ASEGURAMIENTO DE LA CALIDAD MEDIANTE INGENIERÍA DE SOFTWARE

16

OBJETIVO DE APRENDIZAJE

Una vez que haya dominado el material de este capitulo, podrá:

1. **Reconocer la importancia de adoptar un enfoque de calidad total para todo el SDLC.**
2. **Crear diagrama de estructura para diseñar sistemas modulados con un enfoque descendente (de arriba abajo).**
3. **Usar diversas técnicas para mejorar la calidad del diseño y mantenimiento del software.**
4. **Entender la importancia de ejecutar una variedad de pruebas durante el desarrollo de sistemas para identificar problemas desconocidos.**

La calidad ha sido durante mucho tiempo una preocupación para las empresas, como lo debe ser para los analistas de sistemas en el análisis y diseño de sistemas de información. Es demasiado arriesgado emprender todo el proceso de análisis y diseño sin usar un elfoque de aseguramiento de la calidad. Los tres enfoques para el aseguramiento de la calidad mediante ingeniería de software son: (1) garantizar el aseguramiento de la calidad total diseñando sistemas y software con un enfoque modular, descendente (de arriba abajo); (2) documentar el software con las herramientas adecuadas, y (3) probar, mantener y auditar el software.

Dos propósitos guían el aseguramiento de la calidad. El primero es que el usuario del sistema de información es el factor individual más importante en establecer y evaluar su calidad. El segundo es que mucho menos costoso corregir los problemas en sus fases iniciales que esperar hasta que un problema se manifiesta a trabes de las quejas o crisis del usuario.

Ya hemos aprendido acerca de la enorme inversión de mano de obra y otros recursos empresariales que se requieren para desarrollar con éxito un sistema. El uso del aseguramiento de la calidad a lo largo del proceso es una forma de reducir los riesgos, ayuda a garantizar que el sistema resultante será lo que se necesita y de sea, mejorara notablemente algún aspecto del desempeño del negocio. Este capitulo proporciona al análisis tres enfoques principales para la calidad.

ENFOQUE DE ADMINISTRACIÓN DE LA CALIDAD TOTAL

La administración de la calidad total (TQM, por sus siglas en ingles) es esencial a lo Largo de todos los pasos del desarrollo de sistemas. Según Dean y Evans (1994), los principales elementos de la TQM solo son significativos cuando se presentan en un con-

Texto organicional que favorece un esfuerzo integral por la calidad. Es en este contexto donde los elemento de enfoque en el cliente, planificación estratégica y liderazgo, mejora continua, facultar al empleado y trabajo en equipo se unifican con el propósito de cambiar el compartimiento de las empleados y, en consecuencia, el curso de la organización. Observe que el concepto de calidad se ha ampliado con el paso de los años para reflejar un enfoque en toda la organización, y no tan solo en la producción. En lugar de concebir a la calidad como un control del número de artículos defectuosos que se producen, ahora se considera como un proceso evolutivo hacia la perfección que se denomina administración de la calidad total.
Los análisis de sistemas deben estar concientes de los factores que despiertan el interés en la calidad. Es importante comprender que el creciente compromiso de las empresas hacia la TQM encaja sumamente bien en los objetivos generales del análisis y diseño de sistemas.

SEIS SIGMA

La llegada de seis sigmas ha cambiado el enfoque de la administración de la calidad. Cada analista de sistema necesita estar conciente de seis sigmas y aplicar de los principios a sus proyectos de análisis de sistemas. Originalmente desarrollado por motorala en la década de 1980, seis sigmas es más que una metodología; es una cultura basada en la calidad. La meta de seis sigmas es eliminar todo los defectos. Estos se explican a cualquier producto, servicio o proceso. En los libros de texto de administración de operaciones que se publicaron a partir de la década de 1970 y hasta fines del siglo pasado, el control de calidad se expreso en términos de tres desviaciones estándar de la media, o tres sigma lo cual es equivalente aproximadamente 67,000 defectos por millón de oportunidades. Seis sigmas implican una meta de solo 3.4 defectos por millón de oportunidades.

Seis sigmas es un enfoque descendente de arriba abajo. Se requiere que un CEO adopte la filosofía y un ejecutivo funja como campeón de proyecto. Un líder de proyecto de seis sigmas se denomina ***Black Belt*** (cinta negra). Las personas escogidas para ser ***Black Belts*** pueden provenir de diferentes niveles e incluso diferentes niveles salariales, pero deben tener experiencias en el proyecto y contar con capacitación especial. Los ***Black Belts*** se certifican después que han liderado proyectos de manera exitosa. Los miembros del proyecto se denominan ***Green belts*** (cinta verde). Los ***black belts*** maestros son los ***black belts*** que han trabajado en muchos proyectos y están disponibles como un recurso para los equipos de proyectos. (La metáfora de ***Black Belt*** viene de sistema de clasificación de capacidades en las artes marciales. resalta la importancia de la disciplina en todos los ámbitos).

Seis sigmas se pueden resumir como una metodología. En la figura 16.1 se muestran los pasos de seis sigmas. Sin embargo, seis sigma es mucho más que una metodología; es una filosofía y una cultura.

Para mas información sobre seis sigma y administración de la calidad, visite el sitio Web del juran Center en la carlson School of management de la university of minnesota en twin cities (www.csom.umn.edu). En 2002 el juran center emitió un manifiesto para apoyar y fomentar la calidad. Los autores de este libro firmaron el manifiesto en ese momento y sinceramente estamos de acuerdo con sus principios.

