HERRAMIENTAS CASE

INTRODUCCION

Las Herramientas de Ayuda al Desarrollo de Sistemas de Información, surgieron para intentar dar solución a los problemas inherentes a los proyectos de generación de aplicaciones informáticas: plazos y presupuestos incumplidos, insatisfacción del usuario, escasa productividad y baja calidad de los desarrollos. Algunas de estas herramientas se dirigen principalmente a mejorar la calidad, como es el caso de las herramientas CASE (Computer Aided Software Engineering− Ingeniería de Software Asistida por Computadora). Otras van dirigidas a mejorar la productividad durante la fase de construcción, como es el caso de los lenguajes de cuarta generación (4GL−Fourth Generation Language).

En el presente trabajo se describe una de las principales herramientas de ayuda al desarrollo de Sistemas de Información, existentes en la actualidad: CASE. También se describe su funcionalidad y las características más relevantes, con la finalidad de ayudar en la elección de la herramienta adecuada.

Se describen las funcionalidades básicas de los diferentes tipos de CASE. Por último, se analizan las tendencias tecnológicas y del mercado.

HERRAMIENTAS DE AYUDA AL DESARROLLO

Es un software que facilita la producción de aplicaciones a la medida. Existe una amplia gama de posibles productos que se pueden incluir en esta definición, desde los lenguajes de programación (Cobol, FORTRAN, C, C++, o combinación de ellos, etc.), hasta sofisticados y complejos productos como las herramientas CASE.

Se consideran herramientas de ayuda al desarrollo a:

- Herramientas de ingeniería de software asistida por computadora o CASE.
- Lenguajes de cuarta generación o 4GL.
- Otras herramientas: gestión de proyectos, gestión de la configuración, ayuda en las pruebas, control de calidad, bibliotecas de clases de objetos, etc.

Es importante puntualizar que las fronteras entre unas y otras no siempre están claramente definidas.

DEFINICION DE HERRAMIENTAS CASE

Son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

El empleo de herramientas Case permiten integrar el proceso de ciclo de vida:

- Análisis de datos y procesos integrados mediante un repositorio.
- Generación de interfaces entre el análisis y el diseño.
- Generación del código a partir del diseño.
- Control de mantenimiento.

TIPOS DE CASE

No existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase determinada. Podrían clasificarse atendiendo a:

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

Las herramientas CASE, en función de las fases del ciclo de vida abarcadas, se pueden agrupar de la forma siguiente:

- **Herramientas integradas, I−CASE** (Integrated CASE, CASE integrado): abarcan todas las fases del ciclo de vida del desarrollo de sistemas. Son llamadas también CASE workbench.

Las herramientas I−CASE se basan en una metodología. Tienen un repositorio y aportan técnicas estructuradas para todas las fases del ciclo de vida. Estas son las características que les confieren su mayor ventaja: una mejora de la calidad de los desarrollos. Sin embargo, no todas ellas son modernas en el sentido de aprovechar la potencia de las estaciones de trabajo o la utilización de lenguajes de alto nivel o técnicas de prototipo.

- **Herramientas que comprenden algunas fases del ciclo de vida de desarrollo de software:**

- **Herramientas de alto nivel, U−CASE** (Upper CASE − CASE superior) o front−end, orientadas a la automatización y soporte de las actividades desarrolladas durante las primeras fases del desarrollo: análisis y diseño.

Una estrategia posible es utilizar una U−CASE para análisis y diseño, combinada con otras herramientas más modernas para las fases de construcción y pruebas. En este caso, habría que vigilar cuidadosamente la integración entre las distintas herramientas.

- **Herramientas de bajo nivel, L−CASE** (Lower CASE − CASE inferior) o back−end, dirigidas a las últimas fases del desarrollo: construcción e implantación.
- **Juegos de herramientas o toolkits**, son el tipo más simple de herramientas CASE. Automatizan una fase dentro del ciclo de vida. Dentro de este grupo se encontrarían las herramientas de reingeniería, orientadas a la fase de mantenimiento.

Otra posible clasificación, utilizando la funcionalidad como criterio principal, es

la siguiente:

- **Herramientas de planificación de sistemas de gestión.** Sirven para modelar los requisitos de información estratégica de una organización. Proporcionan un "metamodelo" del cual se pueden obtener sistemas de información específicos. Su objetivo principal es ayudar a comprender mejor cómo se mueve la información entre las distintas unidades organizativas. Estas herramientas proporcionan una ayuda importante cuando se diseñan nuevas estrategias para los sistemas de información y cuando los métodos y sistemas actuales no satisfacen las necesidades de la organización.

- **Herramientas de análisis y diseño.** Permiten al desarrollador crear un modelo del sistema que se va a construir y también la evaluación de la validez y consistencia de este modelo. Proporcionan un grado de confianza en la representación del análisis y ayudan a eliminar errores con anticipación. Se tienen:

- Herramientas de análisis y diseño (Modelamiento).
- Herramientas de creación de prototipos y de simulación.

- Herramientas para el diseño y desarrollo de interfaces.
- Máquinas de análisis y diseño (Modelamiento).

- **Herramientas de programación.** Se engloban aquí los compiladores, los editores y los depuradores de los lenguajes de programación convencionales. Ejemplos de estas herramientas son:

- Herramientas de codificación convencionales.
- Herramientas de codificación de cuarta generación.
- Herramientas de programación orientadas a los objetos.

- **Herramientas de integración y prueba:** Sirven de ayuda a la adquisición, medición, simulación y prueba de los equipos lógicos desarrollados. Entre las más utilizadas están:

- Herramientas de análisis estático.

- Herramientas de codificación de cuarta generación.
- Herramientas de programación orientadas a los objetos.

- **Herramientas de gestión de prototipos.** Los prototipos son utilizados ampliamente en el desarrollo de aplicaciones, para la evaluación de especificaciones de un sistema de información, o para un mejor entendimiento de cómo los requisitos de un sistema de información se ajustan a los objetivos perseguidos.

- **Herramientas de mantenimiento:** La categoría de herramientas de mantenimiento se puede subdividir en:

- Herramientas de ingeniería inversa.

- Herramientas de reestructuración y análisis de código.
- Herramientas de reingeniería.

- **Herramientas de gestión de proyectos.** La mayoría de las herramientas CASE de gestión de proyectos, se centran en un elemento específico de la gestión del proyecto, en lugar de proporcionar un soporte global para la actividad de gestión. Utilizando un conjunto seleccionado de las mismas se puede: realizar estimaciones de esfuerzo, coste y duración, hacer un seguimiento continuo del proyecto, estimar la productividad y la calidad, etc. Existen también herramientas que permiten al comprador del desarrollo de un sistema, hacer un seguimiento que va desde los requisitos del pliego de prescripciones técnicas inicial, hasta el trabajo de desarrollo que convierte estos requisitos en un producto final. Se incluyen dentro de las herramientas de control de proyectos las siguientes:

- Herramientas de planificación de proyectos.
- Herramientas de seguimiento de requisitos.
- Herramientas de gestión y medida.

