Club de Investigación Tecnológica

Nuevas Tecnologías de Información

Preparado por: **Juan Ignacio Biehl,**
Róger Mayorga J., y
Roberto Sasso Rojas (editor)

Abril 1992

Editado y Publicado por Rho-Sigma, S.A.
a nombre del Club de Investigación Tecnológica.

San José, Costa Rica
Abril 1992

Resumen Ejecutivo

Casi a diario vemos el anuncio de nuevas tecnolgías, algunas de las cuales serán flor de un día, algunas madurarán y pasarán a formar parte del conjunto de tecnologías utilizadas en nuestra rutina diaria, otras vendrán a transformar el mundo.

En marzo 1990, los afiliados del Club de Investigación Tecnológica decidieron que se realizara una investigación de algunas de las más relevantes tecnologías emergentes.

Para este estudio se escogieron: Multimedia, Orientación a Objetos y AD/Cycle. De estas tecnologías, las primeras dos tenían años de estar madurando y empezaban a dar señales de estabilidad y la tercera estaba recién anunciada.

Para las tres tecnolgías se presenta primero una breve descripción de lo que son, incluyendo los conceptos básicos, luego se discuten las aplicaciones e implicaciones de las tecnologías, así como las ventajas y desventajas de su aplicación en nuestro medio.

La tecnología **Multimedia** ha producido sistemas que integran grandes cantidaes de información en formatos tan dicímiles como el sonido, las imágenes (estáticas y animadas),y el texto. La aplicabilidad de esta tecnología ya ha rebasado los confines del hogar y las instituciones educativas, hoy en día hay aplicaciones reales y valiosas que pueden aprovechar las empresas afiliadas al Club. Se presentan recomendaciones para ayudar al Departamento de Sistemas a apoyar la introducción de esta tecnologíua en las empresas.

La **Orientación a Objetos** es la más antigua de las tecnologías estudiadas, es también la más compleja y posiblemente la que podría tener mayor impacto. La Orientación a Objetos no es un producto, es una filosofía, un enfoque y un paradigma de la ingeniería de software. Hoy en día hay numerosos productos que apoyan o implementan dicho paradigma. Ya la orientación a objetos no se limita únicamente a la programación (como hace veinte años) sino que incluye metodologías de desarrollo de sistemas, sistemas operativos, bases de datos y ambientes de usuario final.

Se demuestra en el informe que la Orientación a Objetos es una alternativa viable para el desarrollo de sistemas en un futuro cercano.

El **AD/Cycle** fue anunciado por IBM en setiembre 1989, más como una dirección y una serie de planes e intenciones que como una tecnología disponible. El AD/Cycle propone una serie de interfases y estándares para la creación de un ambiente integrado y completo para la automatización del proceso de desarrollo de software de apliación.

La relevancia de este anuncio de IBM radica probablemente en la arquitectura abierta que propone, aprovechando así los esfuerzos pioneros de otros proveedores de software en el desarrollo de herramientas conocidas como Ingeniería de Software Asistida por el Computador (CASE).

AD/Cycle es el único planteamiento (de que tenemos conocimiento) que lleva a la integración de todas las herramientas de alta productividad existentes (y en desarrollo) incluyendo las herramientas para la administración del proceso mismo de desarrollo de apliaciones.

Se hace evidente en el informe que AD/Cycle tiene el potencial de cambiar radicalmente, en el mediano plazo, la forma en que desarrollamos software de aplicación.

Es, obviamente, de interés para los afiliados del Club el conocer y entender los conceptos y alcances del AD/Cycle para así prepararse y planear mejor la introducción de la automatización del desarrollo de sosftware en susu organizaciones.

De los Autores

El capítulo de Multimedia fue desarrollado por **Juan Ignacio Biehl**, Vice-Presidente de ICON (representante de Apple en Costa Rica, Panamá y Nicaragua). El Sr. Biehl realizó estudios de electrónica en el Instituto Tecnológico de Costa Rica y ha participado en numeroso cursos y seminarios especializados en Cupertino, California.

El capítulo de Orientación a Objetos fue desarrollado por **Róger Mayorga**, Licenciado en Computación e Informática por la Universidad de Costa Rica, Máster en Computación por la Universidad de Waterloo, Canada y actualmente profesor de la Maestría en Computación del I.T.C.R. y coordinador de Informática del IICA.

El capítulo del AD/Cycle fue desarrollado por **Roberto Sasso**, Doctor en Ingeniería de Software por la Universidad de Oxford, Presidente y Director de Investigación del Club de Investigación Tecnológica quien se desempeña actualmente como consultor independiente tanto en Costa Rica como en el exterior. El Dr. Sasso fue además el responsable de diseñar, coordinar y editar el presente informe.

Contenido ..... Página

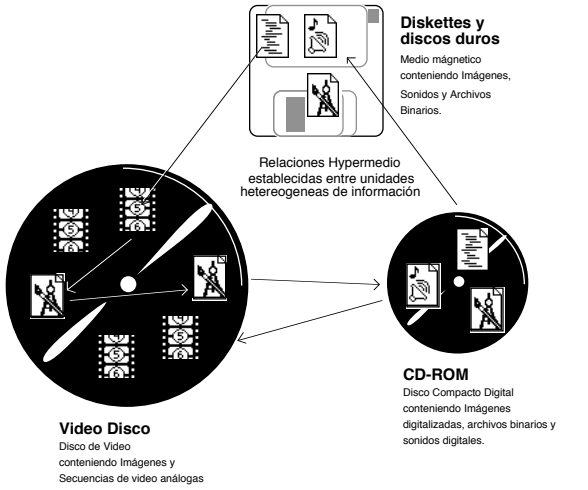
Multimedia

por: Juan Ignacio Biehl

Durante los últimos tres años, el término *Multimedia* ha cobrado popularidad en los círculos tecnológicos; sin embargo, no existe todavía un entendimiento claro de lo que significa y de lo que representa.

La definición de multimedia describe, más o menos acertadamente, un nuevo sistema de comunicación que integra información de varias fuentes mediante estructuras flexibles. Se acepta implícitamente la interactividad como característica de los materiales multimedios.

Para describirlo en función de medios conocidos cabe imaginar un producto, similar a una publicación, que nos permite explorar conceptos expresados en medios tan disímiles como el sonido, las imágenes estáticas o animadas - videos - y el texto, entre los que se establecen relaciones lógicas. Pensemos en un ornitólogo sentado frente a una estación "multimedia", estudiando un ave mediante la lectura de bibliografía en pantalla al tiempo que observa fotografías digitalizadas y escucha los diferentes cantos. En este medio, el usuario tiene acceso a una gran cantidad de información ordenada a la que solo tendría acceso por medios convencionales de una forma aislada e inconexa.



Multimedios. Relaciones y unidades de información en diferentes medios físicos y formatos.

El concepto de multimedia surge pues, para llenar la necesidad de entregar una gran cantidad de información de la manera más eficiente posible. Puede pensarse que es un paso necesario en la historia de la comunicación que funde medios como el libro, la televisión y la radio.

Multimedia

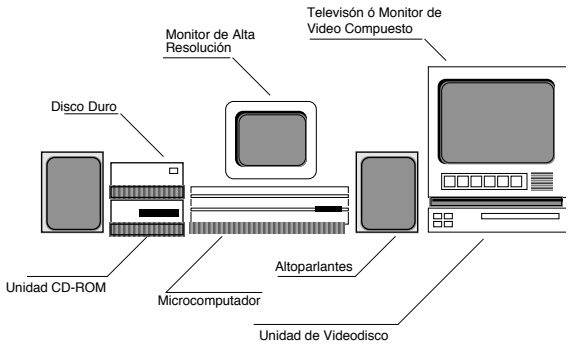
Hoy en día muchos de los materiales multimedios toman la forma de "Enciclopedias Ilustradas" que ya no solo integran ilustraciones para complementar el texto, sino que incluyen segmentos de video y audio digitalizado.

En este punto es importante diferenciar el *material multimedia* de las *aplicaciones o herramientas multimedia*. El primero, se refiere al producto terminado que se distribuye al usuario. Las segundas son las aplicaciones que se emplean para elaborar estos productos ó publicaciones. Es común que el material multimedia se distribuya "encerrado" en la aplicación, lo que permite al usuario modificar en algunos casos la estructura y aprovechar las unidades de información para sus propios fines. Algunos ejemplos comerciales de estos ambientes de desarrollo de material multimedia son HyperCard de Apple, MediaMaker de MacroMind, LinkWay de IBM, SuperCard de Silicon Beach y Course of Action de Authorware Inc.

Conviene también establecer con claridad las diferencias entre *multimedia* e hypermedia. Como se vio, los sistemas Multimedios involucran formatos heterogéneos en las unidades de información mientras que los Hypermedios pueden involucrar solo un medio. Los sistemas hypermedios que emplean exclusivamente texto se llaman *hypertexto*.

De momento, el estado de la tecnología obliga a emplear configuraciones de equipo con cierto grado de complejidad para brindar los servicios que requieren los multimedios.

Una estación típica consiste de un microcomputador que ejecuta el programa multimedios, normalmente operando bajo sistemas gráficos. El microcomputador también controla periféricos de almacenamiento y recuperación como unidad de videodisco y unidad CD-ROM. Esta configuración se complementa usualmente con un monitor de video (televisor) y altoparlantes.



Estación Multimedios. Configuración Típica

La unidad CD-ROM se emplea en algunos casos para almacenar simultáneamente la

aplicación, la información textual, imágenes estáticas y sonidos digitalizados. El video (full motion video) se almacena normalmente en videodiscos de 12". Estos, a diferencia de los discos compactos, almacenan información en formato análogo.

Sin embargo, día con día los fabricantes de productos electrónicos de consumo están unificando estos medios en uno solo que permita la distribución eficiente de material multimedio; Sony y Philips están patrocinando el estándar CD-I, al tiempo que Intel impulsa el formato DVI. Ambos sistemas permiten almacenar y recuperar video, sonido y datos de un mismo disco compacto.