Joseph M. juran dijo: “Toda mejora de la calidad ocurre proyecto tras proyecto y de ninguna otra forma” (juran, 1964). Los análisis de sistemas y gerentes de proyecto deben tomar muy en serio esta afirmación.

RESPONSABILIDAD DE LA ADMINISTRACIÓN DE LA CALIDAD TOTAL

En términos prácticos, gran parte de la responsabilidad por la calidad de los sistemas de información recae en los usuarios de éstos y en los directivos. Para que la TQM se vuelva una realidad en los proyectos de sistemas, deben darse dos condiciones. Primero, debe existir un apoyo organicional incondicional por parte de los directivos, lo cual es distinto a simplemente respaldar el nuevo proyecto de los directivos. Este apoyo significa establecer un contexto para que los directivos consideren seriamente cómo afecta su trabajo la calidad de los sistemas de información y la información misma.

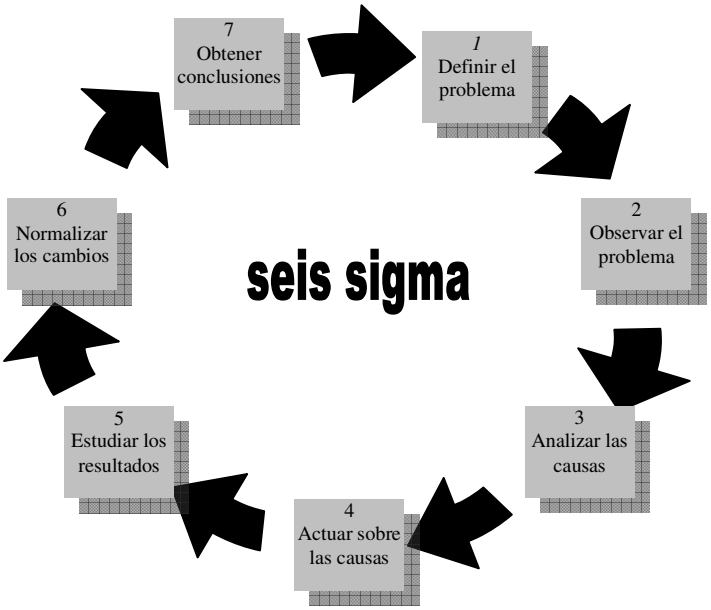


FIGURA 16.1

Cada analista de sistemás debe entender la metodología y filosofía de seis sigmas.

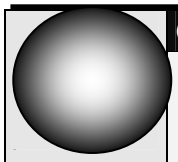
Es necesario que tanto el analista como la empresa se comprometan desde el principio con la calidad para lograr le meta de calidad. Este compromiso da como resultado un esfuerzo uniformemente controlado hacia la calidad durante todo el ciclo de vida del desarrollo de sistemas, y esta en marcado contraste con tener que dedicar gran calidad de esfuerzo para resolver problemas al fin del proyecto.

El apoyo organicional para conseguir calidad en sistemas de información de administración se puede lograr al proporcional el tiempo en el trabajo para los círculos de calidad de SI, los cuales consisten de seis a ocho pares organizacionales específicamente responsable de considerar cómo mejorar los sistemas de información y como implementar las mejoras.mediante el trabajo en los círculos de calidad de SI o a través de otro mecanismos ya colocados, la administración y usuarios deben desarrollar lineamientos para los estándares de calidad de sistemas de información, preferentemente los estándares se diseñaran cada vez que un nuevo sistema o una modificación mayor se proponen formalmente para el equipo de análisis de sistemas.

No es fácil crear los estándares de calidad, pero es posible y se ha hecho. Parte del trabajo del analista de sistema es alentar a usuarios a que cristalicen sus expectativas acerca de los sistemas de información y sus interacciones con éstos.

Los estándares de calidad departamentales se deben comunicar mediante retroalimentación para el equipo de análisis de sistemas. Normalmente el equipo esta sorprendido por lo que se ha desarrollado. Las expectativas generalmente son menos complejas de lo que los analista experimentados saben se podría hacer con un sistema. Además los problemas humanos que se han omitido o menospreciado por el equipo del

Analista se podrían diseñar como extremadamente urgentes en los estándares de calidad para los sistemas de información ayudaran al analista a evitar errores costosos en el desarrollo de sistemas no desarrollado o innecesario.



OPORTUNIDADES DE CONSUTARIA 16:1

LA CALIDAD DE MIS NO ES OBLIGATORIA

ªMerle, ven aquí y echa un vistazo a estos informes de fin de semanas, suplica portia. Como uno de los gerentes del comité de aseguración de la calidad/ grupo de trabajo de SI, compuesto por seis integrantes, portia ha estado examinado la salida del sistema producida por el prototipo para su departamento de marketing. El equipo de análisis de sistemas le ha permitido que revise la salida.

ªMerle Chat se encamina hacia el escritorio de portia y da una mirada a los documentos que ella le extiende. ª ¿Por qué?, ¿Cuál es el problema?ª, pregunta. ªA mi me parece que están bien. Creo que le estas dando demasía- da importancia a esta grupo de trabajo. Se supone que también debemos hacer nuestro otro trabajo, ya sabes, ª Merle da la vuelta y regresa a su escritorio ligeramente perturbado por la interrupción.

