez de la organización,** analizando con especial énfasis el ciclo de vida, las metodologías y las técnicas que se aplican en el desarrollo de sistemas.

Es importante destacar que muchas empresas no se encuentran preparadas para la introducción inmediata de herramientas CASE. En estos casos, puede resultar más aconsejable utilizar herramientas puntuales (por ejemplo CASE front-end en una computadora personal, de bajo costo) con el objetivo de que los profesionales de desarrollo se vayan habituando a las técnicas básicas, para posteriormente ser capaces de asimilar una metodología, y finalmente implantar una herramienta más completa.

En esta fase **se debe valorar económicamente el proyecto,** así como realizar un plan de evaluación de riesgos, que dependerá en gran medida de la situación de la empresa; si se cambia de hardware, sistema operativo, adoptar una nueva metodología y una nueva herramienta CASE, todo prácticamente al mismo tiempo, el riesgo será muy alto.

**Adquisición**

**Esta fase depende en gran medida de la situación en que se encuentre la empresa,** pero en general se deberán abordar tres cuestiones:

*Infraestructura*

En la que podemos incluir tanto el hardware como el software necesarios para la implantación de CASE, así como el nuevo personal que pueda requerirse.

*Proceso de desarrollo de software*

En caso de que existan se deberán adaptar o formalizar, mientras que si no existen se deben definir y escoger:

1.- Ciclo de vida, en el que se especifiquen los pasos, estándares y procedimientos a seguir, con especial énfasis en la garantía de calidad y administración de proyectos.

2.- Las técnicas y las metodologías que se emplearán.

3.- Los productos resultantes de cada fase del ciclo de vida, especificados de acuerdo a las técnicas y metodologías seleccionadas.

*Herramientas*

Se seleccionarán las herramientas capaces de automatizar el proceso, las técnicas y metodologías escogidas, y que puedan soportar los productos definidos mediante un procedimiento.

**Introducción**

Esta es una de las fases que más tiempo ocupa, ya que abarca todas las actividades de la puesta en marcha, y cuyo resultado será crítico para descubrir cualquier cambio que sea necesario implantar, o incluso para determinar el fracaso del proyecto CASE.

Algunas de las actividades más importantes son:

- **Organización del personal**, ya que deben asignarse los nuevos papeles (roles) que supone la tecnología: responsable de metodologías, administrador de la herramienta, responsable de diccionarios, etc.

- **Instalación de la herramienta y su adaptación.**

- Formación, que debe componerse tanto de una sólida base sobre principios de ingeniería de software como de las propias técnicas y metodologías, acompañadas de un entrenamiento adecuado sobre las herramientas. **La formación a lo largo de todo el proyecto es esencial**; lo ideal sería que en esta fase se incluyera un procedimiento de tutoría para la realización del proyecto piloto que permitiera una verdadera transferencia de conocimiento entre los formadores y el personal que vaya a utilizar la tecnología.

- Proyecto(s) piloto(s), **lo ideal sería poder escoger un proyecto** (o un par de ellos) para probar cómo afecta la tecnología al desarrollo de sistemas y **adquirir experiencia práctica**. Debido a la importancia de este punto, posteriormente se dará una explicación más amplia de el.

· Evaluación, aunque tiene que ser una actividad continua, la evaluación adquiere una importancia trascendental en esta fase, ya que **se trata de descubrir los fallos e inconvenientes que pueda tener la aplicación de la metodología o de la herramienta para proceder a su solución**. También es importante que se mida el resultado de los proyectos piloto para compararlos con las medidas de éxito predeterminadas, que hayan sido establecidas en la fase de planificación.

Como es lógico, muchas de las actividades de esta fase van encaminadas a vencer la resistencia de los profesionales del desarrollo y de la dirección, inercia lógica, por otra parte, ante cualquier cambio de la magnitud que supone la implantación de tecnología CASE.

**Utilización**

En esta fase, una vez aplicadas las medidas correctoras obtenidas de la evaluación de los proyectos piloto, se extiende la tecnología a todas áreas y grupos involucrados, formando a todo el personal, instalando herramientas, etc.

Durante la fase de utilización **hay que considerar una actividad continua de evolución, ya que un entorno CASE no puede ser estático**, debiéndose adaptar a nuevos requisitos, métodos plataformas, etc. en un proceso continuo de mejora de software.

Algunos autores afirman que durante esta fase se debe llevar a cabo la migración de aplicaciones anteriores (no desarrolladas con CASE) al nuevo entorno. Aunque esto sería lo ideal, no siempre resulta posible o rentable, debido al enorme volumen de información y a la falta de documentación de estos entornos. Esta recomendación resulta interesante en caso de que se disponga de herramientas de reingeniería muy potentes o que se este aplicando reingeniería de procesos al sistema o área de que se tratara.

**Retirada**

Por último, hay que considerar que llegará un momento en que la tecnología que hemos implantado se quede obsoleta, debido a cambios en el software utilizado o a la elección de nuevas metodologías, con lo que puede ser necesaria su retirada.

## EL PROCESO DE ASIMILACION DE LA TECNOLOGIA CASE

***El proceso de desarrollo de software debe estar completamente entendido para que la tecnología de la ingeniería de software pueda ser completamente aceptada.***[12] En muchos aspectos, se debe haber recorrido el camino del desarrollo del software para apreciar las ventajas introducidas por la tecnología CASE.

Las herramientas CASE proveen primordialmente ayuda en las dos primeras fases de desarrollo de software reduciendo substancialmente el esfuerzo en la implantación y prueba de objetivos. Por esta razón, más discusiones se basan en estas dos primeras fases.

La llave para la asimilación de nueva tecnología en cualquier organización es la presencia de un promotor de esta. Las organizaciones se estancan frecuentemente y necesitan de una fuerza motivadora para redireccionar sus objetivos. El camino para el promotor de la nueva tecnología puede ser largo y arduo, dependiendo del soporte de la empresa.

La gente dentro de organizaciones tecnológicas tiende a guardar perjuicios técnicos, basados generalmente en una experiencia anterior. Después de todo si los procedimientos y políticas actuales están funcionando, es difícil proponer un cambio. Como un promotor de la nueva tecnología, se debe convencer a la gerencia de que CASE puede incrementar la productividad de la organización más allá del presente nivel[13]. Adoptar la tecnología CASE para un extenso uso y, seleccionar herramientas y metodologías particulares, requiere un proceso de asimilación lento y potencialmente riesgoso figura (3.3).

12 FISHER, Alan S., Using software development tools.

13 Para un empleado interesado en trabajar con nueva tecnología es particularmente difícil convencer a la gerencia de los beneficios de su adquisición, para ello se necesita un plan adecuado de acercamiento de esta a la empresa

El juego de asimilación de la tecnología

| INICIO → | Nace un promotor | Asistir a cursos y seminarios | Los vendedores realizan presentaciones |
|---|---|---|---|
| CASE gana la aceptación de la empresa | | QUIZA La gerencia diga NO! | QUIZA |
| Proyecto éxitoso a causa del uso de herramientas CASE | QUIZA El proyecto falle! | | Adquir una herramienta para evaluación |
| QUIZA | Primer proyecto en desarrollo | Elección del primer proyecto | Recibir autorización para iniciar el primer proyecto |

Figura 3.3 Promover una nueva tecnología para el desarrollo de software como lo es CASE puede resultar muy riesgoso.

### Nace un promotor

Es difícil de precisar exactamente cuando un individuo comienza a ser partidario de una nueva tecnología. La semilla quizá sea sembrada después de leer algunos artículos en la prensa tradicional. Más bien, un individuo lentamente se adentra cuanto más aprende acerca de la tecnología y mantiene algunas conversaciones con colegas.

**Promover cualquier tecnología nueva es una tarea difícil, aún para un experto en maniobras organizacionales. La primera tarea es obtener información en otras compañías u organizaciones de desarrollo, acerca de proyectos exitosos en donde hayan aplicado herramientas CASE.** Muchos vendedores de herramientas estarán felices de suministrar información sobre sucesos o historias de proyectos que ocuparon sus productos.

Una segunda opción es persuadir a la gerencia de invertir algo de dinero y esfuerzo para investigar la tecnología; después de todo ningún grupo de manejo quiere prescindir de un buen respaldo tecnológico. Las grandes organizaciones pueden disponer de uno o

dos individuos para un equipo de investigación por algunos meses. Los individuos en compañías pequeñas deben, por supuesto, estar preparados para investigar dentro de su propio tiempo si una asignación formal de investigación no es posible.

El mejor promotor es aquel que tiene una buena reputación y está apostando por un proyecto exitoso, como oposición a un ingeniero de proyecto que no puede controlar otros proyectos de equipo. Además, el promotor no debe descuidar las actividades de su equipo de trabajo, ya que frecuentemente es difícil comunicar los objetivos hacia los niveles inferiores quienes generalmente son los responsables de implementarlos y una nueva tecnología solo exacerba este problema.

La presencia de un promotor de CASE, proporcionará la oportunidad de que el proyecto triunfe. **Este individuo servirá como el mentor del primer proyecto basado en herramientas CASE y asumirá la responsabilidad de vender los conceptos de la ingeniería de software a otras personas en la empresa.**

**Educación y entendimiento.**

El proceso de educación y entendimiento ocurre gradualmente sobre el tiempo, los promotores primero se enteran sobre la tecnología y entonces juegan un papel más activo en la enseñanza. El primer paso probablemente es contactar algunos vendedores y pedir los manuales de sus productos. El segundo quizá sea el organizar una conferencia local de ingeniería donde la tecnología CASE sea discutida. Finalmente, el promotor debe iniciar un curso de entrenamiento. Para este momento, el promotor debe conocer el costo de tiempo y dinero de los cursos.

Puede sugerir a la gerencia el tomar uno o dos cursos de entrenamiento ofrecidos por el vendedor o por una compañía independiente; también sugerir una o dos herramientas **in-house**[14] para evaluación. Otra opción es emplear un grupo de consultoría externo con experiencia en CASE. Este grupo aportará experiencia colectiva adquirida en otros proyectos de software de diferentes compañías a la organización. Algunas veces los gerentes suelen creer más en terceras personas que en su propio personal y esto siempre lleva a consultar opiniones externas.

14 Herramientas prestadas por los vendedores para probarse dentro de la organización.

En este punto es común para el promotor conseguir presentaciones para los grupos interesados dentro de la organización de desarrollo, quizá también invite a los vendedores de CASE para dar presentaciones sobre sus productos. Esto es parte del proceso de compra interno necesario para convencer a otros de la oportunidad.

**Evaluación In-House de la herramienta**

Adquirir una herramienta in-house para su evaluación, es el primer paso en el proceso de asimilación, esta representa un riesgo para el promotor y la organización de desarrollo. El promotor debe ser cuidadoso al seleccionar algunas herramientas par su evaluación. Un error común en esta etapa es el favorecer a un vendedor sobre los otros. **Hay que recordar que el objetivo de una evaluación in-house es educar a la organización, además de tomar una descición de compra.** De esta forma, si una herramienta no es aceptable por alguna razón, la organización no desechará a CASE por ser una novedad, al contrario verá el rechazo de la herramienta como un paquete que no satisface las necesidades de la organizacion porque esta no tiene suficiente capacidad.

El plan para evaluar cada herramienta es tomar una parte pequeña de una aplicación y usar la herramienta para dirigir el análisis de requerimientos y generar las aplicaciones de diseño. Estimar como intervendrá la herramienta en el estilo de la organización y asegurarse que el análisis y las metodologías de diseño sean familiares para la organización

El tiempo gastado en una evaluación sirve para realizar un informe detallado para la decisión de compra. Una vez que el candidato adecuado ha sido seleccionado, se entrega una evaluación completa a la gerencia.

