47

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ACATLAN

TECNOLOGIA CASE PARA LA AUTOMATIZACION DEL DESARROLLO DE SISTEMAS DE INFORMACION

T E S I S P R O F E S I O N A L

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

VICTOR MANUEL VILLASANA MARQUEZ

ACATLAN, EDO. DE MEX. 1996

TESIS CON
FALLA DE ORIGEN

47
2y

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

ESCUELA NACIONAL DE ESTUDIOS PROFESIONALES ACATLAN

TECNOLOGIA CASE PARA LA AUTOMATIZACION DEL DESARROLLO DE SISTEMAS DE INFORMACION

T E S I S P R O F E S I O N A L

QUE PARA OBTENER EL TITULO DE

LICENCIADO EN MATEMATICAS APLICADAS Y COMPUTACION

P R E S E N T A :

VICTOR MANUEL VILLASANA MARQUEZ

ACATLAN, EDO. DE MEX 1996

A mis padres y hermanos
con mucho cariño

A mi esposa y mi hijo con amor
por su cariño, apoyo y comprensión

A la Universidad
por haberme enseñado

JURADO:

PRESIDENTE:	ING. RUBEN ROMERO RUIZ
VOCAL:	ING. JOSE ALFREDO LOPEZ RODRIGUEZ
SECRETARIO:	LIC. JUAN CARLOS RENDON AGUILAR
1er. SUPLENTE:	LIC. JORGE ARTURO LOPEZ PAREYON
2o. SUPLENTE:	LIC. NIELS OMAR GARCIA ESPINOZA

ASESOR DEL TEMA: LIC. JORGE ARTURO LOPEZ PAREYON

TECNOLOGIA CASE PARA LA AUTOMATIZACION DEL DESARROLLO DE SISTEMAS DE INFORMACION

INDICE

INTRODUCCION

El objetivo de este trabajo es conocer la metodología de una de las tecnologías más dinámicas para el desarrollo de sistemas de información basados en computadora llamada C.A.S.E. (Ingeniería de Software Asistido por Computadora).

Para llegar a este fín es necesario conocer la evolución de metodologías que se desarrollaron para proporcionar una aplicación.

Actualmente, son varias las compañias, que mediante su unidad de informática siguen desarrollando sistemas con métodos que satisfacen al usuario pero que no cuentan con una estructura dinámica que permita procesar los requerimientos del mismo, cuando las necesidades para llevar a cabo un proyecto aumentan, CASE es una de las tecnologías más versátiles para solucionar este y otro tipo de inconvenientes. Además, simplifica de manera significativa el tiempo de desarrollo para un sistema, ya que CASE permite la automatización de todas las etapas en el ciclo de vida de un sistema de información.Esta investigación es una herramienta de consulta dirigida a los técnicos, e ingenieros involucrados en el desarrollo de sistemas de información.

Este trabajo analiza tres aspectos:

1. Los antecedentes metodológicos y de software dentro del ciclo de vida del desarrollo de un sistema.
2. Los componentes de un sistema C.A.S.E., aquí se analizará a detalle el entorno de desarrollo de software C.A.S.E, el soprte a los procesos de software y sus plataformas de harware.

3. Se describirá la posible tendencia que seguirá la tecnología C.A.S.E

El término C.A.S.E. no es nuevo, desde hace ya algunos años se hablaba de determinadas técnicas y procedimientos de tipo administrativo así como productos o herramientas destinados a facilitar el desarrollo de sistemas.

Con la aparición de la primeras microcomputadoras a partir de la década de los ochentas, se ha iniciado una verdadera competencia entre las compañias que desarrollan software, procurando que sus productos simplifiquen cada vez más el trabajo del usuario interesado en el desarrollo de aplicaciones. Actualmente muchas herramientas de software son muy potentes, pero al mismo tiempo requieren en algunos casos, de un medio ambiente más dinámico.

Algunas productos CASE, se basan en el diseño automatizado de varias tareas que estan dentro del ciclo de vida del software o de algún proyecto en especifico, es decir, con el apoyo de estas herramientas, los encargados del análisis y desarrollo de aplicaciones pueden crear sistemas interactivamente en computadoras personales. A Esta nueva tecnológia se le conoce como Ingenieria de software asistida por computadora, comunmente llamada tecnología CASE.

La aparición de estas herramientas se ha visto favorecida, por el acercamiento de la informática a un número cada vez mayor de usuarios, lo que ha provocado cambios organizativos a nivel laboral.

A esto hay que añadir que una cultura informática cada vez mayor en los usuarios hace que su interés por resolver problemas a través de la computadora sea muy grande.

El orígen de la tecnología CASE se basa en tecnologías tradicionales de software, como son las metodologías de construcción de sistemas de información así como en el software de tercera y cuarta generación. La diferencia y principal ventaja entre la plataforma CASE y las plataformas de tercera y cuarta generación está, en que la primera se centra en el problema de la productividad en todo el ciclo de vida de un sistema mecanizando. Mientras que las últimas se enfocan a la generación de soluciones, sin tomar en cuenta, en algunos casos, aspectos como una adecuada planificación de recursos humanos y materiales.

Los primeros productos CASE bajo el concepto de la mejora en la productividad, trabajaban sobre la automatización de la documentación y comunicación de un sistema de información. A mediados de los ochentas el campo de acción de estas herramientas se fue diversificando sobre otras tareas englobadas en las etapas de análisis y diseño que forman parte del ciclo de vida de un sistema de información.

I ANTECEDENTES DEL DESARROLLO DE SISTEMAS DE INFORMACION

Introducción

Los que estamos en el medio de la computación reconocemos que existen fuertes obstáculos para implantar con éxito sistemas de información basados en computadora, es decir, que satisfagan eficientemente los requerimientos de los usuarios. La principal dificultad radica en que hemos asimilado tecnología de manera constante, pero no nos hemos actualizado en lo administrativo (manuales de métodos y procedimientos).

Por otro lado sigue habiendo problemas con los usuarios a nivel operativo, seguimos sin documentar nuestros sistemas (cuando los desarrollamos y cuando les damos mantenimiento), seguimos sin buenos controles en la operación de sistemas y en la protección de los archivos vitales de nuestra instalación, seguimos convirtiendo sistemas obsoletos de una máquina vieja a una nueva, sin detenernos a considerar que sería más beneficioso desarrollar un nuevo sistema, lo que provoca los conocidos retrazos en la liberación de los sistemas.

Para evitar caer en el desprestigio se debe de contar con una metodología para realizar el trabajo de manera organizada y que además permita controlar y hacer congruentes los esfuerzos de todo el personal del área de sistemas entre sí y con el personal usuario de los sistemas que se desarrollen e implanten.

I.1 CONCEPTOS BASICOS

Sin importar las organizaciones a las que sirven o la forma en que se desarrollan y diseñan, todos los sistemas de información están compuestos de los siguientes componenetes: Entrada, modelos, salida, tecnología, base de datos y controles, los cuales pueden conjuntarse para obtener sistemas de información funcionales que satisfagan las necesidades de las organizaciones y de sus usuarios. La comprensión de estos componentes, sus relaciones y acoplamiento y su contenido lógico y físico, proporciona los conocimientos básicos para describir, desarrollar y diseñar sistemas de información. Con el fin de conocer estos componentes, se estudiaran brevemente.

Bloque de entrada

La entrada representa a todos los datos, texto, voz e imágenes. Y esta compuesta de transacciones, solicitudes, consultas, instrucciones. Por lo general la entrada sigue un protocolo y un formato para que el contenido, el arreglo y el procesamiento sean adecuados.

En la actualidad, los medios más comunes para la introducción de transacciones y texto son las lectoras de códigos de barras y láser y el teclado, respectivamente. Con frecuencia se puede conseguir mayor eficiencia en la entrada combinando métodos. Por ejemplo, desarrollar sistemas de entrada de voz como una alternativa viable al teclado.

Bloque de modelos

Este componente consta de modelos lógico-matemáticos que manipulan de diversas formas la entrada y los datos almacenados, para producir los resultados deseados.

Un modelo lógico-matemático puede combinar ciertos elementos de datos para proporcionar una respuesta adecuada a una consulta. Puede ser tan simple como

GANANCIAS = INGRESOS - COSTOS

El componente de modelos tambien contiene una descripción de algunas de la técnicas de modelado más populares empleadas para diseñar y documentar las especificaciones de los sistemas. Estas técnicas incluyen tambien árboles y tablas de decisión, diagramas de flujo tradicionales, diagramas de Warnier-Orr.

Bloque de salida

El producto del sistema de información es la salida. Esta es en gran medida, el componente que guía e influye en los otros componentes. Si el diseño de este componente no satisface las necesidades del usuario, entonces los otros componentes tienen poco valor. Con frecuencia la entrada y la salida son interactivas, es decir, la entrada se convierte en salida y la salida en entrada.

De manera lógica, la salida se puede representar mediante estados financieros, facturas, ordenes de compra, reportes de presupuesto. La calidad de esta salida se basa en su exactitud, oportunidad y relevancia.

Bloque de tecnología

La tecnología es la "caja de herramientas" del trabajo en sistemas de información. Captura la entrada, activa los modelos, almacena y acarrea datos y ayuda a controlar todo el sistema. La tecnología consta de tres componentes principales:

La computadora y el almcenamiento auxiliar, las telecomunicaciones y el software.

Las telecomunicaciones comprenden el empleo de medios electrónicos y de transmisión para la comunicación entre nodos a lo largo de una distancia. El software corresponde a los programas que hacen que funcione el hardware de la computadora.

En su esencia misma, la tecnología es un sustituto del esfuerzo humano, de los seis componentes, la tecnología es el más evidente.

Bloque de bases de datos

La base de datos es el lugar donde se almacenan todos los datos necesarios para atender los requerimientos de los usuarios. La base datos se considera desde dos puntos de vista, el físico y el lógico. La base de datos física esta compuesta de los medios de almacenamiento como cintas, diskettes, etc.

La base de datos lógica se desarrolla para buscar, asociar y recuperar los datos almacenados.

Bloque de controles

Todos los sistemas de información están sujetos a una diversidad de riesgos como fallas de los sistemas, errores y omisiones. Algunos de los controles que necesitan diseñarse en el sistema, para asegurar su protección y la integridad de la información, son la instalación de un sistema de administración de registros, el establecimiento de sistemas de respaldo y almacenamiento fuera de las instalaciones, la instalación de sistemas ininterrumidos de energía y dispositivos y controles de acceso.

Existen muchos términos asociados al entorno de los sistemas de información, pero lo más relevante sería entender qúe es un sistema y posteriormente, definir un sistema de información.

a) Un sistema es un conjunto de elementos que forman una actividad o un esquema/procedimiento de procesamiento y buscan una meta común, por medio de la operación con datos y/o energía y/o materia, en una referencia de tiempo para producir información y/o energía y/o materia. 1

b) Un grupo de gente, un conjunto de manuales y equipo de procesamiento de datos (un conjunto de elementos) que seleccionan almacenan, procesan y recuperan datos (operando con datos) para reducir la incertidumbre en la toma de decisiones (buscando una meta común) y producir información en el tiempo cuyo uso es más eficiente (produciendo información en una referencia de tiempo). 2.

De aquí en adelante se usara como referencia la definición de sistema de información establecida anteriormente.

1. Ver nota de Robert G. Murdick y Joel E. Ross en Information systems for modern management, Prentice-Hall, Inc., Englewood-Cliffs, Nueva Jersey, 1975.
2. Se toma como base la definición dada por los autores citados en 4.

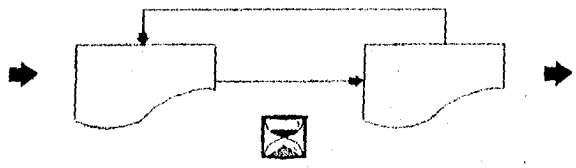


Figura 1.1

(Sistema de informacion manual trabajando adecuadamente)

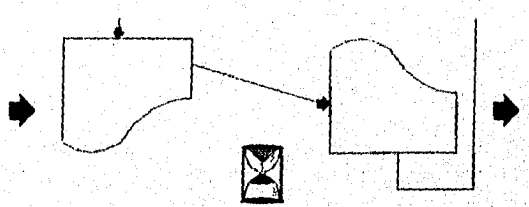


Figura 1.2

(El mismo sistema de información que no funciona adecuadamente)

Las siguientes apreciaciones son las diferncias entre los dos modelos:

1) Más tiempo para producir la información.

2) La información "esta de cabeza" esto es, que el resultado de este proceso no es confiable.

3) El proceso tiene desviaciones no productivas.

4) Los ajustes son mayores y su aplicación es más lenta.

5) Al repetir este ciclo poco a poco se producirá un modelo totalmente diferente al que se planteó originalmente.
Ahora se verá un modelo de un sistema de información basado en computadora (la premisa es que exista un sistema de información y funcione):

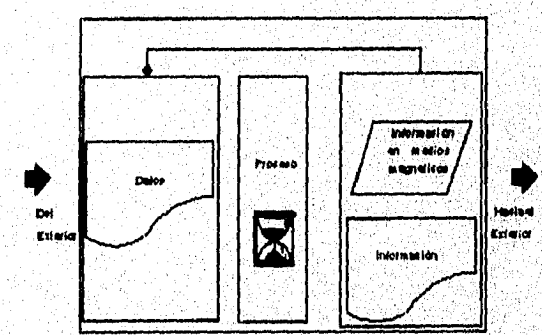


figura 1.3

Lo primero que se puede apreciar en este último modelo (sistema de información automatizado) y un sistema de información, es que hay una separación clara y definida entre la entrada de datos, su proceso en computadora y la salida de información, otra diferencia es el tiempo en que se obtiene la información que se reduce drásticamente (siempre y cuando se haya implantado el sistema).

Por último, es interesante mencionar que en la primera mitad de la década de los ochentas aparecio una tendencia de apoyo a la toma de decisiones gerenciales y en la implantación de las mismas, debido al limitado uso que los sistemas tradicionales de información basados en computadora proporcionaban, ya que estan orientados al tratamiento del dato.

En los sistemas de ayuda a la decisión también, redefine criterios para el diseño, métodos de implantación y desarrollo de los objetivos del sistema orientada a los gerentes. Uno de estos criterios que se usa en el desarrollo de sistemas de apoyo a la decisión es el de desarrollo de arriba hacia abajo (Top-Down) que contrasta con el desarrollo de abajo hacia arriba (Bottom-up) en los tradicionales sistemas de información basados en computadora en los que los niveles de información se integran desde el nivel operativo hacia el directivo. Una representación de los sistemas de información basados en computadora que se mencionaron en las últimas lineas y haciendo referencia a su uso y propósito, es como se indica en la figura 1.4

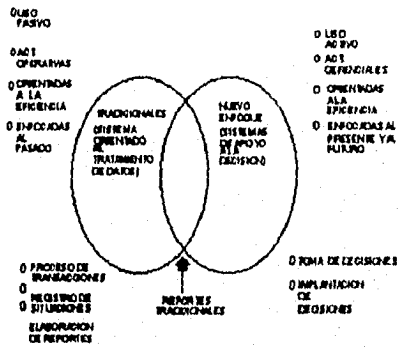


Figura 1.4

(Sistema de información basado en computadora.)

I.2 METODOLOGIA PARA EL DISEÑO DE SISTEMAS

Generalmente se entiende por metodología, a un conjunto de actividades ordenadas de proceder para la obtención de un fin.

Si nos enfocamos en el entorno informático, es decir, en la producción o desarrollo de sistemas computacionales, es evidente que el uso de una metodología en este proceso aporta ventajas que hacen aconsejable su uso.

La metodología que se debe de seguir para el diseño e implantación de sistemas de información basados en computadora debe ser uniforme dentro de una compañia. Es decir, que todo el personal dedicado al desarrollo de sistemas debe seguirla, independientemente de la experiencia del personal, con esto se busca que los sistemas que se vayan a desarrollar, sigan ciertos lineamientos o patrones que permitan una mejor comunicación entre los usuarios y el personal de sistemas. Sin embargo pueden contribuir con sugerencias y comentar modificaciones a esa metodología usando los procedimientos que en ella misma se establezcan.

Estas normas residen normalmente en un manual de estandares del área de sistemas, si en algunas compañias este manual no existe o no se lleva a cabo adecuadamente, los problemas para implantar sistemas serán mayores. La metodología debe indicar los procedimientos que se siguen en lo que se conoce como el ciclo tradicional de desarrollo de un sistema; esto es, investigación, análisis,diseño, desarrollo, implantación y operación. Debe contener también las respectivas políticas y objetivos de cada procedimiento, las descripciones de los puestos del área de sistemas, la ubicación del área de sistemas dentro de la compañía.

Para hacer referencia al manual de estándares esta el manual de anexos que sirve como catálogo de productos finales por obtener(formas, códigos, reportes y procedimientos estándares).

Debido a los grandes avances tecnológicos, es posible que se desarrollen herramientas destinadas a una producción más automatizada de aplicaciones informáticas, las metodologías para el diseño e implantación de sistemas no podian quedar excentos de este avance. Actualmente las metodologías se clasifican en públicas, o sea, aquellas cuya utilización no lleva al usuario al pago de ninguna cantidad a las compañias que la crearon, y las privadas, aquellas desarrolladas por compañias que basan sus beneficios en el cobro de licencias de uso como cualquier otro producto de software.

Dentro de las metodologías públicas pueden distinguirse tres corrientes[3]:

- La francesa que dio como orígen la metodología MERISE, desarrollada por la administración de Francia a partir del año de 1977.
- La americana, basada en las teorías de Edward Yourdon y que tiene algunas variantes aportadas por otros autores como De Marco, Gane y James Martin. El resto de las metodologías existentes, tanto públicas como privadas, deben considerarse como adaptaciones de las citadas anteriormente. Pero el futuro en el uso de estas metodologías depende del soporte de productos de automatización, algunas englobadas actualmente bajo el nombre de CASE.

3 Antonio López Fuensalida Metodologías de desarrollo Macrobit 1990

CONCLUSION

Los problemas que afligen al desarrollo del software se pueden caracterizar bajo muchas perspectivas, pero los responsables del desarrollo de software se centran sobre aspectos de más fondo . La planificación y la estimación de costos son frecuentemente muy pobres.

La calidad del software no corresponde con la demanda del usuario o cliente. Se ha hecho muy poco para mejorar la productividad de los diseñadores de software, los errores en los programas, producen en los clientes insatisfacción y falta de confianza. Frecuentemente los responsables del desarrollo de software han sido ejecutivos de nivel medio y alto, sin conocimentos de software.

El gestor o responsable del desarrollo de software debe comunicarse con todos los implicados en el desarrollo de alguna aplicación (programadores,equipo de soporte y otros..).

La comunicación puede romperse debio a que se comprenden mal las características especiales del software y los problemas particulares asociados con el desarrollo. Los desarrolladores de software han tenido muy poco entrenamiento formal en las nuevas técnicas para el desarrollo de sistemas de información. En muchas organizaciones reina una suave forma de anarquía, cada individuo enfoca su tarea de hacer programas con la experiencia obtenida en trabajos anteriores. Algunas personas desarrollan un método ordenado y eficiente de desarrollo de programas mediante prueba y error, pero otros desarrollan malos hábitos que dan como resultado una pobre calidad en el mantenimiento del software.

Todos de alguna manera nos resistimos al cambio. sin embargo es verdaderamente ironico que mientras el hardware experimenta enormes cambios, la gente del software, se oponga normalmente a los cambios cuando se discuten y se resista al cambio cuando se introduce. Puede que esta sea la causa real de la crisis del software.