Varios fabricantes de computadores personales distribuyen estaciones multimedia en el país. Los más importantes son Commodore, IBM, Tandy y Apple.

Estos fabricantes están integrando en productos únicos -estaciones multimedia- todos los componentes citados arriba. Entre estos, cabe distinguir las estaciones para el empleo de material multimedia *(delivery)* y las estaciones para la producción de material multimedia. Esta distinción se hace sin perjuicio de las capacidades de producción de las primeras.

Actualmente cada fabricante está proponiendo sus propias ideas para la implementación, pero todos comparten criterios sobre los formatos, el almacenamiento y la manipulación. El reciente acuerdo entre IBM y Apple se apoya grandemente en el deseo de las dos compañías de emplear un juego común de herramientas para el desarrollo de sistemas multimedia. La tecnología QuickTime, desarrollada por Apple como plataforma para la integración de medios dinámicos -sonido y video- , promete convertirse en el estándar de los sistemas futuros que desarrollen estas empresas. Hoy en día QuickTime se encuentra comercialmente disponible para Macintosh, y estará disponible a finales de 1992 para Windows. Mas de 200 aplicaciones comerciales ya han adoptado la tecnología.

Nótese que los interfaces "amigables" juegan un papel importante en el desarrollo de sistemas comerciales por que estos productos se orientan al usuario común, no al desarrollador profesional. En Costa Rica, al igual que en otros países altamente informatizados, este tipo de sistemas operativos se están imponiendo lo que facilitará llevar los beneficios de los multimedios a los usuarios finales.

Multimedia y nosotros

Para analizar el impacto de los multimedios en la sociedad es necesario observar las diversas formas que estos van adquiriendo conforme el gran público comienza a tener acceso a ellos. A continuación se describe la aplicación de esta tecnología en diversas áreas.

Un nuevo medio de comunicación

Al lado de los libros, revistas, videos, casetes, y publicaciones en general vamos a encontrar cada día más material multimedios. Tomarán inicialmente la forma de "kits" conteniendo un videodisco, un disco compacto CD-ROM y tal vez algún software complementario en disquettes. En un futuro estos serán sustituidos por un solo disco digital que reúna la información en los diferentes formatos.

Lo más destacado en este punto es entender que los multimedios constituyen un sistema para la distribución de contenido; esto los separa del software, entendido como herramienta. De hecho el desarrollo de material multimedio más un arte interdisciplinario más que una ciencia tecnológica. Con más probabilidad encontraremos que los "autores" importantes de estos productos serán más usuarios promedio (ó artistas en el mejor de los casos) que técnicos en programación.

Educación

Este material es ya bastante popular en medios educativos. En la instrucción escolar formal se ha encontrado dificultad para popularizarlo porque no se ha encontrado consenso sobre la forma que debe tomar el material didáctico. La instrucción a niños y jóvenes plantea problemas de concepto que no ha sido posible resolver a satisfacción de los expertos. Sin embargo estos problemas no son de orden tecnológico sino pedagógico.

Es más común encontrar material educativo en corporaciones y en universidades. Muchas empresas han adoptado este medio para la instrucción de sus empleados. El volumen de información que permiten distribuir estos sistemas acelera enormemente el proceso costoso del entrenamiento. Otra razón para que se haya adoptado en empresas es que la naturaleza interactiva y autoguiada del material prácticamente hace innecesario al instructor. El empleado puede sentarse solo frente a la estación de entrenamiento y aprender a su ritmo para finalmente ser evaluado automáticamente.

En Costa Rica ya se esta progresando aceleradamente hacia la creación de ambientes multimedios de enseñanza y aprendizaje. En la Facultad de Medicina de la Universidad de Costa Rica, bajo el patrocinio del programa NIDES, se ha instalado un completísimo Centro de Multimedia. Actualmente se está trabajando en varias modalidades para el desarrollo de material instructivo. Inicialmente se trabajó en el desarrollo de material para asistir la enseñanza de materias como fisiología y anatomía; actualmente se está comenzando a trabajar en material para el aprendizaje, más enfocado a la enseñanza autoguiada ya al estudio.

También en la Universidad de Costa Rica, se desarrollo material interactivo para guiar a los nuevos empleados en el funcionamiento y procedimientos de la Universidad.

Actualmente la Facultad de Música está implementando un sistema auxiliar para la enseñanza principalmente en historia de la música.

Para enfrentar exitosamente el futuro hay que examinar de cerca el progreso de la tecnología en el área de la educación ya que de ahí nos llegarán las “pistas” de aplicación en otras áreas.

Mercadeo y Ventas

Por razones similares, los multimedios han empezado a aparecer en empresas comercializadoras. El enorme volumen de información que contiene un catálogo por ejemplo, y el tipo de comunicación que se pretende establecer con el cliente, impulsa a las empresas a crear catálogos electrónicos interactivos que sustituyen casi por completo al distribuidor. Varias de las empresas que lo han adoptado emplean sistemas de distribución postal y de esa forma eliminan la necesidad de canales tradicionales de distribución. Otras lo usan para apoyar el trabajo de sus distribuidores.

El material multimedio en aplicaciones comerciales funde lo mejor de los comerciales de televisión con la no opresividad del catálogo. El cliente ve lo que quiere en detalle y evita lo que no le interesa.

Aún está un poco distante el día en que la mayoría de las empresas lleguen a sus clientes por este medio. El costo relativamente alto de una estación multimedia (No así el del material multimedia) mantiene reducido el mercado para estos sistemas. Sin embargo en las ciudades más caras del mundo: Tokio y Nueva York el costo de sostener cadenas de

distribución ordinarias le abre las puertas a sistemas que evitan la "Tienda" y reducen los costos asociados a la propiedad de la tierra y al transporte.

Entretenimiento

También guardando cierto paralelo con la orientación de la tecnología en educación, la aplicación de los multimedios en el entretenimiento es una tendencia perceptible. En la sociedad "hecha a la medida" del siglo XXI, el entretenimiento tiene que poder ajustarse a los gustos de las personas. Los "Invasores Espaciales" ya no ofrecen diversión a las nuevas generaciones; lo predecible y falto de contenido de este tipo de entretenimientos de video abre paso a los mucho más sofisticados juegos multimedios. La posibilidad de participar en juegos interactivos de aventuras con secuencias de video a todo color y con sonido estereofónico hace sonreír a niños y adultos por igual.

En el mercado norteamericano ya han aparecido los primeros productos comerciales de este género; Uno de los pioneros es casualmente un juego para adultos: "Virtual Valerie".

Servicios

Las empresas de servicios están comenzando a ver en los multimedios sistemas adecuados para orientar a sus clientes.

El comercio de intangibles puede aprovechar esa propiedad emocional de este medio para vender vacaciones a países lejanos ó "vender" diseños ya sean industriales o arquitectónicos. El poder traer hasta el comprador este tipo de productos antes de proveerlos mejora la posibilidad de éxito para muchos profesionales, ya sea en turismo, arquitectura ó ingeniería.

Incluso los gobiernos locales empiezan a encontrar razonable proveer al turismo de herramientas más sofisticadas que el clásico mapa. La cantidad de información que puede proveer un mapa turístico es en estos casos muy limitado. En Canadá se está popularizando la estación de información electrónica que ofrece, a través de un interfaz sencillo, acceso a información de bases de datos como clima, información bursátil y al mismo tiempo descripción - con video y sonido interactivo- de la atracciones principales de ciudades como Quebec y Montreal.

También se ven aparecer estaciones especializadas en los grandes centros de convenciones que ofrecen al visitante

orientación en las actividades y exhibiciones.

Implicaciones para el Departamento de Sistemas

La década de los noventas llevará al Departamento de Sistemas corporativo a jugar un papel diferente del que ha desempeñado hasta la fecha.

Hoy día esperamos que el Centro de Cómputo procese datos y entregue reportes y en casos muy calificados, que provea herramientas para que el usuario maneje la información; esta última función es producto del empuje de la computación personal. Mañana esperaremos que el Centro de Cómputo -ó mejor expresado: el Centro de Información-, administre grandes volúmenes de información, distribuidos en imágenes digitales, texto, sonidos digitalizados, etc.

El Departamento de Sistemas de la próxima década podrá contribuir al desarrollo de estos nuevos sistemas siguiendo los siguientes consejos:

Dejar pasar. Los multimedios se están imponiendo en algunas áreas, como las que se describieron arriba, haga lo que haga el Departamento de Sistemas. Lo más acertado es dejar que estos aparezcan en donde el aporte sea innegable y donde se pueda manejar, al menos al principio, como proyectos aislados.

Permitir -y ayudar- a que se adopten en entrenamiento, mercadeo, ventas, y servicios, le permitirá al Administrador de Sistemas aprender a manejar el cambio, y a posicionar dentro de la empresa este tipo de sistemas. También le ayudará a familiarizarse con sistemas para el manejo masivo de información no-textual. Ya es posible empezar a experimentar en el país con estos sistemas y vale la pena hacerlo.

Proveer infraestructura. Otra manera en la que el Departamento de Sistemas puede participar del desarrollo de los multimedios en los ambiente profesionales es supliendo gradualmente la infraestructura que se requiere para permitir que las estaciones departamentales funcionen como un todo. El "costo" de la información exige de sistemas de comunicación que permitan compartirla.

Los sistemas Hipermedio[1] se están desarrollando de manera que las unidades de información (los datos) se separan de la

[1] En este caso se habla de hipermedio como subconjunto de los multimedios, para hacer referencia solo a las estructuras interactivas que conectan las células informativas.

estructura con el fin de reutilizar las primeras. El usuario crea ligas entre estas unidades informativas, que dicho sea de paso pertenecen a medios diversos como audio, video y texto, para producir nuevas unidades informativas. Así, un Informe Anual, contendrá texto proveniente de varios autores, segmentos de audio almacenados individualmente, imágenes y secuencias de video, todas existiendo como unidades aisladas dentro del sistema. Este informe anual no será otra cosa que la descripción de como se relacionan en espacio y tiempo esta serie de células informativas.