**Seleccionar la herramienta CASE correcta**

Seleccionar una herramienta de software es tanto un proceso de establecer una relación con un vendedor, como lo es seleccionar una metodología. A menos que la organización sea muy sofisticada, probablemente se requiera ayuda con el entrenamiento antes de que la herramienta sea instalada y esté dando soporte al proyecto. Una pequeña ayuda de entrenadores capaces puede reducir substancialmente el riesgo en el primer proyecto de CASE. Quizá exista la posibilidad de requerir consultoría especializada para mejorar el desarrollo de la aplicación.

Algunas herramientas CASE son de propósito general, análisis de requerimientos y especificaciones de diseño, algunas enfatizan los diagramas de flujo de datos y carácteres de estructura. Otras proveen apoyo en áreas bien definidas como son interfases con el usuario y generación de reportes para base de datos. **Las siguientes cuestiones son puntos importantes a considerar cuando se analizan herramientas CASE** figura (3.4).

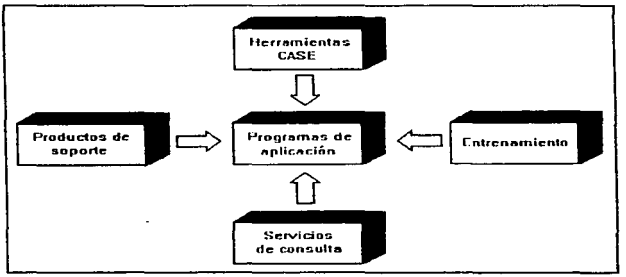



**Figura 3.4 Seleccionar la herramienta CASE correcta, no solo es considerar el producto de software y la tecnología, sino asegurarse de que exista un entrenamiento y un servicio de soporte.**

**¿ La herramienta tiene un alcance bien definido ?**

Después de analizar las necesidades, seria fácil pensar en elegir la herramienta correcta. Esto suena obvio, pero muchas organizaciones venden por la "chispa" y no por la substancia. Si se requiere una herramienta para análisis de requerimientos, hay que considerar herramientas que implementen la metodologia de análisis estructurado.

**¿ La herramienta puede "adaptarse" si es necesario ?**

Muchas organizaciones usan variantes de metodologias para análisis de requerimientos y diseño de especificaciones. Por ejemplo, las organizaciones emplean grandes cantidades de tiempo real en el diseño del sistema, probablemente han extendido

las metodologías estándares o han creado nuevas metodologías para obtener control en el tiempo y en la información. Varias de las herramientas CASE proveen facilidades para instalar metodologías que permitan a los desarrolladores el adherir sus propios símbolos y reglas de conexión al principio del diccionario de datos. Ciertamente muchas organizaciones no necesitan esta capacidad, pero esto es obligatorio para algunas.

**¿ Es suficiente la interacción para disminuir los esfuerzos de diseño y desarrollo ?**

Las herramientas CASE pueden proveer un beneficio suficiente para que el proyecto actual sea entregado. Por ejemplo, si siempre se usa un editor de gráficas para crear diseños de interfases, entonces seleccionar una herramienta para la generación de interfases que no genere código probablemente no mejore la productividad. Sin embargo, si se usa un editor de gráficos para dibujar diagramas de flujo de datos, entonces la implementación de una herramienta de análisis estructurado proveerá mucha flexibilidad de edición con características como elevar el nivel, revisar consistencia y mantener el diccionario de datos.

**¿ La herramienta genera software automáticamente de las especificaciones ?**

Si la herramienta genera código de las especificaciones de diseño, esta deberá generar el código correcto. La relativa dificultad de capacidad para evaluar el resultado del proceso de generación automática de código no esta aún bien entendida. Muchas herramientas generan definiciones de las estructuras de datos de los diccionarios de datos. Otros producen código compilable de las miniespecificaciones escritas con programas de lenguajes de diseño. Si la generación de código es un requerimiento absoluto, entonces hay que insistir en una prueba de evaluación usando un ejemplo real en la organización. Una herramienta que sólo genera una parte del código es básicamente inútil si este debe ser modificado manualmente. Esta elimina los beneficios de la generación automática de código desde la flexibilidad de alterar el diseño hasta la reutilización de código.

## SELECCIONAR CORRECTAMENTE EL PRIMER PROYECTO

Seleccionar correctamente el proyecto para introducir la tecnología CASE en la organización es tan importante como seleccionar la herramienta correcta[15]. Hay muchas formas de evitar seleccionar el proyecto equivocado. Obviamente el objetivo es desarrollar un proyecto exitoso en el menor tiempo y costo posible; ***se debe elegir un proyecto que permita probar por si mismos los beneficios de CASE***[16]. Los siguientes puntos quizá no sean pertinentes para la organización, pero se deben considerar cuando se efectúe la elección del proyecto.

**Seleccionar un nuevo proyecto, que no este en progreso.**

La ingeniería de software asistida por computadora no es fácil de cancelar en un proyecto que ha fallado debido a un pobre análisis de requerimientos, un mal diseño o un manejo diferente, CASE no es un caballero blanco al rescate, al contrario, la tecnología CASE demanda una estructura diferente en el proyecto para obtener óptimos resultados a diferencia de los típicos pasos seguidos en la mayoría de las organizaciones de desarrollo. CASE propone un énfasis especial en el ciclo de vida de desarrollo de software, da la oportunidad de desarrollar un proyecto "fresco no maleado" por elementos que fallan u otros atributos que podrían poner en peligro las oportunidades de éxito del proyecto.

**Seleccionar un proyecto de tamaño moderado.**

Seleccionar un proyecto inalcanzable, con obstáculos de tamaño y complejidad no es recomendable para iniciar el desempeño de CASE dentro de la organización, el objetivo es proveer un lugar a la tecnología CASE dentro del arsenal de desarrollo de software de la organización -escoger un proyecto que establezca claramente la posibilidad de que la tecnología CASE pueda desarrollarse como se debe.

15 Es vista con mejores ojos la tecnología que triunfa desde el inicio de su implantación, por esto es importante que el primer proyecto sea el adecuado para la introducción de CASE en la orgnización
16 FISHER, Alan S., Using software development tools.

**Seleccionar un proyecto técnicamente normal.**

No hay que aceptar un proyecto riesgoso o complejo hasta que la organización haya ganado experiencia en el manejo de la tecnología CASE, así como el entender sus alcances y potenciales. Asegurarse de que no se tengan requerimientos extremos o anormales, tales como fechas de termino ambiciosas o escoger un área donde no exista un experto de desarrollo.

**Asegurar la cooperación de los usuarios finales.**

Una comunidad de usuarios que coopera es esencial para cualquier proyecto exitoso, no importa si la tecnología CASE es empleada o no. Las metodologías de análisis estructurado y modelado de la información son ideales para interactuar con el usuario - tomando ventaja de las oportunidades para encontrar la retroalimentación durante las sesiones de la etapa de análisis de requerimientos. Manejar la relación diseñador/usuario es crítico, ignorar esta oportunidad de comunicación puede entorpecer las oportunidades de triunfo

**Entrenar al equipo del proyecto en las herramientas y metodologías apropiadas.**

Los miembros del equipo de proyecto deben conocer lo más posible las herramientas y metodologías de desarrollo, deben tomar cursos de entrenamiento, además de tener la oportunidad de aprender y practicar con herramientas CASE antes de que el proyecto inicie. Hay que evitar el entrenamiento sobre la marcha, ya que este sólo impide el progreso.

El proyecto debe estar constantemente monitoreado para verificar la eficiencia de la tecnología CASE y las reacciones de los miembros del equipo al trabajar con las nuevas herramientas de desarrollo. Se debe preparar un reporte completo para la gerencia y para comprobar que se tomó la decisión correcta de metodología y herramientas.

## UTILIZANDO UNA HERRAMIENTA EN EL PRIMER PROYECTO

La ingeniería de software aplicada a la computación enfatiza el análisis de requerimientos y las especificaciones de diseño, mientras desenfatiza el código y la implementación. El objetivo de esta nueva tecnología es dejar que la computadora ejecute el trabajo tedioso y consumidor de tiempo, permitiendo la concentración en lo más importante, la recolección de requerimientos y el diseño de las tareas. Con esto en mente, suponemos:

- Mayor enfoque en detalles de requerimiento y diseño
- Más aumento en el análisis de requerimientos
- Más tiempo de entrevistas con los usuarios
- Extender el periodo de diseño
- Varias iteraciones durante el proceso de diseño

Estas espectaciones quizá den un sentido conceptual, pero quizá lleve a un inexperto líder de proyecto a desesperarse con valioso tiempo perdido y un progreso poco tangible (esto es, que el código no ha comenzado a escribirse). Esto resulta especialmente difícil de entender para la gerencia. Esta generalmente se encuentra más tranquila cuando los diseñadores están trabajando en tareas según ellos productivas[17]. Por esto lo mejor es prepararlos con los hechos antes de que comience el proyecto, y así deberán entender que los periodos de análisis y diseño se extenderán a cambio de que los de codificación y mantenimiento se reduzcan figura (3.4). No obstante estar preparado para la tensión, los siguientes beneficios agradarán a la gerencia:

- Corto periodo de implementación
- Pocas sorpresas durante la implementación
- Código más mantenible
- Mejor organización de módulos y separación funcional
- Un diseño de código que se "escribe por si mismo"

17 Cuando se desarrolla el primer proyecto con tecnología CASE, se debe explicar a la gerencia que en este se ampliaran los periodos de análisis y diseño a cambio de reducir los de codificación, implantación y documentación

| | Ciclo de vida tradicional | Ciclo de vida utilizando metodologías estructuradas | Ciclo de vida utilizando herramientas CASE |
|---|---|---|---|
| Análisis | 20% | 30% | 45% |
| Diseño | 15% | 30% | 40% |
| Codificación | 20% | 15% | automática |
| Prueba | 45% | 25% | 15% |

**Figura 3.4 División del ciclo de vida utilizando CASE**

***Es vital que el primer proyecto usando herramientas CASE triunfe, para así cuantificar y medir de alguna forma su potencial***[18]. Los siguientes puntos deberán darnos un parámetro para medir el éxito cuantitativo y cualitativo del proyecto:

- La duración de los periodos de implementación e integración del sistema.

- El total de meses-empleado de esfuerzo requeridos durante el desarrollo para una aplicación aceptable.

- El número de cambios requeridos por los usuarios, en el diseño del proyecto de prueba.

- Número de errores por linea de código

Usar herramientas ayudará a la instalación mejorando el análisis y el diseño dentro de la organización de desarrollo. La tecnología CASE enfatiza la filosofía de separar el proceso de desarrollo de software en segmentos manejables y claramente definidos, con excelente documentación para cada etapa.

18 FISHER, Alan S., Using software development tools.

Completar exitosamente el primer proyecto de desarrollo basado en CASE logrará convencer a la organización de cambiar a esta corriente. El siguiente paso es entrenar a la organización en metodologías y herramientas CASE. Esto incluye identificar programas y seminarios de entrenamiento apropiados, aportados desde el exterior de la organización de diseño. Además de considerar el establecer un programa de entrenamiento adecuado para los nuevos miembros de la organización. Esta es la forma más segura de institucionalizar la tecnología CASE como parte de las prácticas estándar en la organización.

## CONSTRUCCION DE UN EQUIPO DE TRABAJO

Existen diferentes metodos para organizar y manejar equipos de desarrollo. Como una comparación, el equipo de desarrollo organizado funciona como un equipo quirúrgico, con cada miembro desempeñando una responsabilidad especifica. El cirujano sería el programador en jefe, responsable de diseñar los requerimientos del programa y las especificaciones de diseño, los demás miembros del equipo se reportan con el, incluyendo los asistentes administrativos y otros especialistas de software.