II TECNICAS MODERNAS PARA EL DESARROLLO DE SISTEMAS DE INFORMACION

Introducción

La tecnología CASE tiene su orígen en las metodologías tradicionales para el desarrollo de sistemas de información, por ejemplo, el ciclo de vida clásico de algúna aplicación. Sin embargo, al evolucionar la tecnología del harware, fué creciendo el interés por desarrollar técnicas modulares que permitieran incursionar en otras actividades profesionales. A continuación, conoceremos las metodologías más populares, analizando brevemente sus características, principios e interacción, esto permitirá la comprensión de la estructura metodológica de uno de los productos más potentes y dinámicas que existen para el desarrollo de sistemas de información sin importar el tipo de aplicación o el equipo(PC,MINIS o MAINFRAME) en el que se vaya a trabajar, es en estos dos últimos puntos, donde se nota la enorme ventaja que favorece a una herramienta CASE sobre los lenguajes de tercera y cuarta generación. Dentro de las metodologías europeas, las más populares son la francesa MERISE y la inglesa SSADM.[1] Ambas se desarrollaron bajo la supervisión de los respectivos gobiernos con la intención de extenderlas en sus administraciones públicas.

1 Antonio López Fuensalida Metodología de desarrollo Macrobit 1990

Al hablar de metodologías americanas, es importante mencionar que alrededor de la de Yourdon hay multitud de colaboradores que aportan variantes y opciones para determinadas etapas en el desarrollo de sistemas de información. Como por ejemplo:

- La metodología DSSD (Data Structured Systems Development), la metodología de Jackson, metodología de la ingeniería de la información de Martin.

Clasificación de las metodologías estructuradas

Las primeras metodologías relacionadas con las fases en el ciclo de vida de un sistema de información, se efectuaban de manera lineal. Pero a finales de los 60´s se desarrollaron trabajos publicados en varios artículos que presentaban una corriente más evolucionada y consiste en modular las tareas que se dan en cada una de las etapas del ciclo de vida de un sistema de información.

Con el tiempo estos trabajos han sido recopilados y ordenados además de haber sido analizados y probados en situaciones reales para formar una metodología modular o estructurada. No es un método sencillo que se aplica siempre de la misma forma. Más bien es una amalgama que ha evolucinado durante los últimos 20 años.

Probablemente no exista una tecnología tan amplia, que haya despertado tanto interés, como la Ingeniería de Software. Sin embargo, esta, ha prosperado y cada vez es más ampliamente utilizado dentro de la comunidad informática.

La primera tarea dentro de la de etapa análisis en el ciclo de vida de un sistema de información es el establecimiento de la definición de requisitos; esto es, exponer al usuario los servicios proporcionados, así como las restricciones o limitantes a las cuales esta sujeta la operación del sistema.

Es importante distinguir entre las necesidades y los requisitos del usuario.

Una organización puede decidir que necesita un sistema de software para, por ejemplo, apoyar su contabilidad, pero es irreal, presentar esta simple necesidad a un ingeniero de software y esperar que desarrolle un sistema de software aceptable y útil.

Más bien, se debe reunir y analizar la información acerca del problema que se va a resolver, y producir una definición completa a este. También es importante distinguir entre los objetivos y los requisitos del sistema. En escencia, un requisito, es algo que puede probarse, mientras que un objetivo es una característica más general que debe exhibir el sistema, por ejemplo, un objetivo es que el sistema debe ser "favorable al usuario". Esto no se puede probar porque "lo favorable" es un atributo más subjetivo. Un requisito asociado puede ser que toda la selección de mandatos por parte del usuario se haga con menús de mandatos.

El análisis estructurado como parte del análisis de requisitos es una actividad de construcción de modelos o modular. Mediante una notación propia del método estructurado se establecen los modulos que reflejan el flujo y el contenido de información (datos y control).

El análisis estructurado al igual que muchas contribuciones a la Ingeniería de Software no fue definido en un solo artículo o libro clave que incluyera un tratamiento completo del tema.

Los primeros trabajos sobre modelos de análisis aparecieron a finales de 1960 y principios de 1970.[2]

A medida que la información fluye por un sistema de computadora se transforma. El diagrama de flujo de datos (DFD) es una técnica gráfica que representa el flujo de información y las transformaciones que se aplican a los datos al moverse desde la entrada a la salida. Las entradas al sistema pueden ser a través de una gran variedad de formas, por ejemplo mediante elementos de hardware, software y humanos para transformar la entrada en salida.

La figura 2.1 muestra los componentes de un DFD (diagrama de flujo de datos)

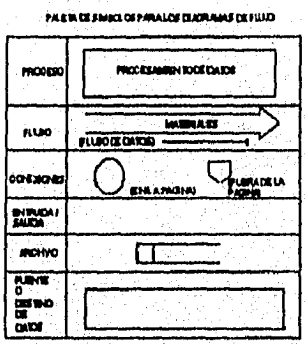


figura 2.1

2 Roger S. Pressman Ingeniería de Software Un Enfoque Práctico Mc. Graw Hill

El sistema basado en computadora se representa como una transformación de información, que en la figura 2.2 aparece como una burbuja, en entidades externas, representadas por cuadros, se originan una o más entradas que aparecen como flechas etiquetadas.

La entrada conduce la transformación que produce información de salida dirigida hacia otras entidades externas. A medida que la información se mueve a través del software es modificada por una serie de transformaciones.

Se puede usar el diagrama de flujo de datos para representar un sistema a cualquier nivel de abstracción. Un DFD de nivel 0 tambien es denominado modelo fundamental del sistema o modelo de contexto y representa al sistema completo como una sola burbuja con datos de entrada y salida representados por flechas de entrada y salida respectivamente. A partir del DFD de nivel 0 aparecen representados procesos(burbujas) y caminos de flujo de información adicionales.

La sencillez de la notación DFD es una de las razones por las que las técnicas de análisis estructurado son ampliamente utilizadas.

Como ya se indicó anteriormente, se puede refinar a cada una de las burbujas en distintos niveles para mostrar un mayor detalle.

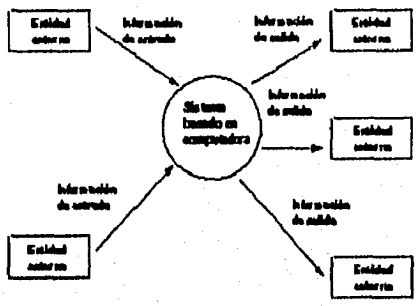


figura 2.2

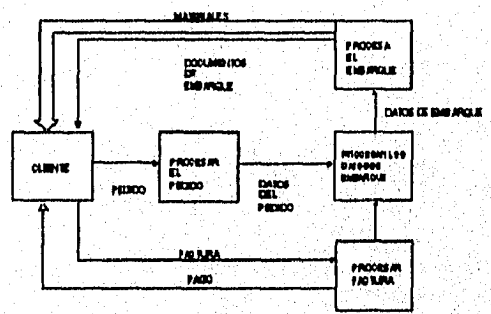


figura 2.3

Los DFD´s se construyen en forma descendente. La figura 2.3, por ejemplo, es un modelo descendente de un sistema de procesamiento de pedidos. Describe el sistema que está considerado en una forma fácil de entender. El diagrama puede llevar a confusión si no se aclara su función con la del Diagrama de Flujo .

Un Diagrama de Flujo de Datos representa el flujo de la información sin representación explícita de la lógica del procedimiento, por ejemplo, condiciones o bucles. No se trata de un Diagrama de Flujo con elementos redondeados. La notación básica que se usa para desarrollar un DFD no es en si misma suficiente para describir los requisitos del sistema.

Por ejemplo una flecha DFD representa un elemento de datos que entra o sale de un proceso. Un almacen de datos representa alguna colección organizada de datos, pero ¿ Cual es el contenido de los datos implicados en las flechas o en el almacen ? Para responder a esta pregunta se aplica otro concepto de la notación básica del análisis estructurado - El diccionario de datos.

El diccionario de datos es un listado organizado de todos los elementos de datos que son pertinentes para el sistema, con definiciones precisas y rigurosas que permiten que el usuario y el analista del sistema tengan una misma comprensión de las entradas, de las salidas y de los cálculos intermedios. Actualmente casi siempre se implementa el diccionario de datos como una herramienta CASE de análisis y diseño estructurados.

Aunque el formato del diccionario varía entre las distintas herramientas, la mayoria contiene la siguiente información:

- **Nombre** : El nombre principal del elemento de datos o de control de una entidad externa.
- **Donde se usa / Como se usa** : Un listado de los procesos que usan el elemento de datos o de control y como lo usan, por ejemplo, como entrada al proceso, como salida del proceso, como

almacen de datos, como entidad externa.

- **Descripción del contenido :** El contenido representado mediante una notación.

- **Información adicional :** Otra información sobre los tipos de datos.

Clasificación del análisis estructurado

Muchas aplicaciones del software son dependientes del tiempo y procesan más información orientada al control de datos.

Un sistema de tiempo real es aquel sistema que controla y es controlado por eventos externos (por ejemplo, señal,disparador) Ejemplos de sistemas reales incluyen: La navegación aerea, las redes de comunicaciones, el control de procesos de fabricación y los procesos químicos. Para especificar los requerimientos de un sistema de tiempo real se manejan los siguientes conceptos para:

- El manejo de interrupciones.
- La comunicación y la sincronización entre tareas.
- El proceso concurrente
- La respuesta oportuna a los eventos externos
- Los requerimientos y las restricciones de los sistemas hardware
- Las interacciones entre el sistema/entorno

El diseño de un sistema de tiempo real debe poder representar procesos que puedan ser interrumpidos por eventos externos y que pueden procesarse en distintos ordenadores debe interactuar con el mundo real en marcos temporales que vienen dados por el mundo real. Por lo anterior se han clasificado las metodologías estructuradas

en pertenecientes a la Ingenieria de Software y a la Ingeniería de la información.

II.1 LA INGEWNIERIA DE SOFTWARE

La Ingenieria de software fue la propúesta por Fritz Bauer como:
"El establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales".

La ingeniería de software abarca un conjunto de tres elementos clave métodos,herramientas y procedimientos, que facilitan al especialista controlar el proceso del desarrollo del software y suministrar a los que practiquen dicha ingeniería las bases para contruir software de alta calidad de una forma productiva.

A continuación describiremos brevemente cada uno de estos elementos.

Los métodos de la Ingenieria de software indican como construir técnicamente el software. Los métodos abarcan un amplio expectro de tareas que incluyen: planificación y estimación de proyectos, análisis de los requisitos del sistema y del software, diseño de estructura de datos. Las herramientas de la Ingeniería de Software suminstran un proceso automático o semiautomático para los métodos. Hoy existen herramientas para soportar cada uno de los métodos mencionados anteriormente. Cuando las herramientas se integran de forma que la información creada por una herramienta pueda ser usada por otra se establece un sistema para el soporte del desarrollo del software, llamado **Ingeniería de Software Asistido por Computadora** (del inglés, CASE).

CASE combina software, hardware y bases de datos.

3 Roger S.Pressman Ingeniería de Software Un Enfoque Práctico Mc Graw Hill

Los procedimientos de la Ingeniería de Software son la unión entre los métodos y las herramientas y facilita un desarrollo amigable y oportuno del software. Los procedimientos definen la secuencia en la que se aplican los métodos, los controles que ayudan a asegurar la calidad y coordinar los cambios y las directrices que ayudan a los gestores del software a evaluar el progreso.

La Ingeniería de Software, como ya se comento esta compuesta por una serie de pasos que abarcan los métodos, las herramientas y los procedimientos. Estos pasos se denominan frecuentemente, paradigmas de la Ingeniería de software. La elección de un paradigma para la Ingeniería de software se lleva a cabo tomando en consideración la naturaleza del proyecto y de la aplicación, los métodos y herramientas a usar. A continuación describiré los modelos o paradigmas de mayor uso en la actualidad.

El ciclo de vida clásico

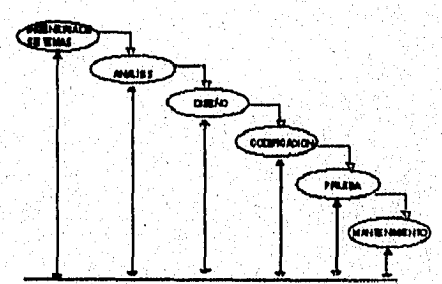


figura 2.4

La figura 2.4 ilustra el modelo del ciclo de vida clásico para la ingeniería de software, algunas veces llamado "modelo en cascada" el modelo del ciclo de vida exige un enfoque sistemático y secuencial del desarrollo del software.

El modelo del ciclo de vida abarca las siguientes actividades:
Debido a que el software es siempre parte de un sistema mayor sea manual o automático, el trabajo comienza estableciendo los requisitos de todos los elementos del sistema y luego asignando algún subconjunto de estos requsitos al software. Este planteamiento del sistema es escencial cuando el software debe interrelacionarse con otros elementos, tales como el hardware, personas o bases de datos. La ingeniería y el análisis del sistema abarca los requsitos globales a nivel del sistema con una pequeña cantidad de análisis y diseño a nivel superior.

La seguna actividad se refiere al proceso de recopilación de los requisitos. Para comprender la naturaleza de los programas que hay que construir, el especialista en software debe comprender el ambito de la información de software así como la función, el rendimiento y las interfases requeridas. Los requsitos tanto del sistema como del software se documentan y se revisan con el cliente.
La siguiente actividad se refiere al diseño del software, que en realidad es un proceso multipaso que se enfoca sobre varios atributos distintos: la estructura de los datos, la arquitectura del software entre otras.

El proceso de diseño traduce los requisitos en una representación del software que pueda ser establecida de forma que obtenga la calidad requerida antes de que comience la codificación. Al igual que los requisitos el diseño se documenta y forma parte de la configuración del software.

La cuarta actividad se refiere a la traducción del diseño en una forma legible para la máquina. Si el diseño se realiza de una manera detallada, la codificación puede realizarse mecánicamente. Una vez que se ha generado el código, comienza la prueba del programa. La prueba se centra en la lógica interna del software, asegurando que todas las sentencias se han probado, y en las funciones externas realizando pruebas que aseguren que las entradas definidas producen los resultados que realmente se requieren.

El mantenimiento constituye el último eslabón en el ciclo de vida clásico de un sistema de información, esto es , el software indudablemente sufrirá cambios después de que se entregue al cliente. Los cambios ocurrirán debido a que se hayan encontrado errores, a que el software deba adaptarse a cambios del entorno externo (por ejemplo un cambio solicitado debido a que se tiene un nuevo sistema operativo o dispositivo periférico) o debido a que el cliente requiera ampliaciones funcionales o del rendimiento. El ciclo de vida clásico es el modelo más antiguo, usado en la ingeniería de software, para el desarrollo de sistemas de información.

Con el paso de unos cuantos años, se han producido críticas al modelo, incluso por seguidores activos, que cuestionan su aplicabilidad a todas las situaciones. Entre los problemas que se presentan algunas veces cuando se aplica el modelo del ciclo de vida clásico se encuentran :

- Los proyectos reales rara vez siguen el flujo secuencial que propone el modelo.
- Normalmente, es difícil para el cliente establecer explicitamente al principio todos los requisitos, el ciclo de vida clásico lo requiere y tiene dificultades en acomodar posibles incertidumbres que puedan existir al comienzo de muchos proyectos.
- El cliente debe tener paciencia, hasta llegar a las etapas finales del desarrollo del proyecto. Finalmente, un error importante no detectado hasta que el programa este funionando, puede ser desastroso.

Cada uno de estos problemas es real. Sin embargo el modelo clásico del ciclo de vida tiene un lugar definido e importante dentro del trabajo realizado en la ingeniería del software.

Suministra una plantilla o base en la que pueden colocarse los métodos para el análisis, diseño, codificación, prueba y mantenimiento. Además los pasos del modelo clásico del ciclo de vida son muy similares a los pasos genéricos aplicables a todos los modelos de la ingeniería de software. A pesar de sus inconvenientes, es significativamente mejor que desarrollar el software sin guía.

Construcción de prototipos

Un prototipo de una aplicación de sistemas de información es un modelo físico de la aplicación que actúa de manera similar al modelo de un automóvil empleado para probar un nuevo automovil entes de que empiece a producirse en serie .

Esta maqueta o modelo de la aplicación de sistemas de información, proporciona al analista de sistemas una retroalimentación temprana por parte de los usuarios acerca de la viabilidad del modelo.
La figura 2.5 se presenta el proceso de elaboración de prototipos. Con base en los requerimientos percibidos del usuario, se construye un prototipo para proporcionar un modelo inicial. Los requerimientos del usuario se comparan con los del modelo. Si el modelo no satisface los requerimientos del usuario, entonces el primer prototipo se descarta y se construye otro, o se modifica, dependiendo de los requerimientos del usuario. Este proceso heurístico, continúa hasta que se llega a un acuerdo y el prototipo es aceptado.

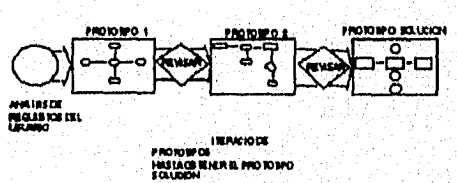


figura 2.5

La elaboración de prototipos es un enfoque de ingeniería para " construir un poco, probar un poco " , antes de construir el sistema final.

Este enfoque elimina la mayor parte de las fallas en el diseño antes de que tengan oportunidad de volverse parte del sistema de producción.

La fuerza de diseño de la interfaz usuario/sistema es un elemento importante en la elaboración de prototipos. Los resultados de un acoplamiento estrecho entre el usuario, el analista de sistemas y los modelos de sistema reducen el vacío entre lo que el usuario dice o piensa que desea del sistema y lo que realmente obtiene. Al usuario se le introduce directamente en el proceso. Esta interfaz más estrecha permite a los usuarios comunicar con mayor precisión sus requerimientos al analista de sistemas para traducir los requerimientos de los usuarios a un sistema funcional proyectos.

Normalmente un cliente define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida.

En otros casos puede no estar seguro de la eficiencia de un algoritmo, de la adaptabilidad de un sistema operativo. En estas y otras muchas situaciones es mejor desarrollar un prototipo.

La construcción de un prototipo o modelo, es un proceso que facilita al especialista la creación de un modelo de software a construir. El modelo tomará una de las siguientes formas:

1)Un procedimiento en papel o un modelo basado en PC´S que describa el primer paso en el desarrollo de prototipos puede consistir en trazar un plan en papel. A continuación se puede utilizar un diagrama de flujo de datos para indicar el flujo la interacción hombre- máquina.

2)Un programa que ejecute parte o toda la función deseada.

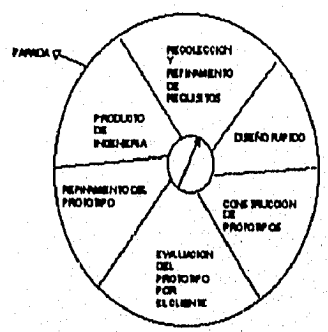


figura 2.6

La figura 2.6 muestra la secuencia de sucesos en la construcción de modelos o prototipos. Como en todos los métodos de desarrollo de sistemas de información la construcción de prototipos, comienza con la recolección de requisitos.

El especialista o analista y el cliente o usuario definen los objetivos globales para el sistema, identifican todos los requisitos conocidos y perfilan las áreas en donde sera necesario una mayor definición, luego se produce un diseño rápido, que se enfoca sobre la representación de los aspectos del sistema transparentes al usuario (por ejemplo, métodos de entrada y formatos de salida).

El diseño rápido conduce a la construcción de un prototipo, el cual es evaluado por el usuario y se usa para refinar los requisitos del sistema a desarrollar. Con esto se da un proceso interactivo en el que el modelo es probado para que satisfaga las requerimientos del cliente,al mismo tiempo que facilita, al que lo desarrolla una mejor comprensión de lo que hay que hacer.