ª Merle, ten un poco de compasión. Es verdaderamente tonto soportar estos informes tal como están. No puedo encontrar nada de lo que necesito, y se supone que tengo que indicarles a todos los demás en el departamento qué parte del informe deben leer. Por una parte, estoy decepcionada. Este informe es descuidado. No le encuentro ningún sentido. Es tan solo una copia de la salida que estamos recibiendo ahora. De hecho, parece peor. Lo voy a presentar en la próxima reunión del grupo de tra- bajo, manifiesta insistentemente portia.

Merle voltea a verla y dice: la calidad es su responsabilidad, portia. Si el sistema no esta dándonos buenos informes, ellos lo agregaran cuando Todo esté junto. Nada más estás provocando problemas. Estas actuando como si ellos valoraran realmente nuestra información. Yo no le dedicaría tiempo de mi día, mucho menos haría su trabajo. Ellos son inteligentes, dales la oportunidad de que deduzcan lo que necesitamos.

Portia mira a Merle inexpresivamente, y poco a poco comienza a enfadarse, dice .tú has asistido a cuatros reuniones. Nosotros somos los únicos que conocemos el negocio. La idea esencial de TQM es decirles lo que necesitamos, entonces no podemos quejarnos. Esto lo plantearé la próxima vez que nos reunamos. ª

¿Qué tan eficaz cree que será merle para comunicar sus normas de calidad al equipo de análisis de sistemas y a los miembros del grupo de trabajo de SI? Responda en un párrafo. ¿si los analistas de sistemas pueden percibir la reunión de merle para trabajar con el grupo de trabajo en el desarrollo de las normas de calidad, qué le diría para convencerlo de la importancia de la participación de los usuarios en la TQM? haga una lista de argumentos que apoye el uso de la TQM. ¿Cómo puede responder el equipo de análisis de sistemas a las inquietudes que plantea portia? explique su respuesta en un párrafo.

REPASO ESTRUCTURADO

Una de las acciones de administración de la calidad mas eficaces que puede emprender un equipo de análisis de sistemas es hacer repasos estructurados de manera rutinaria. Los repasos estructurados son una forma de usar expertos para monitorial la programación y el desarrollo general del sistema, señalar los problemas y permitir al programador o analista responsable de dicha parte del sistema hacer los cambios correspondientes.

Los repasos estructurados involucran por lo menos a cuatros personas: la persona responsable de la parte del sistema o subsistema que se revisará (un programador o analista) o coordinador del repaso, un programador o analista experto y un experto que toma notas acerca de las sugerencias.

Cada persona que participa en un repaso tiene un papel especial que cumplir. El coordinador se encarga de asegurar que otros cumplan los papeles que se le asigne y de que realicen las actividades establecidas. El programador o analista Este para escuchar, no para defender su punto de vista, racionalizar o discutir un problema. El programador o analista experto tiene que señalar los errores o problemas potenciales, sin especificar como se debe resolver. El tomador de notas registra lo que se dice con el fin de que los demás participantes puedan interactuar sin ningún problema.

Los repasos estructurados se pueden hacer siempre que una parte de la codificación, de un subsistema o de un sistema esté terminada. Simplemente asegurarse de que el subsistema bajo revisión sea comprensible fuera del contexto al que pertenece. Los repasos estructurados son especialmente adecuados en un enfoque de administración de la calidad total cuando se realiza durante todo el ciclo de vista de desarrollo de sistemas. El tiempo para realizarlos debe ser de media hora a una hora cuando mucho, lo cual implica que debe coordinarse muy bien. En la figura 16.2 se muestra un formulario útil para organizar el repaso estructurado e informar sus resultados. Debido a que el repasos estructurado tomas tiempo, no abuse de ellos.