Como se ve, las redes locales, y su interconexión con otras serán vitales para que este fluido mantenga vivos los sistemas de comunicación del futuro. El Departamento de Sistemas está en la posición idónea para ayudar en la planificación de estos.

Sobra decir que se requerirán sistemas de transmisión con mayor ancho de banda para permitir la movilización expedita de grandes cantidades de información. Los canales y los depósitos se optimizarán para el manejo masivo de información heterogénea. Los mecanismos de compresión y descompresión, que ya son populares, lo serán mucho más en el contexto de estas redes.

En Costa Rica nos encontramos en una posición de privilegio para llenar estos requerimientos, si es que se consigue despertar la voluntad para implementar una red ISDN (red digital de servicios integrados) a nivel nacional antes de año 2000. La centralización de los servicios de comunicación en nuestro país puede representar una ventaja para el establecimiento de sistemas estandarizados como este.

Administrar la información. Como se mencionó arriba, cada día el MIS responderá más por el contenido de los sistemas que por las herramientas. Para manejar con éxito la telaraña de relaciones que los usuarios establecerán entre la información propia y la de otros usuarios hay que establecer criterios sobre ¿Quien es el "dueño" de la información?, ¿En que punto de una transformación deja una unidad de información de pertenecer al autor y pasa a pertenecer al "transformador"?, ¿Como se controla la autoría, validez, vigencia, etc de intangibles?.

Como se ve, existen problemas complejos que se deberán resolver para que los sistemas multimedios se puedan integrarse a

todo nivel dentro de la compañía. Sin embargo, la determinación de estos criterios no debe atrasar la toma de decisiones en otras áreas como el almacenamiento de la información, los sistemas de búsqueda y recuperación, el intercambio de formatos, y en general todas las áreas que tienen que ver con proveer el servicio básico de acercar la información a los usuarios.

El hito que representa el advenimiento de la era de los multimedios forjarán en definitiva la "cara" de la tecnología de la información en los años venideros, por tanto el interés que presentemos a su desarrollo nos asegurará éxito en el cada día más vertiginoso y globalizado mundo de los negocios.

Bibliografía

Sueann Ambron y Kristina Hooper. "Interactive Multimedia"*Microsoft Press. 1988*

Sueann Ambron y Kristina Hooper. "Learning with Interactive Multimedia"*Microsoft Press. 1988*

Neil McManus y Carolyn Said. "Apple fast-forward video for Macs with QuickTime".*MacWeek Volumen 5 Numero 20. 1991*

Philip Robinson, Tim Shetler y Otros. "Multimedia". *Byte. Febrereo 1990.*

Ric Ford. "QuickTime brings Motion To Desktop". *MacWeek Volumen 5 Numero 21. 1991*

John J. Anderson, Andrew Himes y otros. "Multimedia: Interacting with Information". *MacUser. Marzo 1990*

James Burke. "The Connection Machine"*MacUser. Marzo 1990*

Orientación a Objetos:

una alternativa para el desarrollo de sistemas.

por: Róger Mayorga J.

Introducción

Hoy, "Orientación a Objetos" es la palabra mágica. Para que un producto sea de interés y tenga éxito comercial, debe ser "Orientado a Objetos". Esto puede compararse a lo que sucedió en la década pasada, cuando el éxito (o no) de un producto dependía de si la palabra "Relacional" aparecía en su nombre. La orientación a objetos ha evolucionado hasta convertirse en una tecnología ampliamente aceptada y que ofrece considerables beneficios para el desarrollo de sistemas.

Muchas tecnologías aparentan ser (o proclaman ser) orientadas a objetos. Se tienen: Lenguajes de Programación, Sistemas Operativos, Interfaces de Usuarios, Bases de Datos, y también metodologías para el desarrollo de sistemas orientados a objetos. Se ofrecen nuevos productos, los cuales se insiste que son realmente orientados a objetos, y se tienen viejos ambientes que han sido adaptados (o mejorados) para ofrecer la funcionalidad de la orientación a objetos.

La idea básica que soporta la filosofía de orientación a objetos es sumamente simple: los "objetos" son, a un muy alto nivel, cualquier entidad que existe en tiempo y espacio. Los objetos tienen un "estado", y están caracterizados por las acciones que realizan sobre otros objetos y por las acciones realizadas sobre ellos mismos. Los objetos pueden funcionar como "actores", "agentes" o como "servidores", dependiendo de su relación con otros objetos. Los actores operan sobre otros objetos, los servidores únicamente brindan servicios a otros objetos, y los agentes realizan operaciones en nombre de otros objetos.

La historia (conocida) de "orientación a objetos" comienza a finales de los años 60 con el lenguaje de programación Simula. Sin embargo, empieza a recibir mayor atención hasta los años 70 con el lenguaje de programación Smalltalk, así como con la

aplicación de sus conceptos en el desarrollo de grandes proyectos dentro de la comunidad de usuarios de Ada. Esto permitió reconocer las ventajas de aplicar la orientación a objetos al desarrollo de sistemas.

Se han desarrollado muy diversas clases de lenguajes de programación orientados a objetos; más aún, hasta la mitad de los años 80 gran parte del trabajo en esta área estuvo enfocado principalmente al tema de la programación orientada a objetos. Ahora los conceptos se están aplicando en otras disciplinas. Las interfaces de usuario orientadas a objetos son ampliamente utilizadas, el primer Sistema de Base de Datos Orientado a Objetos está ya disponible comercialmente, existen ya Sistemas Operativos que han implantado algunos de los conceptos, y recientemente se está prestando mucha atención a la aplicación de los conceptos al proceso de desarrollo de sistemas.

Dadas las ventajas que ofrece la tecnología de orientación a objetos, con respecto a: reutilización, extendibilidad, mantenimiento, y facilidad para modelar problemas complejos; así como una mayor productividad, se puede decir que la aplicación de esta tecnología constituye una alternativa muy prometedora para enfrentar la muy conocida "crisis del software".

Dentro de unos pocos años, el término "orientado a objetos" será de uso común dentro de la sociedad computadorizada. Y, posiblemente los sistemas orientados a objetos sufrirán del mismo mal que sufren los sistemas y técnicas "relacionales": "Codd continúa discutiendo que es realmente el modelo relacional". Con muchas compañías (en muy diversas áreas) generando productos orientados a objetos, y cada una utilizando su propia interpretación de lo que "orientación a objetos" significa, no se vislumbra un concenso en el corto plazo.

La primera sección, presenta los fundamentos de la orientación a objetos, las siguientes secciones describen las diferentes aplicaciones que han tenido estos conceptos, así como sus ventajas y desventajas.

Definiciones

En esta sección se dará un rápido vistazo a las definiciones más comúnmente utilizadas de los conceptos de orientación a objetos. Para una descripción detallada de estas definiciones se remite al lector a la bibliografía suministrada.

Conceptos:

Se utilizará el término **"sistemas orientados a objetos"** para agrupar a los lenguajes de programación, herramientas y metodologías que soportan a la tecnología de orientación a objetos.

La definición de objeto se toma de Wegner [Wegner, 1987]:

"Un objeto es un conjunto de operaciones y un estado local compartido (datos) el cual recuerda el efecto de las operaciones (sobre el objeto). El valor regresado por una operación realizada sobre un objeto, puede depender tanto del estado del objeto como de los argumentos de la operación. El estado de un objeto sirve como una memoria local al objeto, la cual es compartida por las operaciones que actúan sobre ella. En particular, otras operaciones realizadas previamente, pueden afectar el valor regresado por una operación dada. Un objeto puede aprender de la experiencia, almacenando el efecto acumulativo de su experiencia -- historia de operaciones -- en su estado."

Esta definición dice que un objeto es más que una estructura de datos con algunas operaciones que actúan sobre su estado. Un objeto es una entidad que muestra algún comportamiento reflejado por las operaciones que actúan sobre su estado. En otras palabras, los objetos son todas las cosas físicas y conceptuales que podemos encontrar en el mundo que nos rodea. Un objeto podría ser "hardware", "software", o un concepto (e.g. velocidad). Los objetos son entidades completas; e.g., no son "simplemente información", o "simplemente información y acciones". Finalmente, los objetos son "cajas negras"; i.e., su implantación interna son ocultados al exterior, y toda interacción con el objeto tiene lugar vía una muy bien definida interfaz.

Se define un sistema orientado a objetos, como uno que soporta **todas** las siguientes propiedades:

1- Los datos y los procedimientos están combinados en "objetos de software". Esto es lo que comúnmente se conoce como **"encapsulación"**. Encapsulación es el nombre técnico para la técnica conocida como ocultamiento de información. Significa además que el estado de los objetos puede ser manipulado únicamente por las operaciones asociadas al objeto. De esta forma la implantación de un objeto (datos y operaciones) no puede ser vista por las aplicaciones que los utilizan.

2- Se utilizan **mensajes** para la comunicación entre los objetos. Esto se conoce como "paso de mensajes". Un mensaje sirve como el medio para solicitar que alguna operación sea realizada sobre un objeto. Constituye una forma indirecta de invocar un procedimiento. Junto con la encapsulación, el envío de mensajes soporta

un principio muy importante en la programación: la abstracción de datos. Este principio establece que los programas no deben hacer suposiciones sobre la implantación y representación interna de los datos.

3- Objetos similares se agrupan en **clases**. Una clase es la descripción de uno o más objetos similares. La clase es donde se almacenan los procedimientos y atributos comunes a todos los objetos del mismo tipo. A cada objeto individual se le conoce como una **instancia** de la clase.