Al igual que en el ciclo de vida clásico, la construcción de prototipos como paradigma para la ingeniería de software, puede ser problemática por las siguientes razones :

1) El cliente ve funcionando lo que parece una primera versión del sistema ignorando que el prototipo se ha desarrollado sin considerar aspectos de calidad o de mantenimiento a largo plazo.

2) El responsable del desarrollo del modelo, frecuentemente impone ciertos compromisos de implementación con el fín de obtener un sistema que funcione rapidamente. Puede que utilice un sistema operativo o un lenguaje de programación inapropiado.

Aunque pueden aparecer problemas, la construcción de prototipos es un paradigma efectivo para la ingeniería del software la clave está en definir al comienzo las reglas del juego; esto es, el especialista y el cliente deben estar deacuerdo en que el prototipo se construya solo para servir como un mecanismo de definición de requisitos. Posteriormente, ha de ser descartado (en parte) y debe construirse el software real con los ojos puestos en la calidad y en el mantenimiento.

El prototipo deberá incluir tanto detalle sobre los componentes estrucurales como sea posible. La entrada y especialmente la salida, deberán definirse junto con la tecnología de apoyo, la base de datos y los controles. Por lo general, la elaboración de prototipos se utiliza mejor en el desarrollo de aplicaciones que no estan definidas correctamente. Asimismo, el desarrollo de prototipos es apropiado en aplicaciones pequeñas de -uno-sobre-uno o uno-sobre-unos-cuantos que requieren una interfaz estrecha usuario/sistema.

Se pueden encontrar probablemente muchas otras aplicaciones de los prototipos, incluso si los modelos simplemente se trabajan en papel. A decir verdad, en cada situación, los prototipos mejoran la visualización y las comunicaciones.

El modelo en espiral

El modelo en espiral para la ingeniería de software ha sido desarrollado para apoyar las mejores características, tanto del ciclo de vida clásico de un sistema de información, como de la creación de prototipos, añadiendo al mismo tiempo un nuevo elemento:
el análisis de riesgo. El modelo representado mediante la espiral de la figura 2.7 define cuatro actividades principales, representados en los cuatro cuadrantes de la figura :

1) Planificación : determinación de objetivos, alternativas y restricciones

2) Análisis de riesgo : análsis de alternativas e identificación /resolución de riesgos

3) Ingeniería : desarrollo del producto.

4) Evaluación del cliente: valoración de los resultados de la ingeniería.

Un aspecto interesante del modelo en espiral se hace evidente cuando consideramos la dimensión radial representado en la figura 2.7

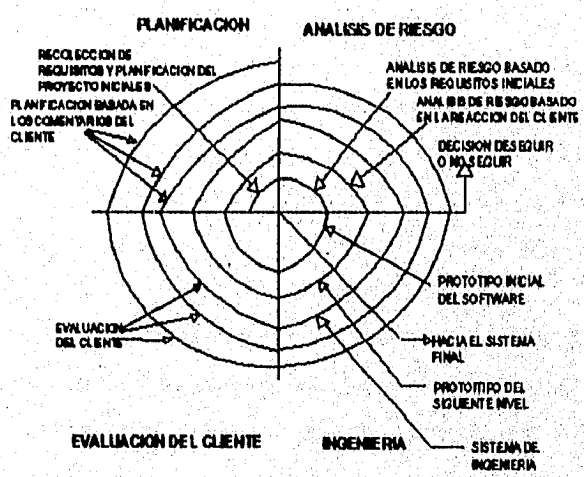


figura 2.7

Con cada iteración, alrededor de la espiral (comenzando en el centro y siguiendo hacía el exterior) se construyen sucesivas versiones del software o sistema, cada vez más completos. Durante la primera vuelta alrededor de la espiral se definen los objetivos, las alternativas y las restricciones y se indentifican y analizan los riesgos.
Si el análisis de riesgo indíca que hay una incertidumbre en los requisitos se puede usar la creación de prototipos, para dar asistencia tanto al encargado del desarrollo como al cliente.
El cliente evalúa el trabajo de ingeniería (cuadrante de evaluación del cliente) y sugiere modificaciones.

En base a los comentarios del cliente, se produce la siguiente fase de planificación y de análisis de riesgo. En cada bucle alrededor de la espiral, la culminación del análisis de riesgo, resulta en una decisión de seguir o no, si los riesgos son demasiado grandes se puede dar por terminado el proyecto. Sin embargo en la mayoría de los casos, se sigue avanzando alrededor del camino de la espiral y ese camino lleva a los desarrolladores hacia un modelo más completo del sistema, y al final al propio sistema operacional.
Cada vuelta alrededor de la espiral requiere ingeniería, que se puede llevar a cabo mediante el enfoque del ciclo de vida clásico o de la creación de prototipos. Debe tenerse en cuenta que el número de actividades de desarrollo que ocurren en el cuadrante inferior derecho aumenta al alejarse del centro de la espiral.

El paradigma del modelo en espiral para la ingeniería de software es actualmente más realista para el desarrollo de sistemas a gran escala. Utiliza un enfoque evolutivo permitiendo al desarrollador y al cliente entender y reaccionar a los riesgos en cada nivel evolutivo. Utiliza la creación de prototipos, como un mecanismo de reducción del riesgo, pero lo que es más importante, permite a quien lo desarrolla aplicar el enfoque de creación de prototipos en cualquier etapa de la evolución del producto,mantiene el enfoque sistemático correspondiente a los pasos sugeridos por el ciclo de vida clásico, pero incorporandola dentro de un marco de trabajo interactivo que refleja de una manera más realista el mundo real.

El modelo en espiral demanda una consideración directa de riesgos técnicos en todas las etapas del proyecto, y si se aplica adecuadamente debe reducir los riesgos antes de que se conviertan en problemáticos. Pero al igual que otros paradigmas el modelo en espiral no es la panacea, puede ser difícil convencer a grandes clientes (particularmente en situaciones bajo contrato) de que el enfoque evolutivo es controlable. Requiere de una considerable habilidad para la valoración del riesgo y cuenta con esta habilidad para el exito.

Por último en sí mismo es relativamente nuevo y no se ha usado tanto como el ciclo de vida clásico o la creación de prototipos.

Técnicas de cuarta generación

El término "técnicas de cuarta generación", abarca un amplio expectro de herramientas de software que tienen algo en común:

Todas facilitan al que desarrolla el sistema en la especificación de algunas características del software a alto nivel, luego las herramientas generan automáticamente el código fuente, basandose en la especificación del especialista. El paradigma técnicas de cuarta generación para la ingeniería de software se orienta hacia la posibildad de especificar el software a un nivel más próximo al lenguaje natural.

Actualmente, un entorno para el desarrollo de software que soporte el paradigma técnicas de cuarta generación, puede incluir, algunas de las siguientes herramientas:

lenguajes no procedurales para consulta a bases de datos, generación de informes, manipulación de datos, interacción y definición de pantallas, generación de código y facilidades gráficas de alto nivel. Todas estas herramientas están disponibles pero solo para ámbitos muy especificos.

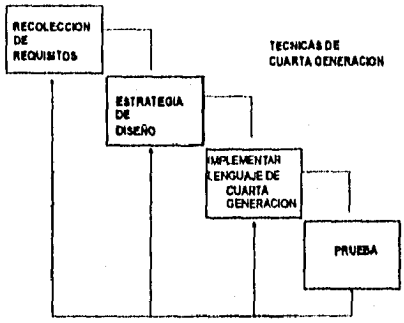


figura 2.8

En la figura 2.8 se describe el paradigma técnicas de cuarta generación para la ingeniería de software.

Al igual que otros paradigmas estas técnicas comienzan con el paso de recolección de requisitos, idealmente, el cliente describe los requisitos que a continuación son traducidos directamente a un prototipo operativo. Sin embargo en la práctica esto no puede ser, por dos razones, una, el usuario puede no estar seguro de lo que necesita y puede no desear o ser incapaz de especificar la información en la forma en que una herramienta incluída en las técnicas de cuarta generación puede aceptarla.

Dos, las herramientas actuales de técnicas de cuarta generación no son lo suficientemente sofisticadas, como para entender el lenguaje natural. En este momento el dialogo cliente-técnico descrito por los otros paradigmas sigue siendo una parte escencial del enfoque de técnicas de cuarta generación.

II.2 LA METODOLOGIA DSSD

La metodología DSSD por sus siglas en ingles (Data structured Systems Development) Metodología del desarrollo de sistemas estructurados de

datos, fué desarrollado por Jean Dominique Warnier a finales de los años 50´s. La metodología de Warnier llamada LCP (Logical Constructions of Programs, Construcción Lógica de Programas) está basada en el desarrollo de programas sobre la estructura de los datos del problema que se va a programar. A finales de los años 70´s, Ken Orr modificó el LCP de Warnier [4], creando una nueva metodología que llamó SPD (Structured Program Design), diseño estructurado de programas, y que ha evolucionado en el DSSD.

El DSSD utiliza la teoría matemática de conjuntos para describir el diseño del programa. Un conjunto es una serie ordenada de objetos que comparten una o varias características comunes. En matemáticas, un conjunto se describe como la lista de sus miembros encerrada entre llaves, por ejemplo, los departamentos de una compañía, forma el conjunto indicado por : { contabilidad,producción y compras } .

4 Roger S Pressman Ingeniería de Software Un Enfoque Práctico Mc Graw Hill

El DSSD utiliza una notación similar para los conjuntos excepto que se elimina la llave de la derecha y que los miembros se listan verticalmente

{ Contabilidad
{ Producción
{ Compras

El diagrama de Warnier-Orr, es la herramienta central de la metodología DSSD, se emplean para mostrar la estructura jerárquica y el flujo de actividades, procedimientos o datos del proceso.

Los pasos básicos del DSSD, aparecen en la figura 2.9

Fase de planificación del proyecto.

Durante esta fase se desarrolla el plan del proyecto, enlazando todos los objetivos del sistema. Se definen los resultados esperados por el sistema así como, las limitaciones comerciales del mismo

1)**Planificación del proyecto** [[Contexto
[**Requerimientos Lógicos** [funciones
[[Resultado
2)**Definición de requerimientos**[[Restricciones
[**Requerimientos físicos**[Alternativas
[Clasificación
[Selección
[Diseño base de
[**Diseño Lógico** [datos lógica
3) **Diseño** [[Diseño de procesos
[[Definición de
[[restricciones físicas
Diseño Físico [Diseños físicos

{alternativos

[Clasificación de

[Diseños alternativos

Construcción y pruebas

Instalación

Operaciones

Utilización

Mantenimiento

Evaluación y auditoría

figura 2.9

Representación de la metodología de desarrollo de sistemas por medio de un diagrama de Warnier-Orr

Fase de definición de requerimientos

Durante esta fase se definen los requisitos del usuario y organizativos para el sistema. Tambien se define el entorno del ordenador necesario para el sistema. Se divide en dos subfases; los requerimientos lógicos y físicos, y a su vez, estas subfases se dividen en pasos. El objetivo principal de esta fase, es definir las salidas principales que va a producir el sistema. Estas salidas son el fundamento del diseño de bases de datos y sistemas durante la fase de diseño del DSSD.

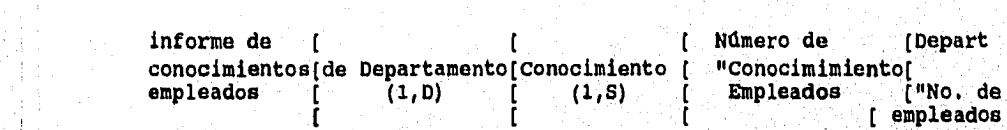
Paralelamente, el diagrama de entidad (utilizado por la metodología de Jackson) se emplea para definir las entidades del sistema y las transacciones que suceden entre ellas.

El diagrama de entidad, se utliza para conocer la amplitud del sistema.

El diagrama de linea de montaje, se utiliza para definir los procesos funcionales, es decir, la transformación del conjunto de entrada en una de salida.

Las salidas necesarias para soportar los procesos y definir la base de salida lógicas se representan en un diagrama de Warnier-Orr

La figura 2.10 es un ejemplo de una base de salida lógica.



"Depart" --- clave de departamento

"Conocimiento" --- Descripción del conocimiento

" Numero de empleados" --- de conocimiento de empleados

figura 2.10

Fase de diseño

La metodología DSSD divide la fase de diseño en dos partes : el diseño lógico y el físico.

El diseño de una base de datos lógica comienza por el desarrollo de la estructura de datos lógica, que contienen los datos mínimos requeridos, para producir una salida. A continuación, se desarrollan los ficheros básicos lógicos y todos sus datos representando una arquitectura de datos estandarizada.

La estructura lógica de los datos, la base de actualización lógica se representa en el diagrama de Warnier-Orr de la figura 2.11 es un ejemplo, de la estructura de datos lógica.

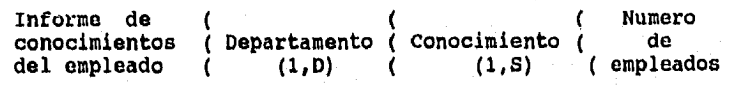


figura 2.11

El diseño del proceso lógico, tiene por objetivo definir las transformaciones necesarias, para producir las salidas del sistema desde las entradas.

Un caso práctico

El método DSSD examina el contexto de la aplicación, es decir, como se mueven los datos entre productores y consumidores de la información, desde la perspectiva de uno de los productores o de uno de los consumidores, a continuación se establecen las funciones de aplicación, finalmente se modelizan, los resultados de la aplicación usando el daigrama de Warnier-Orr. Usando este enfoque DSSD engloba todos los atributos del ambito de información : flujo, contenido y estructura.

Para ilustrar la notación DSSD veamos un sencillo ejemplo.

Un negocio de ventas por correo / telefono denominado Almacen de Software, vende software para computadoras personales. Hemos de especificar para este negocio, un sistema de procesamiento de pedidos basado en computadora.

Para ilustrar el flujo global de la información, para el Almacen del software, observemos el diagrama de flujo de datos de la figura 2.12

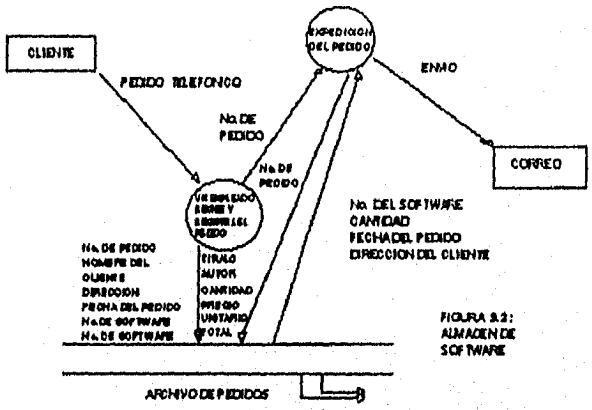


FIGURA 2.2: ALMACEN DE SOFTWARE

figura 2.12
Almacén de software

Los pedidos telefónicos, son recibidos por un empleado, quien registra el pedido en un archivo de pedidos, que esta compuesto por los elementos de datos que aparecen en la misma figura 2.12

A cada pedido se le asigna un número de pedido y se pasa al departamento de expedición, donde se prepara el envío, usando la información que existe en el archivo de pedidos.

Para determinar el contexto de la aplicación en DSSD, debe caracterizarse el problema de forma que nos permita responder a tres preguntas:

1) ¿ Cuales son los elementos de información que han de procesarse ?

2) ¿ Quienes son los productores y consumidores de la información ?

3) ¿ Como ve la información cada productor / consumidor en el contexto de los demás grupos.?

DSSD propone un diagrama de entidades como mecanismo para responder a estas preguntas.

El círculo en un diagrama de entidad, corresponde a un productor o a un consumidor de información (una persona, una máquina o sistema) Los cinco productores y consumidores de información del Almacen de software se muestran en la figura 2.13a, en la figura 2.13b se muestra un diagrama de entidades para el departamento de ventas.

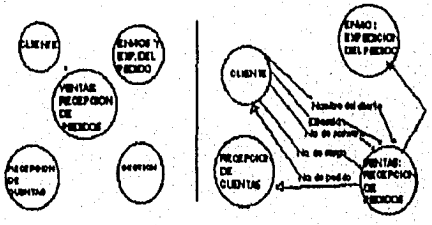


figura 2.13a

figura 2.13b

Despues de revisar cada diagrama de entidades, para comprobar que es correcto, se crea un diagrama combinado de entidades.

Con todos los productores y consumidores de información, las entidades se encuentran dentro de los límites del sistema propuesto (Un sistema de procesamiento automático de pedidos) .

Funciones de la aplicación

Las funciones que deben implementarse para construir el sistema de procesamiento automático de pedidos, pueden encontrarse examinando el flujo de información a través del límite de la aplicación. De acuerdo con la figura 2.14 se anota la secuencia en la que se mueven los elementos de datos a través del límite. Mediante una notación denominada diagrama de linea de ensamblaje, DSSD proporciona un mecanismo para acoplar la información y los procesos que se le aplican, conceptualmente el diagrama de linea de ensamblaje, desempeña el mismo papel que el diagrama de flujo de datos.

Un diagrama de linea de ensamblaje, se desarrolla comenzando con el último flujo de información numerado (figura 2.14) y yendo hacia atrás, mostrando el procesamiento que deriva el elemento de información con el número anterior.

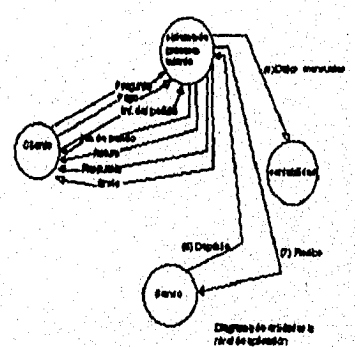


figura 2.14

Resultados de la aplicación

DSSD requiere que el analista construya un prototipo en papel de la salida deseada por el sistema. El prototipo indica la salida primaria del sistema y la organización de los elementos de información que componen la salida. Una vez terminado este proceso, se puede modelizar la jerarquía de la información, usando un diagrama de Warnier-Orr.

El análisis estrucurado

El análisis estructurado propone un enfoque sistemático por pasos para la realización del análisis del sistema y la producción de las especificaciones del sistema.

El análisis estructurado utiliza un método descendente de descomposicion funcional para definir los requerimientos del sistema. La especificación del sistema producido por el análisis estrucuturado es un modelo descendente particionado del sistema a construir. El suministro de una descripción de los requerimientos del sistema a construir es la conexión entre el análisis y el diseño.

La unión entre el análisis y el diseño se proporciona con la descripción de los requerimientos del sistema a construir.

El objetivo primario del análisis es producir una especificación del sistema que defina la estructura del problema a resolver según la visión del usuario. El proposito del diseño es definir la estructura de la solución de forma que sea compatible con la estructura del problema y con los requerimientos del usuario. Los defensores del análisis estructurado sugieren que al utilizar el mismo método de

construcción(descomposición funcional descendente, en la especificación y en el diseño, los dos pueden unirse para representar un sistema que satisfaga las necesidades y expectativas del usuario. La especificación del sistema producida por el análisis estructurado es un modelo lógico,gráfico,particionado,descendente y jerárquico de los procesos del sistema y de los datos utilizados por estos procesos. Se compone de diagrama de flujo de datos, un diccionario de datos y las especificaciones de los procesos.

II.3 METODOLOGIA DE JACKSON

La metodología de diseño de Jackson se basa en el análisis de la estructura de los datos. Su objetivo es obtener la estructura del programa derivandola de la(s) estructura(s) de los datos, asumiendo que se ha especificado totalmente el problema.

Este procedimiento considera al programa como un proceso secuencial. Tiene entradas y salidas, mismas que se le consideran como corrientes secuenciales de registros. El proceso de diseño consiste en definir la estructura de los datos y despues ordenar el proceso lógico(operaciones) para ajustar las estructuras de datos.