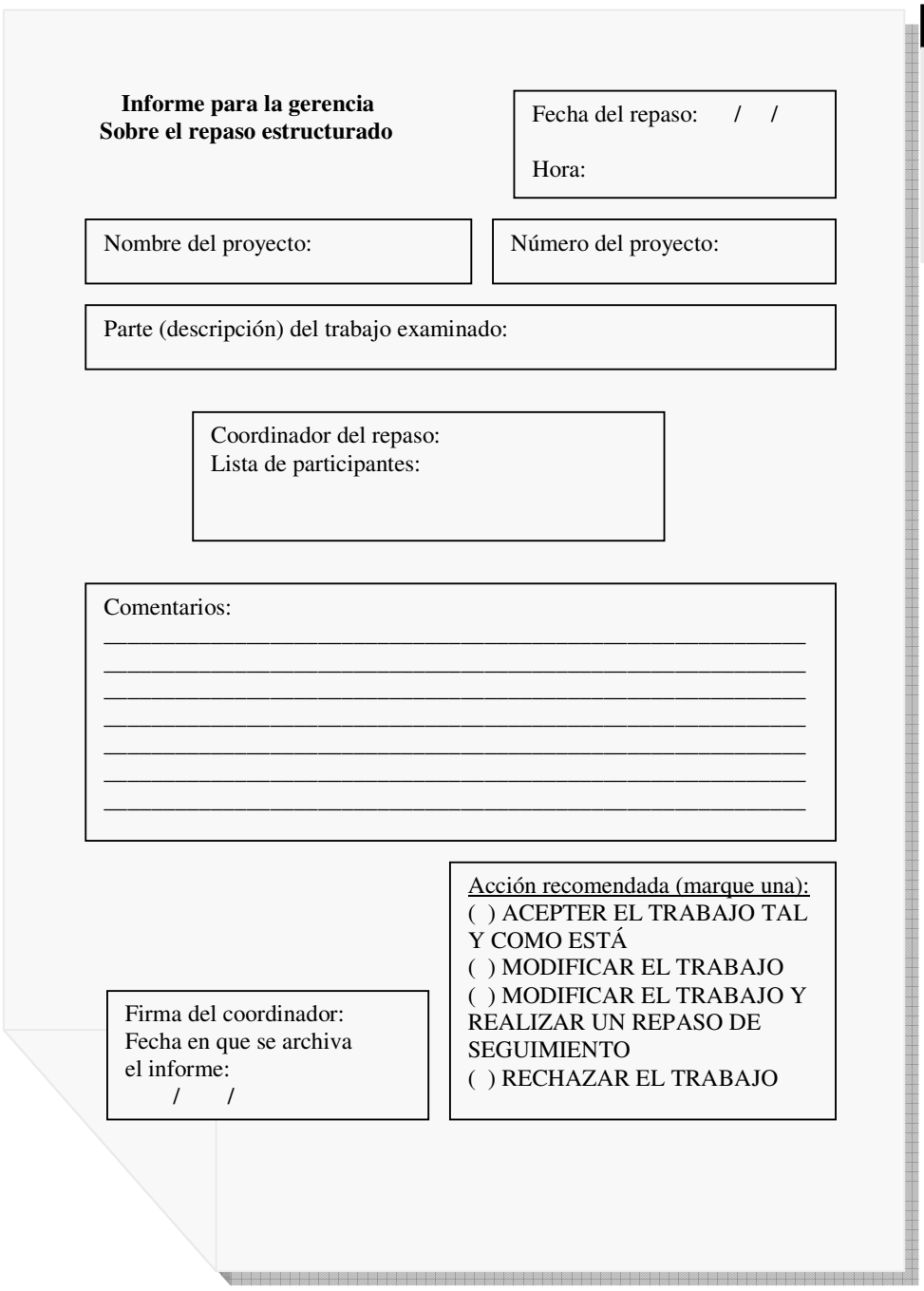


FIGURA 16.2

Formulario para documentar repasos estructurados; los repasos se pueden hacer siempre que se finalice una parte de la codificación, de un sistema o de un subáistema.

Utilice los repasos estructurados para obtener (y después emprender acciones acordes con) retroalimentación valiosa desde una perspectiva que le falte. Al igual que con todas las medidas de aseguramiento de la calidad, el propósito de los repasos es evaluar el producto sistemáticamente de manera continua en lugar de esperar hasta la terminación del sistema.

DISEÑO Y DESARROLLO DE SISTEMAS

En esta sección definimos los diseños de sistemas ascendente (de abajo a arriba o *bottom-up)* y descendente (de arriba abajo o *top-down),* así como también el enfoque modular para la programación. Discutimos las ventanas de cada uno, así como también las preocupaciones que se deben observar al emplear un enfoque descendente o uno modular.

También discutimos las propiedades de los enfoques descendente y modular para ayudar en el aseguramiento de la calidad de los proyectos de sistemas.

Diseño ascendente este diseño se refiere a identificar los procesos que necesitan computarizarse conformes surgen, analizarlos como sistemas y codificar los procesos o comprar software empaquetado para resolver el problema inmediato. Los problemas que requieren computarizarse normalmente se encuentran en el nivel mas bajo de la organizacion. Con frecuencia este tipo de problemas son estructurados y por lo tanto son más sensibles a la computación; también son los más rentables. Por lo tanto, el nombre *ascendente* se refiere al nivel inferior en el cual se introduce primero la computación. Por ejemplo, con frecuencia los negocios toman este enfoque para el desarrollo de sistemás al adquirir software comercial para la contabilidad, un paquete diferente para la programación de producción y otros para el marketing.

Cuando la programación interna se hace con un enfoque de ascendente, es difícil interconectar los subsistemas de manera que se desempeñen fácilmente como sistema. Es muy costoso corregir las fallas de la interconexión y muchas de ellas no se descubren, sino hasta que se completa la programación, cuando los analistas intentan reunir el sistema en la fecha limite señalada para la entrega. En esta situación, hay poco tiempo, presupuesto o paciencia del usuario para la depuración de las interconexiones delicadas que se han ignorado.

Aunque cada subsistema aparenta conseguir lo que quieren, al considerar el sistema global hay series limitantes para tomar un enfoque de ascendente. Una es que hay duplicidad de fuerza software e incluso introducir los datos. Otra es que se introducen datos inválidos en el sistema. Una tercera, y quizás las desventajas mas serias del enfoque ascendente, es que no se consideran los objetivos organicionales globales, y por lo tanto dicho objetivos no se pueden cumplir.

Diseño descendente es fácil visualizar este enfoque: como se muestra en la figura 16.3 significa ver una descripción amplia del sistema y después dividirla en partes más pequeñas o subsistemas. El diseña descendente permite a los analista de sistemas determinar primero los objetivos organicionales globales, así como también determinar como se reúnen mejor en un sistemas global. Después el analista divide dicho sistemas en subsistemas y requerimientos.