4- **Herencia**. Este concepto establece que los datos y operaciones definidas para un tipo estarán automáticamente definidas para sus subtipos. Esto permite definir nuevos objetos tomando como base objetos que ya existen, los nuevos objetos podrán diferir de los objetos base agregando más (o limitando) características.

Cualquier sistema (lenguaje, herramienta, metodología) se dirá que es **orientado a objetos** si soporta los cuatro conceptos anteriores. Si únicamente soporta los dos primeros, se dirá que es "**basado en objetos**". Si soporta los tres primeros se dice que es "**basado en clases**".

Otra idea básica de la orientación a objetos lo constituye el "**polimorfismo**". Los objetos del mundo real responden de diferente forma al mismo mensaje. Los mensajes enviados a los objetos podrían tener esta propiedad. Un ejemplo de esto lo constituye la operación suma. La misma notación, "+", puede ser utilizada para sumar dos enteros, dos números en punto flotante, dos números complejos, etc. En estos casos, la operación actúa apropiadamente de acuerdo con el tipo de argumentos.

Podríamos decir que el principio general detrás de la filosofía de orientación a objetos es [Verharen, 1991]:

"La tecnología de orientación a objetos es una forma de desarrollar y "empacar" software basada fuertemente en la experiencia común y en la forma en la que los objetos del mundo real se relacionan entre si",

o desde un punto de vista más filosófico:

"El paradigma orientado a objetos significa que un sistema a cualquier nivel es considerado como un conjunto de objetos autónomos que tienen suficiente conocimiento para solicitar u ofrecer ciertos servicios a otros objetos de su ambiente, sin interferir en su organización interna".

A diferencia de los datos, los objetos pueden actuar. Un objeto puede decidir que hacer cuando se recibe un mensaje. La idea de utilizar objetos es el producto de muchos años de experiencia tratando de encontrar un mecanismo (natural) de alto nivel para

estructurar los problemas. Aquí, el concepto clave es que una colección de datos y las operaciones que actúan sobre estos datos están muy ligados y deben ser tratados como una sola entidad en vez de cosas separadas.

Tal como se indicó anteriormente, la orientación a objetos no es algo nuevo; estos conceptos tienen ya unos veinte años de estar siendo digeridos. Sin embargo, en este momento algunos factores están orientando hacia una mayor aceptación comercial de esta tecnología. Uno de los factores lo constituye el "hardware". Ahora es posible contar con equipos más poderosos y a un costo menor que en el pasado; los profesionales pueden contar con poderosas estaciones de trabajo a un precio razonable. Este tipo de equipo es necesario para hacer accesible a los usuarios finales todo el poder de los ambientes (por lo general, basados en gráficos) que ofrece esta tecnología.

Sin embargo, no todas las manifestaciones de la orientación a objetos requieren de poderosas estaciones de trabajo con capacidades gráficas, rápidos procesadores y memoria en abundancia. La tecnología de orientación a objetos puede aplicarse aún en aplicaciones más convencionales. Por ejemplo, IBM ya ha implantado un "Sistema Operativo Orientado a Objetos" para sus equipos de la línea AS/400.

Posiblemente, la causa principal para la adopción de esta tecnología lo constituye, la (siempre en aumento) complejidad de los sistemas requeridos. Bajo una enorme presión para desarrollar sistemas cada vez más complejos, y al mismo tiempo reducir el tiempo y el esfuerzo requeridos para lograrlo, la industria de "software" esta adoptando esta tecnología, la cual enfrenta tanto los problemas de complejidad de los sistemas como de productividad. El software debe ser *modular, robusto, extendible, transportable, reutilizable*, y por supuesto, fácil de utilizar.

La siguiente sección presenta una descripción de las principales áreas de aplicación de los conceptos de la orientación a objetos.

Areas de aplicación

Los conceptos de orientación a objetos se han aplicado en muy diversas áreas, para examinar sistemáticamente el estado actual de su aplicación, se consideran cinco grandes áreas (o grupos): Metodologías para el Desarrollo de Sistemas, Lenguajes de Programación, Sistemas Operativos, Bases de Datos, y Ambientes de Usuarios y Aplicaciones.

Metodologías para el desarrollo de sistemas

El impacto de la tecnología de orientación a objetos no está limitado únicamente a la fase de programación o implantación del ciclo de vida de un sistema -- su aplicación puede extenderse a todas las fases (análisis, diseño, implantación). Quizá, uno de los factores más importantes es que proporciona un elemento unificador que es común a todas las fases del ciclo de desarrollo: **el objeto**.

La existencia de lenguajes de programación orientados a objetos ha generado una demanda por metodologías y herramientas para apoyar el desarrollo de sistemas orientado a objetos. Los problemas que presentan las metodologías tradicionales para el desarrollo de sistemas (e.g., la metodología "Catarata" [Verharen, 1991]) están documentados en casi todo libro de ingeniería de software. Entre los principales problemas que presenta el ciclo de vida tradicional tenemos: la no iteración, no hay énfasis en la reutilización, así como la no existencia de un modelo unificador para la integración de las diferentes fases. La diferencia (en puntos de vista) entre seguir flujos de datos en el análisis estructurado y construir jerarquías de tareas en el diseño estructurado siempre han constituido un serio problema. Cada sistema es construido a partir de cero, y los costos de mantenimiento representan un muy alto porcentaje del costo total de un sistema.

La tecnología de orientación a objetos pretende enfrentar cada uno de estos problemas. Un rápido vistazo al "ciclo de vida orientado a objetos" de un sistema, tal como lo presentan diferentes autores ([Booch, 1991], [Brooks,1987], etc.), nos muestra las tres fases tradicionales: análisis, diseño, e implantación. Sin embargo, se ha eliminado (en cierto grado) las rígidas (y muy diferentes) fronteras entre las fases. La principal razón para esto, es que los elementos de interés para cada fase son los mismos: los objetos. Los analistas, diseñadores y programadores trabajan con un mismo conjunto de elementos a partir de los cuales construyen el sistema.

Las metodologías para apoyar las fases de análisis, diseño e implantación de sistemas orientados a objetos se encuentran en un estado primario, existiendo en este momento diferentes modelos. Un modelo que esta teniendo mucha aceptación es el "Modelo Paralelo/Recursivo" propuesto por Grady Booch [Booch, 1991]. En síntesis este modelo se puede describir como la secuencia: analice un poco, diseñe un poco, implante un poco, pruebe un poco. En vez de hacer "todo el análisis", seguido de "todo

el diseño", como en la metodología de "catarata", el Modelo Paralelo/Recursivo sugiere que uno hace análisis cuando es apropiado, diseño cuando es apropiado, etc.

Durante los últimos 5 años han surgido varias metodologías para apoyar el desarrollo de sistemas orientados a objetos, entre ellas: GOOD ("General Object-Oriented Development"), MOOD ("Multiple-view Object-Oriented Design"), HOOD ("Hierarchical Object-Oriented Design"), LVM/OOD ("Layered Virtual Machine/Object-Oriented Design"), ObjectOry, y OOSD ("Object-Oriented Structured Design"), para más información ver [Verharen, 1991].

Algo interesante de algunas de estas metodologías es que el ciclo de vida de un sistema es visto como análisis, diseño, y evolución; donde evolución lleva implícitos todos los posibles cambios (ajustes) requeridos por el sistema durante su existencia. **El cambio es considerado como una parte normal en el ciclo de vida del sistema.**

Lenguajes de programación

Esta es quizá el área donde más y por más tiempo se han aplicado los conceptos de la orientación a objetos. De hecho, muchas personas relacionan la orientación a objetos exclusivamente con la programación. Existen muchos criterios diferentes sobre que es programación orientada a objetos y por lo tanto sobre lo que es un lenguaje o ambiente de programación orientado a objetos.

En síntesis, la "programación orientada a objetos" es una técnica (un paradigma) para escribir "buenos programas" para resolver un conjunto de problemas. Mejorando la forma en que el "software" es organizado, y creando "software" que es reutilizable y flexible, se puede reducir considerablemente no únicamente el costo del desarrollo de aplicaciones, sino que también el costo de su mantenimiento.

La mayoría de los lenguajes de programación soportan el paradigma procedimental (datos-procedimientos): "los procedimientos son entes activos que actúan sobre los datos (entes pasivos) que les son enviados como argumentos". En el desarrollo tradicional de "software" existen procedimientos y estructuras de datos, los cuales son tratados como entidades independientes. Un sistema orientado a objetos tiene un solo tipo de entidad, el objeto, que representa a ambos.

Un objeto es una especie de "caja negra" (similar a un circuito integrado), el cual

puede potencialmente ser reutilizado en varias aplicaciones. Un programador podría seleccionar objetos de diversas fuentes y acoplarlos de muy diversas maneras para crear nuevas aplicaciones.

Existen varios lenguajes de programación que han contribuido considerablemente a la evolución de los conceptos de la orientación a objetos. La siguiente figura [Booch, 1991] muestra el árbol genealógico de los lenguajes de programación orientados a objetos, el ancestro común de todos ellos es el lenguaje Simula, desarrollado en los años 60.

Aquí va la figura del árbol genalógico de los lenguajes de programación orientados a objetos

En esta figura se muestran los cinco lenguajes que han tenido más influencia y son más utilizados para la programación orientada a objetos: Smalltalk, Object-Pascal, C++, CLOS, y Ada (se dejan por fuera otros tales como el Objective-C [Cox, 1986]). Nótese que de acuerdo con lo expresado anteriormente, se hace la distinción entre lenguajes orientados a objetos y lenguajes basados en objetos. Por razones prácticas, únicamente se darán

detalles sobre los lenguajes Smalltalk y C++.

Quizá el mejor ejemplo de un ambiente completamente integrado y orientado a objetos lo constituye el del lenguaje de programación SmallTalk. Smalltalk fue creado en los años 70 por los miembros del Xerox Palo Alto Research Center Learning Research Group como el elemento de software del Dynabook, un visionario proyecto de Alan Kay.