Implícitamente, el equipo entero esta comprometido con trabajar en el proyecto durante la duración de la "operación". Los miembros son entrenados para trabajar como un equipo y cada uno es capacitado para ello, el equipo labora bien en trabajos pequeños y moderados donde los objetivos están bien especificados y el cambio de la gente de desarrollo no genera problemas. Contratar un sólo programador es poco deseable debido a la fatiga y aburrimiento en proyectos a largo plazo. En grandes contratos y en grandes aplicaciones comerciales los proyectos de desarrollo de software pueden tardar de tres a cinco años.

Una alternativa es la construcción de un equipo. Los equipos de software, funcionan mejor, con división de labores y de especialidades. Como una analogía, los edificios son diseñados por arquitectos quienes nunca esperan manejar un martillo. Una vez que los detalles han sido aprobados por el inspector del edificio, se puede empezar su construcción, los constructores no necesitan cuestionarse sobre el diseño, o los detalles del edificio. Ellos simplemente siguen las indicaciones sistemáticamente, piso por piso, cuarto

por cuarto, hasta completar el edificio. Si cualquier miembro del equipo sale, los nuevos pueden tomar el trabajo casi inmediatamente sin entender la arquitectura completa. Esto ocurre similarmente con los proyectos de software, donde el diseño es la estructura para la construcción del sistema.

Como sucede con los miembros de las compañías de construcción que tienen herramientas altamente especializadas, un caso similar está emergiendo en la industria del software figura (3.6) **Los equipos de desarrollo consistirán de profesionales especializados en herramientas específicas, como las de análisis estructurado, las modeladoras de datos y las generadoras de interfaces.** Esta especialización permite a los profesionales hacer contribuciones de gran nivel para cada aplicación, ya que éstas son capaces de hacer más en menos tiempo y con disminución de errores.

| CONSTRUCCION DE EDIFICIOS | CONSTRUCCION DE SOFTWARE |
|---|---|
| ARQUITECTO<br>Trabaja con el constructor para definir los requerimientos y diseñar los planos | ARQUITECTO/DISEÑADOR DE SOFTWARE<br>Trabaja con los usuarios para definir los requerimientos y desarrollar especificaciones. Desarrolla el modelo de datos |
| CARPINTERO<br>Erige la estructura del edificio | ESPECIALISTAS INTERNOS<br>Crean las estructuras de datos, procedimientos y algoritmos. |
| ELECTRICISTA<br>Installa los contactos eléctricos. Conecta el edificio a los servicios de electricidad | ESPECIALISTAS DE BASES DE DATOS<br>Escriben las estructuras de la base de datos para las consultas acostumbradas |
| TAPICEROS Y DECORADORES<br>Ponen el tapiz, pintan, decoran y realizan otros trabajos de interiores | ESPECIALISTAS DE INTERFASES<br>Diseña e implementa la interfase con el usuario. En sistemas interactivos, la interfase con el usuario consume del 40 al 60% del total de software |
| INSPECTOR DE CONSTRUCCION<br>Inspecciona la calidad del edificio durante las fases de construcción | GRUPO DE VERIFICACION DE CALIDAD<br>Revisa que el software tenga claridad, integridad y eficiencia |

**Figura 3.6 El personal de construcción de parece al del desarrollo de software.**

Separar grupos de calidad confiable no es nuevo en las organizaciones de software, durante años éstos han realizado pruebas aceptables en proyectos completos. El concepto nuevo es tener grupos de confianza que juzguen el software en puntos clave durante el ciclo de desarrollo, como después de las fases de análisis y diseño. Este rol era tradicionalmente desempeñado por el mismo equipo de implementación, el cual se encuentra generalmente cansado después de completar una fase de algún proyecto,

quedándole poca energía o deseo de revisar su propio trabajo. Además, los equipos de implementación frecuentemente son apresurados a iniciar sin completar las fases de análisis y diseño. Una perspectiva fresca es siempre mejor y una inspección separada de estas fases debe eliminar substancialmente más errores de diseño -la clase de error más difícil durante las etapas posteriores de implementación.

Un concepto bastante nuevo en ingeniería de software es la fábrica de software, como las fábricas convencionales, trabajan en líneas de ensamblado, con un grupo de trabajadores especializados en aspectos del proceso de manufactura. Para el software esto significa analistas, diseñadores, implementadores, y confiables profesionales de calidad, todos especializados. La fábrica de software también significa tener un conjunto de herramientas de software como soporte en el ambiente de producción.

Implementar el concepto de fábrica de software es necesario para construir instituciones exitosas, debemos ser capaces de producir consistentemente gran calidad, software libre de errores, recolección confiable de requerimientos, todo en una cantidad de tiempo predecible. Esto es sólo para conducir las tareas de análisis y diseño en una forma apropiada y lógica, y que un proyecto pueda estar organizado alrededor de un equipo de construcción. Sin esta división de responsabilidades, y sin metodologías apropiadas, el desarrollo del software está destinado a quedarse como un fino arte, en vez de avanzar como una disciplina de ingeniería.

## COSTO DE LA ADOPCION

***La adopción de una herramienta CASE lleva asociada una importante inversión económica que suele infravalorarse, lo que puede llevar a la paralización del proyecto y al consiguiente rechazo de la tecnología***[19].

El costo más evidente es el de la propia herramienta; sin embargo, en la práctica, este costo representa sólo una pequeña parte del total necesario[20] figura (3.7).

19 PIATTINI Y DARYANANI, Elementos y herramientas en el desarrollo de sistemas de información.
20 Obviamente este es el principal factor a considerar en la adquisición, ya que el costo de una herramienta es alto y un estudio de adopción puede asegurar la inversión

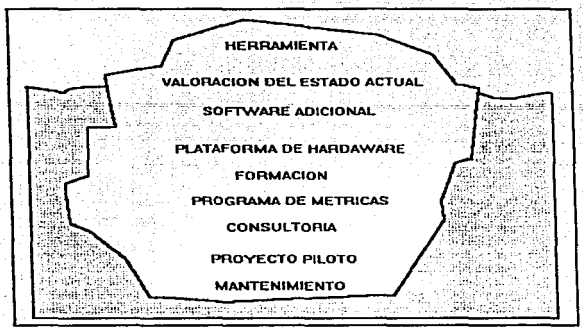



**Figura 3.7 Costos de la adopción de la tecnología CASE**

De hecho, la adopción de una herramienta CASE vendrá acompañada por:

- Valoración del estado actual de las técnicas de desarrollo empleadas en la organización, el nivel de formación del personal, la infraestructura (hardware/software) disponible, etc.

- Infraestructura, conjunto de software y hardware necesarios para soportar el entorno CASE, como estaciones de trabajo, redes de comunicación, sistemas operativos,etc., que pueden exigir determinadas herramientas.

- Formación, sobre metodologías, sobre el mercado CASE en general, sobre la herramienta específica, etc.

- Establecimiento de un programa de mediciones que permita evaluar el impacto de la herramienta y afinar su utilización.

- Consultoría, para integrar y adaptar las metodologías y herramientas.

- Proyecto piloto, y el trabajo que lleva asociado, ya mencionado anteriormente.

- Mantenimiento, que empezó siendo un porcentaje fijo entre el 10 y el 15% del precio del producto, y actualmente se suele ofrecer en distintas modalidades, de precios bastante variables.

En HUFF (1992) se identifican los principales indicadores de costos en la adopción de herramientas CASE:

- Alcance de CASE, esto es, la escala a la que se aplica por ejemplo, proyectos, que influyen no sólo en el coste, sino también en la complejidad y el riesgo.

- Complejidad del entorno, es decir, si la infraestructura hardware/software es compleja o si se limita a un entorno uniforme.

- Organización.

- Cultura de cambio, que se refleja en la historia de la organización.

- Tecnología actual.

- Prácticas actuales, como si existe una metodología documentada que se utilice.

- Nivel actual de habilidades, esto es, si los usuarios de las herramientas utilizan metodologías.

- Capacidad, talento y experiencia del personal de desarrollo.

- Velocidad del avance tecnológico, ritmo de cambio y su influencia en la infraestructura y el personal.

El costo total de la adopción depende en gran medida del producto del que se trate; pudiendo, en determinados casos, llegar a suponer hasta ocho veces el costo de la herramienta, lo que resalta la importancia de considerar todos estos factores e indicadores a la hora de elaborar el presupuesto para la adopción.

# CAPITULO IV:

## Catálogo de herramientas CASE

La gran variedad de herramientas CASE que existen en el mercado provoca confusión y muchas veces una actitud negativa en el usuario al no saber cual puede ser la herramienta que mejor se acople a sus necesidades, esto no es una decisión sencilla y depende de los recursos con los que cuente, así como de la tecnología del momento. Esta necesidad provoca el analizar algunos criterios de las tecnologías existentes, tales como la descripción, los principales módulos con que cuenta, la versión, en cuales plataformas puede ser utilizada, que lenguajes genera y su fabricante.

El objetivo de este capitulo es proporcionar observaciones realizadas por expertos sobre el desempeño de los producto que se muestran en el catálogo.

## DESCRIPCIÓN DEL CATÁLOGO

La presentación de los productos de la lista, se encuentra dividida en cuatro módulos, el primero contiene información sobre datos del fabricante cómo lo son la nacionalidad, dirección y teléfono, además de mostrar el año de la primera versión y la actual, lo anterior con el propósito de facilitar la localización para el acercamiento con el fabricante. El siguiente módulo cuenta con una descripción general de la herramienta, en la cual se señalan los atributos, los objetivos para los cuales se desarrolló, además se especifican las soluciones que ofrece a los usuarios, todo con la finalidad de ofrecer un mejor panorama y así captar el interés del posible comprador. La tercera parte muestra información más detallada sobre el desempeño del producto como lo es: la plataforma en la que "correrá", los módulos con los que cuenta, los lenguajes que genera, los estándares con los que trabaja y las metodologías que soporta, este conjunto de datos pretende mostrar que instrumentos posee la herramienta para su explotación. Finalmente en la cuarta sección se vierten una serie de observaciones con las cuales se resaltan las cualidades del producto CASE, este módulo tiene el objetivo de proporcionar al interesado una serie de comentarios de fondo, emitidos por personas capacitadas los cuales permitan formar una idea clara de la herramienta.

La estructura del catálogo fue diseñada para facilitar el análisis de los productos al clasificar la información presentándola de forma organizada y consistente, para resaltar las características más relevantes de cada herramienta CASE.

# BACHMAN

**FABRICANTE:**
BACHMAN INFORMATION SYSTEMS INC.
New England Executive Park
Burlington, MA01803 Tel. 617 273 9003
ESTADOS UNIDOS
**AÑO PRIMERA VERSION:** 1987 **VERSION ACTUAL:** 4.20 1994

**DESCRIPCION:**
BACHMAN Information Systems fue fundada en 1983 por Charles Bachman, inventor del primer sistema de administrador de Bases de datos.
Es objetivo de BACHMAN Information Systems el desarrollar y comercializar un conjunto integrado de productos de software orientados a la creación y mantenimiento de sistemas de información.
Las soluciones abiertas que hoy ofrece están basadas en la reingeniería, reutilización e integración de información, y su implementación en distintas bases de datos y plataformas incluyendo el entorno, cliente/servidor.
Actualmente, las soluciones ofrecidas por BACHMAN incluyen el diseño, mantenimiento y optimización de bases de datos críticas, el modelado de procesos, metodologías de trabajo y herramientas para la construcción de aplicaciones en tecnologías tradicionales y en cliente/servidor. La arquitectura abierta con la que están diseñados sus productos permiten crear aplicaciones que integran diversas fuentes de información, modificarlas en función de los cambios tecnológicos y de la actividad de la empresa, y ejecutarlas en una gran variedad de plataformas.