- **Herramientas de soporte.** Se engloban en esta categoría las herramientas que recogen las actividades aplicables en todo el proceso de desarrollo, como las que se relacionan a continuación:

- Herramientas de documentación.
- Herramientas para software de sistemas.
- Herramientas de control de calidad.
- Herramientas de bases de datos.

Otra clasificación, diferencia las funciones CASE en cinco grupos:

- **Repositorio.** Funcionan en torno a un repositorio central, siendo éste el núcleo fundamental que contiene todas las definiciones de objeto y sus relaciones. Los objetos pueden ser especificaciones del sistema en forma de diagramas de flujo de datos, diagramas entidad−relación, esquemas de bases de datos, diseños de pantallas, etc. El repositorio es un concepto más amplio que el de diccionario de datos y soporta a los demás grupos de funciones. No es fácil encontrar en el mercado productos Case con funcionalidades estrictamente a las de repositorio, ya que, a pesar de su innegable importancia, tienen un carácter auxiliar de los demás grupos de funciones. Cualquier sistema Case poseerá un repositorio propio o bien, trabajará sobre un repositorio suministrado por otro fabricante o vendedor.

- **Reingeniería.** Los sistemas Case permiten establecer una relación estrecha y fuertemente formalizable entre los productos generados a lo largo de distintas fases del ciclo de vida, permitiendo actuar en el sentido especificaciones−código (ingeniería "directa") y también en el contrario (ingeniería "inversa"). Ello facilita la realización de modificaciones en la fase más adecuada en cada caso y su traslado a las demás. Al conjunto de facilidades proporcionadas por la ingeniería «directa» e "inversa" se le denomina "reingeniería".

- **Soporte del ciclo de vida.** El ciclo de vida de una aplicación o de un sistema de información se compone de varias etapas, que van desde la planificación de su desarrollo hasta su implantación, mantenimiento y actualización. Aunque el número de fases puede ser variable en función del nivel de detalle que se adopte, pueden de modo simplificado, identificarse las siguientes:

- Planeamiento.
- Análisis y Diseño.
- Implantación (programación y pruebas).
- Mantenimiento y actualización.

Los sistemas Case pueden cubrir la totalidad de estas fases o bien especializarse en alguna(s) de ellas. En este último caso se pueden distinguir sistemas de "alto nivel" ("Upper Case"), orientados a la autonomía y soporte de las actividades correspondientes a las dos primeras fases y, sistemas de "bajo nivel" ("Lower Case"), dirigidos hacia las dos últimas. Los sistemas de "alto nivel" pueden soportar un número más o menos amplio de metodologías de desarrollo.

- **Soporte de proyecto.** Este tipo de funciones hace referencia al soporte de actividades que se producen durante el desarrollo, derivadas fundamentalmente del trabajo en grupos, tales como facilidades de comunicación, soporte a la creación, modificación e intercambio de documentación, herramientas personales, controles de seguridad, etc. Los sistemas Case pueden conceder a estas cuestiones una importancia variable por lo cual el soporte de proyecto constituye un factor de diferenciación.

- **Mejora continua de calidad.** Aunque frecuentemente se asocia a los sistemas Case con la mejora de la productividad en el desarrollo de aplicaciones, debe tenerse en cuenta que una de las principales ventajas estriba también, en la mejora de la calidad de los desarrollos realizados. Determinados sistemas Case enfatizan más sobre este punto que sobre el anterior, introduciendo herramientas que permiten ejercer un control intenso de garantía de calidad del software desarrollado desde las primeras fases de su ciclo de vida.

BENEFICIOS DE LAS HERRAMIENTAS CASE

Entre los beneficios ofrecidos por la tecnología CASE se encuentran los siguientes:

- **Facilidad para la revisión de aplicaciones**

La experiencia muestra que una vez que las aplicaciones se implementan, 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 las especificaciones más que por los 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 los datos, encabezados y mensajes. Los ajustes necesarios al diseño se hacen con rapidez para alterar la presentación y las características de la interface. Sin embargo, no se prepara el código fuente, de naturaleza orientada hacia procedimientos, como una parte del prototipo.

Como disyuntiva, el desarrollo de prototipos puede producir un sistema que funcione. Las características de entrada y salida son desarrolladas junto con el código orientado hacia los procedimientos y archivos de datos.

Muchas herramientas CASE soportan las primeras etapas del desarrollo del prototipo. Muy pocas brindan apoyo durante todo el proceso de desarrollo del prototipo. Las que proporcionan la capacidad para generar código soportan de hecho todo proceso, ya que el código puede ser generado al inducir la actividad de generación después de cambiar las especificaciones o requerimientos.

- **Generación de código**

Como ya se mencionó, algunas herramientas CASE tienen la capacidad de producir el código fuente. La ventaja más visible de esta característica es la disminución del tiempo necesario para preparar un programa. Sin embargo, la generación del código también asegura una estructura estándar y consistente para el programa (lo que tiene gran influencia en el mantenimiento) y disminuye la ocurrencia de varios tipos de errores, mejorando de esta manera la calidad. Las características de la generación del código permiten volver a utilizar el software y las estructuras estándares para generar dicho código, así como el cambio de una especificación modular, lo que significa volver a generar el código y los enlaces con otros módulos. Ninguna de las herramientas que existen en el presente es capaz de generar un código completo en los dominios.

- **Mejora en 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 practicas de desarrollo. Parece ser que las herramientas CASE disminuyen el tiempo de desarrollo, una característica que es importante para los usuarios. Las herramientas afectan la naturaleza y cantidad de interacción entre los encargados del desarrollo y el usuario. 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.

- **Soporte interactivo para el proceso de desarrollo**

La experiencia ha demostrado que el desarrollo de sistemas es un proceso interactivo. Las herramientas CASE soportan pasos interactivos al eliminar el tedio manual de dibujar diagramas, elaborar catálogos y clasificar. Como resultado de esto, se anticipa que los analistas repasarán y revisarán los detalles del sistema con mayor

frecuencia y en forma más consistente.

DEBILIDADES DE LAS HERRAMIENTAS CASE

Las herramientas CASE tienen puntos débiles significativos, que van desde la confiabilidad en los métodos estructurados hasta su alcance limitado, 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 de desarrollo de sistemas. Por si sola, esta característica puede convertirse en la principal limitante ya que no todas las organizaciones emplean métodos de análisis estructurado.

Los métodos estructurados, introducidos en la década de los setenta, 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 lato entre los profesionales de sistemas de información − de acuerdo con algunas estimaciones (Yourdon), casi el 90% de todos los analistas esta 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 del 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 tanto, debe tener precaución al seleccionar una herramienta de este tipo.

Existen dos significados para las palabras soporte de la metodología. Una herramienta puede: 1) dar soporte a los diagramas que emplea una metodología o 2) soportarlos e imponer la metodología, sus reglas y 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.
- 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 (por ejemplo: diagramas de flujos de datos, un diccionario de datos y facilidades para la descripción de procesos), pero no el marco de referencia, reglas y procedimientos que en realidad constituyen el núcleo de la metodología. Aunque se puede llevar a cabo acciones básicas para la validación de diseños y diagramas para detectar componentes faltantes, éstas son sólo funciones mecánicas. Por otra parte, esta clase de herramientas no puede proporcionar ayuda metodológica o pedir al usuario que realice tareas necesarias para la metodología que aún esta sin terminar.