Smalltalk representa tanto un lenguaje de programación como un ambiente de desarrollo. Ofrece un lenguaje de programación orientado a objetos "puro", en el sentido de que todo es visto como un objeto, tanto números enteros como las mismas clases. Esto tiende a una total consistencia en el lenguaje, tanto al nivel práctico como conceptual. Ofrece una interfaz al usuario consistente con la filosofía de orientación a objetos; hizo popular el ahora común ambiente de múltiples ventanas, "icons", integración de texto y gráficos, menús "pull-down" y "mouse", haciéndolo atractivo a una gran cantidad de desarrolladores e investigadores. Es justo mencionar que mucho antes de que la compañía Apple ofreciera este tipo de interfaz, ya Smalltalk lo utilizaba.

La última versión Smalltalk-80 ha sido implantada en múltiples plataformas, y ya se encuentra disponible comercialmente, especialmente para microcomputadores y estaciones de trabajo.

El lenguaje de programación C++, ha recibido especial atención en los últimos años. Diseñado por Bjarne Stroustrup para los Laboratorios Bell (1980-1983, ...), combina los conceptos de orientación a objetos con el (tradicional) lenguaje de programación C. C++ constituye un superconjunto del lenguaje C. Como el mismo Stroustrup dice [Stroustrup, 1986a, 1988, 1989], C++ corrige muchas de las deficiencias del lenguaje C, además de agregar al lenguaje el soporte para las clases, herencia múltiple, chequeo de tipos, sobrecarga de operadores (polimorfismo), administración de memoria, tipos constantes, referencias, funciones "inline", funciones virtuales, etc.; lo cual hace que este lenguaje soporte el paradigma de orientación a objetos.

Las últimas versiones se encuentran disponibles en muy diversas plataformas, incluyendo UNIX y MS/DOS.

En la actualidad, hay un debate sobre cual es el lenguaje de programación orientado a objetos que será el estándar para el

desarrollo de aplicaciones a nivel comercial en el futuro. Aparentemente, la alternativa es entre lenguajes orientados a objetos "puros", tales como Smalltalk y lenguajes híbridos, tales como C++.

Posiblemente los programadores de diferentes bandos (Smalltalk, C++) podrían pasar años discutiendo cual debe ser el lenguaje de programación orientado a objetos más apropiado. Sin embargo para los administradores del desarrollo de software, el problema se presenta en forma diferente. Posiblemente, lo más relevante para la adopción de esta tecnología reside en la capacidad de los programadores, y posiblemente existen más programadores con experiencia en C que en Smalltalk.

Es importante mencionar que la programación orientada a objetos no es (únicamente) programación utilizando Smalltalk o C++ (u otro lenguaje orientado a objetos), sino que es parte de toda una filosofía para el desarrollo de sistemas, la cual debe ser comprendida y utilizada coherentemente para poder aprovechar todas las ventajas que ofrece.

Sistema Operativo Orientado a Objetos

Un sistema operativo orientado a objetos por definición trabajará sobre objetos en vez de tipos específicos de archivos. Esta abstracción -- tratar con objetos generales en vez de tipos específicos de archivos -- proporciona una gran flexibilidad al sistema. Algunos ejemplos lo constituyen los sistemas Biin-OS, el IBM OS/400, y el NeXT.

Biin es una compañía formada por Siemmens Information Systems Inc. e Intel Corporation. Esta compañía esta ofreciendo un sistema operativo orientado a objetos (similar a UNIX) para equipos de múltiples procesadores RISC.

En el sistema operativo IBM OS/400, todo lo que puede ser almacenado o recuperado (leído) esta contenido en un objeto, esto libera al usuario de conocer los detalles de las técnicas de implantación, o estructuras de direccionamiento específicas a la máquina. Por ejemplo, cuando el OS/400 almacena una imagen, todo lo que sabe es que está almacenando un objeto. El sistema operativo no conoce que contiene el objeto. De la misma forma el mismo objeto no tiene que hacer consideraciones especiales cuando ocurran cambios en la máquina o sistema operativo.

Los conceptos de orientación a objetos han sido aplicados en el diseño del "NextStep", ambiente orientado a objetos para los computadores NeXT. De igual forma estos conceptos se han aplicado (en algún grado)

en el diseño de los sistemas OS/2 y Windows.

Una de las características claves de un sistema operativo orientado a objetos es el ofrecer una interfaz consistente al usuario, la cual permita la operación y uso de los recursos del equipo a través de objetos, manteniendo al mismo tiempo la independencia de la máquina. De esta forma es más fácil soportar los (inevitables) cambios en el "hardware".

Bases de Datos

La necesidad de manejar datos poco estructurados (y de tamaño variable), muy diferentes a los de las tradicionales aplicaciones administrativas, ha generado la necesidad por contar con sistemas de bases de datos que puedan manejar este tipo de datos. Aplicaciones en las áreas de geografía, climatología, automatización de oficinas, medicina, CAD/CAM, etc., requieren manejar altos volúmenes de datos no estructurados, tales como texto, gráficos, imágenes, voz, etc. Los manejadores de bases de datos relacionales no pueden satisfacer a cabalidad los requerimientos que imponen este tipo de aplicaciones.

Dadas las características de estas aplicaciones, y la generalidad y el poder que ofrece la tecnología de orientación a objetos, se han aplicado sus conceptos a la tecnología de las bases de datos para producir lo que se conoce como los "sistemas de bases de datos orientadas a objetos". El desarrollo de estos sistemas de bases de datos ha recibido considerable atención en los últimos años.

Un "sistema de base de datos orientado a objetos" debe soportar los conceptos de la orientación a objetos; i.e., debe soportar los conceptos de Objeto, Tipo (Clase), Encapsulación y Herencia. De igual forma que los lenguajes de programación orientados a objetos, las bases de datos orientadas a objetos facilitan el manejo de la complejidad en los sistemas aprovechándose de los conceptos de clase y herencia.

Nuevos vendedores, tales como Servio Logic y Ontologic, están comenzado a ofrecer sistemas de bases de datos orientadas a objetos; al mismo tiempo, vendedores ya establecidos, tales como ORACLE Corp. e Informix Inc., están trabando fuertemente en la incorporación de los conceptos de la orientación a objetos a sus productos.

Algunos de los productos ofrecidos están orientados hacia las aplicaciones de tipo científico o de ingeniería (CAD/CAM), grupos que han recibido muy poco apoyo por parte de los vendedores de los sistemas tradicionales de bases de datos relacionales.

Otros, están aplicando la tecnología para producir "front-ends" para los sistemas de bases de datos tradicionales. Esto puede verse como una forma de manejar la migración (implantación) hacia esta tecnología.

Las bases de datos orientadas a objetos parecieran ser la siguiente etapa en la evolución de los sistemas de bases de datos, y esto ha surgido más que nada como respuesta al incremento en la complejidad de las aplicaciones requeridas por los usuarios.

Ambientes de usuarios y aplicaciones

Esta es un área sumamente densa y plagada de muy diversos criterios. En esencia, el objetivo de aplicar los conceptos de la orientación a objetos para la interfaz con los usuarios finales es el proporcionar un ambiente que soporte la manipulación directa por parte del usuario. Por ejemplo, para imprimir un documento un usuario "pondría" (utilizando el "mouse") el "icon" del documento sobre el "icon" de una impresora; en vez de invocar al procesador de palabras, cargar el documento deseado, y decirle a la aplicación que se imprima el documento. La manipulación directa le permite al usuario concentrarse en la tarea a realizar, en vez de los detalles y la mecánica de la aplicación

Existen buenos ejemplos de este tipo de interfaz. El Macintosh de Apple, por ejemplo, representa archivos discretos como "icons", pero se queda corto en ofrecer una interfaz que soporte una manipulación directa y completa.

El paquete NewWave de Hewlett Packard, proporciona un gran nivel de funcionalidad, ofreciendo manipulación y administración de objetos sobre algunos de los sistemas operativos actuales; ya está disponible para MS/DOS. Se espera que pronto estará disponible para UNIX y OS/2.

Ventajas de la orientación a objetos

En esta sección revisaremos algunas de las ventajas de la aplicación de los conceptos de la orientación a objetos. La orientación a objetos ha sido descrita como la última alternativa para enfrentar la "crisis del software". Sin embargo, no se debe olvidar que tales afirmaciones han sido hechas antes para otras tecnologías.

El propósito del paradigma orientado a objetos es proporcionar una forma directa y natural de describir los conceptos del mundo real, ofreciendo (permitiendo) la flexibilidad de expresión necesaria para capturar la naturaleza variable del mundo siendo modelado y la habilidad para representar las

situaciones cambiantes. Una parte fundamental de la naturalidad de expresión proporcionada por el paradigma orientado a objetos es la habilidad para compartir datos, código y definiciones.

Se ha dicho que tecnología orientada a objetos es atractiva debido a que su uso aumenta la productividad a través de todo el ciclo de vida del desarrollo de un sistema. Este incremento en productividad es alcanzado a través de las cuatro principales características de los sistemas orientados a objetos y los beneficios que ofrecen estas características:

- La utilización de objetos como los módulos básicos ayuda al diseñador a modelar más fácilmente los sistemas complejos del mundo real. (Los sistemas orientados a objetos permiten enfrentar la complejidad de los sistemas)

- La flexibilidad del código (orientado a objetos) generado permite una rápida respuesta a los cambios en los requerimientos de los usuarios. (Los sistemas orientados a objetos están diseñados para el cambio)

- La reutilización de componentes estándar reduce tanto el tiempo de desarrollo requerido para una nueva aplicación, como la cantidad de código generado. (Los objetos son reutilizables)

- La mayor facilidad para el mantenimiento de los sistemas, los hace más confiables y reduce considerablemente los costos de mantenimiento. (Los sistemas orientados a objetos son mantenibles)

El aumento en productividad surge en forma natural debido a estos beneficios.