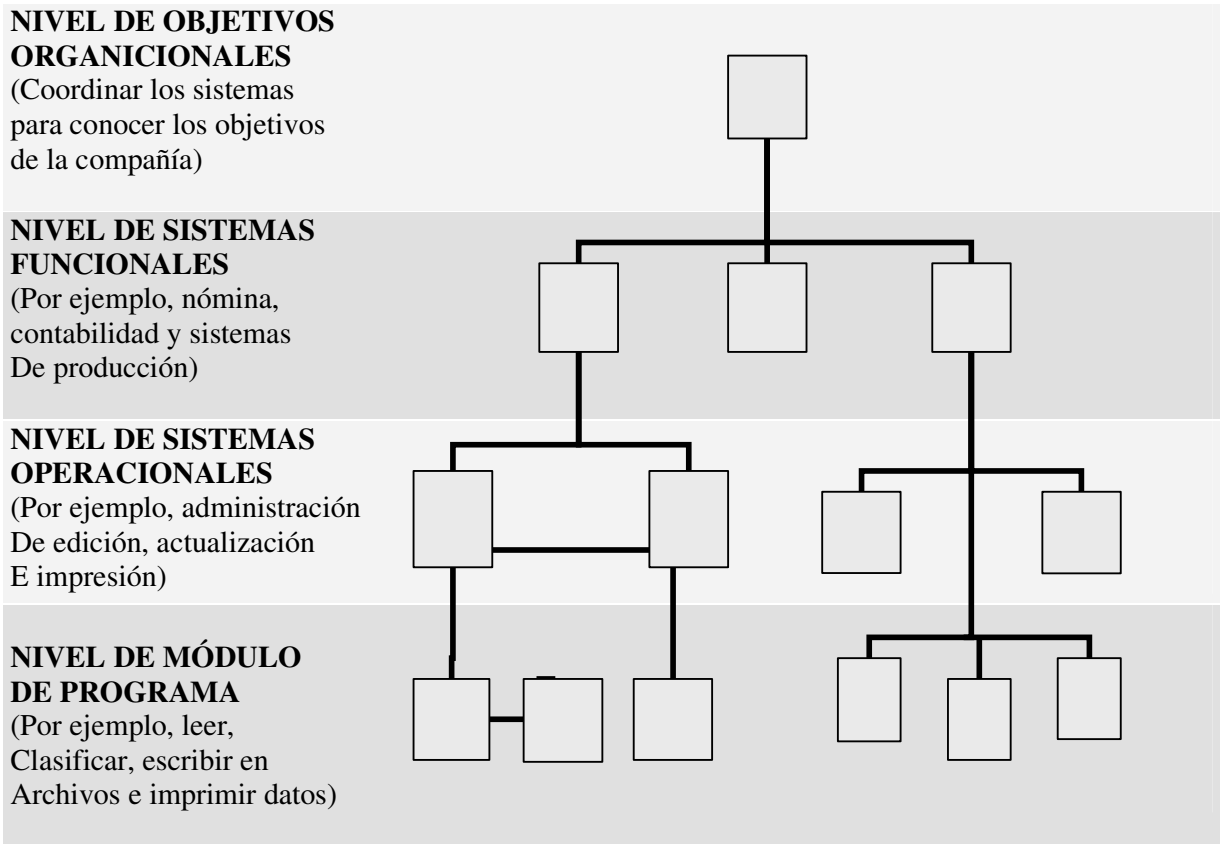
El diseño descendente es compatible con el pensamiento general de sistema que se discutió en el capitulo 2. Cuando los analistas de sistemas utilizan un enfoque descendente están pensando en que las interrelaciones e interdependencia de subsistemas se adaptan a la organización de existencia. el enfoque descendente también proporciona el énfasis deseable en la colaboración o interconexiones que los sistemas y sus subtemas necesitan, cual falta en el enfoque ascendente.

Las ventanas de usar un enfoque descendente para el diseño de sistemas incluyen evitar el caos de intentar diseñar un sistema de repente. Como hemos visto, planear e implementar sistemas de información y administración es increíblemente complejo intentar colocar todos los subsistemas en un lugar ejecutable enseguida es casi un fracaso seguro.

Una segunda ventana de tomar un enfoque descendente para diseñar es que permiten separar a los equipos de análisis de sistemas para trabajar en paralelo en diferentes subsistemas, lo cual puede ahorrar mucho tiempo. El uso del equipo para el diseño de subsistemas se ajusta particularmente bien a un enfoque de control de calidad total.

FIGURA 16.3

Uso de un enfoque descendente para determinar primero los objetivos organicionales generales.



Una tercera ventaja es que un enfoque descendente evita un problema mayor asociado con un enfoque ascendente; evitar que los analistas de sistemas se metan tanto en los detalles que pierdan de vista lo que se supone que el sistema hace.

Hay algunas dificultades con el diseño descendente que el analista de sistema necesita saber. La primera es el riesgo de que el sistema se divida en subsistemas ªerroresª. Se debe por atención a las necesidades que se traslapen y a la comparticion de recursos de manera que la partición en subsistemas tenga sentido para todos los sistemas. Además, es importante que cada subsistema solucione el problema correcto.

Un segundo riesgo es que una vez que se hacen las decisiones de un subsistema, sus interfaces se podrían descuidar o ignorar. Es necesario destallar de quien es la responsabilidad de las interfaces.

Una tercera advertencia es acompaña al uso de un diseño descendente es que los subsistemas se deben reintegrar eventualmente. Los mecanismos para la reintegración se necesitan poner en funcionamiento desde el principio. Una sugerencia es negociar información regular entres los equipos del subsistema; otra es usar herramientas que permiten flexibilidad si se requieren cambios para los subsistemas interrelacionados.