Estas herramientas mejoran la productividad al efectuar tareas tediosas y de documentación, aunque ellas no puedan asegurar buenos resultados. Desde el punto de vista funcional, las capacidades que brindan para garantizar la calidad son mínimas.

- **Conflictos en el uso de los diagramas**

Las herramientas difieren en el uso que hacen 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 puede 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 en una organización.

- **Diagramas no utilizados**

En general, los productos CASE emplean gráficas para modelar y generar informes sobre el análisis y 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, propician 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 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.

- **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 especifico. 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 en torno a la base 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 de información. 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 computadoras incluyen la capacidad de verificar las especificaciones para determinar su complementes 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 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 poco satisfactorios. No ofrecen o evalúan, soluciones potenciales para los problemas relacionados con sistemas. Y tampoco existe una

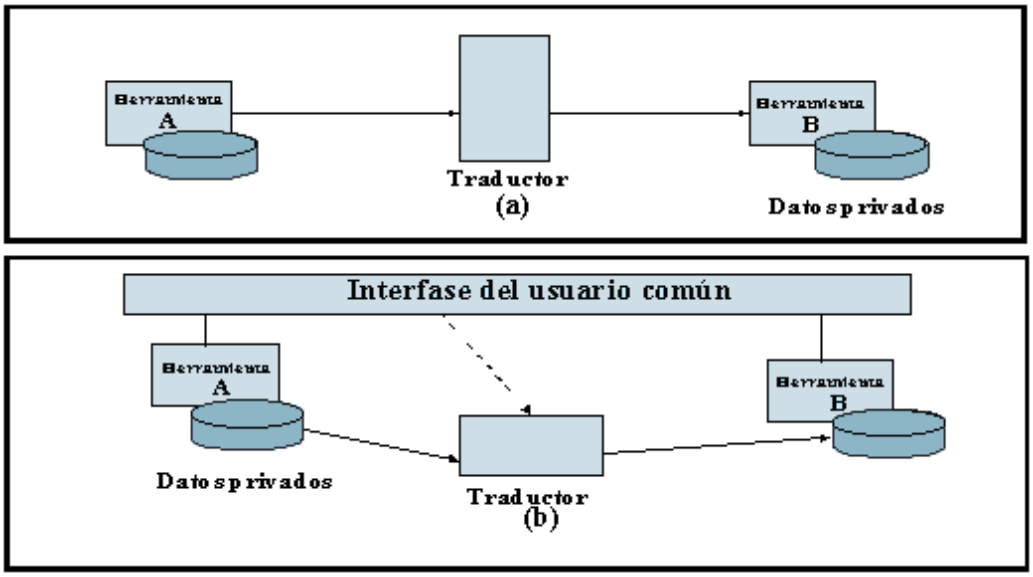
garantía clara para que dos analistas que utilicen los mismos métodos aplicados a información idéntica, formulen recomendaciones igualmente aceptables.

OPCIONES DE INTEGRACION

Las herramientas Case pueden ser integradas de muchas formas. En un extremo se utiliza una herramienta CASE de forma aislada. Se crea un número limitado de elementos de configuración de software (documentos, programas o datos) que se manipulan mediante una única herramienta y cuya salida tiene el formato de copia de pantalla y/o documentación gráfica. En cierto sentido, el enlace con el resto del entorno de desarrollo se realiza mediante copias en papel que gestiona el ingeniero.

Pocas herramientas CASE se utilizan en forma aislada. Se suele disponer de las siguientes opciones:

- Intercambio de datos.
- Acceso común a herramientas.
- Integración de datos.
- Integración total.



a) Intercambio de datos. La mayoría de las herramientas permiten exportar datos en forma de archivo sin estructura con un formato conocido. Esto permite un intercambio de datos punto a punto entre las distintas herramientas CASE, utilizando normalmente un "filtro" de transmisión intermedio.

La desventaja del intercambio de datos punto a punto está en que, a menudo, sólo parte de los datos exportados es utilizable por la herramienta receptora, ya que no fue diseñada para ser totalmente compatible. Además, a medida que evoluciona el software, la necesidad de transferir archivos cada vez que se hace un cambio pequeño puede llevar mucho tiempo. Las versiones pueden quedar "desfasadas" fácilmente, perdiéndose la posibilidad de transferencia, la cual suele ser en un único sentido. No hay posibilidad de que los cambios se reflejen en ambos sentidos y, es difícil hacer comprobaciones cruzadas de documentos y mantener la integridad de la configuración a través de las distintas herramientas que se estén utilizando.

b) Acceso común a herramientas. Permite al usuario utilizar distintas herramientas de forma similar, por ejemplo a través de un menú desplegable del gestor de ventanas del sistema operativo. En un entorno multitarea, un usuario podría abrir simultáneamente varias herramientas, coordinando manualmente sus entradas y comparando las representaciones de diseño a medida que evolucionan. Por ejemplo, el usuario podría visualizar un diagrama de flujo de datos, un diagrama de estructura, un diccionario de datos y un

segmento de código fuente, todos mantenidos por diferentes herramientas. En estos entornos, el intercambio de datos de herramienta a herramienta podría simplificarse llamando al procedimiento de traducción a través de un simple menú o de la selección de una macro. No es la opción más adecuada.

c) Integración de Datos.

- **Gestión común de datos.** Los datos de distintas herramientas se pueden mantener en una única base de datos lógica, que puede estar físicamente centralizada o distribuida. Hay una modalidad de fusión que permite combinar el trabajo de varias personas trabajando en diferentes partes de una aplicación.

Aunque los datos generados por las distintas herramientas se gestionan de forma conjunta en el nivel de gestión de datos comunes, las herramientas no conocen de forma explícita las estructuras de datos y la semántica de representación del diseño de las demás. Consecuentemente, se requiere una etapa de traducción (normalmente ejecutada manualmente) para permitir que una herramienta utilice la salida generada por otra.

- **Datos compartidos.** Las herramientas del nivel de datos compartidos tienen estructuras de datos y semántica compatible, pudiendo intercambiar datos sin necesidad de una etapa de traducción. Cada herramienta se diseña para ser compatible con las demás. Por esta razón, la mayor parte del intercambio de datos se da entre herramientas de un único fabricante o en casos en los que se han establecido relaciones estratégicas, entre distintos fabricantes para generar un conjunto de datos integrado, a veces, a petición de clientes importantes.

- **Interoperabilidad.** Las herramientas que combinan las características de acceso común y la capacidad de compartir datos, tienen la capacidad de interoperación. Esto representa el mayor nivel de integración entre herramientas diferentes. Sin embargo, hay otras propiedades del entorno global CASE que se pueden añadir para mejorar la efectividad del proceso de desarrollo de software.

d) Integración total. Para alcanzar la integración total del entorno CASE se necesitan dos características más: gestión de metadatos y capacidad de control. Los metadatos representan información sobre los datos de ingeniería generados por las distintas herramientas CASE. Esta información incluye:

- Definiciones de objetos (tipos, atributos, representaciones y relaciones válidas).
- Relaciones y dependencias entre objetos de granularidad arbitraria (p. ej.: un proceso en un diagrama DFD, una entidad única o un fragmento de código de una subrutina).
- Reglas de diseño del software (p. ej.: las distintas formas válidas de dibujar y equilibrar un diagrama de flujo de datos).
- Procedimientos (fases estándar, hitos, informes, etc.) y sucesos (revisiones, finalizaciones, informes de problemas, peticiones de cambios, etc.) del flujo de trabajo (proceso).