**PRINCIPALES MODULOS:**
BACHMAN/Analyst BACHMAN/DBA
BACHMAN/Designer BACHMAN/Ellipse
BACHMAN/NETRON

**PLATAFORMAS:**
OS/2, WINDOWS, AIX, UNIX, MVS

**FACES SOPORTADAS:**
Todas

**TECNICAS Y METODOLOGIAS:**
Reingeniería, análisis orientado a objetos, modelado gráfico, E/R,DFD,FDD,PSD,GANE & SARSON,BACHMAN, etc RDSM & Adaptable a muchas metodologías y técnicas

**LENGUAJES GENERADOS:** COBOL (multiplataformas),CPS,APS,TELON etc.

**ESTANDARES SOPORTADOS:** ESF

**OBSERVACIONES:**
Herramientas BACHMAN se destaca por modelación y su facilidad de uso, reutilización a través de técnicas de reingeniería y análisis orientado a objetos, integración entre datos, procesos y lógica que elimina muchas tareas repetitivas e independencia de la implementación para especificar la actividad empresarial.

# CONCERTO

**FABRICANTE**
Sema Group sa y Sema Group sae
C/Albarracín 25
28035 Madrid
**AÑO PRIMERA VERSION:** 1989 **VERSION ACTUAL:** R4

**DESCRIPCION:**
CONCERTO es una factoría de software dirigida especialmente a compañías en las que se desarrolla software técnico a gran escala, principalmente aeroespacial y de telecomunicaciones, donde ha tenido una gran aceptación.
CONCERTO/ESSDE, constituido por las herramientas SADT, DOC, HOOD, ADAM, Configurator y Organizer, ha sido elegido como entorno de desarrollo de software por la Agencia Espacial Europea. Por otra parte, la configuración formada por Organizer, Configuración, TRACE, LFS y AUDIT, proporciona al usuario un paquete para gestión y configuración de proyectos de software. A este entorno se le llama CONCERTO/SKIPPER.
CONCERTO proporciona, por tanto, un variado conjunto de herramientas de las que el usuario podrá elegir aquellas que mejor se adapten a sus necesidades.
Organizer: manejo y control de actividades dentro del entorno, definición y soporte de los procedimientos.
Configurator: manejo de versiones y configuraciones.
AUDIT: métricas, control de calidad y verificación de los estándares de programación.
DOC: manejo y producción de documentación, referencias detalladas.
LFS: administración lógica de ficheros y directorios UNIX creados por cualquier herramienta.
TRACE: matrices de seguimiento entre los requisitos y cualquier tipo de objeto.

**PRINCIPALES MODULOS:** ORGANIZER, CONFIGURATOR, DOC, LFS, AUDIT, TRACE,FORTRAN, C/C++,ASA.
HOOD: metodología de diseño jerárquico orientada a objetos.
ADAM: manejo de código fuente Ada permite una organización flexible de las bibliotecas.
REVE: herramienta de ingeniería inversa de ADA a HOOD.
TTCN: lenguaje de descripción de baterías de pruebas.

**PLATAFORMAS:** SUN/SOLARIS, DEC/VMS, DEC/ULTRIX, HP/UX

**FASES DEL CICLO DE VIDA SOPORTADAS:** Análisis, diseño, implementación, prueba y mantenimiento

**TECNICAS Y METODOLOGIAS:**
HOOD: metodología de diseño jerárquico orientada a objetos.
SDL: lenguaje de diseño de especificaciones.
C/C++: manejo y edición de programas C y C++, así como OOD.

**LENGUAJES GENERADOS:** ADA, C/C++
**DOCUMENTACION/BIBLIOGRAFIA:** CONCERTO NOTES, Manuales de usuario por cada herramienta, Manual de referencia, Manual de administración.

**OBSERVACIONES:**La arquitectura abierta de CONCERTO permite la integración de herramientas externas necesarias para la producción y control de software industrial.

# INTEGRATED SYSTEM DEVELOPMENT (ISD)

**FABRICANTE:** BULL S.A.
Tour Bull, Cedex 74, 92039 Paris La Defense, Tel: 33(1)46 96 90 90
Fax: 33(1)46 96 90 92
France
AÑO PRIMERA VERSION: 1993 VERSION ACTUAL: V1.2

**DESCRIPCION:**
ISD proporciona a los desarrolladores una plataforma de servicios basada en dos modelos de referencia: el DCM (distribuited Computer Model), desarrollado por Bull para conectividad de entornos de información heterogéneos, y el modelo general para entornos de desarrollo PCTE especificado conjuntamente por ECMA (European Computer Manufacturers Association) y NIST (U.S. National Institute for Standards in technology).
Es un entorno de integración en el que una amplia variedad de herramientas de proveedores diversos pueden trabajar juntas, compartir información, intercambiar peticiones y mensajes y aparecer de una forma consistente al desarrollador, ofreciendo integración de presentación, datos y control.
ISD construye modelos de cómo herramientas y subsistemas estructuran las descripciones de recursos (metadatos) que contienen y transforman estas descripciones en forma neutral y compartible. El repositorio ISD acoge información sobre: planificación objetivos y requisitos, analisis, diseño, hardware y sistemas operativos, redes, modelos de datos, modelos de procesos, reglas de negocio, código fuente, documentación, administración de proyectos, auditoria y usuarios.
ISD está compuesto por dos grupos de componentes: un grupo es una implementación de PCTE, el segundo grupo, construido sobre PCTE, es una series de productos de intercambio que traducen la salida de herramientas en forma neutral, la almacenan en el repositorio, y permite su compartición entre herramientas.

**PRINCIPALES MODULOS:**
- Integrated Desktop
- Integrated PCTE Options
- Integrated Exchange Server
- Integrated PCTE Graphical
- Integrated Exchange Options
- Integrated PCTE Server
- Integrated Repository Tools
- Integrated Repository Model

**PLATAFORMAS:** UNIX, PC's

**FACES SOPORTADAS:** Es integración de herramientas de diversas de cualquier fase y/o de ciclo completo

**TECNICAS Y METODOLOGIAS**
El repositorio integrado y los modelos de información neutral han sido implementados usando el sistema de gestión de objetos PCTE, una avanzada base de datos orientada a objetos que incluye servicios e interfases para asegurar su portabilidad. Es independiente de toda metodologia, está abierto a la integración fácil de herramientas.

**HERRAMIENTAS SOPORTADAS:**
Hay bridges disponibles para herramientas como Intersolv Excelerator IS, Intersolv Exelerator SSADM, Bachman Analyst, Knowledgwere IEW/ADW, MSP Datamanager.

**ESTANDARES SOPORTADOS:** ECMA Portable Common Toll Environment - NIST Integrated Software Engineering Environment

**DOCUMENTACION/BIBLIOGRAFIA:**- Articulos en PCTE Newsletter

**OBSERVACIONES:**
No es una herramienta sino un integrador de herramientas CASE. Aporta beneficios a usuarios de paquetes CASE integrados que aprecian deficiencias en el soporte de sus herramientas para algunas parte del ciclo de vida, usuarios de componentes CASE de fases específicas diferentes, constructores de herramientas, productores de herramientas CASE y especialistas en integración de sistemas.

# Aplication Development Workbench (ADW)

**FABRICANTE:**
Knowledge Ware
Atanta, USA
**AÑO PRIMERA VERSION:** 1984 **VERSION ACTUAL:** 2.7

**DESCRIPCIÓN:**
El ADW es una herramienta CASE gráfica integrada que funciona en PC. Asiste al usuario en todas la fases del ciclo de vida. El trabajo fluye de una manera eficiente desde la planificación estratégica hasta el diseño detallado y la construcción.
Toda la información recogida durante al ciclo de vida del desarrollo es almacenada en una enciclopedia central, la cual es el punto de integración de todos los productos de la línea ADW.
La enciclopedia asegura la consistencia de las especificaciones y diseño, y provee la verificación de errores en tiempo real para garantizar la calidad del proceso.
Debido a que el ADW ayuda al análisis sistemático de los requerimientos, las aplicaciones desarrolladas cubren los requerimientos de la empresa desde el momento que son implantadas. La facilidad para entender los diagramas, tablas y matrices, aumenta la comprensión y la comunicación con todas la personas involucradas en el proceso de desarrollo. Además, ya que todos los requerimientos han sido capturados en la enciclopedia, no existe el riesgo de olvidarlos o extraviarlos. Los datos y procesos lógicos almacenados pueden ser reutilizados por otro equipo de proyecto, reduciendo el tiempo de desarrollo y asegurando la consistencia de los mismos a través de la organización.
Con ADW, se analizan y definen los requerimientos sin tomar en cuenta la plataforma tecnológica de la organización. Posteriormente se pueden desarrollar las especificaciones del diseño y la generación de código para diferentes entornos de hardware.

**PRINCIPALES MODULOS:**
ADW/Planning ADW/Analysis
ADW/RAD ADW/Desing
ADW/Construction ADW/Doc

**PLATAFORMAS:** PC con OS/2

**FASES DEL CICLO DE VIDA SOPORTADAS:**
Planificación estratégica, análisis, diseño, construcción y mantenimiento

**TECNICAS Y METODOLOGIAS:** ADW es una herramienta abierta y no está condicionada por ninguna metodologías especial. Las técnicas soportadas son las siguientes:
E/R, DFD, diagramas de descomposición, matrices(asociación, propiedad, seguimiento), control de normalización hasta la forma normal 1, 2 ó 3, diagramas de estructuras.

**LENGUAJES ORDENADOR:**Cobol MVS, Cobol 400, rpg-400, Cobol-II para Presentatio Manager, Cobol-II para Windows

**DOCUMENTACION/BIBLIOGRAFIA:** Manuales necesarios para su utilización

**OBSERVACIONES:**
La próxima versión de ADW será la 3.8, la cual se encuentra en estos momentos en beta-test. A corto plazo se podrá disponer del ADW en el entorno Windows NT y Unix

# OBSYDIAN

**FABRICANTE:** Synon Inc
1100 Larkspur Landing Circle
Suite 300, Larkspur, CA 94939
**AÑO DE LA PRIMERA VERSION:** 1994 **VERSION ACTUAL:** 1.0

**DESCRIPCIÓN:**
Obsydian es un entorno de aplicaciones que acelera significativamente la implantación de sistemas empresariales a nivel global, mediante su capacidad de definir y reutilizar objetos abstractos de negocio.
Obsydian es único en el uso de la ingeniería de objetos -una poderosa fusión de la ingeniería de la información y la orientación a objetos. Obsydian combina la abstracción de los modelos empresariales de la ingeniería de la información con los principios de reutilización del software de la orientación a objetos para mejorar la calidad y la productividad del desarrollo de software.
Obsydian da soporte a entornos de desarrollo de aplicaciones muy complejos y de alto rendimiento, facilitando:
·Capacidad de diseño y construcción, totalmente integradas.
·Reutilización de objetos para acelerar la entrega de aplicaciones.
·Herramientas que facilitan el trabajo en grupo en grandes desarrollos
·Construcción de diseños de complejidad ilimitada
Obsydian esta diseñado para crear aplicaciones que soporten:
·Nuevos modelos de empresa basados en arquitectura cliente servidor, host y computadoras personales
·Demandas masivas de datos y transacciones
·Los mayores niveles de integridad, control y seguridad
·Ejecución y optimización en múltiples plataformas
Obsydian desarrolla aplicaciones criticas para la empresa, adaptándose a las condiciones cambiantes del negocio explotando completamente las nuevas tecnologías.