La administración de calidad total y el enfoque descendente para diseñar puede estar relacionada. El enfoque descendente proporciona el grupo de sistemas con una decisión más natural de usuarios en grupos de trabajos para subsistemas. Los grupos de trabajos establecidos de esa forma después pueden servir a una función dual como círculos de calidad para el sistema de información de administración. La estructura necesaria para el control de calidad está en el lugar, así como la motivación apropiada para obtener el subsistema para lograr las metas departamentales que son importantes para los usuarios involucrados.

DESARROLLO MODULAR

Una vez que se toma el enfoque del diseño, el enfoque modular es útil en la programación. Este enfoque implica dividir la programación en partes lógicas y manejables llamada módulos. Este tipo de programación funciona bien con el diseño descendente por da énfasis a las interfaces entre los módulos y no los descuida hasta el final del desarrollo de sistema. Idealmente, cada modulo individual debe ser funcionalmente cohesivo de manera que encargue de realizar una sola función.

El diseño de progre modular tienes tres ventajas principales. Primero, los módulos son más fáciles de escribir y de depurar porque prácticamente son independientes. Rastrear un error en un modulo es menos complicado, debido a que un problema en un modelo no debe causar problemas en otros.

Una segunda ventaja del diseño modular es que los módulos son más fáciles de mantener. Normalmente las modificaciones se limitarán a unos módulos y no seguirán en todo el problema.

Una tercera ventaja del diseño modular es que los módulos son más fáciles de entender, debido a que son subsistemas independientes. Por lo tanto, un lector puede adquirir una lista del código de un modulo y entender su función.

Algunos lineamientos para la programación modular incluyen lo siguiente:

1. mantener cada modulo de un tamaño manejable (incluir a la perfección una solo función).
2. poner particular atención a la interfaces críticas (los datos y variables de control que se pasan a otros módulos).
3. minimizar el numero de modulo que el usuario debe modificar al hacer los cambios.
4. mantener las relaciones jerárquicas establecidas en las fases descendentes.

MODULARIDAD EN EL ENTORNO DE WINDOWS

La modularidad se está volviendo muy importante. Microsoft desarrolló dos sistemas para vincular los programas en un entorno de Windows. El primero se llama intercambio dinámico de datos (DDE), el cual comparte códigos al usar archivos de bibliotecas de vínculos dinámicos (DLL). Al usar DDE, un usuario puede almacenar datos de un programa quizás en una hoja de cálculos tal como Excel— y después usar dichos datos en otros programas, por decir, en un paquete de procesamiento de texto tal como Word para Windows. El programa que continúe los datos originales se denominan servidor y el programa que los usa se llama cliente, los datos se actualicen automáticamente y se refleje los cambios hechos en los archivos de hoja de cálculos del servidor desde la última vez que se abrió dicho archivo de procesamiento de texto. (Véase el capitulo 17 para una discusión amplia del modelo/servidor).

Uno de los archivos de la DLL normalmente usados es COMMDLG.DLL, el cual contiene los cuadros de diálogos de Windows para **Abrir Archivos, Guardar Archivos, Buscar** e **Imprimir.** Una ventaja de usar este archivo es que los programas tendrán la misma apariencia y funcionamiento que otros programas de Windows. También acelera el desarrollo, debido a que los programadores no tienen que escribir el código contenido en los archivos DLL más comunes.

Un segundo enfoque para vincular programas en Windows se denomina vinculación e incrustación de objetos (OLE). Este método de vincular programas es superior a DDE porque está ligado a los datos y gráficos de la aplicación. Mientras que DDE utiliza un enfoque de cortar y pagar para vincular datos y no retiene el formato, OLE retiene todas las propiedades de los datos creados originalmente. Este enfoque orientado a objetos (Véase en el capitulo 18 para una discusión de principios orientados a objetos) permite al usuario final pertenecer a la aplicación del cliente y evitar los datos originales en la aplicación del servidor. Con OLE, cuando un usuario final hace clic en el objeto incrustado, se despliega una barra de herramienta que permite la edición visual.

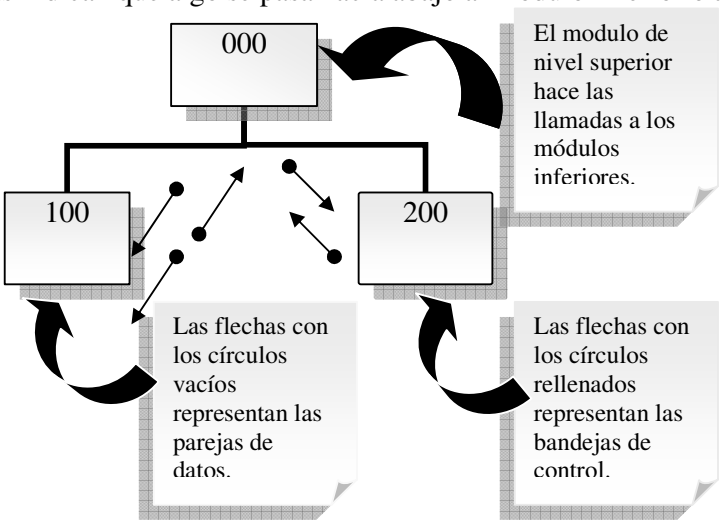
……..

USO DE DIAGRAMAS DE ESTRUCTURA PARA DISEÑAR SISTEMAS

Las herramientas recomendadas para diseñar un sistema modular descendente se denomina diagrama de estructura. Este grafico simplemente es un diagrama que consiste de cuadros rectangulares, los cuales representan los módulos, y de flechas de conexión.