Normalmente, la parte de reglas y procedimientos de los metadatos se definen en forma de base de reglas, para facilitar su modificación según evoluciona el proceso de desarrollo del software. Por ejemplo, un nuevo método de diseño podría alterar las reglas de representación y cambiar los estándares del proceso de trabajo seguido hasta el momento.

La capacidad de control permite que cada herramienta pueda notificar al resto del entorno (a otras herramientas, al gestor de metadatos, al gestor de datos, etc.) la ocurrencia de sucesos significativos, así como enviar peticiones para la realización de acciones a otras herramientas y servicios por medio de un activador. Por ejemplo, una herramienta de gestión de configuración que haga una comprobación cruzada de la consistencia de documentos. La capacidad de control ayudará a mantener la integridad del entorno y proporcionará, también, un medio para automatizar procesos y procedimientos estándar. El activador puede estar incorporado en un entorno cerrado o puede estar visible para las distintas herramientas, a través de una interface de programación y un mecanismo de paso de mensajes.

La tecnología Case tendrá el mayor impacto si se integra a proyectos de innovación tecnológica que hoy en día contemple:

- Interfaces de programación visual.
- Soluciones cliente−servidor.
- Manejo de múltiples Bases de Datos.
- Independencia de la plataforma de hardware y software.
- Reingeniería de proceso de negocios.

COMPONENTES Y FUNCIONALIDADES DE UNA HERRAMIENTA CASE

A continuación se describen los principales componentes de una herramienta CASE y sus funcionalidades:

Repositorio. Base de datos central de una herramienta CASE. El repositorio amplia el concepto de diccionario de datos para incluir toda la información que se va generando a lo largo del ciclo de vida del sistema, como por ejemplo: componentes de análisis y diseño (diagramas de flujo de datos, diagramas entidad − relación, esquemas de bases de datos, diseños de pantallas), estructuras de programas, algoritmos, etc. En algunas referencias se le denomina Diccionario de Recursos de Información.

La mayoría de las herramientas CASE poseen un repositorio propio o bien trabajan sobre un repositorio suministrado por otro fabricante o vendedor.

Apoyándose en la existencia del repositorio se efectúan comprobaciones de integridad y consistencia:

- Que no existan datos no definidos.
- Que no existan datos autodefinidos (datos que se emplean en una definición pero que no han sido definidos previamente).
- Que todos los alias (referencias a un mismo dato empleando nombres distintos) sean correctos y estén actualizados.
- Las características más importantes de un repositorio son:

- **Tipo de información.** Que contiene alguna metodología concreta, datos, gráficos, procesos, informes, modelos o reglas.
- **Tipo de controles.** Si incorpora algún módulo de gestión de cambios, de mantenimiento de versiones, de acceso por clave, de redundancia de la información. La gestión de cambios y el mantenimiento de versiones, ayudarán en el caso de que convivan diferentes versiones de la misma aplicación o se tengan que realizar cambios en la versión en producción y en la de desarrollo, simultáneamente.
- **Tipo de actualización.** Si los cambios en los elementos de análisis o diseño se ven reflejados en el repositorio en tiempo real o mediante un proceso por lotes (batch). Esto será importante en función a la necesidad de que los cambios sean visibles por todos los usuarios, en el acto.
- **Reutilización de módulos para otros diseños.** El repositorio es la clave para identificar, localizar y extraer código para su reutilización.
- **Posibilidad de exportación e importación** para extraer información del repositorio y tratarla con otra herramienta (formateo de documentos, mejora de presentación) o incorporar al repositorio, información generada por otros medios.
- **Interfaces automáticas con otros repositorios** o bases de datos externos.

Módulos de diagramación y modelización. Algunos de los diagramas y modelos utilizados con mayor frecuencia son:

- Diagrama de flujo de datos.
- Modelo entidad − interrelación.

- Historia de la vida de las entidades.
- Diagrama Estructura de datos.
- Diagrama Estructura de cuadros.
- Técnicas matriciales.

Algunas características referentes a los diagramas son:

- **Número máximo de niveles** para poder soportar diseños complejos.
- **Número máximo de objetos** que se pueden incluir para no encontrarse limitado en el diseño de grandes aplicaciones.
- **Número de diagramas distintos en pantalla** o al mismo tiempo en diferentes ventanas.
- **Dibujos en formato libre** con la finalidad de añadir comentarios, dibujos, información adicional para aclarar algún punto concreto del diseño.
- **Actualización del repositorio por cambios en los diagramas.** Siempre resulta más fácil modificar de forma gráfica un diseño y que los cambios queden reflejados en el repositorio.
- **Control sobre el tamaño, fuente y emplazamiento de los textos** en el diagrama.
- **Comparaciones entre gráficos de distintas versiones.** De esta forma será más fácil identificar qué diferencias existen entre las versiones.
- **Inclusión de pseudocódigo** que servirá de base a los programadores para completar el desarrollo de la aplicación.
- **Posibilidad de deshacer el último cambio** facilitando que un error no conlleve perder el trabajo realizado.

Herramienta de prototipado. El objetivo principal de esta herramienta es poder mostrar al usuario, desde los momentos iniciales del diseño, el aspecto que tendrá la aplicación una vez desarrollada. Ello facilitará la aplicación de los cambios que se consideren necesarios, todavía en la fase de diseño.

La herramienta será tanto más útil, cuanto más rápidamente permita la construcción del prototipo y por tanto antes, se consiga la implicación del usuario final en el diseño de la aplicación. Asimismo, es importante poder aprovechar como base el prototipo para la construcción del resto de la aplicación. Actualmente, es imprescindible utilizar productos que incorporen esta funcionalidad por la cambiante tecnología y necesidades de los usuarios.

Los prototipos han sido utilizados ampliamente en el desarrollo de sistemas tradicionales ya que proporcionan una realimentación inmediata, que ayudan a determinar los requisitos del sistema. Las herramientas CASE están bien dotadas, en general, para crear prototipos con rapidez y seguridad.

Generador de código. Normalmente, se suele utilizar sobre ordenadores personales o estaciones de trabajo, por lo que el paso posterior del código al host puede traer problemas, al tener que compilar en ambos entornos.

Las características más importantes de los generadores de código son:

- **Lenguaje generado.** Si se trata de un lenguaje estándar o un lenguaje propietario.
- **Portabilidad del código generado.** Capacidad para poder ejecutarlo en diferentes plataformas físicas y/o lógicas.
- **Generación del esqueleto del programa o del programa completo.** Si únicamente genera el esqueleto será necesario completar el resto mediante programación.
- **Posibilidad de modificación del código generado.** Suele ser necesario acceder directamente al código generado para optimizarlo o completarlo.
- **Generación del código asociado a las pantallas e informes de la aplicación.** Mediante esta característica se obtendrá la interfase de usuario de la aplicación.