Para llevar a cabo un DSJ por sus siglas en ingles (Development System Jackson) el especialista sigue los siguientes pasos:

Paso de entidad acción Primero se identifican las entidades (personas, objetos u organizaciones que necesita el sistema para producir o usar información) y las acciones (los sucesos que ocurren en el mundo real que afectan a las entidades.

Como ejemplo, analizarémos los requisitos de un sistema de control por software para un servicio transbordador de turistas

Un gran parque de diversiones, esta dividido en dos secciones que estan separadas por una milla. Para ayudar a los turistas y visitantes, que desean ir de una sección a otra, se tiene planeado instalar un servicio de transbordador.

Solamente se utilizará un único trasbordador. Cada estación tiene un botón de llamada que los visitantes pueden usar para solicitar un viaje a la otra sección.

Las entidades se seleccionan, examinando todos los nombres de la descripción, por ejemplo, parque de diversiones, visitantes, transbordador, estación y botón. Sin embargo las entidades que atañen directamente al ejemplo y que son las más relevantes son : transbordador y botón.

Una acción ocurre en un instante especifico de tiempo y se aplica a una entidad. Las acciones se seleccionan, examinando todos los verbos de la descripción, como por ejemplo, viaja, llega, pulsa, sube, parte y espera, algunas acciones, se relacionan directamente con entidades que no son importantes, o que se salen del contexto del objetivo. Por lo tanto las acciones que más se ajustan a las entidades seleccionadas son : llega, pulsa y parte.

Al ir enmarcando nuestro modelo y dejando fuera las entidades y acciones candidatas, hemos delimitado al ámbito del sistema a desarrollar.

Por ejemplo, al rechazar visitante, hemos imposibilitado posteriores mejoras como la generación de información sobre el número de visitantes que utilizan el sistema de transbordador un determinado día. Esto no quiere decir, que a lo largo del análisis, se puede modificar la lista de entidades y acciones.

Paso de estructura de entidad

Cuando se usa el contexto del DSJ, la estructura de la entidad está fuertemente influenciada por las acciones asociadas a la misma, a lo largo del tiempo. Para representar la estructura de las entidades. A una entidad, se le aplican acciones como secuencia, como parte de una selección o repetitivamente (una iteración).

En todo programa se consideran tres componentes básicos que, debidamente combinados, resuelven la lógica del mismo. Estos componentes son : Secuencia, Iteración y Selección.
La secuencia entre varias operaciones establece la realización ordenada y consecutiva de las mismas.
La iteración significa la repetición de una operación mientras se cumpla una determinada condición que maneja el fín del bucle. Hay dos posibles representaciones según se plantee la condición de salida del bucle.
Por selección entenderemos la ejecución opcional de una u otra operación dependiendo del cumplimiento de una condición.
Estos tres componentes se han visto representados mediante símbolos de organigramas ordinarios. El diagrama de estructura es una especificación ordenada en el tiempo, de las acciones ejecutadas sobre o por una entidad. Por esta razón, se trata de una

representación más precisa del mundo real que lo pueda ser una simple lista de acciones y entidades. Para cada entidad se crea un diagrama de estructura, que puede ir acompañado de un texto indicativo.

Paso del programa

Este paso se enfoca en combinar las estructuras de datos en una sola estructura de programa. Existen dos partes en el paso de programación, una se refiere a la correspondencia o relación entre los componentes de la estructura de datos, que se identifican encontrando la relación consumo/producción entre los componentes, luego se diseña la estructura del programa, especificando la función de cada uno de los procesos, es decir cuales son de consumo y cuales de producción. A continuación, se verifica la correccción a la estructura del programa reduciendo la estructura del programa a sus estructuras de datos individuales. El proceso de verificación se aplica a cada proceso. Para demostrar que el programa se puede reducir a cada una de las estructuras de datos, si esto puede hacerse se asume que el diseño es correcto.

Paso de operaciones

El paso de operaciones se compone de tres partes:

1.Se listan las operaciones ejecutables requeridas por el programa para convertir su entrada en salida; esto se hace empezando en la salida y yendo hacia atrás hasta la entrada.

2.Cada operación se asigna al lugar apropiado de la estructura del programa.

3.Se verifica la corrección comprobando si se producen todas las salidas y se consumen todas las entradas.

Paso del texto

El último paso en el diseño de Jackson es el de texto, en este paso el diagrama de estructura del programa, junto con sus correspondientes operaciones, se traduce a un texto de estructura, es decir, a un programa en pseudocódigo (ver figura 2.20).

```
INFORME-CONOCIMIENTOS seq
    open fichero-conocimientos;
    open fichero-informe;
    read registro-conocimiento;
    CUERPO-INFORME iter while (not end-of-file)
      GRUPO-CONOCIMIENTOS-DEPART seq
         num-de-empleado :=0;
         depart-en-curso := depart;
         conoci-en-curso :=conoci;
         REGISTRO-CONOCIMIENTO iter while (depart = depart-en-
              curso and conoci = conoci en curso)
              num-de-empleado = num-de-empleado +1;
              read registro-conocimiento
         REGISTRO-CONOCIMIENTO end
         move num-de-empleado to linea-conocimiento;
         write linea-conocimiento
      GRUPO-CONOCIMIENTOS-DEPAR end
   CUERPO-INFORME end;
   close fichero-conocimientos;
   close fichero-informe;
INFORME-CONOCIMIENTOS end
```

figura 2.20

II.4 DISEÑO ESTRUCTURADO DE YOURDON

La metodología de diseño estructurado de Yourdon proporciona un proceso para el diseño paso a paso de sistemas y programas detallados. Unos pasos implican el desarrollo del diseño, otros, la medición y las mejoras de la calidad del diseño. Cada paso es soportado, por un conjunto de estrategias de diseño, guías y técnicas de documentación.

Por ejemplo, en el diagrama de flujo de datos (DFD's) se usan los siguientes elementos:

Proceso Se representan por medio de círculos poniendo en el interior el nombre del proceso, representan las operaciones manuales o mecanizadas que se realizan con los datos.

En una primera etapa deben ser muy generales, con la idea de dar una visión global del sistema y de los objetivos a cubrir. Posteriormente cada círculo o proceso dara lugar en un DFD (diagrama de flujo de datos) de mayor nivel de detalle . Al DFD más general se le llama tambien, Diagrama de Contexto.

Debe tener, al menos, un flujo de entrada de datos y uno de salida.

Para indicar la secuencia de los procesos, en el interior del círculo, además del nombre del proceso, se pondra un número de orden de ejecución del mismo.

Flujo de datos

Representa el movimiento de información o de objetos entre las personas o departamentos contemplados. Gráficamente se dibujan como

lineas que unen al emisor con el receptor de la información u objeto, indicando el sentido del movimiento por medio de una punta de flecha.

Entidades

Son las personas o servicios que reciben o emiten algún flujo de información, se representan como rectangulos, en cuyo interior figura el nombre de la entidad.

Almacenamiento

Su representación gráfica corresponde con dos lineas paralelas en cuyo interior se pone el nombre del archivo o fichero.

Corresponden a los conjuntos básicos de información de la empresa y podran ser manuales o informatizadas y serán origen o destino de un flujo de datos.

A su vez proporcionarán entradas de información a los procesos o serán salida de los mismos.

En cuanto a los simbolos empleados para la representación de los (DFD´s) existen dos variantes principales : La de Yourdon/De Marco y la Gane/Sarson.

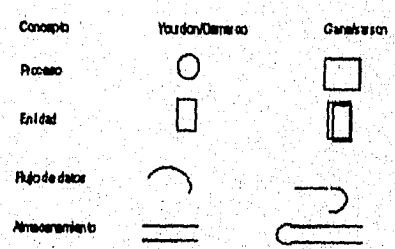


figura 2.15

Simbolos de los DFD´s según Yourdon

El primer paso en el proceso de diseño estructurado es representar el problema de diseño como el flujo de datos a través de un sistema. El sistema se compone de procesos que transforman los datos. Estos procesos y los datos que los enlazan forman las bases para definir los componentes del programa. Se utiliza un conjunto por niveles de diagramas de flujo de datos que los enlazan para representar la primera visión del diseño del programa.

El segundo paso consiste en representar el diseño del programa, como una jerarquía de componentes de procedimiento. Se emplea un diagrama de estructura para mostrar esta visión del diseño.

El diagrama de estructura se deriva del diagrama de flujo de datos obtenido en el paso 1 . El diseño estructurado proporciona dos estrategias de diseño para guiar la transformación de uno o varios diagramas de flujo de datos a un diagrama de estructura: los análisis de transformación y de transacción.

El análisis de transformación es un modelo de flujo de información que divide el diagrama de flujo de datos en tres partes : la entrada que recibe el nombre de rama eferente, el proceso lógico llamado transformación central y la salida llamada rama eferente. Como se muestra en las figuras 2.16, 2.17 el diagrama de estructura se deriva de la identificación de estas dos partes básicas en el diagrama de flujo de datos.

El análisis de transacción se emplea cundo se diseñan programas con proceso de transacciones.

Evaluación del diseño

El tercer paso en el diseño estructurado es la medición en la calidad del diseño. El acoplamiento y la cohesión, son las dos técnicas de medición del diseño proporcionadas por el diseño estructurado.

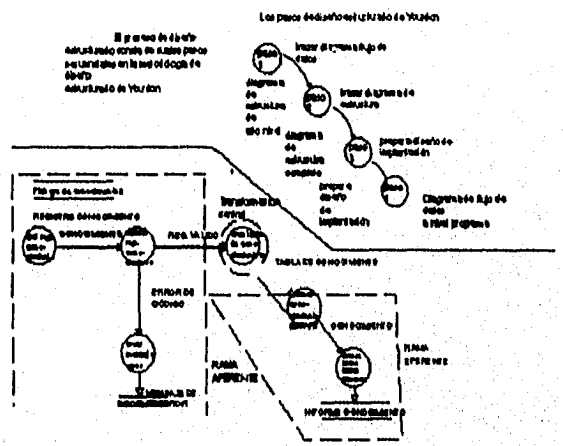


figura 2.16 y figura 2.17

El acoplamiento mide el grado de independencia entre los componentes del procedimiento (modulos) en el diagrama de estructura.
Cuando existe poca interacción entre dos modulos, éstos se describen como ligeramente acoplados, cuando sucede lo contrario, se describen como estrechamente acoplados.

Un diseño de alta calidad, significa que los modulos están lo más ligeramente posible acoplados.

La cohesión mide la fuerza de las relaciones entre los elementos, dentro de un módulo, mientras más estrecha es mejor.

Preparación del diseño para la implementación

El último paso de un diseño estructurado es la preparación del mismo para la implementación, es decir, el proceso de dividir el diseño de programa lógico en unidades físicas, llamadas unidades de carga.

II.5 METODOLOGIA DE LA INGENIERIA DE LA INFORMACION DE MARTIN

La metodología de la Ingeniería de la información de Martin, es una formulación descendente por pasos para la construcción de sistemas de información. Los primeros pasos de esta metodología se centran en la información y en el modelo de la empresa a alto nivel.

El objetivo de esta metodología es gestionar el desarrollo del sistema y la interacción a través del control de los datos. Los datos compartidos en el sistema de información se definen en un modelo lógico y se controlan centralmente. Así, la metodología de Martin revisa todo el sistema en el intento de identificar como se utiliza y se comparte la información. En pasos posteriores, se definen con más detalles los datos y las funciones. En el último paso , se definen e implantan los datos y lógica del programa.

La figura 2.18 muestra los cuatro pasos principales de la metodología de la Ingeniería de la Información de Martin.

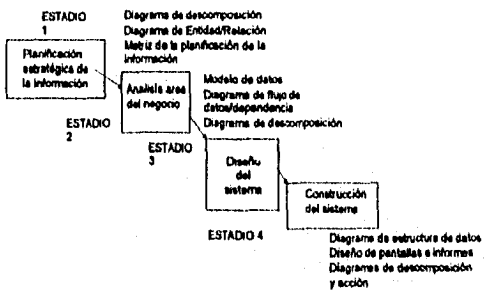


figura 2.18

Paso 1 Planificación de la estrategia de la información

Este estadío establece la creación de un plan estratégico de los sistemas del negocio, en el que se definen los objetivos del mismo. Junto con el plan estratégico, se construyen los modelos de alto nivel, es decir, esta etapa permite conceptualizar el negocio o compañía de una manera general, pero importante, ya que si conocemos los objetivos, los lineamientos organizativos bajo los cuales se guía la compañía y las funciones de la misma, podremos entender, ajustar y evitar posibles errores en el diseño de algún procedimiento para automatizar.

Para definir los tipos de datos de la organización se emplea un diagrama de entidad / relación ver figura 2.19

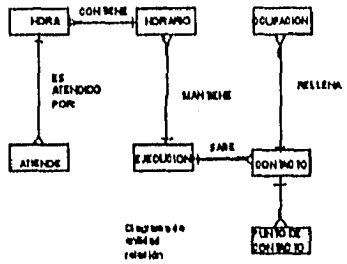


figura 2.19

Una entidad es cualquier información de la empresa que se pueda almacenar (como empleados, departamentos, clientes, facturas, etc.) . Las entidades se descubren llevando a cabo pláticas con los usuarios del sistema de información. Las visiones de los usuarios se analizan y luego se mezclan y clasifican para crear un modelo compuesto.
Finalmente, se determina en este estado, el orden en el cual,se van a desarrollar los sistemas de información que satisfagan los objetivos de la empresa. El usuario tiene una importante función activa en este estadío y en todo el proceso de desarrollo de los sistemas de la Ingeniería de la Información.

Paso 2 El análisis del área del negocio

Las funciones del negocio definidas en el **paso 1** se usan para dividir la empresa en áreas lógicas del negocio (por ejemplo, VENTAS). Después se definen y construyen los sistemas de información necesarios para soportar un área del negocio en particular.
Este paso se concentra en la definición de los datos y procesos necesarios para satisfacer un área del negocio en particular.

Una parte del diagrama entidad / relación desarrollada en el paso 1 que se enfoca a un área en particular del negocio, se extrae y se particulariza lo más posible, es decir, el diagrama se definen completamente las entidades, sus atributos y relaciones recíprocas.

Paso 3 El diseño del sistema

Este paso se ocupa del diseño lógico, es decir se diseñan los procedimientos necesarios para realizar los procesos y las estructuras lógicas en sus datos. Se utiliza la descomposición funcional descendente, es decir, partiendo de la(s) funciones generales de la empresa, se van desarrollando los procesos. Es en este paso, donde se notan similitudes en la fase de diseño entre la metodología tradicional de la Ingeniería de software para el desarrollo de programas (La metodología Clásica para el desarrollo de sistemas de información) y la metodología de la Ingeniería de la Información, el sistema diseñado se representa con diagramas de estructura de datos y con diagramas de acción.

Paso 4 La construcción

Mientras que el **paso 3** se enfoca al diseño del sistema lógico, el **paso 4** se ocupa del diseño del sistema físico, es decir, transformar lo expresado en el diseño lógico mediante código así como la implementación del mismo. Durante el **paso 4**, toda la información del sistema se entrega como una base de datos lógica, una base de datos física y un código documentado del trabajo.

II.6 ANALISIS ORIENTADO A LOS OBJETOS

A medida que se acercaban los años 80, el término orientado a los objetos comenzaba a madurar como un enfoque de desarrollo de software.

Actualmente, el análisis orientado a los objetos (AOO) va progresando poco a poco. En lugar de examinar un problema mediante el modelo clásico de entrada-proceso-salida (flujo de información) o mediante un modelo derivado exclusivamente de estructuras jerárquicas de información, EL AOO introduce varios conceptos nuevos.

Cualquier discusión sobre el análisis orientado a los objetos ¿ Qué significa el término orientado a los objetos ? ¿ Por qué existe un método considerado orientado a los objetos ? ¿ Qué es un objeto.

Para comprender el punto de vista orientado a los objetos, consideremos un ejemplo de un objeto del mundo real - la cosa sobre la que está ahora mismo sentado- una silla. **Silla** es un miembro o "instancia" de una clase de objetos mucho mayor que denominamos mueble. Se puede asociar un conjunto de atributos genéricos a cada objeto de la clase mueble. Por ejemplo, todo mueble tiene **precio, dimensiones, peso, situación y color**, entre muchos atributos posibles. Se aplican estemos hablando de una mesa o de una silla. Dado que silla es un miembro de la clase mueble, hereda todos los atributos definidos para la clase.

Si miramos alrededor en una habitación, veremos un conjunto de objetos físicos que pueden ser identificados fácilmente, así como clasificados y definidos (en términos de atributos y operaciones).

Pero ¿ qué debemos observar una vez que ya hemos aislado todos los nombres?

Los objetos se manifiestan en una de las formas que a continuación se se detallan

Entidades externas (por ejemplo dispositivos, gente otros sistemas)

Cosas (por ejemplo informes, visualizaciones, cartas, señales)

Ocurrencias o sucesos (por ejemplo una transferencia de una propiedad o la terminación de una serie de movimientos de un robot) que ocurren en el contexto de operación del sistema)

Papeles que juegan las personas que interactúan con el sistema (por ejemplo gestor, ingeniero, vendedor)

Unidades organizativas (por ejemplo división, grupo, equipo) que son relevantes para la aplicación.

Lugares (por ejemplo sala de facturación o muelle de descarga) que establecen el contexto del problema y del funcionamiento general del sistema.

Especificación de atributos

Los atributos describen un objeto que haya sido seleccionado para ser incluido en el modelo de análisis. Esencialmente, son los atributos los que definen un objeto son los que aclaran el significado del objeto en el contexto del espacio del problema. Por ejemplo, si fueramos a desarrollar un sistema que gestionara las estadísticas de béisbol para los jugadores profesionales, los atributos del objeto **jugador** serían bastante diferentes de los que tendría si ese objeto

se usara en el contexto de un sistema de pensiones de jugadores profesionales de beisbol.

En el primero, pueden ser relevantes atributos como el nombre, la posición, la media de bateo, los años en la liga o los partidos jugados. En el segundo caso, algunos de esos atributos seguirían siendo significativos, mientras que otros serían reemplazados por atributos como el salario medio, el credito total, la dirección postal.

Definición de las operaciones

Una operación cambia un objeto de alguna forma. Más concretamente, cambia valores de uno o más atributos que están contenidos en el objeto.

Aunque existen muchos tipos distintos de operaciones, generalmente se pueden dividir en tres grandes categorías: (1) operaciones que manipulan los datos de alguna forma (adiciones, eliminaciones, cambios de formato) (2) operaciones que realizan algun cálculo; (3) operaciones que monitorizan un objeto frente a la ocurrencia de algún suceso de control.

Comunicación entre objetos

La definición de los objetos del contexto del modelo de análisis puede ser suficiente para establecer una base para el diseño, pero se debe añadir algo más(durante el análisis o durante el diseño) para que se pueda construir el sistema - se debe establecer un mecanismo

para la comunicación entre los objetos. Este mecanismo se denomina mensaje.

Modelización de datos

Los conceptos de AOO son realmente extensiones de una técnica de análisis que se ha usado durante muchos años en aplicaciones de procesamiento masivo de datos. Esta técnica, denominada modelización de datos o modelización de información, se centra únicamente en los datos. La modelización de datos es útil para aplicaiones en las que los datos y las relaciones que gobiernan los datos son complejos. A diferencia del enfoque de análisis estructurado, la modelización de datos considera los datos independientemente del procesamiento que transforma los datos.

La modelización de datos se usa ampliamente en aplicaciones de bases de datos. Proporciona al analista y al diseñador de la base de datos una a,plia visión de los datos y de las relaciones que gobiernan los datos. Dentro del contexto del análisis estructurado se puede usar la modelización de datos para representar el contenido de los almacenes de datos y de las relaciones que existen entre ellos.