**PRINCIPALES MÓDULOS:**
Obsydian Módulo de Diseño: Incluye el modelador de datos, de procesos, diccionario de grupo, ojeador de objetos y editor GUI
Obsydian Biblioteca de Clases: Incluye, objetos de diseño básico y objetos de diseño de negocios abstractos y básicos (ABO). Esto es, objetos con sus atributos, estructuras, comportamientos y componentes de diseño e implantación reutilizables para aplicaciones empresariales comunes.
Obsydian Generadores de Entorno: Todos los generadores incluyen módulo de acceso a la base de datos, lenguajes, interfases y generadores de ayuda textual para:
·Generador cliente servidor Microsoft Windows - AS/400 (C++rpg/400)
·Generador cliente servidor Microsoft Windows - HP/9000 (C++C)
·Generador cliente servidor Microsoft Windows - RS/6000 (C++C)
·Generador para AS/400 NPT 5250 (RPG/400)

**PLATAFORMAS:** Microsoft Windows 3.1 o superior

**FASES SOPORTADAS:** Análisis, diseño, construcción mantenimiento.

**TECNICAS Y METODOLOGIAS:** Orientación a objetos, Ingeniería de la información TCSM de James Martin

**LENGUAJES GENERADOS:** C++, C RPG
DB2/400, Oracle, Sybase, Informix, ODBC, EDA/SQL

## OMW (Objet Management Workbench)

**FABRICANTE:** INTELLICORP
1975 El Camino Real West
Mountain View CA 94040 - 2216
TEL. 415 965 5700
FAX 415 96555647
**AÑO PRIMERA VERSION:** 1993

**DESCRIPCION:**
OMW implementa la metodología de Martin/Odell Objet-Oriented Information Engineering (OOIE) y tiene una original visión de CASE: modela el negocio que ejecuta.
OMW es una nueva clase de entorno de desarrollo visual que proporciona soporte a todo el ciclo de vida, desde el análisis y el diseño hasta la entrega.
Las herramientas gráficas de OMW modelan los problemas de negocios y así los usuarios, analistas y desarrolladores pueden contribuir a construir el diseño.
Los modelos son completamente ejecutables y se pueden usar directamente como la base para el desarrollo de aplicaciones.
Las herramientas de OMW son totalmente orientadas a objetos y permiten usar una variedad de técnicas de modelado como son:
·Object Diagramer, permite modelar los objetos y las asociaciones entre ellos.
·Event Diagramer, presenta información del flujo de los procesos. Muestra relaciones entre los procedimientos de negocio, operaciones e interfases gráficos de usuarios.
Al ser modelos ejecutables, existe una animación gráfica que proporciona un flujo visual del proceso.
·Business Rule Editor, convierte las reglas del negocio en explícitas, visibles y fáciles de modificar y validar.
·Scenario Manager, permite ejecutar y validar los modelos de negocios gráficos creados en OMW. Se pueden analizar pruebas como situaciones de la vida real.
·Documentación e informes proporcionados automáticamente para aplicaciones OMW.

**PLATAFORMAS:** UNIX

**FASES DEL CICLO DE VIDA SOPORTADAS:** Análisis, diseño, construcción y prueba.

**LENGUAJES GENERADOS:** C, PROTALK (parecido al SmallTalk)

**OBSERVACIONES:**
Por tratarse de una metodología de orientación a objetos, entrega unas soluciones reutilizables y mantenibles. Es muy potente en cuanto al modelado de los eventos.

# ORACLE *CASE

**FABRICANTE:** Oracle Corporation
500 Oracle Parkway
Redwood City, CA
94065 USA
TEL: 415 506 70000, FAX: 415 506 72000
**AÑO PRIMERA VERSION:** 1987 **VERSION ACTUAL:** 5.1

**DESCRIPCION:**
Oracle*Case es un conjunto de herramientas integradas que cubren todo el ciclo de vida de los sistemas de información.
Oracle*Case cuenta con una metodología propia, denominada Case*Method, que desglosa el ciclo de vida en la fases de estrategia, análisis, diseño e implementación.

**PRINCIPALES MODULOS:**
Oracle*Dictionary. Interfaz del repositorio activo, multiusuario, multiversión, extensible.
Oracle*Designer. Herramienta gráfica que soporta diagramas entidad/relación, DFD, descomposición funcional y diagramas de matriz.
Oracle*Generator. Generadores de programas interactivos e informes basados en Oracle*Forms y Oracle*Reports

**PLATAFORMAS:** Cliente/servidor.
La práctica totalidad de plataformas Unix, así como VMS,MVS, Netware, Windows NT, OS/2 etc. pueden comportarse como servidores RDBMS Oracle.
Oracle*Case puede ser cliente en
MS-Windows, Unix/Motif, OS/2

**FASES SOPORTADAS:** Todas las fases del ciclo de vida están soportadas.

**TECNICAS Y METODOLOGIA:**
Oracle*Case incorpora diagramas entidad/relación para el modelo de datos y descomposición de funciones y diagramas de flujo de datos para el modelo de procesos. A la vez, dispone de diagramas matriciales abiertos para obtener e incorporar información cruzada sobre cualquier clase de objeto del repositorio

**LENGUAJES GENERADOS:**

| | | |
|---|---|---|
| Oracle*Forms | Oracle*Reports | SQL*Forms |
| SQL*ReportWriter | SQL*Plus | PL*SQL |
| SQL*TextRetrieval | | |

**ESTANDARES SOPORTADOS:**
Oracle participa activamente en los comités

| | | |
|---|---|---|
| IRDS (ANSI e ISO) | CDIF | PCTE |
| ISO Database Panel | OMG | |

**DOCUMENTACION BIBLIOGRAFIA:**
La metodología Oracle*Method está recogida en tres libros de Addison-Wesley
Case*Method Entity Relationship Modelling (Barker R 1989)
Case*Method Task and Deliverables (Barker R. 1990)
Case*Method Function and Process Modelling (Barker R 1992)
Además, existe una gran cantidad de publicaciones sobre Oracle*Case en prensa especializada.

# PCBASE/CS

**FABRICANTE:** CGI-INFORMATIQUE (An IBM-Company)
30,rue du Chateau des Rentiers, F75640 PARIS CEDEX 13
TEL: 140 77 20 00, FAX 1 40 77 26 66
**AÑO PRIMERA VERSION:** 1974 **VERSION ACTUAL:** 8.02

**DESCRIPCION:**
PACBASE/CS es un producto de soporte lógico integrado, destinado a la asistencia y automatización de la ingeniería de aplicaciones informáticas en ciclo completo, desde la concepción hasta el mantenimiento. Dicho de otra forma, PACBASE/CS es lo que en términos actuales se denomina I-CASE. Pero lo que genuinamente PACBASE/CS es, puede expresarse del siguiente modo.
PACBASE/CS gira en torno a una base de especificaciones única, donde las informaciones se definen una sola vez y desde donde pueden utilizarse tantas veces como sea necesario, quedando así automáticamente interrelacionados todos los componentes de los sistemas desarrollados. La base de especificaciones está abierta a la integración de informaciones externas (aplicaciones ya descritas, otras herramientas, etc.).
Los generadores PACBASE/CS producen, a partir de la base de especificaciones, todos los componentes completos de las especificaciones descritas: documentación, programas, descripciones de la base de datos, etc. Estos componentes -una vez generados, son independientes de PACBASE/CS. Todo lo producido por PACBASE/CS hacia una plataforma destino es transportable hacia cualquier otra.
Las estaciones de trabajo acceden en modo visual e intuitivo a las informaciones de la base de especificaciones, realizando con eficacia cualquier consulta o manipulación. Las estaciones de trabajo están dotadas de dispositivos metodológicos que guían y normalizan el trabajo de todos los usuarios. PACBASE/CS dispone de las metodologías más frecuentes del mercado, siendo posible su personalización en cada instalación así como la integración de metodologías propias.

**PRINCIPALES MODULOS:** PACDESING (concepción de aplicaciones, apoyadas en diversas metodologías). PACBENCH (Desarrollo de aplicaciones clásicas), PAC/CS (Desarrollo de aplicaciones cliente/servidor para entornos visuales), PACREVERSE (Integración de aplicaciones existentes), DSMS (Control integrado del mantenimiento)

**PLATAFORMAS:** DESARROLLO: MVS, VSE, AIX, CCOS7, GCOS88, DPX/2, DPX/200SX, VMS, ULTRIX, HP-UX, OS2, WINDOWS, etc.
**FASES SOPORTADAS:** Análisis, diseño, desarrollo y mantenimiento.

**TECNICAS Y METODOLOGIAS:** Las técnicas utilizadas son las propias de cada metodología ya incluidas. Por otro lado, PACBASE/CS está abierto a la personalización de cualquiera de las metodologías de que dispone, siendo posible adaptarlas a cada caso, llegando, si es necesario, a especificar e incluir nuevas metodologías.
**LENGUAJES GENERADOS:** COBOL, C, DDL de las bases de datos.

**DOCUMENTACION/BIBLIOGRAFIA:** Note - December 13 1993
Applications Development & Management Strategies-GARTNER GROUP -Research
IBM -Doc Number ADWENTCS- "Enterprice 4GL Client/Server Application Environment. White Paper" May 24 1994

**OBSERVACIONES:** PACBASE/CS mantiene un nivel muy alto de evolución en sus especificaciones. CGI ofrece, en las direcciones indicadas, toda la información que por razones obvias no aparece en esta ficha.

# PREDICT CASE

**FABRICANTE:** SOFTWARE AG
**AÑO PRIMERA VERSION:** 1989 **VERSION ACTUAL:** 2.4

**DESCRIPCION:**
PREDICT CASE es una herramienta de ayuda al análisis, diseño y construcción de sistemas de información dentro del entorno de cuarta generación NATURAL. Ofrece una solución integrada al desarrollo de aplicaciones comerciales y de administración que utilicen esta tecnología.
El centro neurálgico de PREDICT CASE es la base de datos de desarrollo integrada, multiusuario, multiproyecto, que recoge la información manejada durante el desarrollo de los sistemas. El repositorio almacena todos los datos relativos a los objetos definidos en las distintas fases del proyecto (entidades, funciones, flujos de datos, etc.), los datos necesarios para la correcta administración del sistema (estándares de documentación de diseño, ciclos de vida,etc.) El repositorio tiene una metaestructura pública y es controlado por un administrador de base de datos de tipo entidad/relación, que permite una mayor riqueza semántica que los administradores convencionales.
En torno a este repositorio se articulan una serie de subsistemas para mantener y explotar el contenido del mismo. Entre los principales, podemos citar los siguientes: mantenimiento de objetos, generación de aplicaciones NATURAL (incluyendo soporte al prototipado), generación de documentación y control de calidad.

**PRINCIPALES MODULOS:**
-PREDIC CASE: Parte Host (Diccionario central)
-NATURA ENGINEERING WORKBENCH: Estación de trabajo PC.