Módulo generador de documentación. El módulo generador de la documentación se alimenta del repositorio para transcribir las especificaciones allí contenidas.

Algunas características de los generadores de documentación son:

- **Generación automática** a partir de los datos del repositorio, sin necesidad de un esfuerzo adicional.
- **Combinación de información textual y gráfica,** lo que hace más fácil su comprensión.
- **Generación de referencias cruzadas.** Con ello se podrá localizar fácilmente en qué partes de la aplicación se encuentra un determinado objeto o elemento, con el fin de analizar el impacto de un cambio o identificar los módulos afectados por un determinado error.
- **Ayuda de tratamiento de textos.** Facilidad para la introducción de textos complementarios a la documentación que se genera de forma automática.
- **Interfase con otras herramientas:** procesadores de textos, editores gráficos, etc.

Módulo de gestión de proyectos. Algunos productos CASE incorporan un módulo para la gestión del proyecto de desarrollo de sistemas. Sus características más importantes serán analizadas en el apartado de otras herramientas.

ESTRATEGIAS DE IMPLANTACION DE UNA HERRAMIENTA CASE

- Identificar la magnitud de problemas a resolver en la Institución.
- Identificar el nivel estratégico que deben tener los sistemas.
- Evaluar los recursos de hardware y software disponibles en la Institución y el medio.
- Evaluar el nivel del personal.
- Efectuar un estudio de costo−beneficio definiendo metas a lograr.
- Elegir las herramientas apropiadas para la Institución.
- Establecer un programa de capacitación de personal de sistemas y usuarios
- Elegir una aplicación que reúna la mayor parte de los siguientes requisitos:

- Gran impacto de resultados.
- Disponibilidad de recursos.
- Mínimo nivel de riesgos.
- Máxima colaboración de usuarios.
- Tamaño reducido de solución.

Se establecerá interfases de compatibilidad de los nuevos sistemas que deben convivir con los sistemas anteriores.

CONSIDERACIONES PARA LA ELECCION DE CASE

La elección del Case va a depender de sus estrategias de desarrollo:

- Si tiene un gran volumen de aplicativos desarrollados, es conveniente contrastar lo realizado versus las técnicas de Análisis y Diseño.

- Si tiene presión por resultados a corto plazo, el empleo de un Lower Case le será de utilidad, si se basa en modelos de datos y procesos claros y definidos.
- Si desea realizar proyectos de gran envergadura es recomendable aplicar Upper y Lower Case.
- Si trabaja con archivos de grandes dimensiones, es recomendable que el Case soporte el Diseño de Bases de Datos.
- Si no tiene formación y experiencia en el manejo de metodologías es recomendable contar con asesoría especializada, que capacite al personal y supervise los avances de Análisis y Diseño.

- Evalúe la eficiencia del producto, en las pruebas unitarias y de integración, y fundamentalmente en las pruebas de sistemas.
- Considere los recursos apropiados para usar el Case, de HW (memoria, disco, concurrencia), de SW (versión de Sistema Operativo).

PROCESO DE ADQUISICION DE CASE

En la definición del objeto del contrato y los requisitos inherentes al mismo, así como en la valoración y comparación de ofertas de los proveedores, pueden intervenir muchos factores y de muy diversa índole, los cuales deberán estar recogidos dentro del conjunto de cuestionarios disponibles a tal efecto:

- De empresa o Institución.
- Económicos.
- Técnicos particulares.

No obstante, y a título orientativo en este apartado se hace mención de aquellos factores que, entre los anteriores, pueden intervenir en el proceso de adquisición de herramientas de ayuda al desarrollo y cuyo seguimiento debe efectuarse exhaustivamente.

- **Consideraciones en el Contrato de Adquisición**. Aparte de las cláusulas que son comunes a todos los contratos, se considerarán las siguientes:

- Requerimientos para el funcionamiento del Case.
- Incumplimiento de los requerimientos.
- Entrega e instalación de la herramienta.
- Instalación del Case.
- Certificación de la Instalación.
- Prueba de funcionamiento.
- Informe de fallas durante la prueba de aceptación.
- Responsabilidad de fallas.
- Penalidad en caso de no alcanzar el nivel de funcionamiento mínimo.
- Constancia de aceptación del equipo.
- Garantía de la herramienta.
- Asesoría técnica.
- Capacitación.
- Información técnica.

- **Estrategia de implantación**. Se debe comenzar aplicando la herramienta al desarrollo de un proyecto piloto, que no afecte a ningún área crítica y que sea de poca envergadura. Con la experiencia adquirida en este proyecto piloto, se podrá acometer el desarrollo de otros más complejos. Es importante asegurarse de poder utilizar la nueva herramienta sin tener que volver a escribir las aplicaciones existentes. En el caso particular de implantar por primera vez una herramienta CASE, es un factor crítico el apoyo del suministrador o de consultores con experiencia en las etapas iniciales.
- **Requisitos físicos**. Expresado en el Modelo de Tecnología de Arquitectura y características del puesto de desarrollo (procesador, memoria RAM, espacio en disco) y características del puesto de producción para las aplicaciones desarrolladas. Con ello se asegura que se dispone de los equipos necesarios o se prevé la necesidad de compra. Es posible que este factor obligue a la remodelación de todos los equipos y que su coste no sea asumible.

- **Requisitos lógicos**. Expresado en el Modelo de Tecnología, se debe analizar con especial atención la necesidad de otros módulos, no incluidos en el producto ofertado por el vendedor, para el correcto y completo funcionamiento de la herramienta (compiladores, módulos para trabajo en grupo, etc.).

Es fundamental comprobar si la herramienta tiene los módulos que incorporan las funcionalidades ofrecidas. Hay que tener cierta precaución cuando se analice un módulo ofertado, ya que hay casos en que para el funcionamiento de dicho módulo, es necesario adquirir otros módulos que, a veces, no se incluyen en la oferta.

- **Prueba en condiciones reales**. Si se va a instalar una herramienta CASE, se debe exigir al suministrador una prueba anterior a la adquisición de la herramienta CASE. Esta prueba debe realizarse en la propia instalación de destino.

La prueba se debe realizar en las condiciones más parecidas a las reales que se puedan conseguir e intentando simular el acceso de un número de usuarios, parecido al esperado. Durante la prueba se deberán evaluar conceptos objetivos fácilmente medibles.

No todas las herramientas cumplen con las prestaciones indicadas en los manuales, por lo que es aconsejable establecer un período de prueba para explorar la herramienta que se pretende adquirir. Una vez que en las especificaciones técnicas se hayan definido la plataforma física y lógica y las necesidades funcionales, mediante este período de prueba se podrá probar que la herramienta puede ser instalada en esa plataforma y soporta dichas funcionalidades.

- **Dependencia del proveedor**. Hay que evitar esta dependencia. A veces las herramientas llevan integradas partes de la plataforma operativa, lo cual las hace cerradas y propietarias. En el contrato de adquisición se debe contemplar la asesoría técnica, la capacitación y la información técnica.

Se debe encontrar el equilibrio entre la productividad de la herramienta y su carácter abierto, por ejemplo: independencia del proveedor.