La figura 16.4 muestra tres módulos que se etiquetan como 000, 100 y 200 y se conectan mediante líneas de ángulos rectos. Los módulos del nivel superior se numeran por 100s o 1,000s y los módulos de nivel inferior se numeran por 10s o 100s. Esta enumeración permite a programadores insertar módulos que se usan un número entre los números de módulos adyacentes. Por ejemplo, un modulo insertado entre los módulos 110 y 120 recibiría el numero 115. Si se insertarán dos módulos, los números podrían ser 114 y 117. Estos esquemas de numeración varían, dependiendo de los estándares organicionales usados.

A los datos de las líneas de conexión, se dibujan dos tipos de flechas. Las flechas con los círculos vacíos se denominan parejas de datos y las flechas con los círculos rellenados se denominan bandejas de control o interruptores. Un interruptor es lo mismo que una bandeja de control acepto por que está limitado por dos valores: si o no. Esta flechas indican que algo se pasa hacia abajo al modulo inferior o arriba al superior.



FIRURA 16.4

Un diagrama de estructura muestre el control que se mueve de forma descendente y también muestras los módulos no funcionales.

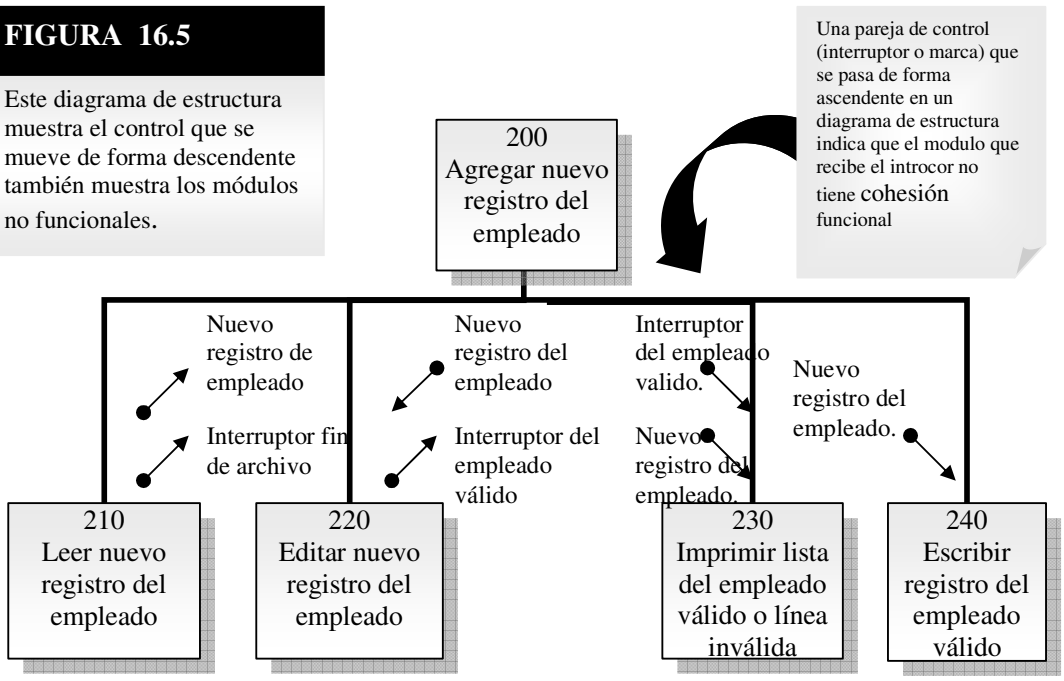
El analista debe mantener a la perfeccionaste acoplamiento al mínimo. Cuando hay pocas parejas de datos y bandejas de control en el sistema, lo más fácil es cambiar el sistema. Cuando finalmente se programan estos módulos, es importante pasar el menor número de parejas de datos entre los módulos.

Aun mas importante es que se debe evitar las bandejas de control numerosas. El control se diseña para ser pasado de los módulos del nivel inferior a los del nivel superior en la estructura. Sin embargo, en rara ocasiones será necesario pasar el control hacia abajo en la estructura. Las bandejas de control deciden qué parte de un modelo se ejecuta y están asociado con las instrucciones IF…THEN…ELSE… y otros tipo similares de instrucciones. Cuando el control se pasa en forma descendente, se permiten que un modulo de nivel inferior tome una decisión y el resultado en un modulo que decenpeña dos tareas diferentes. Este resultado rompe con el modelo de modulo funcional: un modulo solo debe desempeñar una tarea.

La figura ilustra que parte de un diagrama de estructura para agregar nuevos empleados. El programa lee un archivo de TRANSACCION DE EMPLEADO y verifica que cada registro en el archivo únicamente tenga datos aceptables. Los informes se imprimen por separados para los registros válidos e inválidos, proporcionando un rasgo para auditoria de todas las transacciones.

FIGURA 16.5

Este diagrama de estructura muestra el control que se mueve de forma descendente también muestra los módulos no funcionales.



El informe que contiene los registros inválidos se envía al usuario para la corrección de errores. Los registros que son validos se ponen en un archivo de transacción válido, el cual se pasa a un programa separado para actualizar el archivo MAESTRO DE EMPLEADOS. El modulo 200, AGREGA NUEVOS REGISTRO DEL EMPLEADO, representa la lógica de agregar un registro. Debido a que el modulo 230 se usa para imprimir ambos informes, se debe enviar una bandera de control hacia abajo para indicar al modulo que informe imprimir. De esta manera la lógica del modulo 230 se controla por completo mediante una instrucción IF, la cual se ilustra en la figura 16.6.