**PLATAFORMAS:**
-Parte Host: IBM o compatible: SIEMENS, UNIX. -Parte PC: PC 486 con Windows 3.1

**FASES DEL CICLO DE VIDA SOPORTADAS:**
-Análisis de requerimientos -Diseño -Implantación
-Construcción -Mantenimiento

**TECNICAS Y METODOLOGIAS:**
-Entidad/Relación -Descomposición funcional
-Flujo de datos. -Redes de procesos
-Diseño lógico de B.D. -Estructura de módulos
-Prototipos

**LENGUAJES GENERADOS:** NATURAL

**ESTANDARES SOPORTADOS:** CDIF

**DOCUMENTACIO/BIBLIOGRAFIA:**
User's Guide Reference Manual Administration Manual
Apliaction Programming Installation & Operation Manual

**OBSERVACIONES:** PREDICT CASE es una herramienta orientada al desarrollo industrial del software. Esto se traduce en un énfasis en las técnicas de construcción de aplicaciones basadas en la estandarización de código a gran escala. Dentro de esta línea, las futuras versiones del producto incluirán un soporte exhaustivo de las técnicas de análisis, diseño y construcción orientada a objetos.

# PRINCIPIA

**FABRICANTE:**
Sema Group sa
16 Rue Barbes
92126 -Mountrouge Cedex
FRANCE
AÑO PRIMERA VERSION: 1990 VERSION ACTUAL: 4.11

**DESCRIPCIÓN:**
Principia es un taller de ingeniería de software de administración. El ciclo de vida de un sistema desarrollado con Principa está soportado en dos elementos:
1. El puesto de concepción, diseño y especificaciones generales y detalladas, que trabaja en los contextos de Merise, SSADM y AXIAL.
2. El elemento en host, IBM/VMS y BULL/GCOS7 pata estudio técnico y generación de código.
Está en desarrollo, con entregas previstas para final del 1996 de una versión UNIX, tanto para el puesto de concepción como para el generador de código.

**PLATAFORMA:**
Puesto de concepción: DOS/Windows, OS/2-PM, UNIX
Host: IBM-MVS, BULL-GCOS7

**FASES DEL CICLO DE VIDA SOPORTADAS:**
Diseño, análisis, prueba y generación de código

**TECNICAS Y METODOLOGIAS:** Merise, Merise/2, SSADM y AXIAL

**LENGUAJES GENERADOS:** COBOL

**ESTANDARES SOPORTADOS:** SGML en documentador

**DOCUMENTACION/BIBLIOGRAFIA:**
Merise/2 editado por Les Editions d'Organisation
Manuales varios de la herramienta, el contexto y el puesto de trabajo

**OBSERVACIONES:**
Merise/2 es una ampliación de Sema Group de Merise, e incluye diagramas de flujo de dominios, de procesos y arquitectura cooperativa, entre otras mejoras.

## PRO IV Workbench

**FABRICANTE:** McDonell Douglas
Information Systems Limited
Maylands Park South
Hemel Hempstead
Herts HP2 7Hu
TEL: 232424
FAX 244896
**AÑO PRIMERA VERSION:** 1990 **VERSION ACTUAL:** DOS 2.2,Win 1.1

**DESCRIPCION:**
PRO-IV es un paquete de herramientas avanzadas e integradas de desarrollo para las fases de análisis, diseño y desarrollo del ciclo de vida del software. La aplicación de este CASE proporciona un marco efectivo y eficiente para traducir los objetivos estratégicos de la compañía en un software que satisfaga los requerimientos establecidos.
PRO-IV Workbench utiliza directamente las técnicas fundamentales del desarrollo de software. El módulo Analyzer soporta las fases de análisis, desde la creación del DFD hasta la generación de la documentación asociada. El prototyper permite la generación de prototipos del sistema. El módulo de Data Modeler soporta la definición de la estructura de datos. El designer provee del entorno necesario para el diseño de diagramas de estructuras. Un diccionario integrado, un sistema de generación de Reports, y el mantenimiento de diferentes proyectos y versiones, completan al ciclo de las herramientas. Además, cuenta con herramientas adicionales para funciones auxiliares, tales como importación y exportación, backup, etc.

**PRINCIPALES MODULOS:**
Analyzer - Modelo conceptual DFD
Data Modeler - Modelo de datos (entidad/relación)
Prototyper - Diseñador con navegador de entradas/salidas
Designer - Modelo físico (estructuras. Yourdon)
Report - Generador de informes

**PLATAFORMAS:**
MS DOS / NOVELL. Disponible versión para Windows.

**FASES DEL CICLO DE VIDA SOPORTADAS:** Análisis y diseño.

**TECNICAS Y METODOLOGIAS:**
E/R, DFD, Relacional, Batchman, Chen, Searson, Gane y estructuras Yourdon.

**LENGUAJES GENERADOS:**
Pro-IV 4GL, Telon, Transform
CICS/IMS screen maps

**DOCUMENTACION/BIBLIOGRAFIA:**
McDowell Douglas Information systems.

**OBSERVACIONES:**
Posee un módulo de Prototyper, a partir del cual se pueden generar los mapas CICS/IMS.
El producto soporta arquitectura Cliente /Servidor.

## SILVERRUN

**FABRICANTE:** CSA Research Ltd
445, Av. St-Jean-Baptiste, Suite 100
Quebec Canada
G2E 5N7
**AÑO DE LA PRIMERA VERSION:** 1986 **VERSION ACTUAL:** 2.3

**DESCRIPCION:**
SILVERRUN es una herramienta CASE Front-end que facilita el diálogo, la integración y la coordinación de los diferentes elementos del sistema a lo largo de las distintas etapas del proyecto, mediante la producción rápida de esquemas de alta calidad, y con un enfoque de trabajo orientado hacia los datos o los procesos, según se requiera un enfoque u otro.
Partiendo de las especificaciones de los requerimientos, el usuario de SILVERRUN puede realizar los diagramas de flujo de datos, los diagramas de entidad relación y trabajar con éstos para generar los esquemas de la base de datos de forma automática.
SILVERRUN fue el primero en introducir en 1986, el concepto de herramienta de modelado para Machintosh. Desde entonces, el producto ha crecido, incorporando el Workgroup Repository Manager (WRM), ha diversificado plataformas, ha añadido las herramientas de modelado de procesos (DFD), las herramientas de modelado físico de los datos (RDM) y el sistema experto de entidad/relación (ERX). Y en toda su historia ha hecho hincapié en maximizar el uso de interfases gráficas para garantizar el aprendizaje rápido de los usuarios. El resultado es un entorno de análisis y diseño potente, flexible y fácil de usar en todos los niveles de experiencia.

**PRINCIPALES MÓDULOS:**
DFD (Diagramas de flujo de datos), permite la creación de diagramas de forma rápida y bajo diferentes notaciones [Yourdon and De Marco, Merisse y Gane and Sarson]
ERX (entidad relación experto) es un sistema experto que ayuda a los modeladores de datos a crear modelos correctos y normalizados a partir de los datos y de las reglas que definen el negocio
RDM (Modelador de datos relacional), es un módulo de modelaje avanzado, con funciones de generación que ayudan la producción de esquemas de base de datos de alta calidad y totalmente ajustadas.
WRM (Gestor del repositorio), realiza la coordinación y comunicación entre todos los modelos creados durante el desarrollo del sistema.

**PLATAFORMAS:**
Windows 3.0 o superior, OS/2 Presentation Manager 1.3 o superior, Machintosh 6 o superior

**FASES DEL CICLO DE VIDA SOPORTADAS:** Análisis, diseño y construcción.

**TECNICAS Y METODOLOGIAS:**
Entidad/relación
Gane & Sarson
Yourdon-DeMarco
Merise
Ingeniería de la información

**LENGUAJES GENERADOS:**
Ansi-SQL, DB2/2, DB2, DBase IV, Informix, Ingres, NonStop SQL, Oracle, Ebd, SQL/400, SQL/400, SQLBase, SQL-DS, SQL Server, Sybase, Teredata, Xdb, Uniface, Progress, Synon.

# SNAP

**FABRICANTE:** SNAPCO
SAN JOSE DE COSTA RICA

**DESCRIPCION:**
SNAP es una herramienta para el desarrollo de aplicaciones en AS/400 de IBM. SNAP proporciona al analista un ambiente integrado de trabajo, brindándole la posibilidad de construir sistemas de inmejorable calidad, adherido a los estándares S.A.A. totalmente documentados y ajustados a los requerimientos específicos de la organización en una fracción del tiempo y costo que se invertiría si se utilizarán herramientas tradicionales.
SNAP genera aplicaciones de alta calidad 100% nativas AS/400, totalmente documentadas y libres de errores. SNAP produce resultados a muy corto plazo, la rápida curva de aprendizaje hace que los tiempos sean menores.
SNAP provee de interfaz gráfica en la modalidad MS WINDOWS, utilizando una estación de trabajo inteligente como un cliente servidor AS/400.

**PRINCIPALES MODULOS:**
Modelo de datos
Programación automática
Método desarrollo acelerado (Pseudocódigo)
Utilidades
Seguridad

**PLATAFORMAS**
Pantalla carácter y AS/400
PC/WINDOWS y AS/400 (CLIENTE/SERVIDOR)
PC/WINDOWS y UNIX (Sybase)

**FASES DEL CICLO DE VIDA SOPORTADAS**
Diseño de entidades participantes en la aplicación, con comportamiento automático en la creación.

**TECNICAS Y METODOLOGIAS:**
Técnica en entidad/relación en bases de datos.
Programación estructurada

**LENGUAJES GENERADOS**
AS/400 _____ RPG/400
PC_____ VISUAL BASIC (CLIENTE/SERVIDOR)

**ESTANDARES SOPORTADOS:** C.U.A. (Acceso común de usuarios), incorporando los estándares de la arquitectura de aplicaciones de sistemas S.A.A.

**DOCUMENTACION BIBLIOGRAFIA:**
Manuales de SNAP (Documentación): introducción, instalación, tutorial, modelo de datos, programación automática, programación en seudocódigo, SNAP/WIN (Manual para SNAP en WINDOWS) Documentación IBM AS/400 Application Development Program

**OBSERVACIONES:**
Seudocódigo de programación y documentación en español, fácil de aprender y manejar. Integración con CASE Front-end. Reconstrucción automática de aplicaciones, ficheros y programas, programas resultantes con el más alto rendimiento.

## StP/OMT

**FABRICANTE:** IDE
Interactive Development Environments
San Francisco, California, EEUU
**AÑO PRIMERA VERSION:** 1993 **VERSION ACTUAL:** 2.0

**DESCRIPCION:**
StP/OMT es un entorno de desarrollo integrado multiusuario que da soporte de forma completa a la metodología orientada a objetos.
La navegación que se puede realizar entre los distintos editores de StP/OMT permite ir completando las especificaciones del sistema con metodología orientada al objeto. En definitiva, se puede ir pasando de editor en editor, que dan soporte a los tres modelos que componente la metodología modelo de objetos, modelo dinámico y modelo funcional, e ir definiendo cada uno de los componentes del sistema. Asociados a los editores, existen, de forma complementaria, tablas que se pueden llenar y así completar la información para cada modelo.
Añadiendo información especifica a cada elemento representado en los diagramas o tablas (mediante anotaciones), se puede controlar la generación de código o de informes, así como la navegación entre editores o el control de la sintaxis de los diagramas.
Dentro de la metodología, también da soporte a la extensión para tiempo real. Por último hay que destacar que posee un módulo de captura de clases C++, lo que permite realizar tareas de ingeniería inversa desde código fuente Asimismo, existe la opción para ingeniería inversa para Ada.
En cuanto a la generación de código, existen entornos específicos para generación de C++, Ada, Smalltalk e IDL.
La comunicación con otros productos de la familia StP/T (generador de pruebas) y StP/IM (implementación de modelado de la información) y la comunicación con productos de terceros es bastante amplia; herramientas de seguimientos de requisitos, de control de configuración, puentes para manejo de bases de datos relacionales, etc.