- **Coste límite de adquisición**. En este apartado hay que analizar las posibilidades que ofrece el suministrador en cuanto a disponer de licencias individuales, grupos de licencia o licencias corporativas. Los costes varían considerablemente en función del tipo de licencia.

- **Coste de instalación de las aplicaciones generadas**. Hay que averiguar si una vez generada la aplicación y a la hora de distribuirla entre los usuarios, es necesaria la instalación de un módulo propiedad del suministrador (runtime). Este módulo en ocasiones no es de libre distribución y es preciso comprarlo. Hay que dejar claro este punto desde un principio.

- **Capacidad de integración**. Hay que tener en cuenta la plataforma o plataformas diferentes en que va a ser instalada la herramienta en cuestión, su tipología (fabricante, modelo y sistema operativo) y las características de la red de interconexión, cuando exista. Igualmente habrá que asegurar la integración con el software ya instalado. La necesidad de la integración con una herramienta CASE determinada, condiciona de forma decisiva la elección de un 4GL.
- **Portabilidad de la aplicación generada**. Cuando se pretende ejecutar la aplicación generada en diferentes plataformas, es un factor muy importante la portabilidad, tanto del código generado como de las especificaciones del diseño. En el caso particular de 4GL, este factor puede convertirse en decisivo si se tiene la intención de instalar la aplicación generada en entornos técnicos diferentes: sistemas operativos, plataformas físicas, interfases gráficas y protocolos de red. Un 4GL será realmente portable si el código generado se ejecuta en diferentes plataformas sin necesidad de adaptar los programas.

- **Capacidad técnica de la empresa y de la asistencia técnica que presta**. Es recomendable pedir referencias a otros usuarios de la Administración de este tipo de productos.

Además de los factores relevantes anteriores, en las herramientas CASE hay que prestar especial atención a:

- **Metodología y técnicas soportadas**. Para obtener éxito en la utilización de una herramienta CASE, es necesaria la existencia en la organización de una metodología. Si todavía no se cuenta con ninguna, la instalación de una herramienta CASE es un buen momento para implantarla.

Si ya se está utilizando una metodología, la herramienta CASE deberá ser capaz de adaptarse con el menor esfuerzo posible por parte del usuario, a dicha metodología y a las técnicas empleadas en cada una de sus fases.

- **Módulos que componen la herramienta CASE**. Las herramientas CASE suelen necesitar varios módulos, que se venden como productos independientes, para alcanzar su plena funcionalidad. Por lo tanto, en las especificaciones técnicas se deben señalar las funcionalidades a cubrir por la herramienta CASE, las cuales deben estar totalmente cubiertas por los módulos ofertados.

Igualmente se debe exigir que se detallen cuáles son las funcionalidades que cubre cada módulo y, para cada uno de ellos, cuáles de los otros son un pre−requisito para poder utilizarlo.

- **Licencias de Explotación y Desarrollo**. Es posible que para algunos o todos los módulos ofertados, existan dos versiones distintas. Una versión completa conocida normalmente como versión de "Desarrollo" y otra, con alguna de las funcionalidades restringidas o inexistentes, usualmente llamada versión de "Explotación" o "Producción" (a veces se utiliza runtime).

Es necesario que el suministrador detalle cuál de las dos versiones está ofreciendo para cada una de las licencias que se compren y, si alguna de ellas fuese una versión limitada, que especifique claramente cuáles de las funcionalidades ofertadas no se encuentran presentes en la versión restringida. Se debe especificar en el contrato de adquisición.

- **Funcionalidad del repositorio**. Los cambios que se hagan en el repositorio se deben realizar automáticamente en los programas. Por ejemplo: tenemos definido en nuestro repositorio el campo *sueldo* de longitud 5 y es cambiado a 9, luego en nuestro programa este campo ya tendrá longitud 9.

- **Costes indirectos**. Es muy importante tener en cuenta:

- La formación del personal y el efecto de la curva de aprendizaje. Para minimizar el coste de la curva de aprendizaje son importantes factores como la calidad de la formación inicial, la calidad de la documentación de la herramienta, la existencia de ayuda interactiva y la disponibilidad de la herramienta en castellano.
- La definición de estándares de uso y mantenimiento de la herramienta.
- La adaptación de la herramienta a las necesidades de la organización, tanto desde el punto de vista metodológico como tecnológico.
- La adaptación de las aplicaciones ya existentes al nuevo entorno.

CAUSAS DEL FRACASO DE 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 grandes apartados.

Deficiencias de la propia tecnología

Como ya hemos señalado, 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 puede destacar:

- Soporte parcial del ciclo de vida, lo que permite automatizar sólo parte de las actividades de

desarrollo, mientras que las 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 hardware y software sobre las que actúan.
- Escasa integración entre herramientas y el resto del entorno: SGBD, 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.
- 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 multiusuarios, ya que muchas herramientas nacieron para ordenadores personales y no han tenido una versión disponible para entornos UNIX o redes de área local hasta bastante tiempo después. Incluso aquellas que sí soportan esta forma de trabajo, no siempre gestionan adecuadamente la concurrencia entre diferentes desarrolladores. Es de esperar que la aplicación de técnicas de grupo de trabajo (groupware) ayuden en este sentido.
- Poca capacidad de adaptación (customización)
- Un alto coste, no sólo en la herramienta sino en la plataforma que ésta conlleva.

Todas estas deficiencias pueden ser superadas actualmente, en mayor o menor medida, evaluando varias herramientas, considerando, sí fuera posible, un cambio en hardware/software utilizado, intentando cuantificar el coste 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 de fiasco se debe a la utilización de herramientas CASE en problemas para los que no están preparadas, debido a que:

- Soportan una sola metodología (por ejemplo, especializada en el desarrollo de aplicaciones de gestión) y que se pretende emplear4 para construir sistemas en tiempo real.
- No soportan la técnica más adecuada; por ejemplo, en el diseño de bases de datos muy grandes puede ser conveniente emplear la integración de vistas, que muchas herramientas CASE no soportan.
- Metodologías y herramientas que funcionan relativamente bien en proyectos pequeños o medianos, puede fracasar en proyectos grandes.
- La selección se centra sólo en factores técnicos, por lo que la herramienta resulta insuficiente para los aspectos relativos a la gestión que todo lleva consigo.

Las medidas más eficaces 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 de la propia organización

A pesar de las deficiencias citadas anteriormente, la mayor parte de los fracasos en la adopción de herramientas CASE son debidos a las 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 del 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 gestión adecuada de las expectativas, siendo realista (conociendo la cultura de la empresa y su historia frente a cambios tecnológicos) y con una buena gestión.

TENDENCIAS TECNOLOGICAS Y DEL MERCADO DE LAS HERRAMIENTAS CASE

Las principales líneas de evolución hacia las que parecen encaminarse las herramientas CASE son:

- **CASE para sistemas bajo arquitectura cliente/servidor.** No hay que confundir el hecho de que una herramienta CASE funcione en un entorno de arquitectura cliente/servidor, con que el sistema desarrollado mediante una herramienta CASE vaya a funcionar bajo dicha arquitectura.