Diagramas de entidad- relación

La notación principal de la modelización de datos es el diagrama de entidad-relación (E-R).

La notación de diagrama E-R es relativamente sencilla. Los objetos de datos se representan como rectángulos etiquetados. Las relaciones se indican mediante rombos. Las conexiones entre los objetos de datos y las relaciones se establecen mediante variadas líneas especiales de conexión. Por ejemplo la relación entre objetos de datos coche y

fabricante se puede expandir representando un diagrama E-R excesivamente simplificado de los elementos de distribución del mercado del automovil. Se han introducido nuevos objetos de datos, transportista y distribuidor. Además, aparecen nuevas relaciones - transporta, contrata , licencia y almacena- indicando cómo están asociados entre sí los objetos de datos.

Finalmente el método de análisis orientado a los objetos proporciona una notación para construir un modelo de AOO . Para construir una especificación gráfica de un sistema basado en computadora, se usan estructuras, temas, conexiones de instancia y caminos de mensajes. El objetivo principal del AOO es identificar las clases de las que se instanciarían los objetos.

Se puede ver la modelización de datos como un subconjunto de AOO. Usando el diagrama de entidad-relación como notación principal, la modelización de datos se centra en la definición de objetos de datos y en la manera en que se están relacionados entre sí.

La modelización de datos se usa en aplicaciones con procesamiento de datos masivos y se puede aplicar como notación complementaria para el análisis estructurado.

La modelización de datos y el diagrama de entidad-relación se convierten para el analista en una notación concisa para examinar los datos dentro del contexto de la aplicación de procesamiento de datos. En la mayoría de los casos, se usa el enfoque de modelización de datos conjuntamente con el análisis estructurado, aunque se puede usar tambien para el diseño de bases de datos y como soporte de cualquier otro método de análisis de requisitos.

Los objetos modelizan casi cualquier aspecto identificable del ámbito del problema: entidades externas, cosas, sucesos, papeles, unidades organizativas,lugares y estructuras pueden ser representados como objetos. Como punto importante, los objetos encapsulan datos y procesos. las operaciones de procesamiento son parte del objeto y son iniciadas pasando un mensaje al objeto. Una definición de una clase forma la base para la reusabilidad en los niveles de modelización, diseño e implementación.

II.7 DISEÑO ORIENTADO A LOS OBJETOS

El diseño orientado a los objetos (DOO), al igual que otras metodologías de diseño orientadas a la información, crea una representación del campo del problema del mundo real y la hace corresponder con el ámbito de la solución, que es el software. A diferencia de otros métodos, el DOO produce un diseño que interconecta objetos de datos (elementos de datos) y operaciones de procesamiento en una forma que modulariza la información y el procesamiento, en lugar de dejar aparte el procesamiento.

El análisis orientado a los objetos, el diseño orientado a los objetos y la programación orientada a los objetos comprenden un conjunto de actividades de ingeniería del software para la construcción del sistema orientado a los objetos.

Orígenes del diseño orientado a los objetos.

Los objetos y las operaciones no son conceptos nuevos de programación pero el diseño orientado a los objetos sí lo es. Al aparecer los

lenguajes de programación de alto nivel (ALGOL, COBOL Y FORTRAN), se pudieron modelizar los objetos y las operaciones del espacio del problema del mundo real con estructuras de datos y de control predefinidas que estaban disponibles como parte del lenguaje de alto nivel.

Conceptos del diseño orientado a los objetos

La terminología orientada a objetos es similar en el análisis como en el diseño, solo que en ésta última se introduciran algunos conceptos adicionales que son relevantes para el diseño.

Métodos de diseño orientados a los objetos.

Esencialmente, el análisis orientado a los objetos (AOO) es una actividad de clasificación. Es decir se analiza un problema con le fin de determinar clases de objeto que sean aplicables en el desarrollo de la aplicación. El diseño orientado a los objetos permite al ingeniero de software indicar los objetos que se derivan de cada clase y las interrelaciones entre ellas.

Los métodos de DOO iniciales estan caracterizados por los siguientes pasos:

1. Definir el problema.
2. Desarrollar una estrategia informal (narrativa de procesamiento)para la realización en software del campo del problema del mundo real.
3. Formalizar la estrategia mediante los siguientes subpasos:

   Identificar los objetos y sus atributos.

   Identificar las operaciones que se les puede applicar a los objetos.

Establecer interfases qur muestren las relaciones entre los objetos y las operaciones

Decidir aspectos del diseño detallado que proporcionen una descripción de la implementación de los objetos.

4. Volver a aplicar los pasos 2,3 y 4 recursivamente.
5. Refinar el trabajo realizado en el AOO, buscando las subclases, las características de los mensajes y otros detalles de elaboración.
6. Representar la(s) estructura(s) de datos asociados con los atributos del objeto.
7. Representar los detalles procedimentales asociados con cada operación.

El diseño orientado a los objetos crea un modelo del mundo real que puede ser realizado en software. Los objetos proporcionan un mecanismo para representar el ámbito de información, mientras que las operaciones describen el procesamiento asociado con el ambito de información. Los mensajes (un mecanismo de interfaz) proporciona el medio por el que se invocan las operaciones. La característica distintiva de AOO/DOO es que los objetos "saben" qué operaciones se puede aplicar sobre ellos. Esta conocimiento se consigue combinando abstracciones de datos y de procedimientos en un solo componenete de programa.

AOO/DOO ha evolucionado como resultado de una nueva clase de lenguajes de programación orientada a los objetos, tales como C++, OBJETIVE-C, CLU. Consecuentemente, las representaciones del diseño

orientado a los objetos son más adecuadas que otras que dependen del lenguaje de programación.

La metodología AOO/DOO consiste en tres pasos que requieren que el diseñador establezca el problema, defina una estrategia informal de resolución y formalice la estrategia, identificando objetos y operaciones, especificando interfases y proporcionanado detalles de implementación para las abstracciones de datos y procedimientos.. El papael del DOO es tomar las clases y los objetos básicos definidos en el AOO y refinarlos con detalles adicionales de diseño..

II.8 ESTUDIO COMPARATIVO ENTRE ELLAS

Evidentemente, en las metodologías expuestas hay puntos comunes y técnicas similares en determinadas fases del desarrollo.

Para establecer una comparación entre ellas, hay que tomar como base los siguientes puntos comunes que toda metodología debe contemplar y después como son abordados estos puntos desde cada una de las metodologías tomaremos como referencia lo siguiente :

ETAPAS DEL CICLO DE VIDA.

MODELIZACIÓN DE DATOS

MODELIZACIÓN DE PROCESOS.

1 Etapas del ciclo de vida

En general todas estas metodologías estructuran el desarrollo en diversas etapas.

La metodología de Yourdon cuya estructura se basa en la ingeniería de información, es quiza la de mayor precisión ya que proporciona un procedimiento paso a paso para construir sistemas de información. Se podría decir tambien que la metodología de Yourdon es la que más trascendencia da al estudio organizativo de la empresa como base de todo el sistema informático.

Por otro lado la metodología de Jackson, no es tan comercial como la de Yourdon, ya que, mientras la última se pude aplicar a otras áreas de manejo de información, la primera se centra en resolver problemas específicos que no sean muy complejos.

2 Modelización de datos

Es común en todas las metodologías la definición de los niveles conceptual, lógico y físico en la definición de los datos.

A nivel coceptual, se admite por parte de todas el modelo de entidad relación como el más adecuado para reflejar la información del sistema. También es una tendencia general del mercado la utilización de bases de datos relacionales como soporte del nivel lógico del modelo de datos.

3 Modelización de procesos

La mayor parte de las metodologías actuales realizan esta modelización a través de los DFD´s de diferentes niveles de detalle.

Los DFD's resultan claros para el usuario y relativamente fáciles de realizar. Evidentemente, si hay un apoyo en una herramienta CASE de diagramación, esta técnica resulta cómoda de realizar y de interpretar por el usuario.

CONCLUSION

Al igual que en otros temas relacionados con el software, la tendencia va dirigida hacía la unificación de criterios y al a creación de estandares, a lo largo de todo el capítulo se ha detectado una cierta analogía entre las diferentes metodologías ya sea en las técnicas como en las fases de desarrollo.

A veces, las diferencias entre las principales corrientes metodólogicas que hemos visto y otras alternativas, son pequeños matíces sobre determinados aspectos y no cambios sustanciales de fondo.

Concretamente en Europa, las metodologías Merise Ssaam serán la base para la creación del Eurométo do como éstandar a seguir por organismos dependientes de la comuidad europea. Si el carácter comunitario que afecta a otros ámbitos, tanto económicos como sociales sigue aumentando, es de esperar que aspectos relacionados con la informática no escape a esa tendencia. En lo que se refiere al aspecto de las metodologías, a pesar de los intereses de las casas comerciales, no es difícil que a nivel de las diferentes administraciones públicas de la comunidad, se llegue a acuerdos en la creación y uso de metodologías comunes.

La automatización de la producción pasa por la generación de código fuente a partir de las especificaciones de funciones a realizar.
Este aspecto puede modifcar la visión actual del ciclo de vida de un sistema ya que las técnicas de programación vendran implementadas en las heramientas y no serán usadas por los programadores.

III CASE UNA ALTERNATIVA PARA LA AUTOMATIZACION DEL DESARROLLO DE SISTEMAS DE INFORMACION

Introducción

Es de todos conocido que no se puede soportar una gran carga de sistemas en espera de desarrollo . Una forma de reducir esta carga de espera es aumentar el número de personas que desarrollan software.

Según se han ido sucediendo los nuevos avances tecnológicos en hardware y software, el entorno de desarrollo ha ido cambiando desde un entorno individual en los años cincuenta y principios de los sesentas hacia el modo de proceso por lotes (batch) en los años sesenta y principios de los setenta, y después al modo de tiempo compartido (time sharing). A finales de los años setenta y comienzos de los ochenta, un nuevo modo, basado en las computadoras personales se ha impuesto como entorno de desarrollo de software. La introducción de potentes herramientas en las estaciones de trabajo con el acompañamiento de metodologías estructuradas transforma y automatiza el proceso de desarrollo de software.

III.1 LA AUTOMATIZACION DEL SOFTWARE

CASE propone una nueva formulación del concepto de ciclo de vida del software, basada en la automatización. La idea básica que subyace en CASE es proporcionar un conjunto de herramientas bien integradas y que ahorren trabajo, enlazando y automatizando todas las fases del ciclo de vida del software.

Las tecnologías tradicionales del software son de dos tipos: herramientas y metodologías. Estas categorías incluyen herramientas de tercera y cuarta generación y más recientemente de quinta generación. La mayoría de estas herramientas son autónomas y dirigidas a la implementación del ciclo de vida del software.

En la categoría de metodologías de construcción de sistemas software, se incluyen las metodologías de desarrollo manual, como el análisis estructurado, el diseño estructurado y la programación estructurada. Estas metodologías definen un diciplinado proceso para el desarrollo del software paso a paso.

La tecnología CASE es una combinación de herramientas de software y de metodologías. CASE es diferente de las primeras tecnologías del software porque se centra en el problema de la productividad del software y no solamente en la implantación de soluciones. Extendiendose a todas las fases del ciclo de vida del software, aborda además la productividad del software durante todo el ciclo de vida, automatizando muchas de las tareas del análisis y del diseño como también las tareas de la implantación y el mantenimiento de los programas.

Como las metodologías estructuradas manuales son demasiado tediosas y de un trabajo muy intenso, en la práctica raramente se siguen a un nivel más detallado. CASE hace prácticas las metodologías estructuradas manuales al automatizar el dibujo de diagramas estructurados y la generación de la documentación del sistema.

III.2 HISTORIA DE CASE

El fracaso de varios proyectos grandes de software, reveló los problemas peculiares de la administración del software. Esos proyectos no fallaron debido a la incompetencia de los administradores o de los programadores. De hecho, la naturaleza de esos grandes proyectos era tal que atrajo a gente por encima de la habilidad promedio. El fallo residió en las técnicas de administración empleadas. En muchos de estos proyectos para el desarrollo de software se usaron técnicas derivadas de proyectos de desarrollo de pequeña escala que resultaron inadecuadas. La entrega del software era lenta, el software era incierto, costaba varias veces lo estimado en un principio y a menudo exhibía características de rendimiento pobres.

Podemos decir que a finales de los 60´s y finales de los 70 ´s , se llevaba a cabo una metodología de tipo lineal para desarrollar alguna aplicación, influenciada por el modelo clásico del ciclo de vida. Un enfoque secuencial siempre será efectivo en aquellos problemas en los que los requisitos estén bien definidos, la complejidad sea relativamente baja y tanto el proyecto global como los riesgos técnicos se comprendan razonablemente bien.

Pero ¿ que ocurre con los problemas que no son así ?

Es muy probable que la metodología a utilizar se desplace hacia un modelo evolutivo para el desarrollo de dicha aplicación.

El modelo evolutivo abarca cuatro etapas: planificación de la gestión, análisis formal de riesgos, ingeniería y atención al cliente

Recordando el estudio del modelo en espiral, cada bucle a través de los cuadrantes acerca más al ingeniero al sistema final.

Para el cuadrante de ingeniería en el enfoque evolutivo (en espiral) es muy probable que las perspectivas futuras apunten hacía sistemas orientados a los objetos.

Utilizando una combinación de técnicas orientadas a los objetos y varias herramientas CASE sofisticadas para desarrollar prototipos de aplicaciones, aumentará drasticamente la importancia del cuadrante de asistencia al cliente. Debido a la naturaleza iterativa del enfoque evolutivo, el usuario se vera obligado a examinar los progresos regularmente, haciendo comentarios y sugerencias que sirvan de retroalimentación. El incremento de participación del cliente, motivado po el enfoque evolutivo, puede llevar a una satisfacción mayor del usuario final y a una mayor calidad del software.

Como se comentó anteriormente, la experiencia obtenida del fracaso de esos proyectos fue la causa directa del nacimiento de herramientas cuya función es administrar adecuadamente el desarrollo de un sistema desde su conceptualización hasta la instalación y mantenimiento.

Estas herramientas se han desarrollado para la especificación, diseño, aplicación y validación del software así como en la compresión de las dificultades de la administración del software. Sin embargo aún no es posible producir un conjunto de principios generales que describan cómo administrar la producción de software en conjunto.

1 Roger S Pressman Ingeniería de Software Un Enfoque Práctico MC GRAW HILL

III.3 El ENTORNO DE DESARROLLO DE SOFTWARE CASE

CASE es un entorno completo, que incluye hardware y software, cuya función es proporcionar asistencia por ordenador para la producción, mantenimiento y gestión de los sistemas de software.

Además, la estación de trabajo proporciona el procesamiento de textos, el almacenamiento y la recuperación de la información, el correo electrónico y las funciones de ayuda y calendario.

El objetivo principal de la utilización de estaciones trabajo CASE es incrementar la productividad del software. Inmediatamente después, un segundo objetivo es incrementar la calidad del software.

Un amplio conjunto de herramientas de software integradas llamadas banco de trabajo CASE compone el software del entorno de la estación de trabajo.

Los hay a las medidas de las necesidades del desarrollador de software; de asistencia a tareas especializadas, como a la gestión del proyecto, el diseño y el mantenimiento.

Los bancos de trabajo CASE difieren de los primitivos entornos de programación, como UNIX e INTERLISP, en su amplia cobertura del ciclo de vida del sofware. UNIX es un sistema operativo de tiempo compartido en el cual, un formato uniforme de fichero es el principal medio de integración en un amplio rango de herramientas de programación. Además UNIX es un entorno de programación de propósito general que no soporta un lenguaje de programación en particular ni una metodología de desarrollo de software. El INTERLISP, por otra

parte, es un entorno de programación que soporta solamente la programación en lenguaje LISP.

Este da a INTERLISP la ventaja de adecuar y optimizar las herramientas a un único lenguaje de programación. Todas las herramientas INTERLISP están escritas, en LISP y proporcionan un sistema operativo fuertemente integrado.

Mientras que los entornos de programación de los años setenta y ochenta se concentraron en herramientas para las fases de codificación e implantación, los bancos de trabajo CASE proporcionan potenetes herramientas para la especificación, el diseño, la implantación, la comprobación y la documentación. Son un entorno de soporte de software de propósito general destinado a soportar toda la gama de tareas del software y su gestión.

Para cumplir los objetivos de mejorar la productividad y la simplicidad de los procesos del desarrollo de software, el banco de trabajo CASE no puede consistir solamente en una colección con las mejores herramientas. Lo que hace un banco de trabajo CASE es ensamblar las herramientas que pueden proporcionar un soporte funcional completo al proceso del software:

* Creación de los requerimientos gráficos de sistema y de las especificaciones de diseño.
* Validación, análisis y referencias cruzadas de la información del sistema.
* Almacenamiento, gestión y comunicación de la información del sistema y de la gestión del proyecto.
* Construcción del prototipo de sistemas y simulación de sistemas.
* Generación del código de sistemas y de documentación.

* Imposición de los estándares y procedimientos.
* Prueba, validación y análisis de los programas.

Para proporcionar un soporte total de software, un banco de trabajo CASE completo debe tener las siguientes prestaciones:

* Gráficos
* Comprobación de errores.
* Depósito de información.
* Juego de herramientas totalmente integradas
* Cobertura total del ciclo de vida.
* Soporte de prototipos.
* Generación automática de código.
* Soporte de metodología estructurada.

CONCLUSION

En realidad, pocas herramientas CASE se utilizan de forma aislada. La mayoría de las herramientas permiten exportar datos en forma de archivo sin estructura, con un formato conocido. Esto permite un intercambio de datos entre las distintas herramientas CASE, utilizando normalmente un filtro de transmisión intermedio.

Esto preserva la infomación contenida en una herramienta, eliminando la necesidad de reintroducir elementos ya existentes del diseño o la especificación y evitando errores tipográficos.

La desventaja del intercambio de datos está, en que a menudo, soló 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" facílmente, perdiendose la posibilidad de transferencia.

Cuando se utilizan muchas herramientas en un proyecto, el número de transferencias puede hacerse muy grande. Finalmente, la transferencia 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 esten utilizando.

Cuando se considera un entorno CASE, los distintos mecanismos de integración de herramientas se implementan de forma diferente dependiendo de la arquitectura, la plataforma y la filosofía del diseñador del entorno. Sin embargo, todos los entornos CASE implementan mecanismos de ejecución y mecanismos de comunicación. Para ilustrar las características de estos mecanismos seguiremos el entorno de Herramientas Portables Comunes (del inglés, PCTE) - uno de los diferentes estándares para entornos CASE integrados.

La mayoría de los entornos I-CASE se diseñan para ser ejecutados sobre sistemas operativos multitarea, de tal forma que pueda trabajar con varias herramientas simultáneamente. Por ejemplo, se puede compilar un módulo al mismo tiempo que se realizan modificaciones sobre el diseño de otro módulo. Además, la finalización de una tarea por una herramienta puede arrancar automáticamente otra tarea sobre una herramienta distinta, si existen los mecanismos apropiados de activación.

IV IMPLEMENTACION DE CASE

Introducción

En los años 50´s casi todas las ramas de la ingeniería, incluyendo la de sistemas, trabajaban con herramientas manuales y tablas que contenian las formulas y algoritmos que necesitaban para el análisis de un problema de ingeniería; (calculadoras mecánicas, no eléctricas) para realizar los cálculos necesarios y asegurar que el producto iba a funcionar. Se hizo mucho trabajo bueno, pero se hizo a mano, una decada despues, y con mucho excepticismo se comenzo a experimentar con la ingeniería basada en computadora.

En 1975. Las formulas y los algoritmos que el ingeniero necesitaba, se incorporaron a programas de computadora, que se utilizaba para analizar una gran variedad de problemas de ingeniería, por ejemplo, las estaciones de trabajo gráficas, conectadas a potentes computadoras, estuvieron en uso en varias compañias y sustituyeron a las mesas de trabajo y otras herramientas.