**PRINCIPALES MODULOS:**

| | | |
|---|---|---|
| -Editor del modelo de objeto | -Editor del modelo funcional | -Editor del modelo dinámico |
| -Editor de escenarios | -Editor de Tablas de clases | -Editor de tablas de estados |
| -Browser de clases | -Browser de subsistemas | -Libreria de reutilización |
| -Control de impresión | -Generador de clases OMT a | partir de código C++ |
| -Control de versiones | -Altas de usuarios en la herramienta (control de acceso) | |

**PLATAFORMAS:** Sun Spare, SunOS Solaris, HP 700/800, HP-UX, DEC Alpha, OSF/1 RS/600, AIX

**FASES SOPORTADAS:** Análisis, Diseño, Implementación, (Ingeniería Inversa)

**TECNICAS Y METODOLOGIAS:** OMT (Objet Modelling Technique)

**LENGUAJES GENERADOS:** C++, Smalltalk, Ada, SQL e IDL

**OBSERVACIONES:**Dos características a destacar de este producto:
1. Adaptabilidad: El usuario puede personalizar todos los aspectos de los editores y del entorno de trabajo mediante un lenguaje común de definición no gráfica incluible, generación de informes y documentación.
2. Actualización:Debido a la estrecha colaboración entre IDE, Rumbaugh y ACC, mentores de la metodología, StP/OMT estará siempre por delante con nuevas versiones.

## StP/SE

**FABRICANTE:** IDE
Interactive Development Environment
San Francisco, California, EEUU
**AÑO PRIMERA VERSION:** 1983 **VERSION ACTUAL:** 6.0

**DESCRIPCION:**
StP/SE es un entorno de desarrollo integrado multiusuario que da soporte a las metodologías de análisis y diseño estructurado: los métodos SA (DeMarco/Yourdon o Gane/Sarson y Jackson) y SD según Yourdon/Constantine. Igualmente implementa la extención de tiempo real siguiendo a Hatley/Pirbahi
StP/SE integra editores de diagramas de flujo de datos, de estructura jerárquica de datos y structured chart para la parte de diseño. Si se trabaja con la extensión para tiempo real, se manejan tanto editores de diagramas (de transición de estados) como editores de tablas o matrices de activación (de eventos, de acciones, de procesos, etc) Se desataca, por su gran utilidad, la función collapse. Esta función permite trasladar o crear un diagrama a partir de un grupo de procesos seleccionados, trasladandose todos éstos, junto a los flujos asociados, al nuevo diagrama. Existe la función dual explode, que traslada al diagrama la descomposición del proceso seleccionado. Integraciones Desde el modulo de diseño, se puede realizar la comunicación con StP/T (producto de generación de pruebas), lo que permite un paso más en el desarrollo de software. En ingeniería inversa, su integración se realiza con el producto StP/C (ingeniería inversa para C)
Desde los editores de diagramas de flujo de datos, de estructura de datos o de diseño, es posible comunicarse con el producto de la familia StP, que da soporte a la técnica entidad/relación. Este producto es StP/IM (Information Modelling). Esta integración permite no solo completar el modelo de datos del sistema sino, además generar esquemas y tablas para Oracle, DB2, Ingres, Sybase, Informix, etc. lo que hace de StP/SE, junto a StP/IM, una herramienta especialmente caracterizada para especificación y desarrollo en entornos de bases de datos. Además, comunicación con productos de terceros es bastante amplia: herramientas de seguimientos de requisitos, de control de configuración, ingeniería inversa (Fortran, Cobol, etc.)

**PRINCIPALES MODULOS:**
-Editor de flujos de datos -Editor de diagramas entidad/relación
-Editor de diagramas de cuadros -Editor de tablas CRUD
-Altas de usuarios en la herramienta -Activación de scripts de QRL
-Editor de tablas de especificaciones -Editor de estructura de datos
-Editor de diagramas de transición de estados-Editor de matrices
-Control de impresión de diagramas -Control de inversiones
-Definición y activación de filtros sobre diagramas
**PLATAFORMAS:** Sun Spare, SunOS y Solaris, HP 700/800, HP-UX, DEC Alpha OSF/1, RS/6000, AIX
**FASES DEL CICLO DE VIDA SOPORTADAS:** Análisis y diseño

**TECNICAS Y METODOLOGIAS:**-SA Diagramas de flujo de datos DeMarco/Yourdon)-Extensión de iempo real, Diagramas de datos estructurados, ntidad/relación
**LENGUAJES GENERADOS:** Oracle, C, DB2, Ingres, Sybase, Informix (SQL: DDL, DNL y DCL.

**OBSERVACIONES:**
Dos características a destacar de este producto:
1. Adaptabilidad. El usuario puede personalizar todos los aspectos de los editores y del entorno de trabajo mediante un lenguaje común de definición. Son modificables la representación de objetos, la relación entre los editores (navegación), información no gráfica incluible (anotaciones).
2. Integración. La arquitectura abierta de la herramienta permite la comunicación fácil y fluida con gran cantidad de productos de terceros relacionados con las distintas fases del ciclo de vida (control de configuración, ingeniería inversa, etc.)
Esta integración se hace a través de StP/Core (área funcional común a todos los módulos StP) permitiendo esto realizar cualquier integración en un único paso.

## SYNON

**FABRICANTE:** Synon Inc.
1100 Larkspur Landing Circle
Suite #00
Larkspur, CA 94939
**AÑO DE LA PRIMERA VERSION:** 1986 **VERSION ACTUAL:** 5.0

**DESCRIPCION:**
Synon es una herramienta para el desarrollo de aplicaciones que permite diseñar, desarrollar, implementar y mantener las aplicaciones más eficientes, con más rigor y de forma más efectiva que lo que le permiten los métodos empleados con lenguajes de tercera generación.

**PRINCIPALES MODULOS:**
Synon/2E - Producto núcleo principal que es CASE BAKC-END
Synon CSG- Generador cliente/servidor para entornos Windows y OS/2 con AS/400
Synon Open -Generador entornos Unix, con HP9000 y R/6000
Synon CM - Gestor de cambios para los objetos de AS/400, con total gestión de configuración y versiones
Synon RW - Generador de informes para usuarios finales
Synon PE - Analizador experto de los modelos Synon con informes de mejoras
Synon Gateway 3X - Módulo de reingeniería para aplicaciones heredadas y antiguas en AS/400

**PLATAFORMAS:**
AS/400
Unix, Aix
Windows, OS/2

**FASES DEL CICLO DE VIDA SOPORTADAS:**
Análisis, Diseño, Construcción, Mantenimiento.

**TECNICAS Y METODOLOGIAS:**
Ingeniería de la información

**LENGUAJES GENERADOS:**
RPG y Cobol para AS/400
Cobol Microfocus para Windows y OS/2
Cobol para Aix y Unix
Oracle e informix para Aix y Unix

## SYSTEM ARCHITECT

**FABRICANTE:** POPKIN SOFTWARE & SYSTEM, INC
11 PARK PLACE, NEW YORK TEL : 212-5713434
**AÑO PRIMERA VERSION:** 1988 **VERSION ACTUAL:** 3.0

**DESCRIPCION:**
SYSTEM ARCHITECT es una herramienta de desarrollo de aplicaciones basada en PC con interfaz gráfica. Es una herramienta CASE para el mundo real para los desarrolladores que la tiene que utilizar cada día.
Es un producto versátil, potente y fácil de usar. Totalmente parametrizable y adaptable al usuario. Para cada organización o cada proyecto, permite escoger las técnicas más útiles y las características del diccionario y del entorno de trabajo más adaptado a las necesidades. El equipo del proyecto comparte la información de diagramas y diccionarios de datos a través de la conexión en red.
SA trabaja con múltiples metodologías, y es igualmente apropiado para pequeñas o grandes compañías destinadas a PC, Mainframe o Cliente/Servidor.
Dispone de un conjunto de opciones que cubren la mayoría de las necesidades de una herramienta de desarrollo. Su interfaz con los entornos de desarrollo más comunes permite una construcción integrada de aplicaciones.

**PRINCIPALES MODULOS:** SA-OBJET ORIENTED (permite trabajar con las técnicas de análisis y diseño orientados a objetos). SA-SCREEN PAINTER (Creación de pantallas en modo carácter o gráficos y edición de las capturadas automáticamente). SA-SCHEMA GENERATOR (traductor de modelos de datos a esquema de base de datos relacional, Cobol oC. Generación de Triggers). SA-REVERSE DATA ENGINEER (reingeniería de base de datos y pantallas gráficas para generación automática del modelo de datos). SA-PROJECT DOC FACILITY (capacidad de autoedición entre generación de informes desde la enciclopedia de SA). POWER BUILDER/LINK (intercambio de información de tablas, atributos extendidos y pantallas GUI entre SA y PowerBuilder). SQL WINDOWS LINK: Enlace con SQL/Windows de Gupta.
**PLATAFORMAS:** Versión monousuario: WINDOWS 3.1, OS/2 PM 2.1WINDOWS NT Versión red: NETWARE, LAN MANAGER, VINES
**FASES DEL CICLO DE VIDA SOPORTADAS:**
Planificación, análisis, diseño y construcción.
**TECNICAS Y METODOLOGIAS:** Ingeniería de la información IDEF0 & IDEFX, Yourdon/DeMarco, Gane & Sarson, Shaler/Mellor (OO), Coad/Yourdon(OO), Boch '91, SSADM IV, Ward & Mellor (Tiempo Real), modelo entidad relación, Diaframa de flujo de datos, Flow-Chart, diagrama de descomposición, diagrama de transición de datos, vistas lógicas de base de datos, modelo físico de base de datos, diagramas de estructuras, pantallas modo carácter, GUI y menús.
**LENGUAJES GENERADOS:**
Cobol, C, esquema de bases de datos para DB2, Oracle, Informix, SQL Server, Rdb, Progres, Paradox, SQL Base, Ingres, Dbase III, Sybase, AS/400 (SQL & DDS)
**DOCUMENTACION BIBLIOGRAFIA:** Artículos en revistas y seminarios CASE DATABASE MAGAZINE "Quality of CASE TOOLS", DATAMATION USA, CASE TRENDS MAGAZINE

**OBSERVACIONES:** Elección de las técnicas a utilizar. Repositorio potente, integrado y extensible para el usuario. Diversas técnicas de modelado de datos (Global/Asociativa). Líder del mercado en torno gráfico. Permite la construcción de aplicaciones de gestión y de tiempo real en diversidad de entornos. Soporte cliente/servidor. Intersección con herramientas de desarrollo. Control de versiones. Editor de Triggers.