En la actualidad ya hay ejemplos de los dos casos, herramientas CASE que funcionan bajo un entorno cliente/servidor, en red y con un repositorio centralizado en un servidor y herramientas CASE que generan aplicaciones que funcionan en un entorno cliente/servidor, en las cuales se puede indicar dónde deben residir los componentes de la aplicación en tiempo de ejecución, liberando al programador de aspectos referidos a los protocolos de comunicaciones, seguridad, interfases gráficas de usuario, etc.

La línea de evolución, en este caso, vendrá marcada por versiones mejoradas de la herramienta, que faciliten cada vez más la distribución de los elementos de una aplicación entre los diferentes clientes y servidores y una mayor liberalización del programador, de todos los aspectos que no sean propios de la aplicación (protocolos de red, seguridad, etc.).

- **CASE multiplataforma.** Estas herramientas soportan las combinaciones dominantes de diferentes plataformas físicas, sistemas operativos, interfases gráficas de usuario, sistemas de gestión de bases de datos, lenguajes de programación y protocolos de red. En este sentido el futuro podrá ser de apertura creciente a nuevas plataformas y portabilidad más generalizada.

- **CASE para ingeniería inversa y directa.** Ya existen algunas herramientas de este tipo. Su evolución marcará notables mejoras en la obtención de los diseños a partir del código ya existente (ingeniería inversa) y la regeneración del mismo, una vez optimizado el diseño (ingeniería directa).
- **CASE para trabajo en grupo (groupware).** Estas herramientas se centran en el proceso de desarrollo más que en el producto a desarrollar, facilitando la integración de diferentes grupos humanos, pertenecientes incluso a empresas diferentes, trabajando conjuntamente en un gran proyecto. Deberían incorporar las facilidades clásicas de ofimática: correo electrónico, calendarios en línea, planificación de actividades, preparación de documentos, actas de reuniones, etc.
- **CASE para desarrollo de sistemas orientados a objetos.** En la actualidad existen algunas herramientas que cubren alguna de las fases del ciclo de vida de desarrollo de aplicaciones orientadas a objetos (interfase de usuario, análisis, diseño, programación, etc.). El objetivo futuro podría ser cubrir el ciclo de vida completo. Aunque hoy en día, la mayor efectividad se consigue con las herramientas CASE para métodos estructurados, en un futuro no muy lejano esta situación se invertirá a favor de las que soportan objetos.

La proliferación de este tipo de herramientas podrá verse retrasada debido al gran número de notaciones y metodologías de orientación a objetos distintas que existen en la actualidad.

En general, puede afirmarse que aquellas herramientas que soportan muchas notaciones, no consiguen realmente ayudar en la aplicación de una metodología con todo su proceso y validaciones correspondientes, sino que suelen quedarse, más bien, en un nivel exclusivamente gráfico. Por el contrario, las que cuentan con una sola metodología consiguen recoger prácticamente toda su semántica y ayudar al diseñador en la validación de los sistemas, además de generar un código de mayor calidad.

Es importante resaltar que las herramientas actuales permiten generar objetos: modelo "estático" y modelo "funcional", mas no el modelo "dinámico".

Todas estas herramientas CASE suelen generar código C++. Algunas simplemente la definición esquemática de las clases mientras que otras, pueden llegar a generar más de la mitad del código del sistema.

La programación orientada a objetos puede cambiar la forma que tienen las empresas de hacer negocio y como tal, necesita ser tratada cuidadosamente, tanto por las empresas u organismos, como por los fabricantes de tecnologías que proporcionan las soluciones. Los fabricantes tienen que ofrecer herramientas eficaces para ayudar a las empresas a explotar todas las potentes prestaciones de la tecnología de objetos (código reutilizable, programación modular y capacidad de modelización), para construir aplicaciones críticas y eficaces. Dentro de estas herramientas, tendrán un papel fundamental las herramientas CASE.

Una atención especial merecen las herramientas CASE adaptables, algunas de las cuales permiten que sea el propio usuario quien defina su metodología y los símbolos de las notaciones a utilizar. Estas herramientas se denominan **"meta−CASE"**.

A mediano y largo plazo otras posibles líneas de evolución serán:

- La utilización de la tecnología multimedia.
- La incorporación de técnicas de inteligencia artificial.
- Sistemas de realidad virtual.

CONCLUSIONES

Actualmente, la tendencia en el desarrollo de software está enfocada hacia las microcomputadoras como plataformas de ingeniería de software, que se interconectan mediante redes para que puedan comunicarse de forma efectiva. La base de datos del proyecto (también denominada biblioteca del proyecto o depósito de software), está disponible a través de un servidor de archivos en red que es accesible desde todas las estaciones de trabajo. Un sistema operativo que gestiona el hardware, la red y las herramientas, mantiene todo el entorno unido.

La arquitectura de entorno, compuesta por la plataforma hardware y el soporte del sistema operativo (incluida la red y la gestión de la base de datos), constituye la base del CASE. Pero el entorno CASE, en sí mismo, necesita otros componentes. Un conjunto de servicios de portabilidad constituyen un puente entre las herramientas CASE y su marco de integración y la arquitectura de entorno. El marco de integración es un conjunto de programas especializados que permite a cada herramienta CASE comunicarse con las demás, para crear una base de datos de proyectos y mostrar una apariencia homogénea al usuario final (el ingeniero de software). Los servicios de portabilidad permiten que las herramientas CASE y su marco de integración puedan migrar a través de diferentes plataformas hardware y sistemas operativos, sin grandes esfuerzos de adaptación.

La mayoría de las herramientas Case no han sido construidas utilizando todos los bloques componentes. Muchas de éstas son soluciones puntuales, esto es, una herramienta se utiliza como ayuda en una actividad concreta de ingeniería de software (por ejem.: modelización del análisis), pero no se comunica directamente con otras herramientas, porque no está unida a una base de datos de proyectos.

Aunque esta situación no es la ideal, una herramienta Case puede ser utilizada eficientemente, aún siendo una solución puntual.

En el nivel más bajo del espectro de integración está la herramienta individual (solución puntual). Cuando las herramientas proporcionan facilidades para el intercambio de datos (la mayoría lo hace), el nivel de integración aumenta ligeramente. Estas herramientas generan una salida en un formato estándar compatible con otras herramientas que puedan leer ese formato. En algunos casos, los que construyen herramientas CASE complementarias trabajan juntos para establecer un puente entre ellas (p. ej.: una herramienta de análisis y diseño que se une a un generador de código). Utilizando este enfoque, la compatibilidad entre herramientas puede generar productos finales que serían difíciles de desarrollar utilizando cada herramienta por separado. La integración por fuente única se da cuando un constructor de herramientas CASE integra diferentes herramientas y las vende como un único paquete. Aunque este enfoque es bastante efectivo, la mayoría de los entornos provenientes de una misma fuente tienen una arquitectura cerrada que hace difícil añadir nuevas herramientas de otros vendedores.

Al final del espectro de integración está el entorno de soporte de proyectos integrado (del inglés IPSE). Se crean estándares para cada uno de los bloques componentes. Los vendedores de herramientas CASE utilizan estos estándares IPSE para construir herramientas entre sí.

La principal ventaja de la utilización de una herramienta CASE, es la mejora de la calidad de los desarrollos realizados y, en segundo término, el aumento de la productividad. Para conseguir estos dos objetivos es conveniente contar con una organización y una metodología de trabajo además de la propia herramienta.