La filosofía de orientación a objetos para el diseño de sistemas reduce el "gap" conceptual entre el mundo real y el modelo ha utilizar para el computador. Ayuda a los analistas y diseñadores a pensar claramente sobre la estructura de un sistema. La flexibilidad de los sistemas orientados a objetos es una ventaja particularmente para los diseñadores en campos muy dinámicos tales como CAD/CAM. Uno de los mayores beneficios es que ayuda a los desarrolladores a responder rápidamente a las nuevas tendencias de un mercado cada vez más competitivo. La reutilización de código reduce el tiempo de desarrollo y le permite a los diseñadores enfrentar las áreas difíciles con mayor seguridad.

En el pasado, grandes bibliotecas de subrutinas han estado disponibles a los desarrolladores de sistemas para enfrentar los problemas comunes, tales como los

cálculos matemáticos. La orientación a objetos provee la posibilidad de reutilización a una escala mucho mayor. Según estudios, el mantenimiento representa cerca del 80% del costo total para el ciclo de vida de un sistema. Los desarrolladores de sistemas grandes y complejos, que requieren de frecuentes modificaciones (ajustes), están utilizando la tecnología de orientación a objetos como un medio para reducir los costos de mantenimiento y mejorar la confiabilidad de sus productos.

Estos beneficios surgen de la forma como el "software" orientado a objetos está empacado. El "software" tradicional está hecho de datos y procedimientos que accesan y modifican los datos. Los datos y procedimientos son empacados en forma separada; un cambio en una estructura de datos afectará a muchos módulos diferentes, (y por lo general) escritos por muchos programadores diferentes. En un sistema orientado a objetos, los datos y procedimientos que actúan sobre esos datos son considerados en conjunto, como parte de un paquete -- un objeto. Si se requieren cambios en un dato, todos los procedimientos afectados son fácilmente identificados y cambiados al mismo tiempo. Debido a que el cambio está limitado a una área específica del sistema, su impacto se reduce.

Las ventajas que ofrece la aplicación de los conceptos de la orientación a objetos han sido corroboradas por el éxito alcanzado en mútliples proyectos de desarrollo de software a nivel comercial. Un ejemplo lo constituye el caso de la compañía Borland y su paquete Quattro, el cual ha sido desarrollado utilizando una metodología (y herramientas) orientadas a objetos [PC Week, Julio 29, 1991]. Esto le ha permitido a esta compañía generar nuevas versiones en muy corto tiempo y a un costo mucho menor que con una metodología (y herramientas) convencionales. La siguiente sección presenta algunos de los problemas que representa la aplicación de estos conceptos.

Desventajas de la orientación a objetos

Aún cuando en la sección anterior, y a lo largo del documento, se han mencionado importantes ventajas que ofrece la tecnología de orientación a objetos, las cuales podrían orientar a muchos hacia su aplicación. Con la intención de ofrecer una perspectiva más real y objetiva es conveniente considerar las limitaciones y problemas que presentaría hoy día su aplicación.

Quizá la mayoría de los problemas que enfrenta esta tecnología son debido a su

inmadurez. La mayor limitación que enfrentan los sistemas orientados a objetos en la actualidad es la resistencia al cambio por parte de los desarrolladores de sistemas (administradores y personal técnico), lo cual es natural debido a la inmadurez de muchos de los productos orientados a objetos disponibles en el mercado. La inmadurez es evidente en ciertos problemas que (en realidad) son comunes a cualquier nueva tecnología (y que están siendo superados):

- Disponibilidad limitada en las plataformas de desarrollo estándar. ("main- frames", sistemas propietarios, etc.)

- Poca integración con los sistemas actuales. (lenguajes de programación tradicionales, sistemas de bases de datos relacionales, etc.)

- Falta de soporte para el desarrollo de sistemas a gran escala. (herramientas para apoyar el desarrollo de sistemas)

Se requiere entrenamiento a todos los niveles (administradores, programadores), desarrollo de metodologías apropiadas, así como tecnología para la migración. Para que esto suceda, los problemas antes mencionados deben ser superados.

Sin embargo, superar estos problemas será una labor costosa que requiere de bastante tiempo. Requerimientos establecidos por los usuarios, tales como transportabilidad, sistemas abiertos y estándares están siendo considerados por la mayoría de los proveedores de sistemas orientados a objetos. Para que puedan ser aceptados por los usuarios, los principales ambientes y lenguajes de programación orientados a objetos deberán estar disponibles en las plataformas ("hardware/software") principales. Esto es cierto para lenguajes tales como el C++ y Smalltalk, pero no existe un estándar, lo cual impone problemas para los desarrolladores que deben proporcionar compatibilidad y transportabilidad a través de diferentes sistemas. Por ejemplo, un producto de software que esta disponible en la mayoría de las plataformas de "hardware" deberá funcionar bajo diferentes interfaces de usuario.

Además de los problemas de inmadurez descritos anteriormente, existen dos problemas técnicos que deben ser resueltos antes de que la tecnología de orientación a objetos pueda ser ampliamente adoptada. La primera, y posiblemente más importante, limitación de la presente generación de los sistemas orientados a objetos, es que los objetos no pueden ser fácilmente compartidos por varios usuarios, y que podrían perderse cuando el programa que los

creó termina. Se dice que tales objetos son "persistentes", y por lo general se almacenarán en una base de datos orientada a objetos.

La segunda se refiere al hecho de que las implantaciones de los lenguajes de programación orientados a objetos varían considerablemente en las demandas que imponen sobre los sistemas de cómputo, lo que implica (por lo general) un aumento en el "overhead", conduciendo a problemas de rendimiento. La solución a estos problemas constituyen tópicos de investigación en este momento.

Conclusión

El éxito de la tecnología de orientación a objetos estará marcado por su aceptación a gran escala en la industria de desarrollo de sistemas (comercial, científico, etc.). No es aventurado decir que los conceptos de orientación a objetos son fundamentales para los desarrolladores de sistemas del futuro, pero que son relativamente nuevos a la mayoría de ellos y podría tomar algún tiempo su asimilación.

Dentro de cada una de las áreas de aplicación, existen algunos tópicos de interés a todos los profesionales involucrados en el desarrollo de sistemas:

- **Implantación y migración**: para algunas áreas de aplicación, la implantación de una base de datos orientada a objetos no constituye una labor poco factible, de hecho la comunidad científica y la de las aplicaciones en áreas de la ingeniería (e.g., CAD/CAM) no han podido ser satisfechas por los manejadores de bases de datos relacionales, teniendo que recurrir a soluciones específicas. Por otra parte, tomar una aplicación compleja desarrollada para una base de datos relacional (e.g., IDMS/R, o DB2) y tratar de migrar los datos y la aplicación siguiendo la tecnología de orientación a objetos podría convertirse en una misión imposible.

- **Puro vrs. híbrido**: aún cuando la orientación a objetos no tiene fronteras claramente definidas, siempre surgen disputas sobre la implantación más apropiada y funcionalmente pura de los conceptos, y esto para cualquiera de las áreas: bases de datos, lenguajes de programación, aplicaciones, etc.

- **Compatibilidad y co-existencia**: debido a que no es posible que todo el mundo adopte esta tecnología de un día para otro, los responsables del desarrollo de sistemas deberán (por largo tiempo) soportar la carga de ofrecer alguna forma de compatibilidad y co-existencia entre las nuevas aplicaciones y

las antiguas. Esto podría ser particularmente difícil en el área de los ambientes de usuarios.

- **Interfaz consistente**: se debe llegar a un estándard sobre el tipo de interfaz a ofrecer a los usuarios, y esto debe aplicarse a todas las plataformas.

Como con todo nuevo sistema, los sistemas orientados a objetos deberán "calzar" dentro del actual universo de sistemas procedimentales tradicionales. Deberá ser posible accesar programas escritos en diferentes lenguajes. Existen muchas aplicaciones escritas en C, Pascal, Fortran, o Cobol, las cuales no es posible (factible) reescribir. Las nuevas aplicaciones escritas en los lenguajes orientados a objetos deberán ser capaces de comunicarse con esas aplicaciones.

Mientras la mayoría de los ambientes y lenguajes disponibles en este momento son excelentes para proyectos individuales, la presente generación de sistemas orientados a objetos disponibles en el mercado proporciona poco soporte para el desarrollo en gran escala: múltiples desarrolladores, proyectos convencionales. Por otra parte, las herramientas CASE disponibles no soportan los conceptos de orientación a objetos, ni las metodologías y los lenguajes de programación orientados a objetos. Hasta ahora, están surgiendo comercialmente las primeras herramientas para apoyar el análisis y diseño de sistemas orientado a objetos.