Con esto se estaba construyendo un puente entre la ingeniería y el trabajo de manufactura, creando el primer enlace entre el diseño asistido por computadora (CAD) y la fabricación asistida por computadora (CAM).

Se continuaban realizando progresos, pero ahora dependian del software. La informática y la ingeniería se estaban conjuntando en uno solo.

Actualmente para desarrollar un producto se requiere de tres características principales :

- Un conjunto de herramientas útiles.
- Un panel organizado que permita encontrar la herramienta rapidamente
- Un especialista en el uso de estas herramientas

Hoy en día, cualquier proyecto requiere, no solamente de herramientas manuales, que en algunos casos no satisfacen los requisitos de los sistemas basados en computadora, sino de herramientas automatizadas que simplifican tiempo de esfuerzo y los margenes de error son mínimos.

Actualmente CASE esta, donde estaban CAD/CAM en 1975.[1] Algunas compañias utilizan herramientas individuales; el uso en la industria, se esta extendiendo rapidamente y se esta llevando a cabo un serio esfuerzo para integrar las herramientas individuales en un entorno concistente. Es importante comentar que existen diferencias entre CASE y CAD/CAM, por ejemplo, el CAD/CAM desarrollo prácticas de ingeniería que habían sido probados durante los ultimos cien años. El CASE sin embargo proporciona un conjunto de herramientas semiautomatizadas y automatizada que estan desarrollando una nueva cultura de ingeniería para muchas empresas.

El CAD/CAM se centra en la resolución de problemas y el diseño así como a los procesos de fabricación.

Componentes de CASE

La ingeniería de software asistido por computadora, puede ser tan simple como una única herramienta que permita desarrollar una actividad específica, o tan compleja como un "entorno" que integre

distintas herramientas, una base de datos, una red, sistemas operativos y muchos otros componentes.

Es importante analizar los lineamientos que dan forma a una herramienta CASE.

En la siguiente figura se representan los bloques, que componen el CASE, cada bloque constituye la base del siguiente.

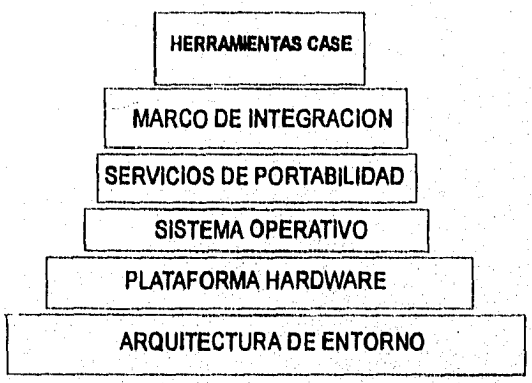


figura 4.1

La arquitectura de entorno, es la plataforma hardware y el software del sistema operativo incluida la red y la gestión de la base de datos constituyen la base del CASE, pero además necesita otros componentes un conjunto de servicios de portabilidad, constituyen un puente entre las herramientas CASE, su marco de integración y la arquitectura de entorno.

El marco de integración es un conjunto de programas especializados que permiten a cada herramienta CASE comunicarse con las demás, para crear una base de datos de proyectos y mostrar una apariencia homogénea al usuario final.

1 Roger S Pressman Ingeniería de Software Un Enfoque Práctico MC GRAW HILL

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.

Los bloques representados en la figura 4.1 representan una base para la integración de las herramientas CASE. Sin embargo, la mayoría no han sido construidas utilizando todos los bloques componentes comentados anteriormente.

Muchas de estas son herramientas que se utilizan como ayuda en una actividad concreta de ingeniería de software, pero se comunica directamente con otras herramientas.

Para comprender mejor el alcance del CASE y para apreciar mejor dónde se pueden aplicar estas herramientas en el proceso de ingeniería de software.

Herramientas de planeación de sistemas de gestión, proporcionan un modelo tan versátil, que se pueden obtener sistemas de información , específicos.

En lugar de centrarse en los requisitos de una aplicación específica, la información de gestión se modeliza según va pasando a través de las distintas entidades organizativas de una compañía. El objetivo principal de las herramientas de esta categoría es ayudar a comprender mejor como se mueve la información entre las distintas unidades organizativas.

Es importante aclarar que este tipo de herramientas son son adecuadas para todas las compañías.

Estas requieren un compromiso importante en cuanto a recursos y un compromiso riguroso en la gestión, para producir un modelo completo y actuar después según la información obtenida de éste. Sin embargo, 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 para la gestión de proyectos Son herramientas que podrían tener una fuerte repercución en la calidad de gestión de proyectos de gran tamaño. Utilizando este tipo de herramientas CASE, el director del proyecto, puede hacer estimaciones útiles de esfuerzo,costo y duración de un proyecto, definir una estructura de partición del trabajo, hacer una planificación realista del mismo y hacer el seguimiento de proyectos de forma continua. Para aquellos directores que dirigen proyectos de desarrollo de software bajo contrato existen herramientas CASE para hacer un seguimiento que va desde los requisitos de la petición de propuesta inicial del cliente, hasta el trabajo de desarrollo que convierte estos requisitos en producto final. Las herramientas para estimación del esfuerzo y el costo de un proyecto de software, le permite al director del proyecto, estimar el tamaño del mismo, utilizando una medida indirecta (lineas de código, y puntos de función) y describir las

características globales del proyecto (complejidad, experiencia del personal, madurez del proceso).

Las herramientas de estimación calculan el esfuerzo estimado, la duración del proyecto, muchas herramientas de esta categoría permiten alguna forma de juego. Por ejemplo, el director puede cambiar la fecha de entrega del proyecto y examinar su repercución sobre el costo final.

Las herramientas de planificación de proyectos permiten al director definir todas las tareas, crear una red de tareas (utilizando normalmente una entrada gráfica. La mayoría de las herramientas utilizan el método de planificación del cámino crítico para determinar la repercución de un retraso en la fecha de entrega.

Cuando se desarrollan sistemas grandes suelen aparecer problemas, por ejemplo, el sistema entregado no satisface plenamente los requisitos especificados por el cliente.

El objetivo de las herramientas de seguimiento de requisitos es proporcionar un enfoque sistemático para aislar requisitos, comenzando con las especificaciones del cliente. La herramienta típica de seguimiento combina la evaluación interactiva de texto con un sistema de gestión de base de datos que almacena y categoriza cada requisito del sistema extraido de las especificaciones originales. La extracción de requisitos puede ser tan sencilla como encontrar cada ocurrencia del verbo "deber" (indicativo de un requisito) que resalta en la pantalla la frase en la que aparece. Las herramientas de gestión (excluyendo las herramientas de planificación y de estimación de proyectos) ayudan a los directores de sistemas de

información a establecer prioridades entre los diferentes proyectos que compiten por un conjunto limitado de recursos.

Basándose en los requisitos y prioridades del cliente, en las restricciones de la organización y en los riesgos técnicos y financieros, estas herramientas utilizan un sistema experto para sugerir el orden en el que se debe llevar a cabo un proyecto.

La categoría de herramientas de soporte engloba las herramientas de aplicación y de sistemas que complementan el proceso de ingeniería del software. Estas incluyen herramientas de documentación, herramientas para gestión de redes y software del sistema, herramientas de control de calidad y herramientas de gestión de base de datos y de configuración de software.

Las herramientas de producción de documentación se utilizan en casi todos los aspectos de la ingeniería de software, las herramientas de documentación suelen estar unidas a otras herramientas CASE por medio de un interfaz de datos suministrada. Por ejemplo, muchas herramientas de análisis y diseño están unidas a uno o varios sistemas de autoedición, de tal forma que los modelos y textos creados durante el análisis y el diseño pueden ser trasmitidos a una herramienta de documentación.

El CASE es una tecnología que se aprovecha en computadoras personales. Por esto, el entorno CASE debe soportar software de redes de comunicación de alta calidad, correo electrónico y otras posibilidades de comunicación, aunque el sistema operativo más empleado en las estaciones de trabajo de ingeniería es el UNIX.

Las herramientas de análisis y diseño permiten al ingeniero de software crear un modelo del sistema que se va a construir.

Estas herramientas permiten la creación de un modelo y también la evaluación de la calidad del modelo. Mediante la comprobación de la validez y la consistencia del modelo, estas herramientas proporcionan al ingeniero cierto grado de confianza en la representación del análisis y ayudan a eliminar los errores antes de que se propaguen al diseño, o lo que es peor, al código mismo.

Las mayoría de las herramientas de diseño y análisis se basan en el método de análisis y diseño estructurado. Permite al ingeniero de software crear progresivamente modelos más complejos de un sistema, comenzando en el nivel de requisitos y concluyendo con un diseño de la arquitectura. Estas combinan una notación específica, heurísticas de análisis y diseño y un proceso de transformación.

Las herramientas de creación de prototipos y de simulación proporcionan al ingeniero de software la capacidad de predecir el comportamiento de un sistema de tiempo real antes de que sea construido. Además, le permiten desarrollar prototipos de sistemas de tiempo real que proporcionen al cliente una visión general de la función, de la operación y de la respuesta, antes de la codificación final.

La mayoría de estas herramientas suministran al ingeniero, medios para crear modelos funcionales y de comportamiento de un sistema. Además muchas tienen la capacidad de generar código, hacen uso de un lenguaje casi formal de especificación. Las herramientas de programación abarcan los compiladores, los editores y los depuradores que se utilizan con los lenguajes de programación convencionales.

Además, también entran en esta categoría los entornos de programación orientados a objetos, los lenguajes de cuarta generación, los generadores de aplicaciones y los lenguajes de consulta a bases de datos.

Durante casi 30 años[2] las únicas herramientas disponibles para los desarrolladores de sistemas eran las herramientas convencionales de programación, y por esto, cualquier problema, desde la concepción de la aplicación, hasta la finalización del producto, era un problema de programación. Hoy en día, las herramientas convencionales siguen existiendo en primera linea del desarrollo del software, pero estan respaldadas por todas las herramientas CASE.

La tendencia hacia la representación de aplicaciones de software en niveles más altos de abstracción ha hecho que muchos diseñadores utilicen herramientas de codificación de cuarta generación.

Los sistemas de consulta de bases de datos, los generadores de código y los lenguajes de cuarta generación han cambiado la forma en que se desarrollan los sistemas. No hay duda de que el objetivo final del CASE es la generación de código automático, esto es, la representación de sistemas a un nivel muy alto de abstracción más alto que el de los lenguajes de programación convencionales.

Aunque los lenguajes de cuarta generación, los generadores de código y los generadores de aplicaciones permiten que un ingeniero de software especifique un sistema a un nivel muy alto de abstracción cada una de estas herramientas difiere en aspectos importantes:

2 Roger S. Pressman Ingeniería de Software Un Enfoque Práctico Mc Graw Hill

Un lenguaje de cuarta generación es la entrada directa a un interprete de lenguaje de cuarta generación. El interprete traduce el código en lenguaje de cuarta generación a código ejecutable. La entrada del generador de código es un lenguaje de especificación procedimental. Este, es procesado por uno o varios módulos de generación de código que traducen el lenguaje de especificación procedimental al lenguaje de programación apropiado.

Las herramientas CASE para el mantenimiento del software abarcan una actividad que actualmente ocupa, aproximadamente, el 70 % del esfuerzo total dedicado al software. La categoría de herramientas de mantenimiento puede subdividirse en :

Herramientas de ingeniería inversa a especificaciones realizan el análisis de un programa una vez que éste ha sido concluido. Para realizar este trabajo,se pueden manejar dos métodos:

- La ingeniería inversa estática, que utiliza como entrada el programa fuente para extraer y analizar su arquitectura, su estructura de control, el flujo lógico.
- Las herramientas dinámicas monitorizan el software durante su ejecución y utilizan la información obtenida para construir un modelo del comportamiento del programa.
- Las herramientas de reingeniería son muy prometedoras, existen muy pocas en uso con calidad industrial. Las herramientas existentes aceptan como entrada código fuente, realizan el análisis de ingeniería inversa y reestructuran el código ajustándolo a los conceptos modernos de programación estructurada.

Las herramientas de reingeniería de datos trabajan con el otro extremo del espectro del diseño. Analizan las definiciones de los datos o una base de datos descrita en un lenguaje de programación. A continuación, traducen esta descripción a una notación gráfica que puede ser analizada por el ingeniero de software.

Finalmente, al trabajar interactivamente con la herramienta de reingeniería, el ingeniero, puede modificar la estructura lógica de la base de datos, normalizar los archivos resultantes y generar automáticamente un nuevo diseño físico de la base de datos.

Estas herramientas pueden utilizar un sistema experto y una base de conocimientos para optimizar el software revisado y aumentar así, el rendimiento.

IV.1 ANALISIS ESTRUCTURADO

herramientas para el análisis

A continuación describiré el módulo de análisis, ya que, los módulos de programación y la implementación por ejemplo, dependen para su desarrollo del análisis que se lleve a cabo para alguna aplicación. Mientras que el módulo de planeación revisa aspectos más generales, es decir, toda la información se maneja a un nivel más alto.

La estación de trabajo de análisis consta de las siguientes herramientas:

Diagramas de acción

Es una herramienta de gráficas y textos que se utiliza para describir sencilla y claramente la lógica detallada de un proceso secuencial.

En estos diagramas se manejan una serie de corchetes de distintos tipos, que nos ayudan en la construcción de bloques de rutinas. Estos pueden manejar una o más labores, o pueden ser repetitivos, mostrando además otros procesos y accesos a depósitos de datos que son lanzadas por la ejecución de decisiones y condiciones lógicas.

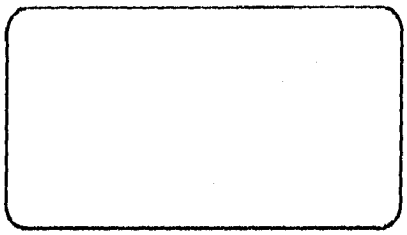
Los diagramas de acción nos permiten describir la lógica detallada del nivel más bajo de proceso en el sistema.

Diagramas de flujo de datos

Estos nos ayudan al tratar de describir como funciona un área del negocio. Estos muestran como fluye la información hacia y desde algún nivel de la empresa, efectivamente, muestran como fluyen los datos entre procesos, como un proceso transforma la información, y como es que los procesos accesan los depósitos de datos e interactuan con agentes externos.

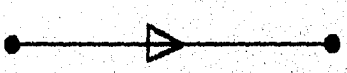
La estación de trabajo de análisis de IEW nos ofrece consistencia entre los diagramas de flujo de datos, asegurando que los flujos que entran y salen de cada proceso aparezacan en niveles superiores e inferiores del sistema.

Procesos representan procesos que manipulan la información y realizan tareas para alcanzar el objetivo del sistema Reciben, procesan y transmiten información.

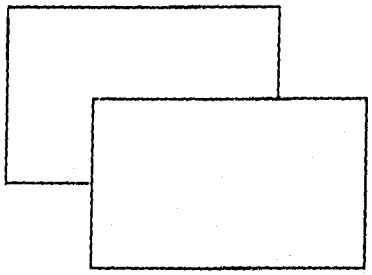


Uniones.- representan puntos de unión o separación de flujos que nos ayudan a hacer más claros los procesos

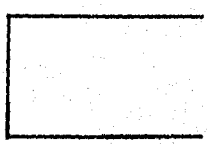
Flujos- representan el flujo de información entre procesos, almacenes de datos y agentes externos



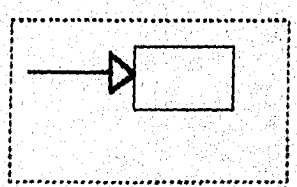
Agentes Externos.- representan agentes que se encuentran fuera del sistema



Almacenes de datos.- representan depósitos de información producida en el proceso, y que sera utilizada por procesos posteriores.



Uniones de contexto.- representan flujos de información entrando o saliendo del proceso en cuestión.

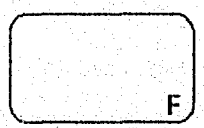


Descomponer el más alto nivel de definición de un proceso en detallados procesos que pueden a su vez ser descompuestos en niveles inferiores más adelante.

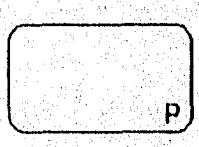
Un diagrama de descomposición muestra cuales son los procesos que se componen de objetos de niveles inferiores o que objetos manejan a otros objetos.

Los símbolos empleados en este tipo de diagrama son los siguientes:

1. Funciones. representan procesos raíz quea su vez pueden ser descompuestos en subprocesos



2. Procesos. representan procesos del negocio



3.Procesos secuenciales- representan procesos del nivel más elemental

4. Contracción de objetos. al encontrar estos, sabemos que el objeto está descompuesto en niveles inferiores.

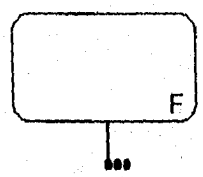


Diagrama de entidades

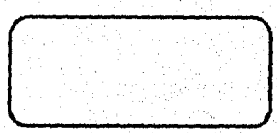
El diagrama de entidades o tambien conocido como diagrama entidad-relación, muestra los diferentes tipos de entidades y las relaciones entre ellas. Estos diagramas son utilizados para desarrollar modelos de datos sobre el negocio en su conjunto, o sobre áreas determinadas dentro de éste. Para cada entidad en un diagrama, es posible abrir otras ventanas que almacenan más información acerca de ésta.

Las entidades pueden ser gente, lugares cosas o ideas sobre las que se quiere manejar información. EL diagramador de entidades provee una manera de describir los requerimientos de información que tiene la empresa.

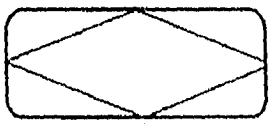
Los diagramas de entidades tienen dos tipos de ventanas tanto en planeación como en diseño, una de ellas es el diagrama de entidades propiamente dicho y otra para la definición de los atributos de cada entidad.

Los simbolos utilizados en este tipo de diagramas son los siguientes:

1. Entidad atributiva.- representan personas, lugares, cosas o conceptos que tienen características de interés para la empresa, el siguiente símbolo se utiliza para representar tanto a entidades fundamentales como atributivas.



2. Entidades asociativa. representan asociaciones entre entidades atributivas.



3.- Cardinalidad.- representa la relación de número entre entidades y se expresa de la siguiente manera:

Mínimos : 0=cero
I=uno

Máximos : I=uno
<=muchos

IV.2 DISEÑO ESTRUCTURADO

El diseño es el primer paso de la fase de desarrollo de cualquier producto o sistema. El objetivo del diseñador es producir un modelo o representación de una entidad que se construirá más adelante. El proceso por el cual se desarrolla el modelo combina: la intuición y los criterios en base a la experiencia de construir entidades similares.

Para diseñar, necesariamente hay que tomar en cuenta la información. Las metodologías de diseño de software surgen de la consideración del campo de la información.

En la actualidad las metodologías de diseño conservan un enfoque sistemático para la obtención de las estructuras de programa, una visión global del software y la base del diseño. Estas caraterísticas forman el diseño estructurado.

Las metodologías de diseño que actualmente de trabajan, se encuentran dentro de este esquema, sin embargo la que mejor se ajusta a éstas caraterísticas es la orientada al flujo de datos por considerar en sus orígenes los conceptos de modularidad, el diseño descendente y la programación estructurada.

En este capítulo se presentará un método de diseño orientado al flujo de datos.

DISEÑO Y FLUJO DE LA INFORMACION

La representación del flujo de la información es como un flujo contínuo que sufre una serie de transformaciones (procesos) conforme va de la entrada a la salida. El diagrama de flujo de datos (DFD) se utiliza como herramienta gráfica para la descrpción del flujo de información. El diseño orientado al flujo de datos define varias representaciones que transforman el flujo de la información en la estructura del programa.