La mejora de calidad se consigue reduciendo sustancialmente muchos de los problemas de análisis y diseño, inherentes a los proyectos de mediano y gran tamaño (lógica del diseño, coherencia, consolidación, etc.).

La mejora de productividad se consigue a través de la automatización de determinadas tareas como la generación de código y la reutilización de objetos o módulos

BIBLIOGRAFIA

Castañeda G., Víctor. Revista Tecnología de Punta.

Herramientas para el desarrollo de Sistemas de Información.

http://www.inei.gob.pe/cpi/bancopub/libfree. Instituto de Estadística e

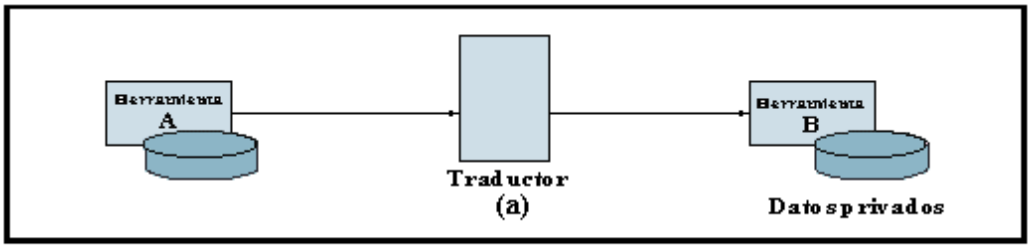
Informática. Lima Perú. Julio, 1997.

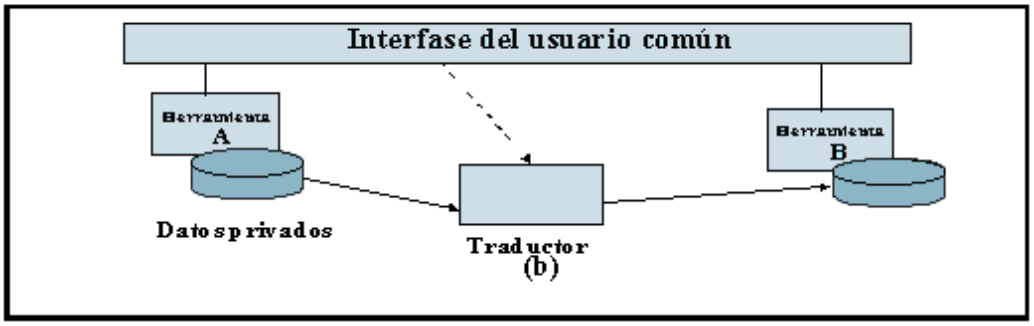
Ingeniería del Software.

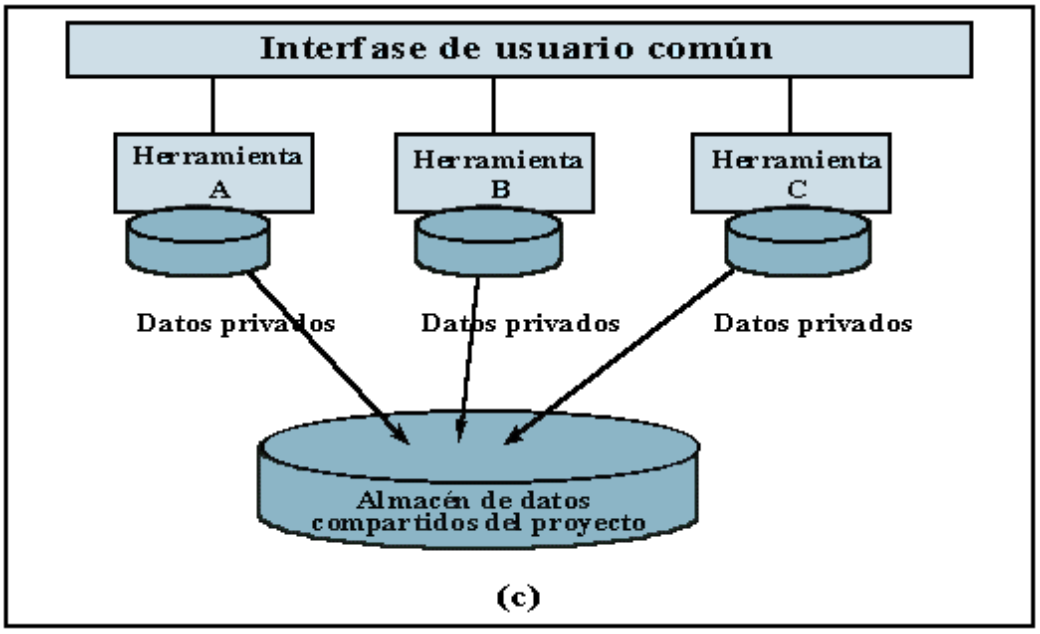
http://www.geocities.com/SiliconValley/lab/7538/

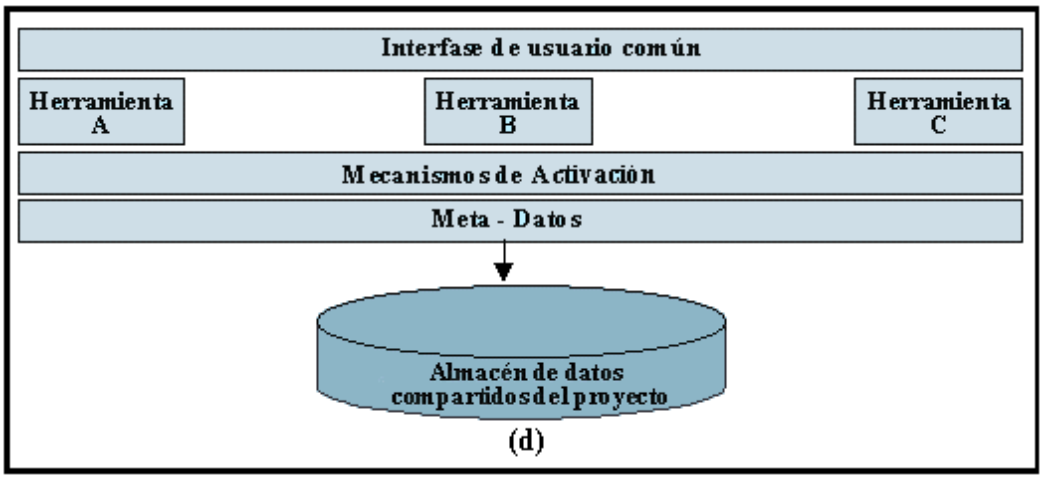
2

39


Herramienta A
Traductor
Herramienta B
Datos privados
(a)


Interfase del usuario común
Herramienta A
Herramienta B
Datos privados
Traductor
(b)


Interfase de usuario común
Herramienta A
Herramienta B
Herramienta C
Datos privados
Datos privados
Datos privados
Almacén de datos compartidos del proyecto
(c)


Interfase de usuario común
Herramienta A
Herramienta B
Herramienta C
Mecanismos de Activación
Meta - Datos
Almacén de datos compartidos del proyecto
(d)