Bibliografía

[ACM]. "Object-Oriented Design", (Varios artículos), *Communications of the ACM, Setiembre 1990, Vol. 33, No. 9.*

[Booch, 1986]. Booch, Grady, "Object-Oriented Development", *IEEE Transactions on Software Engineering, Vol. SE-12, No. 2, Febrero 1986.*

[Booch, 1991]. Booch, Grady, "Object-Oriented Design", *The Benjamin/Cummings Publishing Company, Inc., Cal. 1991.*

[Brooks, 1987]. Brooks, F.P. Jr., "No Silver Bullet: Essence and Accidents of Software Engineering", *IEEE Computer, Vol. 20, No. 4, Abril 1987.*

[BYTE]. "Object-Oriented Programming", (Varios artículos). *BYTE, August 1986, March 1989, October 1990.*

[Cox, 1986]. Cox, Brad J., "Object-Oriented Programming, An Evolutionary Approach", *Addison-Wesley, 1986*

[Goldberg, 1983]. Goldberg, A., Robson, D., "Smalltalk-80: The Language and its Implementation", *Addison-Wesley, Reading, Mass., 1983.*

[OOPSLA]. "Object-Oriented Programming Systems, Languages and Applications", OOPSLA, Conference Proceedings (1986, 1987, 1988, 1989, 1990), *ACM Press, 1986, 1987, 1988, 1989, 1990.*

[Stroustrup, 1986a]. Stroustrup, B., "An Overview of C++", *SIGPLAN Notices, October 1986.*

[Stroustrup, 1986b]. Stroustrup, B., "The C++ Programming Language", *Addison-Wesley, Reading, Mass., 1986.*

[Stroustrup, 1987]. Stroustrup, B., "The Evolution od C++:1985 to 1989", *Proceedings USENIX C++ Conference, November 1989.*

[Stroustrup, 1988]. Stroustrup, B., "What is Object-Oriented Programming", *IEEE Software, Mayo de 1988.*

[Verharen, 1991]. Verharen, Ergon M., "Object-Oriented System Development; An Overview". *Infolab, Tiburg University, The Netherlands, 1991.*

[Wegner, 1987]. Wegner, P., "The Object-Oriented Classification Paradigm", in *Research Directions in Object-Oriented Programming, MIT Press, 1987.*

AD/Cycle

por: Roberto Saso Rojas

Introducción

En setiembre 1989, IBM anunció AD/Cycle (ciclo de desarrollo de aplicaciones) como una extensión del SAA (arquitectura de sistemas de aplicaciones). AD/Cycle es un enfoque, un marco de referencia y un conjunto de productos (tanto de IBM como de otros proveedores) orientados al desarrollo de aplicaciones.

AD/Cycle está diseñado para soportar el desarrollo de software de aplicaciones y ayudar a reducir el inventario de aplicaciones pendientes, a travez del uso de técnicas de automatización avanzadas. Esta área tan importante ha sido recientemente el objeto de sistemas y herramientas conocidas genéri-camente como Ingeniería de Software Asistida por el Computador (CASE).

AD/Cycle es vital para la comunidad interesada en los productos CASE puesto que define un marco de referencia común, a través de todo el ciclo de vida de desarrollo de sistemas, incluyendo la administración del ciclo mismo.

En una encuesta realizada entre 1000 empresas en Estados Unidos durante el mes de agosto 1987 (publicada por Software News' Sentry Research Division) indicó que existía un inventario de apliaciones pendientes en mainframes de 4 años - con una demanda insatisfecha de aplicaciones en minis y micros un poco menor. Además del gran núnero de necesidades insatisfechas de sistemas se debe considerar el alto costo del mantenimiento de las apliaciones actuales y la escasez de programadores experimentados.

Hoy en día existe consenso acerca de la necesidad de automatizar el proceso de desarrollo de sistemas (de ahí el auge que han tenido las herramientas CASE). No existe, sin embargo, concenso alguno respecto al modelo de información a utilizar en la aplicación de herramientas de alta producti-vidad en el desarrollo de sistemas

(como generadores de programas, lenguajes de cuarta generación e implementaciones gráficas de análisis y diseño).

AD/Cycle es un conjunto de planes, interfases y estándares orientados a permitir la integración de herramientas individuales en un ambiente completo de desarrollo de aplicaciones. Un ambiente que provea definiciones de almacenamiento, manipulación y reutilización de toda la información relativa al proceso de desarrollo de sistemas.

Con el AD/Cycle, IBM ha reconocido los esfuerzos pioneros de los proveedores de herramientas CASE, y ha decidido no detener el progreso logrado ofreciendo alternativas propietarias e incompatibles. En esta ocasión, IBM decidió buscar abiertamente el input, la cooperación y el apoyo de casi todos los participantes de la industria -proveedores, consultores, grupos de usuarios y clientes estratégicos.

A continuación se presenta una breve descripción de la estrategia y arquitectura detrás del AD/Cycle, seguido de comentarios y recomendaciones, orientados a la aplicabili-dad del AD/Cycle en los departamentos de sistemas en nuestro medio.

Estrategia

El enfoque seguido durante el final de la década de los 70 y durante los años 80 consistió en el desarrollo de herramientas automatizadas para el desarrollo de sistemas (herramientas CASE). Estas herramientas han sido todas orientadas a automatizar actividades específicas dentro de cada fase del ciclo de vida del desarrollo de aplicaciones (CVDA).

El CVDA tradicional agrupa las actividades en cinco fases: Requerimientos, Análisis y Diseño, Desarrollo, Construcción y Pruebas, y Producción y Mantenimiento. Además existen otras actividades que atraviesan o son comunes a varias fases del ciclo (ver figura 1).

Figura 1 Application development life cycle

Cross Life-Cycle Activities

Process Management	Project Management	Documentation	Perfomance Monitoring	Editing	Query/ Reporting	Prototyping	Reuse	Impact Analysis

Life-Cycle Tool Activities

Requirements	Analysis and Design	Produce	Build and Test	Production and Maintenance
Business Enterprise Modeling	Design Application Structure	Activities: Create Logic	Build Procedures	Package System
Define Processing Requirements	Define Logic	Produce Help Messages, and Screens	Build Programs	Release and Control
Define I/O Requirements	Define Help and Messages	Produce File I/O	Define Test Data	Maintain/Change Existing Code
Define External Data	Design Screens	Produce Report I/O	Create Test Cases	Code Information Extract
	Design Reports	Produce Database I/O	Test and Debug	Restructure
	Design Files	Methods:	System Test	
	Design Internal Data	Compiler Based	Validate Results	
	Design Database	Code Generation	Test Management	
	Define Environment	Knowledge Based	Test Coverage	

Fuente: IBM Systems Journal Vol. 29 No. 2 Página 171

El enfoque de herramientas individuales para actividades específicas ha resultado en una proliferación de herramientas y metodologías no integradas que han producido únicamente mejoras incrementales, niguna de las cuales ha sido suficiente para reducir efectivamente el inventario de necesidades insatisfechas. Algunas razones para esto son:

- Las metodologías y las herramientas para soportar las actividades del ciclo de vida por lo generral no comparten datos.

- La falta de consenso en estándares ha hecho difícil la integración de las herramientas.

- En los casos en que las herramientas han sido integradas en sistemas de desarrollo de aplicaciones, los sistemas han sido

típicamente construidos en arquitecturas cerradas con interfases de datos privadas, obstaculizando así el agregar herramietas o mejorar las existentes para incorporar nuevas metodologías y/o tecnologías.

La administración del CVDA usualmente requiere que cada fase esté casi terminada antes de que la próxima fase pueda iniciarse. Requerimientos y especificaciones son pasadas en papel de planificadores a diseñadores y de diseñadores a codificadores, a menudo requiriendo intervención manual entre fases y actividades. Esta proliferación de herramientas, metodologías y transformaciones manuales de datos ha resultado en un proceso consistente de una serie de etapas discretas con fronteras en cada etapa caracterizadas por una tranferencia manual o en papel de una etapa a la próxima (ver figura 2). Este proceso es muy ineficiente y require una utilización intensiva de gente y papel.

AD/Cycle fue desarrollado buscando mejoras significativas en la productividad, calidad y manejabilidad del desarrollo de aplicaciones. Basado en la implemantación paulatina de herramientas integradas, facilidades de almacenamiento de datos, e interfases de arquitectura abierta, AD/Cycle sigue una estrategia cuya intención es mejorar la habilidad de los usuarios de satisfacer sus necesidades de procesamiento de datos. Los objetivos del AD/Cycle son:

- Llevar a los clientes lo último en tecnología de desarrollo de aplicaciones a través herramientas suplidas por IBM y otros proveedores.

- Facilitar a los clientes la administración y el control centralizado de la información del desarrollo de aplicaciones.

- Establecer una arquitectura abierta, y un conjunto de servicios, para la integración de las herramientas de desarrollo de aplicaciones a través de todo el ciclo de vida.

AD/Cycle

Figura 2 Desarrollo de aplicaciones hoy

Fuente: IBM Systems Journal Vol. 29 No. 2 Página 172 - 173

Arquitectura

Conceptualmente, el AD/Cycle se puede definir como un marco para, y un conjunto de, herramientas de desarrollo. El marco es provisto por una plataforma diseñada para soportar la integración de las herramientas a través de:

- Una interfase consistente al usuario ,
- servicios para estaciones de trabajo,
- un modelo de información,
- servicios para herramientas,
- Servicios de Repositorio, y
- Servicios de Biblioteca.

Las herramientas para el desarrollo de aplicaciones pueden provenir de varias funtes: de IBM, de otros suplidores y del mismo cliente. IBM ha establecido relaciones con Bachman Information Systems, Index Technology Corporation, y KnowledgeWare Inc. en las cuales se han seleccionado productos que son mercadeados a través de un programa complementario de mercadeo de IBM.

La arquitectura del AD/Cycle identifica las tecnologías de desarrollo de sistemas de acuerdo a las fases del ciclo de vida. A continuación se describen las características de las herramientas en cada categoría, así como los productos que ocupan cada categoría.

Requerimientos

El AD/Cycle facilitará la focalización del desarrollo de aplicaciones en el modelo de la empresa y el análisis y diseño - es decir al principio del ciclo de vida del desarrollo. Modelar la empresa es una técnica que permite a los profesionales del negocio definir, relacionar y validar los datos y procesos utilizados. En la implementación del modelo de la empresa en AD/Cycle, los datos del modelo de la empresa (requerimientos de datos y procesos) son almacenados centralmente como datos de entidad-relación por el Repository Manager, poniendo así estos datos a disposición de las herramientas subsiguientes en el ciclo de vida.

El método de modelación de la empresa a partir de definiciones de procesos permite la construcción de prototipos de dichos procesos. La construcción de prototipos permite la evaluación de la interfase con el usuario y la funcionalidad de los procesos, al principio del ciclo de vida -evitando así los costosos retrocesos que a la fecha han caracterizado el proceso de desarrollo de aplicaciones.