Cada metodología de diseño de software tiene sus puntos débiles y fuertes. Un factor importante para la elección de un método de diseño es la variedad de áreas en las que puede aplicarse. El diseño orientado al flujo de datos puede utilizarse en un amplio rango de áreas de aplicación. El método de diseño orientado al flujo de datos es particularmente útil cuando la información se procesa secuencialmente y no existe una estructura de datos jerárquica. Por ejemplo, los complejos procedimientos de análisis numérico, el control de procesos y muchas otras aplicaciones de ingeniería y científicas caen dentro de esta categoría.

El diseño orientado al flujo de datos permite una cómoda transición de las representaciones de la información (por ejemplo,el diagrama de flujo de datos) contenido en una especificación de requsitos del software a una descripción de diseño de la estructura del programa. La transición desde el flujo de información a la estructura, se realiza como parte de un proceso de cinco pasos: (1) establecer el tipo de flujo de información; (2) determinar los límites del flujo; (3)convertir el DFD en la estructura del programa; (4) definir la jerarquía de control (5) refinar la estructura resultante usando medidas heurísticas.

FLUJO DE TRANSFORMACION

Recordando el modelo fundamental del sistema (diagrama de flujo de datos de nivel 0), la información entra y sale en una forma mediante las teclas pulsadas sobre un teclado de una terminal, los tonos sobre una linea telefónica y los dibujos de un monitor gráfico de computadora. Tales datos externos deben ser convertidos a una forma interna adecuada para el procesamiento. La información entra al sistema mediante caminos que transforman los datos externos a una forma interna y se identifica como flujo entrante. En el interior del software se produce una transición.

Los datos entrantes pasan a través de un centro de transformación, moviéndose a lo largo de caminos que conducen ahora hacia la salida. Los datos que se mueven por estos caminos se llaman flujo saliente, el flujo de datos global ocurre de forma secuencial. Cuando un

segmento de un diagrama de flujo de datos exhibe estas características, lo que tenemos es un flujo de transformación.

FLUJO DE TRANSACCION

El modelo fundamental de muchos sistemas implica un flujo de transformación; por lo tanto, todo flujo de datos se puede clasificar en esa categoría . Sin embargo, a menudo el flujo de información está caracterizado por un único elemento de datos, denominado transacción, que desencadena otro flujo de datos a través de uno o varios caminos. El flujo de transacción se caracteriza por el movimiento de datos a través de un camino de llegada que convierte la información que se introduce mediante las teclas de una terminal en una transacción. Se evalua la transacción y de acuerdo con su valor, el flujo sigue por uno de los muchos caminos de acción.El centro de flujo de información desde el que emanan los caminos de acción se denomina centro de transacción.

Hay que tener en cuenta que en el DFD de un gran sistema, pueden estar presentes los dos tipos de flujos, de transformación y de transacción. Por ejemplo, dentro de un flujo de transacción, el flujo de la información a través de un camino de acción puede tener caracterísiticas de flujo de transformación.

El diseño del método general de diseño orientado al flujo de datos, comienza con una evaluación del diagrama de flujo de datos a nivel 2 o 3 . Se establece el tipo de flujo de información (es decir, flujo de transformación o de transacción y se definen los límites del flujo que delimitan el centro de transformación o de transacción. De

acuerdo con la situación de los límites, se convierten en transformaciones en módulos,como estructuras del programa. La organización y la definición precisas de los módulos se realiza mediante una distribución descendente del control en la estructura (llamada factorización) y la aplicación de criterios para conseguir una modularidad efectiva.

IV.3 GENERACION DE CODIGO

Las diferentes herramientas CASE se enfocan hacia el soporte de diferentes fases del ciclo de vida del software o al desarrollo de diferentes tipos de sistemas. Los Bancos de trabajo CASE se componenen de un conjunto de herramientas integradas que automatizan las tareas a lo largo de todo el ciclo de vida de alguna aplicación. En otras palabras es un ensamblaje de herramientas integradas cuya función es automatizar el desarrollo y mantenimiento de los sistemas.Un banco de trabajo CASE contiene las herramientas que integran las fases básicas del ciclo de vida del software. Las herramientas se integran de forma que la salida de una fase del ciclo de vida pasa directa y automáticamente a la fase siguiente el producto final es una aplicación ejecutable acompañado de su documentación.

Los bancos de trabajo CASE difieren de los primitivos entornos de programación como UNIX e INTERLISP, en su amplia cobertura del ciclo de vida del software.El UNIX es un sistema operativo de tiempo compartido en el cual un formato uniforme de fichero es el principal medio de integración en un amplio rango de herramientas de programación, es además un entorno de programación de proposito general que no soporta un lenguaje de programación en particular ni

una metodología de desarrollo. El INTERLISP, por otra parte, es un entorno de programación que soporta solamente la programación en lenguaje LISP, esta característica le da la ventaja de optimizar y adecuar las herramientas a un único lenguaje de programación.
La siguiente lista de herramientas incluidas en un juego de herramientas CASE de programación son para soportar la implementación de un programa.

proposito obtener un programa documentado y probado que satisfaga todas las especificaciones de los requisitos del programa.

Herramientas

* Herramientas de diagramación estructurada de árbol jerárquico con comprobador de sintaxis y de consistencia.

* Diagramación de la lógica del procedimiento y editor en linea.

* Depósito CASE con gestor de información.

* Generador de código.

* Analizador de código.

* Manipulador de código.

* Generador de datos de prueba.

* Analizador de la cobertura de prueba.

* Comparación de fichas.

* Analizador de correlaciones.

* Control de rendimiento.

* Generador y verificador del entorno de ejecución.

* Simulador del entorno de salida/destino.

Muchas de estas herramientas ya son familiares y éstan muy extendidas entre los programadores.La diferencia está en que las herramientas de los juegos de herramientas están integradas para ser compatibles entre sí.

En otras palabras, han sido diseñadas para ser interfases comunes y compartidas y ser llamadas, utilizadas y alimentadas recíprocamente. Los entornos de programación UNIX e INTERLISP son dos buenos ejemplos de los primeros juegos de herramientas de programación.

Los generadores de código

Una herramienta muy importante en los juegos de herramientas CASE de programación es el generador de código. Este posibilita la producción automática del código desde el diseño de programación. Los generadores de código ofrecen varias ventajas:

Incremento de la productividad sobre la escritura manual del programa en código.

Incrementa la fiabilidad del programa porque el código se produce automáticamente.

Los prototipos se soportan mediante generadores de pantallas de informes.

Generación de código compilado y estructurado, como COBOL, PL/1, FORTRAN, C, o Ada.

Reducción de los costos del mantenimiento, pues el código está mantenido por el generador.

Generadores de código Cobol

En general, el generador de código COBOL soporta el diseño de programa a través de las fases de mantenimiento del ciclo de vida del software.

Tiene tres posibilidades:

1. Gestión de la especificación del programa y diseño de la información.
2. Generación de código y documentación del programa.
3. Soporte del mantenimiento del programa.

El proceso típico de desarrollo de un programa con la utilización de un generador de código COBOL para la construcción de una aplicación comercial es como sigue. Primero el profesional de desarrollo crea las pantallas y los informes requeridos por la aplicación. El programador almacena en el diccionario del generador la información sobre el formato del informe y de los datos que aparezcan en el informe. De modo similar, el programador almacena en el diccionario la información sobre los datos y la ordenación de las pantallas. Tal información se utilizará para generar el código fuente y para construir modelos prototipos del programa.

Seguidamente, el programador define el contenido de cada pantalla y/o informe con más detalle. Por ejemplo, cada elemento de la pantalla se identifica como entrada o salida. Se especifica la edición, validación y formato interno de cada dato elemental de las pantallas e informes. Después se especifica toda la lógica adicional del programa. La lógica del cliente se añade a través del empleo de subrutinas,macros, dispositivos de alto nivel proporcionados por el generador. Cuando toda la información necesaria relativa al programa está completa, el generador ya puede producir el código fuente, definiciones de las bases de datos, sentencias del JCL(Job Control Language) y documentación.

Al otro extremo del prototipo en el ciclo de vida del software está la generación de código Un CASE workbench automatiza la fase de implementación del programa partiendo de las especificiones de diseño del programa. Se genera un esquema o un programa completo. En el caso de la generación del esquema, se generan automáticamente código para una base de datos,fichero,pantalla y descripción de informes, más una salida comentada de la lógica de procedimiento. La lógica adicional del program debe codificarse manualmente para completar el programa.
En el caso de la generación completa del programa, desde las especificaiones del diseño, se genera automáticamente el programa completo más su documentación. Lo cual incluye:

* Código ejecutable
* JCL(Job Control Languaje)
* Pantallas de ayuda.
* Mensajes de error.
* Documentación del usuario y del programa.
* Casos de prueba.

Todos estos componentes del programa completo se juntan y se almacenan en el depósito CASE para su fácil acceso,actualización sincronizada y posterior mantenimiento.
El código generado puede ser código objeto o código fuente. Muchos programadores prefieren el código fuente, porque en caso de necesidad puede ser comprendido y programado por programadores humanos. Además es más portátil y puede ser compatible con el sistema de software existente.

El código objeto ofrece la ventaja de una mayor eficiencia, y probablemente llegue a ser el más extendido en el futuro entre los usuarios de CASE a medida que se convencen de la fiabilidad y eficiencia de la generación automática de código.

```
           (GROUP-OUTPUT-POS)
       COMPUTE LAST-STATMENT-NUM = 0000000064
*      MOVE 192 TO LAST-NAME-DYN1 OF LAST-NAME-ATTR OF ATTENDEE OF
           OUTPUT-Z OF GROUP-OUTPUT (GROUP-OUTPUT-POS)
       ADD 32 TO LAST-NAME DYN1 OF LAST-NAME-ATTR OF ATTENDEE OF
           OUTPUT-Z OF GROUP-OUTPUT (GROUP-OUTPUT-POS)
       MOVE 'B7' TO LAST -NAME-DYN2 OF LAST-NAME-ATTR OF ATTENDEE OF
           OUTPUT-Z OF GROUP-OUTPUT (GROUP-OUTPUT-POS)
       COMPUTE LAST-STATMENT-NUM = 0000000065
*      MOVE 192 TO FIRST-NAME-DYN1 OF FIRST-NAME-ATTR OF ATTENDEE OF
           OUTPUT-Z OF (GROUP-OUTPUT-POS)
       ADD 32 TO FIRST-NAME-DYN1 OF FIRST-NAME-ATTR OF ATTENDEE OF
           OUTPUT-Z OF GROUP-OUTPUT (GROUP-OUTPUT-POS)
       MOVE 'B7'TO FIRST-NAME-DYN2 OF FIRST-NAME-ATTR OF ATTENDEE
           OF OUTPUT-Z OF GROUP-OUTPUT (GROUP-OUTPUT-POS)
PARA -0008651125-EXIT.
       EXIT.
PARA-0008651138.
*      COMPUTE LAST-STATEMENT-NUM = 0000000070
       MOVE GROUP-REFD TO GROUP-OUTPUT-REF
       MOVE FUNCTION-CODE OF WORK-AREA OF INPUT-Z OF INPUT-GROUP
           (INPUT-GROUP-POS)
       TO FUNCTION-CODE OF WORK-AREA OF OUTPUT-Z OF GROUP-OUTPUT
           (GROUP-OUTPUT-POS)
       COMPUTE LAST-STATMENT-NUM = 0000000071
       MOVE GROUP-REFD TO GROUP-OUTPUT-REF
       MOVE LAST-NAME OF ATTENDEE OF INPUT-Z OF INPUT-GROUP
```

```
      (INPUT-GROUP-POS)
    TO LAST-NAME OF ATTENDEE OF OUTPUT-Z OF GROUP-OUTPUT
      (GROUP-OUTPUT-POS)
    MOVE FIRST-NAME OF ATTENDEE OF INPUT-Z OF INPUT-GROUP
      (INPUT-GROUP-POS)
    TO FIRST-NAME OF ATTENDEE OF OUTPUT-Z OF GROUP-OUTPUT
      (GROUP-OUTPUT-POS)
    MOVE MIDDLE-INITIAL OF ATTENDEE OF INPUT-Z OF INPUT-GROUP
      (INPUT-GROUP-POS)
    TO MIDDLE INITIAL OF ATTENDEE OF OUTPUT OF OUTPUT-Z OF GROUP-
    OUTPUT
      (GROUP-OUTPUT-POS)
    MOVE AREA-CODE OF ATTENDEE OF INPUT-Z OF INPUT-GROUP
      (INPUT-GROUP-POS)
    TO AREA-CODE OF ATTENDEE OF OUTPUT-Z OF GROUP-OUTPUT
      (GROUP-OUTPUT-POS)
    MOVE PHONE-NUMBER OF ATTENDEE OF INPUT-Z OF INPUT-GROUP
      (INPUT-GROUP-POS)
    TO PHONE-NUMBER OF ATTENDEE OF OUTPUT-Z OF GROUP-OUTPUT
      <8GROUP-OUTPUT-POS)
    MOVE CORPORATION OF ATTENDDE OF INPUT-Z OF INPUT-GROUP
      (INPUT-GROUP-POS)
    TO CORPORATION OF ATTENDEE OF OUTPUT-Z OF GROUP-OUTPUT
      (GROUP-OUTPUT-POS)
    COMPUTE LAST-STATMENT-NUM = 0000000072
*   MOVE 192 TO LAST-NAME-DYN1 OF LAST-NAME-ATTR OF ATTENDEE OF
```

Figura 3.4 Ejemplo de código generado automáticamente en COBOL por INFORMATION ENGINEERING FACILITY de Texas Instruments Incorporated.

El proposito de los juegos de herramientas de programación son obtener un programa documentado y probado que satisfaga todas las especificaciones de los requisitos del programa.

*Herramientas de diagramación estructurada de árbol jerarquíco con comprobador de sintaxis y de consistencia.

*Diagramación de la lógica del procedimiento y editor en linea.

*Generador de código,Analizador de código, Manipulador de código.

La diferencia entre este juego de herramientas y las que comunmente manejamos es que las primeras están integradas para ser compatibles entre sí. En otras palabras, hab sido diseñadas o particularizadas paracompartir interfases comunes además para ser llamadas, utilizadas y alimentadas reciprocamente. Los entornos de programación UNIX e INTERLISP son dos buenos ejemplos de los juegos de herramientas de programación.

Los generadores de código.
Una herramienta muy importante en los juegos de herramientas CASE de programación es el generador decódigo. Este posibilita la producción

automática del código desde el diseño de programación.Los generadores de código ofrecen varias ventajas.

IV.4 CONSIDERACIONES SOBRE LA HERRAMIENTA CASE UTILIZADA (I. E. W.)

Information Engineering Workbench

El incremento de la productividad del software

La clave de la productividad es la automatización. Esto se aplica lo mismo al desarrollo de software que a cualquier otro tipo de producto fabricado.

La ayuda por computadora a la ingeniería de de software (CASE) es la automatización de muchas de las tareas del ciclo de vida del software.

La prueba de que CASE funciona, ya ha llegado para muchas compañias que han conseguido un incremento substancial de la productividad. En cada caso particular, han elegído herramientas que contribuyen a automatizar las metodologías estructuradas. Las metodologías estructuradas, muchas de las cuales se desarrollaron en los años setenta, proporcionan un método probado para producir sistemas de software de alta calidad y fiables. El fallo de la metodología estructurada ha sido, sin embargo, el gran esfuerzo manual y el consumo de tiempo que requieren. Las herramientas automáticas resuelven éste problema problema, encargandose de gran parte de los pesados detalles y el papeleo de la aplicación de estas metodologías. Esto libera al analista para concentrarse en la parte creativa del proceso de desarrollo del software.

La idea básica de CASE es soportar cada fase del ciclo de vida con un conjunto de herramientas que economicen el trabajo. Algunas herramientas se concentran en el soporte de las primeras fases del ciclo de vida, ofrecen asistencia automática en forma de diagramas dibujados automáticamente, generadores de pantalla. Otros se enfocan a las fases de implantación del ciclo de vida, incluyendo los generadores automáticos de código.

En algunos casos, se utilizan estas herramientas en combinación con lenguajes de tercera y cuarta generación. En otros casos los reemplazan permitiendo el desarrollo de las especificaciones de alto nivel del programa con las que puede generarse el código.

En este cápitulo veremos cómo algunas empresas han incrementado la productividad del software con el empleo de herramientas CASE, específicamente de **Information Engineering Workbench de KnowledgeWare.**

Experiencias de productividad con Information Engineering Workbench

El Information Engineering Workbench conocido como (I. E. W.) de KnowledgeWare es un banco de trabajo CASE para el análisis de sistemas comerciales y un compañero de metodología que automatiza una metodología estructurada llamada Ingeniería de Información.

El IEW tiene tres utilidades básicas:

1. Una herramienta de diagramación automática para el dibujo de diagramas estructurados.

2.Un depósito CASE, con un sistema inteligente de la gestiónde la información llamado enciclopedia, para almacenar, analizar y coordinar toda la información del sistema.

3.Una herramienta automatizada de análisis, llamada Knowledge coordinator, para verificar la integridad y consistencia de toda la información del análisis y del diseño del sistema.

El IEW se emplea principalmente para proporcionar un soporte automatizado de los pasos de análisis del sistema y de los datos de metodología de la Ingeniería de Información.

Como usuario de IEW, he encontrado tres beneficios importantes. **Primero**, el IEW es un compañero de metodología, además de automatizar tareas como el dibujo y la revisión de diagramas estructurados, el IEW fuerza y normaliza la utilización de la metodología de la Ingeniería de Información.

Considero que la imposición de una metodología estructurada es esencial en la construcción de sistemas de calidad.

Segundo El IEW aumenta la productividad del desarrollo de software, reduciendo el tiempo de los plazos de análisis, diseño y construcción de los sistemas de información. Por ejemplo en una entrevista con el usuario final que normalmente me lleva 4 horas, con la asistencia de IEW sólo necesito una hora.

Tercero El IEW mejora la comunicación entre los analistas por medio de la documentación automatizada.

El empleo de diagramas estructurados para explicar la especificación del sistema mejóra la comunicación entre los usuarios finales, los gestores y los programadores.

Experiencias de una compañia que usa el I.E.W.

1. Deere & Co. con I.E.W.[3]

En Deere & Co. hay 39 grupos descentralizados para el desarrollo de sistemas de información y no hay normas generales para la compañia. Debido a la depresión en la industria agrícola, hubo que congelar la contratación en Deere & Co. desde 1979. Durante éste período, la adición de múltiples sistemas de gestión de bases de datos como (DB2, ORACLE y DBASE III).ha significado un incremento de la complejidad en las organizaciones.

Además Deere & Co. ha desarrollado su propia metodología de desarrollo basado en la metodología de la ingeniería de la información.

Las funciones del departamento de Planeación de Sistemas y Administración de Datos de Deere & Co. son :

1. Realizar un análisis preliminar de los requerimientos de los proyectos que implica compartir datos corporativos.
2. Mejorar la administración de los datos y su integración en el desarrollo de sistemas.
3. Orientar a la organización hacia la mejora del desarrollo de sistemas, soportando la metodología Deere & Co. con herramientas automatizadas.

3 Carma Mc Clure CASE La Automatización del Software Addison Wesley Iberoamericana

El principal reto productivo de Deere & Co. es conseguir un aumento de 30 % en la productividad del desarrollo del software. Se desea que menos personal, tenga más trabajo en menor tiempo. Se pretende poder trasladar más fácilmente a los analistas de un proyecto a otro. Se pretende realizar un trabajo mejor en la especificación de los requerimientos del sistema.