# VISIBLE ANALYST WORKBENCH

**FABRICANTE:** Visible Systems Corp
300 Bear Hill Roaad Waltham, MA 02154
TEL: 617 890 2273, FAX: 617 890 8909
**AÑO PRIMERA VERSION:** 1985 **VERSION ACTUAL:** 5.3

**DESCRIPCION:**
VAW es un conjunto de herramientas CASE diseñadas para soportar proyectos en equipo para el desarrollo de sistemas utilizando técnicas estructuradas sobre una metodologia de desarrollo. Algunas de las metodologias soportadas por VWA son: Information Engineering, Yourdon Structured Method, SDM, etc y en general cualquiera que utilice las técnicas de análisis y diseño soportadas por VAW (DFD, ME/R, diag. de descomposición, mapas de estructuras etc.).
El VAW tiene una excelente relación prestaciones/costo. Los requerimientos de hardware son los mínimos de la actual tecnología PC, trabajando con MS-DOS y WINDOWS. En su implementación multiusuario el VAW opera con Novell Netware y otras LANs proporcionando a múltiples usuarios de CASE conectados a la red: acceso concurrente, gestión de bloqueos de ficheros y registros, distintos niveles de seguridad de proyectos, etc.
La arquitectura del VAW ha sido realizada según la filosofia OPEN CASE, permitiendo la importación/exportación con numerosos entornos de desarrollo, generadores de aplicaciones y otras herramientas CASE. El VAW ofrece importantes ayudas al desarrollo, generando programas fuente en ANSI COBOL y C, además de SQL DDL para los principales RDBMS (DB2, Oracle, Informix, Gupta, SQL Server, etc.), DDS (IBM AS/400). También incorpora potentes funciones que permiten la reingeniería de modelos de datos (generando M E/R a partir de la implantación de SQL DDL, DDS) y de procesos (Ingeniería inversa para aplicaciones Cobol)

**PRINCIPALES MODULOS:**
Módulo de diagramación
Generador de código
Interfases con otros productos
Diccionario integrado
Módulos de importación exportación
**PLATAFORMAS:** MS DOS, Windows

**FASES DEL CICLO DE VIDA SOPORTADAS:** Análisis, diseño y construcción
**TECNICAS Y METODOLOGIAS:**
DFD, E/R, diagramas de descomposición funcional, diagramas de estructuras, SDM, Yourdon Structured Method, Information Engineering, SSADM.
**LENGUAJES GENERADOS:**
ANSI COBOL, ANSI C, SQL DDL (DB2, Oracle, Informix, Gupta, SQL server, Sybase).
**DOCUMENTACION BIBLIOGRAFIA:**
Analysis and Desing of Business
Merle P. Martin
Macmilllan Publishing Company
Herramientas CASE: metodología
estructurada para el desarrollo de sistemas
Williams S Davis, Editorial Parainfo.

**OBSERVACIONES:**
VAW es una herramienta de altas prestaciones y de manejo realmente sencillo, siendo el ciclo de aprendizaje mínimo. Por otra parte, el VAW es un producto en constante evolución que se adapta continuamente a las nuevas tecnologías. Las proximas versiones incorporarán técnicas de análisis y diseño orientado a objetos, soporte para UNIX, OS/2, ISO 9000, Total Quality Management, etc.

# WESTMOUNT I-CASE

**FABRICANTE:** WESTMOUNT ECHNOLOGY B.V.
Olaf Palmestrat 24, P.O. Box 5063, 2600 GB Delft
The Netherlands
TEL 15 12 02 67, FAX 15 12 02 67
**AÑO PRMERA VERSION:** 1989 **VERSION ACTUAL:** 3.1

**DESCRIPCION:**
WESTMOUNT I-CASE es una herramienta que trabaja en entornos gráficos X y OSF/Motif. Posee versiones que soportan Yourdon, Ward/Mellor, SSADM y OMT. Cubre el ciclo completo de vida del desarrollo de un sistema de información. Toda la información se almacena en un diccionario integrado en INGRES, INFORMIX, ORACLE o SYBASE. WESTMOUNT I-CASE genera aplicaciones cliente/servidor para los entornos de desarrollo 4GL de INGRES, Windows 4GL de INGRES, 4GL de INFORMIX, RDS de INFORMIX, ORACLE, UNIFACE, C, C++. WESTMOUNT I-CASE incluye un módulo para la generación automática de la documentación completa del sistema desarrollado. WESTMOUNT I-CASE trabaja en entorno UNIX.

**PRINCIPALES MODULOS:**
Módulo de análisis, permite la definición del modelo esencial mediante la utilización de técnicas estándar (DFD, M E/R, etc.). Módulo de arquitectura permite la definición del diseño global mediante el modelado de procesos y datos. Módulo de diseño, permite la definición del modelo detallado previo a la generación de SQL y 4GL. Módulo de programación, que permite la generación automática, a partir de los diagramas definidos en los módulos anteriores de: SQL, C y 4GL (Ingres, Informix, Uniface, Oracle). Módulo DOCWRITER, permite la generación automática de documentación asociada a un proyecto, pudiendo trabajar con: Framemaker, Interleaf o WordPerfect para UNIX.

**PLATAFORMAS:** UNIX y VMS

**FASES SOPORTADAS:** Análisis, Diseño, Generación de código y Mantenimiento

**TECNICAS Y METODOLOGIAS:** DFD, M E/R, Miniespecificaciones, DSD, SAD, Análisis estructurado de Yourdon, SSADM, WARM/MELLOR

**LENGUAJES GENERADOS:** SQL de Ingres/ Informix/ Oracle /Sybase, 4GL de Ingres, Windows/4GL de Ingres, 4GL de Informix, RDS de Informix, Uniface, SQL-Forms y SQL-Menu de Oracle, Pro-C de Oracle, C, y C++
**ESTANDARES SOPORTADOS:** CDIF -Case Data Interchange Forma
**DOCUMENTACION/BIBLIOGRAFIA:**
Existe un conjunto completo de manuales como documentación de la herramienta.

**OBSERVACIONES:**
Como principales características de WESTMOUNT I-CASE, podemos destacar que es una herramienta que está en la linea de sistemas abiertos (UNIX y VMS), arquitectura cliente servidor; sistemas de administración de base de datos relacionales (ingres, Informix, Oracle y Sybase); Cubre el ciclo completo de vida de un S>I> llegando a generar código especifico sw RDBMS con el que se integra; dispone de una versión OMT orientada a objetos.

## CONCLUSION

Hoy en México probablemente no exista ninguna industria que se precie de tal que no tenga algún tipo de computadora o soporte informático. Y sobre todo en lo referente a software, las opciones parecen multiplicarse a diario. En realidad la cuestión ya no es si una industria usa recursos informáticos o no -un asunto fuera de discusión-, si no el grado de integración de estas herramientas, la profundidad de su aprovechamiento y el monto de las inversiones dirigidas a implementarlas y mantenerlas vigentes.

Cuando alguna empresa continua haciendo lo mismo y por milagro se esperan resultados diferentes y esto no ocurre, el empresario debe preguntarse que cambiar para obtener mejores resultados y no esperar a que estos lleguen sino buscarlos. La respuesta: mejores equipos y tecnologías, y una capacitación enfocada a obtener mejorias en la operación.

En la década pasada, la industria mexicana no estaba en condiciones de incorporar este tipo de equipos, y muchas empresas no veían la necesidad de mejorar porque se movían en mercados virtualmente cautivos, que no premiaban la eficiencia ni las inversiones en tecnología. Ahora con la apertura de las fronteras, la competencia irrestricta y la inserción en los mercados externos, esta necesidad se ha vuelto urgente y ha dado lugar a un verdadero boom en la oferta y la adquisición de hardware y software.

La actualización acelerada no siempre a dado buenos resultados. Hay algunos expertos que hablan de "exceso" de oferta, sobre todo en materia de software, que en ocasiones se ofrece y se vende con argumentos no funcionales, como que algunos programas son utilizados por grandes empresas transnacionales o que sirvieron para desarrollar proyectos de la NASA.

El punto crucial, según esta visión, es que el futuro usuario debe tener una idea clara de la problemática y las necesidades de su negocio, y en función de ellas escoger la herramienta que mejor sirva. En el trabajo hablamos de herramientas automatizadas, de herramientas CASE (Front-end, Back-end e integrales) -se lamenta- pero no las conocemos. Nos parecen asuntos ajenos a nuestra realidad, mientras en los mercados

globalizados se están implementando estándares para adoptarlas dentro de la industria. Tomando esto en cuenta uno de los objetivos de la tesis consistía en facilitar elementos que permitieran aumentar el conocimiento sobre tecnologia CASE, principalmente enfocandos hacia aspectos cómo describir el ambiente en el cual se desenvuelven, mencionar su definición y caracteristicas fundamentales, proponer una clasificación, aportar un método de adopción y ofrecer una lista con un conjunto de productos existentes en el mercado.

Sin embargo durante el desarrollo del trabajo se me aconsejó realizar la evaluación de cada producto del catálogo, a través de una tabla de propiedades, que mostrara: el costo, adaptabilidad, portabilidad, tiempo de implantación, mantenimiento, etc., a estas se les otorgaria una calificación con la finalidad de señalar los puntos a favor de cada herramienta y posteriormente utilizar esta evaluación para facilitar la elección al usuario. Lo anterior con el propósito de mejorar la investigación.

La idea no parecia mala, por lo que de inmediato me di a la tarea de localizar las fuentes de información para efectuar las mejoras al trabajo. Sin embargo después de consultar en publicaciones sobre el tema sin tener ningún éxito, intente contactar a distribuidores de software dándome cuenta que de no ser el investigador de una revista especializada o el líder de proyecto de alguna empresa interesada no conseguiria ninguna información de ellos. Agotados estos recursos opté por localizar alguna persona especializada en tecnologia CASE, fue asi que conoci al Ing. Edmundo Valenzuela Rodriguez que tuvo la oportunidad de tomar cursos en Japón, además de ser iniciador de los cursos de herramientas CASE en el DGSCA de Ciudad Universitaria, al consultarle el problema me explicó que la investigación que llevaba a cabo era poco recomendable, debido principalmente a la dificultad de conseguir cada uno de los productos del catálogo, además de no contar con el medio ambiente de una empresa para evaluar su impacto en la organización. Esto lamentablemente retarda y limita considerablemente la investigación, además de incrementar el costo y condicionarla a los recursos que se encuentran a su alcance.

Sin embargo este proceso de investigación sirvió para llegar a la siguiente conclusión: si alguna empresa desea mejorar su área de desarrollo de sistemas a través de la adopción de una herramienta CASE, en lo primero que debe pensar es en los recursos (económicos, humanos, técnicos, etc.) con los que cuenta, por ejemplo: una aplicación sencilla la cual será desarrollada por un solo programador, no requiere de un CASE. Una

aplicación para una empresa mediana puede apoyarse en un CASE del tipo FRONT-END el cual beneficie a las fases de análisis y diseño, además de ser accesible para su economia, finalmente una aplicación para una gran corporación (Bancos, Casas de Bolsa, Multinacionales, etc.) las cuales tienen una capacidad financiera solvente pueden obtener un CASE integrado que los ayude durante todo el ciclo de desarrollo.

Tomando en consideración lo anterior asi como la forma en que se amoldan las caracteristicas de las herramientas del catálogo al perfil de la organización, se puede facilitar la decisión sobre la compra del producto correcto, esto sin dejar pasar por alto el proceso de asimilación que la tecnología CASE lleva consigo.

**La selección de una herramienta, por lo tanto, requiere conocimiento de las necesidades propias de cada usuario, de las potencialidades del software escogido y sobre todo de un plan de adopción e implantación dentro de la empresa, el cual ofrecemos con este trabajo.**

BIBLIOGRAFÍA

SEEN, James A.
Análisis y Diseño de Sistemas de Información
McGraw-Hill

TEAGUE y PIDGEON
Structured Analysis Methods for Computer Information Systems
Macmillan Publishing Company

KENDALL y KENDALL
Análisis y diseño de sistemas
Prentice-Hall

DAVIS, Williams S.
Herramientas CASE
Editorial Parainfo

FISHER, Alan S.
CASE Using Software Development Tools
Jonh Wiley & Sons Inc.

MARTIN, James
Estructured Techniques the Basis for CASE
Prentice-Hall

PIATTINI y DARYANANI
Elementos y herramientas en el desarrollo de sistemas de información
Editorial Parainfo

MC CLURE, C
CASE: la automatización del software
Ed. Rama Madrid 1992

HUFF, C. C.
Elements of a realistic CASE tool Adopcion Budget
CACM Vol. 35, No. 4 Abril
pp. 45-54

NORMAN, R. J. y FORTE, G.
Automating the software development process, CASE in the 90's
CACM Vol. 35, No. 4 Abril
pp. 27-32