La información del modelo del la empresa es creada por los profesionales de la empresa y no por los analistas y programadores. Este cambio de funciones trae dos beneficios:

- El conocimiento de los procesos del negocio no tiene que ser transferido a los técnicos para iniciar el proceso de desarrollo. Esto reduce costos significativamente y minimiza los errores ocasionados por falta de conocimiento de los procesos del negocio.

- El tiempo de desarrollo puede ser reducido a traves de la iniciación directa del desarrollo de apliaciones.

Las herramientas del AD/Cycle que se pueden utilizar para definir el modelo de la empresa y para validar el modelo a partir de prototipos son:

- DevelopMate de IBM que facilita la definición, refinamiento y validación del modelo a base de prototipos.

- Information Engineering Workbench (IEW) / Planning Workstation de KnowledgeWare Inc. sirve para capturar, modelar y analizar datos acerca de la organización y sus usos de información.

- PC PRISM de Index Technology Corp. sirve para desarrollar y analizar modelos de la empresa y otros modelos que ayudan a armonizar los objetivos del negocio con los del sistema.

Análisis y Diseño

Durante la etapa de análisis y diseño, los requerimentos del negocio definidos en el modelo de la empresa pueden ser utilizados para desarrollar el diseño de la aplicación.

La selección de las herramientas para esta fase depende de la metodología definida por la empresa. La arquitectura abierta del AD/Cycle soporta la integración de herramientas que utilizan todas la metodologías modernas:

- IEW/Análisis Worksation y IEW/Design Worksation de KnowledgeWare Inc., sirven para analizar y refinar los requerimeintos del usuario final y para realizar la definición lógica de los sistemas a partir de los requerimientos del usuario final.

- Excelerator de Index Technology sirve para desarrollar modelos de procesos y datos, validar información de diseño, hacer

prototipos de pantallas y reportes, y para generar documentación del sistema.

- BACHMAN Re-Engineering Product Set de Bachman Information Systems incluye herramientas para elaborar modelos de datos y diseñar bases de datos.

Construcción y Pruebas

Las herramientas para Construcción y Pruebas incluyen lenguajes, generadores y sistemas basados en conocimiento (knowledge - based systems).

Existe un compromiso de IBM de mejorar los lenguajes tradicionales incluidos en SAA (PL/1, COBOL, C, RPG, FORTRAN y REXX) para dotarlos de funciones integradas de desarrollo (editores, pre-procesadores, "debuggers" dinámicos y estáticos etc.). Está planeado que información capturada por herramientas de planeación, análisis y diseño, tales como: prototipos, esqueletos de código, estructuras de datos, definiciones de pantallas y formatos de impresión podrán ser accesados por los usuarios de herramientas que soportarán los lenguajes de tercera generación.

Al generador de aplicaciones Cross Systems Product (CSP), IBM le ha agregado una facilidad llamada "External Source Format" (ESF). ESF permite que la información capturada por las herramientas de Planeación, Análisis y Diseño sea pasada a CSP para generar el código automáticamente. Los usuarios además podrán combinar código producido de la manera tradicional con el código generado por CSP. En 1990, los siguientes proveedores de herramientas CASE anunciaron que sus productos se comunicarían con CSP a traves de ESF: Andersen Consulting; Index Technology; KnowledgeWare; McDonnel Douglas; Nastec; Sage Software y Texas Instruments.

Se espear que eventualmente, el ESF será redundante al madurar la tecnología de repositorio.

IBM ha incluido tecnología de Sistemas Expertos para facilitar la construcción de aplicaciones que combinan la tecnología de sistemas basados en conocimeinto con el tradicional código procedimental, ofreciendo para esto una transición transparente de herrameinta a herramienta. Se ha anunciado la convergencia de los dos productos de IBM: Expert Systems Environment y Knowledge Tool, adicionalmente se incorporará a AD/Cycle el Knowledege Engineering Environment de Intellicorp Inc.

Producción y Mantenimiento

IBM planea crear nuevas herramientas que faciliten el análisis, la creación, la administración y la cobertura de los datos de prueba, como parte de un exhaustivo ambiente de verificación. Se proveerán además herramientas como el COBOL Structuring Facility para ayudar a usuarios a entender y dar mantenimiento a aplicaciones exitentes. También habrán herramientas integradas que permitan al programador determinar las consecuencias de una modificación en el resto del sistema (esta es una de las tareas de mantenimiento que más tiempo consume).

Actualmente IBM ofrece herramientas cono INSPECT que permite controlar la ejecución de programas en C y PL/1, inspeccionando y modificando variables durante la ejecución, y Software Analysis Test Tool que realiza un análisis de la cobertura de las pruebas de programas en PL/1 y COBOL.

Administración del Proceso

Las herramientas para la administración del proceso de desarrollo de aplicaciones identifican la secuencia de las tareas, los puntos de control, los tiempos requeridos y los entregables en cada etapa. En otras palabras, constituyen la metodología del ciclo de vida. Estas herramientas no son específicas de una fase del ciclo e incluyen documentación, comunicaciones dentro del proyecto, control de cambios y análisis de impacto.

Históricamente, la adminsitración del proceso ha sido uno de los aspectos más ignorados del ciclo de vida. Obviamnete es uno de los más vitales para la automatización del desarrollo de software. En AD/Cycle este es uno de los aspectos más prominentes.

Es importante destacar que IBM no ha intentado, en esta ocasión, dictar a sus clientes cual metodología deben utilizar, sino que ha provisto una herramienta que permite a los usuarios definir sus propios estándares de desarrrollo. La herramienta en cuestión es el Aplication Development Project Support, el cual permite a los usuarios definir el modelo del proceso de desarrollo de aplicaciones, salvar el modelo, para luego guiar el proceso ejecutando las actividaes definidas en el modelo.

Comentarios

La piedra angular del AD/Cycle es, obviamente, el Repository Manager, o sea la base de datos donde se almacenan todos los datos relativos al proceso de desarrollo de aplicaciones. La figura 3 muestra (en contraste con la figura 2) como se espera

que sea el proceso una vez implementadas todas las herramientas.

Figura 3 Desarrollo de aplicaciones en el futuro

Fuente: IBM Systems Journal Vol. 29 No. 2 Página 186

Si bien el AD/Cycle fue anunciado hace ya más de dos años, todavía es más un visión y una dirección que una tecnología disponible. En particular se debe mencionar que la primera versión del Repository Manager no satisfizo a los pocos clientes que recibieron una copia en demostración, principalmente por no contener manejo de versiones y por estar limitada a correr bajo MVS (sistema operativo de IBM en Main Frames).

El AD/Cycle representa la tecnología CASE en su máxima expresión: como un sistema totalmente integrado cubriendo todo el ciclo de vida de desarrollo de sistemas, dirigido por un modelo del negocio a partir de una base de conocimientos.

Dada la solidez, la apertura y lo completo del planteamiento de IBM, la pregunta que deben plantearse los Directores de

Tecnología es ¿Qué debemos hacer para estar preparados para aprovechar la tecnología cuando esté disponible?.

Recomendaciones

Es obvio que nadie va a correr a adquirir los productos mencionados aquí por el simple hecho de ser parte del AD/Cycle, sin embargo es también casi obvio que no se debe ignorar esta visión y dirección que muy probablemente tendrá el desarrollo de software de aplicación en el futuro.

En el Computer World del 3 de febrero de 1992, el Editor Ejecutivo publicó un editorial titulado "Vicious Cycle", en el cual critica fuertemente a IBM por el atraso sufrido en la entrega de los productos que harán del AD/Cycle una realidad, en particular los problemas enfrentados por el Repository Manager (la semana anterior al editorial renunció el Director del desarrollo del Repository Manager). Concluye el editorial poniendo muy en duda las posibilidades de éxito de este proyecto de IBM.

No consideramos que los comentarios de este editorial deban disuadir a nadie del futuro del desarrollo de aplicaciones. Por un lado se continuan publicando declaraciones de IBM y de sus socios (incluso en el mismo ComputerWorld) respecto al compromiso total de todas las partes en este proyecto, y por otro lado, el sentido común nos dice que este proyecto está tan bien concebido que, por lo tanto, en el eventual caso de que IBM lo abandonara, es seguro que otro lo retomaría.

Si en los Estados Unidos menos de un 25% de los Centros de Cómputo ha adquirido herrameintas CASE, en nuestro medio el porcentaje debe ser cercano al 1%. Esto podría verse como una ventaja ya que al incursionar en el mundo de la tecnología CASE se podrá hacer con un panorama completo de a donde se pretende llegar con la automatización del desarrollo de aplicaciones.

Las empresas en nuestro medio deberían empezar por definir primero sus metodologías de desarrollo de una manera rigurosa (independiente de las herramientas a utilizar) e implantar sistemas de medición que proveean la retroalimentación necesaria para mejorar y refinar el proceso hasta llegar a un punto estable (y estándar) que sería sujeto a la automatización.

También será provechoso realizar ejercicios de modelación de la empresa y empezar a utilizar herrameintas CASE que mejoran algunas de las etapas del proceso.

Obviamente, al escoger dichas herramientas no sólo se debe considerar el precio y la posibilidad de correrlas en computadores personales (bajo un sistema operativo estándar) sino también el que estén incluidas dentro del plan del AD/Cycle.

Bibliografía

Mercurio, Meyers, Nisbet and Radin, AD/Cycle Strategy and Arquitecture, *IBM Systems Journal Vol 29 No. 2 1990, pp 170 - 187.*

Merlyn and Boone, The Ins and Outs of AD/Cycle, *Datamation march 1 1990, pp 59 - 64.*

Gillin Paul, Vicious Cycle (editorial), *ComputerWorld March 2 1992.*

Ambrosio Johanna, Developers exit could further slow Repository, *ComputerWorld March 2 1992.*

Hamilton Rosemary, AD/Cycle revival bid targets old programas, new methods, *ComputerWorld August 19 1991.*