Deere & Co. eligio el IEW como ayuda para conseguir estos objetivos. Desde 1985 se han instalado diez copias de IEW en esta empresa, lo que supone un promedio de una computadora personal por cada 2 analistas. Se necesitaron 2 semanas para integrar el uso de IEW en la metodología propia de Derre & Co. El usuario principal de IEW en Deere &Co. es el analista de sistemas y el empleo principal es el diseño de sistemas. El promedio de tiempo para llegar a un nivel eficiente con las herramientas es de 2 días, asumiendo que el usuario está familiarizado con las técnicas de análisis estructurado.

Las herramientas automáticas de diagramación, logran que la diagramación del análisis sea práctica y ayudan a aumentar la comunicación con los usuarios. Además se informó que la comprobación en los diagramas estructurados fué el factor más importante del incremento de la productividad. La depuración temprana de los errores en el proceso de desarrollo simplificaba el trabajo de las últimas fases.

3 Carma Mc Clure CASE La Automatización del Software Addison Wesley Iberoamericana

Un aspecto importante en el uso de la herramienta CASE IEW, es la participación del usuario final en el desarrollo de las aplicaciones, mediante modelos lógicos de datos, es decir, la influencia del cliente en la participación de la aplicación es cada vez más activa, evitando los malos entendidos, perdida de tiempo y mejorando la comunicación con el analista, Además en algunas compañías, los usuarios finales construyen sus propios diagramas estructurados con herramientas automatizadas de diagramación de IEW.

Finalmente, la elección de la metodología de desarrollo correcta es la base sobre la que se asienta la estrategia de la alta productividad, ya que es la metodología la que guía el proceso de desarrollo del software.

Otras herramientas CASE de uso generalizado son :

EXCELERATOR de Index Technology Corporation : dirigida al diseño y la documentación de sistemas de información, maneja las metodologías estructuradas, como el análisis estructurado de DeMarco y el diseño estructurado de Yourdon.

APPLICATION FACTORY de Cortex Corporation : Su principal importancia radica en su generador automático de código, maneja prototipos iterativos, permitiendo desarrollar metodologías propias que se ajustan a las necesidades de la compañía.

CONCLUSION

El diseño orientado al flujo de datos es una metodología que utiliza las características del flujo de información para derivar la estructura del programa. Un diagrama de flujo de datos se convierte

en una estructura de programa usando una de dos técnicas de correspondencia - análisis de transformación o análisis de transacción.

El análisis de transformación se aplica a un flujo de información que exhibe unos límites claros para los datos entrantes y los salientes. El DFD se convierte en una estructura que asigna control a la entrada, al procesamiento y a la salida de tres jerarquías de modulos factorizadas por separado.

El análisis de transacción se aplica cuando cuando un elemento de información hace que el flujo se bifurque hacia uno de entre muchos caminos. El DFD se convierte en una estructura que asigna el control a una subestructura que toma y evalúa una transacción.

V DISPONIBILIDAD DE SISTEMAS DE INFORMACION CASE

Introducción

Actualmente, hay casi cien distribuidores en todo el mundo que ofrecen herramientas CASE. Además es una de las áreas de mayor crecimiento en la industria de las computadoras. En 1985 solamente se habían vendido unas miles de copias de herramientas CASE en Estados Unidos y fuera no había prácticamente mercado. El mercado CASE estaba dominado por pocos distribuidores.

En $1986,_1$ el numero de copias CASE llego a diez mil, el mercado alcanzó los 50 millones de dolares y se extendio a todo el mundo. Entraron nuevos distribuidores en el mercado, algunos de los cuales capturaron una parte sustancial del mismo.

En $1987,_2$ el mercado total de herramientas CASE fúe miles de dolares. Las compañias compraron varias copias de las herramientas CASE para su departamento de software.Muchas compañias han implementado la tecnología CASE y la han impuesto como base de su estrategia de productividad del software. En el futuro, las tasas de crecimiento del mercado de herramientas CASE se cree que serán de al menos un cien por ciento anual.

Aunque es probable que la tecnología CASE evolucione en el futuro hacia una tecnología mucho más potente y completa, en la actualidad ya tiene capacidades muy potentes. . Tres son, particularmente potentes presentes hoy en las herramientas CASE

1. CASE I integrado (del inglés, I-CASE)
2.- El depósito CASE.
3.- La generación automática de código

En los años noventa,[3] las herramientas CASE y las estaciones de trabajo para el desarrollo serán tan comunes para el desarrollo de software como lo han sido durante tres décadas los lenguajes de programación y los compiladores. La ingeniería de software asistida por ordenador tendrá una posición dominante entre las tecnologías de software.En el mercado actual, se pueden citar, entre otras, algunas herramientas distribuídas por fases de desarrollo:

1,2,3 Carma Mc Clure Case La Automatización del Software Addison-Wesley

I-CASE

La ingeniería de software asistida por computadora (CASE) está cambiando el enfoque en el mercado del desarrollo de software. Aunque se obtienen beneficios utilizando herramientas aisladas que realicen determinadas labores de la ingeniería del software.

Los beneficios del CASE integrado (del inglés, I-CASE) incluyen: (1) la transferencia fluida de información (modelos, programas, documentos,datos) entre herramientras y entre etapas de la ingeniería del software; (2) la reducción del esfuerzo requerido para realizar actividades de soporte como la gestión de la configuración del software, el control de calidad y la generación de documentos; (3) mejora de la coordinación entre los miembros del equipo que trabajan en un proyecto grande de software.

Sin embargo, el I-CASE también presenta retos significativos. La integración exige representaciones consistentes de la información, interfaces estandarizadas entre las herramientas, un mecanismo homogéneo para la comunicación entre el ingeniero de software y cada herramienta y un enfoque efectivo que permita al I-CASE ejecutarse sobre distintas plataformas hardware y sistemas operativos. Aunque se han propuesto soluciones a los problemas relacionados con estos retos, los entornos I-CASE están sólo en su fase de nacimiento.

La mayoría de los entornos I-CASE se diseñan para ser ejecutdos sobre sistemas operativos multitarea, de tal forma que se pueda trabajar

con varias herramientas simultáneamente. Por ejemplo, se puede compilar un módulo al mismo tiempo que se realizan modificaciones sobre el diseño de otro módulo.

Los mecanismos de comunicación gestionan la comunicación entre procesos, estableciendo colas de mensajes que permiten comunicarse a las diferentes herramientas. Por ejemplo, la realización de una tarea a cargo de la herramienta CASE A puede generar un "suceso" que active la herramienta CASE B.Para ejecutar B, se debe utilizar un mecanismo de arranque, pero para pasar información de A a B se requiere un mecanismo de comunicación.

El deposito CASE

El depósito es una "cosa" -una base de datos que funciona como centro de acumulación y almacenamiento de la información. El papel del ingeniero de software es el de interactuar con el depósito a través de herramientas CASE con las que está integrado.

El depósito de un sistema I-CASE es el conjunto de mecanismos y estructuras de datos que permiten la integración datos-herramienta y datos-datos.

Estándares de Deposito

Se están realizando muchos esfuerzos de estandarización de entornos I-CASE y de depósitos CASE. En Estados Unidos, varios estándares compiten por predominar. En Europa se ha adoptado uno sólo. Japón y otros estados orientales han adoptado otro diferente.
A continuación se describe brevemente uno de estos estándares.

Estándar de Diccionario de Recursos de información (IRDS), ANSI (X3.I38-1988). El único estándar oficialmente aprobado por el ANSI fue desarrollado originalmente como una definición estándar de dicciopnarios de requisitos. Se centra en la gestión de recursos de información corporativa. Este estándar ayuda a la creación de "puentes" entre herramientas complementarias, tales como las herramientas de análisis/diseño y los generadores de código, así como en la portabilidad de herramientas CASE entre diferentes plataformas

Además del. estándar comentado anteriormente, todos los grandes fabricantes de computadoras han propuesto " soluciones" para el I-CASE. A continuación se enumeran algunos de los entornos más conocidos.

DSEE (Apollo Hewlett-Packard

Cohesion (Digital E quipment Corporation

HP-Softbench (Hewlett-Packard)

AD/Cycle (IBM)

NSE (Sun Microsystems

Finalmente , la utilización de herramientas CASE ayuda en todas las tareas y en la mayoría de las actividades. Pero el beneficio real de estas herramientas no se alcanza hasta que están integradas, es decir, hasta que la información generada por una herramienta puede ser utilizada fácilmente por el resto.

Análisis y Diseño

Porkit*Workbench de McDonell-Douglas

DesignAid de Nastec

Analyst/Designer toolkit de Yourdon

Exelerator de Index Technology

POSE de Computer Systems Advisers

CONCLUSION

Lo más frecuente , al hablar de la reutilización del software es pensar en la reutilidad del código de programa, tanto completo como algunas de sus partes.Los paquetes de software son un ejemplo de reutilidad a nivel de programa. En el caso de los paquetes, el usuario toma el programa completo en su forma genérica, ajusta el programa por medio de los parámetros o cambia profundamente parte del programa para particularizarlo. Los paquetes son muy populares en áreas como nominas, banca, seguros, contabilidad, redes de comunicaciones, control de inventario

Quiza el mayor peligro con los paquetes de aplicaciones es la dificultad de mantenimiento. Muchas aplicaciones comerciales cambian

mucho con el tiempo y se hace necesario modificar los paquetes. Si las modificaciones se han de hacer en lenguajes como Cobol, Fortran o PL1, requieren de mucho trabajo. Para facilitar el mantenimiento se requiere una excelente documentación y un diseño claro del paquete. Estas características deben examinarse al adquirir el paquete.

VI. EL FUTURO DE CASE

Introducción

En la actualidad las tecnologías como CASE determinarán la arquitectura de las comutadoras, en donde el analista construye sus aplicaciones en la pantalla de una estación de trabajo y la máquina se encarga de la generación de código ejecutable. La máquina puede aplicar todos los sorteos, cruces de datos y validaciones necesarias; además puede automatizar varias de las tareas que consumian tiempo en el pasado. Un reto del CASE, es hacer el proceso de planeación , Análisis y Diseño tan amigable como para que el usuario final con un conocimiento básico, desarrolle sus propias aplicaciones. Los usuarios finales necesitan tener la habilidad de resolver sus problemas por medio de la computadora, desde un ingeniero haciendo cálculos complicados o un supervisor de linea implementando sus procesos hasta un alto ejecutivo tomando decisiones complejas, con la ayuda de información computarizada.

Las personas y la forma en que construyen sistemas

A medida que nos adentramos en el nuevo siglo, las tecnologías de sistemas y de software seguiran siendo un desafío para cada profesional y para cada compañia que construya sistemas basados en computadora.

Al avanzar las tecnologías de hardware y de software, cambia la estructura del lugar de trabajo.En lugar de utilizar una estación de

trabajo como herramienta, el hardware y el software se convierten en un ayudante, realizando tareas auxiliares, coordinando la comunicación hombre-hombre.

Si nos guiamos por el pasado, podemos decir que la gente en sí misma no va a cambiar. Sin embargo, las formas en que se comuniquen, el entorno en que trabajen, los métodos que utilicen, la disciplina que empleen y, por tanto, la cultura global de desarrollo de software, serán significativa y profundamente distintos.

Nuevas formas de la representación de la información

Utilizando una combinación de técnicas orientadas a los objetos y varias herramientas CASE sofisticadas para desarrollar prototipos de aplicaciones, aumentará drásticamente la importancia del cuadrante de asistencia al cliente. De hecho , es muy probable que según vaya cambiando el punto de vista secuencial por el evolutivo, los clientes y los usuarios se involucren mucho más en el proceso de ingeniería de software. Debido a lo anterior, el usuario se vedrá obligado a examinar los progresos regularmente,haciendo comentarios y sugerencias que sirvan de realimentación.. El incremento de participación del cliente, motivado por el enfoque evolutivo, puede llevar a una satisfacción mayor del usuario final y a una mayor calidad global dcel software.

La tecnología como conductor

Las personas que construyen y utilizan software, el proceso de ingeniería que se aplica y la información que se genera, se ven afectadas por los avances en la tecnología del software y del hardware. Históricamente, el hardware ha funcionado como un conductor de la tecnología informática.
La tecnología RISC (computadoras con conjunto reducido de instrucciones) ha evolucionado actualmente hasta la fase de producción, pero todavía no constituye un mercado maduro.

Los cambios reales podrían darse en otra dirección. El desarrollo de arquitecturas no tradicionales, por ejemplo procesadores ópticos, máquinas neuronales) pueden causar cambios radicales en el tipo de software que se construya. Debido a estos enfoques no tradicionales se encuentran en la primera fase del ciclo de 15 años, es difícil determinar cuál alcanzará la madurez, y es todavía más difícil predecir cómo cambiará el mundo del software para incorporarlos.

Mientras nos acercamos hacia el próximo milenio, el desafío - los ingenieros de software - tengan los conocimientos de desarrollar sistemas que mejoren la condición humana.

En la manera en que se haga más fácil la construcción de aplicaciones, estas serán más importantes para el usuario final enriqueciendo la planeación automatizada de la información. Es

necesario así un diseño limpio y estable el cual puede ser desarrollado con una diversidad de herramientas de desarrollo como las siguientes:

Ayudas de diseño de pantallas para el desarrollo rápido y sencillos de formatos de pantalla.

Ayudas de diseño de diálogo en pantalla: Es una extensión del diseño de pantallas que ayuda en el desarrollo del diálogo interactivo entre usuarios y máquina.

Generadores de reportes que hacen fácil y sencillo el diseño de reportes, impresiones y estructuras complejas.

Editores de diagramas de acción que permiten el desarrollo de diagramas de lógica complicada, para la construcción de los mismos sin ciclos abiertos, uso incorrecto de estructuras CASE o varios finales en una rutina..

La facilidad de conversión de los diagramas de acción a código en lenguajes específicos especialmente para cuarta generación.

Ayudas en soporte de decisiones tales como herramientas de hoja de cálculo para examinar y graficar datos multidimencionales, herramientas para la exploración con sentencias what-if.

Otras herramientas gráficas para el análisis del sistema con las que es posible convertir Estructuras gráficas en Estructuras de acción automáticamente.

IEW (Information Engineering Workbench/Workstation)
La navegación en base de datos ayuda en la transportación dentro de complejas bases de datos y como también en la generación de código relacionado con dicha navegación.
La ingeniería de software asistida por computadora (CASE) está cambiando el enfoque en el mercado del desarrollo de software. Aunque se obtienen beneficios utilizando herramientas aisladas que realizen determinadas labores de la ingeniería de software, la potencia real de CASE sólo `puede conseguirse a través de la integración

Una de las herramientas CASE más populares para desarrollar algunas fases en la generación de alguna aplicación es el **IEW**(Information Engineering Workbench/Workstation)
paquete de CASE .

IEW es una herramienta de trabajo que nos ayuda en el desarrollo de software asistido por computadora, consta de tres módulos básicos planeación, análisis y diseño de sistemas de información.

Cada herramienta consta de una serie de diagramas integrados para captura y revisión de información, la entrada de información al paquete es revisada lógicamente por una unidad del sistema conocida

como Knowledge Coodinator, y almacena en una base de conocimientos compartida conocida como Encyclopedia..

Los tres niveles de información que manejan las herramientas de la estación de trabajo, pueden dividirse como sigue:

** Planning Workstation (IEW/PWS) en la que se captura la información de alto nivel en la administración del negocio, para la definición de metas y factores críticos de éxito, ademas de que nos ofrece una perspectiva más amplia de la empresa, sus funciones, los datos y la información que requiere. Su propósito central es definir y dar prioridad a proyectos para estudios futuros de implementación.

** Analysis Workstation (IEW/AWS) Esta refina el proceso y la información del modelo. Se determina que procesos se requieren para que funcione una area determinada del negocio, como se interrelacionan dichos procesos y que datos estan involucrados.

** Design Workstation (IEW/DWS) Diseña la implementación de datos y procesos al nivel de máquina. Define como algunos procesos en áreas del negocio serán implementados en procedimientos específicos y cómo trabajan dichos procedimientos. El diseño de procedimientos requiere de participación directa del usuarios final

La información definida en una etapa previa de la estación de trabajo, puede ser utilizada en herramientas de más bajo nivel, por ejemplo, entidades y sus atributos pueden describirse en el módulo de

planeación y posteriormente refinarse en la fase de análisis; especificaciones muy detalladas de algunos procesos pueden ser diagramados en la parte de análisis y refinadas como código de programación en el módulo de diseño.

CONCLUSION

Hemos visto procedimientos de gestión y métodos técnicos, principios básicos y técnicas especializadas, actividades orientadas a la gente y tareas que se pueden automatizar, notaciones y herramientas CASE. Hemos visto que la medida, la diciplina, la preocupación continua por la calidad, dará como resultado un software que satisfaga las necesidades del cliente, eficiente y fácilmente mantenible; un software mejor.

Según nos adentramos en el nuevo siglo, las tecnologías de sistemas y de software seguirán siendo un desafío para cada profesional y para cada compañía que construya sistemas basados en computadora.

Los cambios en las diferentes tecnologías de ingeniería son realmente notables, pero, al mismo tiempo, el progreso avanza lentamente.

Cuando se toma la decisión para adoptar un nuevo método o una nueva herramienta.

CONCLUSIONES Y RECOMENDACIONES

Cada empresa que construye sistemas basados en computadora en los noventa se enfrenta a un dilema. El software requerido por los sistemas de alta tecnología se hace cada vez más complejo y el tamaño de los programas resultantes aumenta proporcionalmente. Hace tiempo , un programa con 100,000 líneas de código era considerado como una aplicación grande. Hoy en día, el programa medio para una aplicación sobre una computadora personal, suele tener un tamaño dos o tres veces mayor. Los programas construidos para utilizarse en diseño asistido por computadora, sistemas de información y cualquier otra aplicación industrial, exceden con frecuencia el millón de líneas de código.

El rápido crecimiento del tamaño medio de los programas nos acarrearía bastantes problemas sino fuera porque al aumentar el tamaño del programa, aumenta el número de gente que debe trabajar en él. La experiencia nos muestra que al aumentar la cantidad de gente que debe trabajar en un proyecto de software, se resiente la productividad media del grupo de trabajo. Una forma de solucionar este problemaes crear varios equipos de trabajo, con especialistas en las diferentes fases del desarrollo de la aplicación. Sin embargo, según aumenta el número de equipos de trabajo, la comunicación entre ellos se puede dificultar y lleva más tiempo, al igual que la comunicación entre individuos y se corre el riesgo de perder la información importante.

Si la comunidad informática quiere resolver el dilema de la comunicación, las perspectivas futuras deben incluir cambios radicales en la forma en que los individuos y los equipos se comunican entre sí.

Al avanzar las tecnologías de hardware y de software, cambia la estructura del lugar de trabajo. En lugar de utilizar una estación de trabajo como herramienta, el hardware y el software se convierten en un ayudante, realizando tareas auxiliares, coordinando la comunicación hombre-hombre y en algunos casos, aplicando un conocimiento específico del campo para potenciar la capacidad del ingeniero.

B I B L I O G R F I A

- CASE LA AUTOMATIZACION DEL SOFTWARE

  CARMA MC. CLURE

  ADDISON-WESLEY IBEROAMERICANA 1993

- METODOLOGIAS DE DESARROLLO

  PRODUCCION AUTOMATICA DE SOFTWARE CON HERRAMIENTAS CASE

  ANTONIO LOPEZ FUENSALIDA

  MACROBIT 1990

- COMPUTER-AIDED SOFTWARE ENGINEERING

  The methodologies. the products. and the future

  CHRIS GANE

  PRENTICE HALL, ENGLEWOOD, CLIFFS. NEW JERSEY 1991

- LOS EXPERTOS EN SISTEMAS

  JESUS PEREZ PEREGRINO

  LIMUSA 1986

- TEORIA GENERAL DE SISTEMAS

  BERTALANFFY LUDWIG VON.

  FONDO DE CULTURA ECONOMICA 1990