La figura 16.7 muestra la forma correcta de diseñar la estructura por debajo del modulo 200, AGREGAR NUEVO REGISTRO DEL EMPLEADO. Aquí, cada función de impresión se ha puesto en un modelo separado y las banderas de control solo se pasan a la estructura al modulo del nivel superior.

FIGURA 16.6

El pseudocódigo para el módulo 230 ilustra el efecto de pasar de forma descendente un interruptor.

```
Modulo 230─imprimir lista del empleado válida o línea inválida

IF INTERRUPTOR DEL EMPLEADO VÁLIDO = ºYº

        Mover NÚMERO DEL EMPLEADO a LISTA DEL EMPLEADO VÁLIDA
        Mover NOMBRE DEL EMPLEADO a LISTA DEL EMPLEADO VÁLIDA
        Mover TRABAJO DEL EMPLEADO a LISTA DEL EMPLEADO VÁLIDA
        Mover DEPARTAMENTO DEL EMPLEADO a LISTA DEL EMPLEADO VÁLIDA
        Mover NÚMERO TELEFÓNICO DEL TRABAJO DEL EMPLEADO a LISTA DEL
            EMPLEADO VÁLIDA
        Mover FECHA DE CONTRATO DEL EMPLEADO a LISTA DEL EMPLEADO VÁLIDA DO
        IMPRIMIR LISTA DEL EMPLEADO VÁLIDA
ELSE
        Mover NÚMERO DEL EMPLEADO a LINEA INVÁLIDA
        Mover NOMBRE DEL EMPLEADO a LINEA INVÁLIDA
        Mover TRABAJO DEL EMPLEADO a LINEA INVÁLIDA
        Mover DEPARTAMENTO DEL EMPLEADO a LINEA INVÁLIDA
        Mover NÚMERO TELEFONICOO DEL TRABAJO DEL EMPLEADO a LINEA INVÁLIDA
        Mover FECHA DE CONTRATO DEL EMPLEADO a LINEA INVÁLIDA
        DO IMPRIMIR LINEA INVÁLIDA
ENDIF
```

También se debe examinar los datos que se pasan a través de las parejas de datos. Es mejor pasar sólo los datos requeridos para realizar la función del módulo. Este enfoque se denomina acoplamiento de los datos. El paso excesivo de datos se denomina acoplamiento de sello, y aunque es relativamente imnofencivo, reduce las posibilidades de crear un módulo reutilizable.

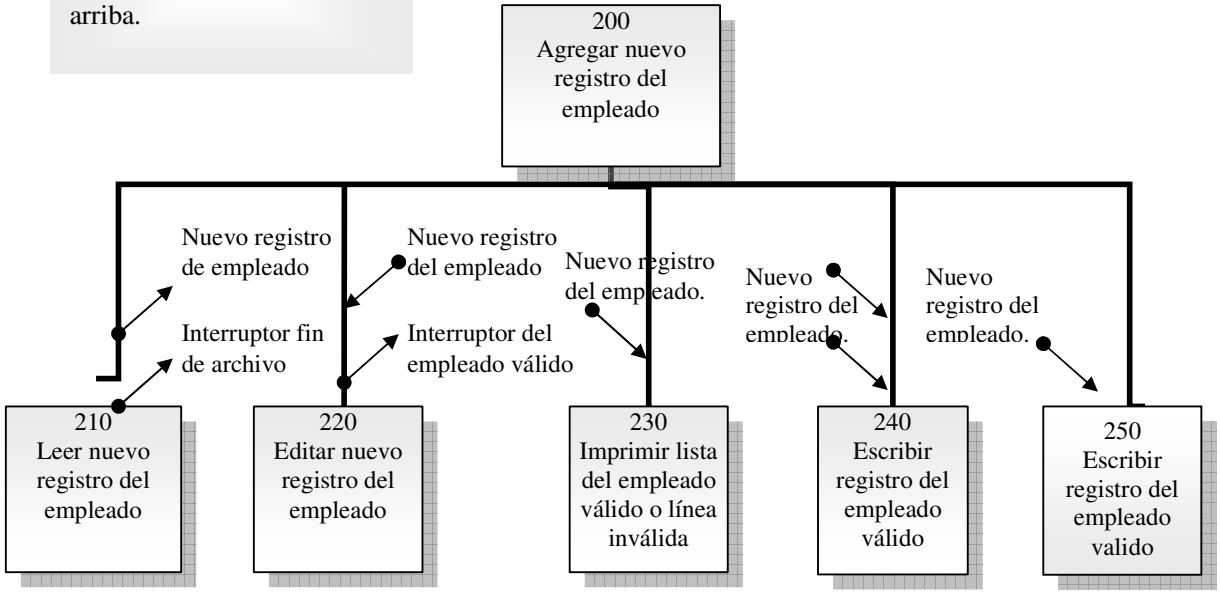
La figura 16.8 ilustra este concepto. Aquí, el módulo EDITAR NUEVO CLIENTE pasa el REGISTRO DEL CLIENTE al módulo EDICTAR NÚMERO TELEFONICO DEL CLIENTE, donde NUMERO TELEFONICO, un elemento encontrado en el REGISTRO DEL CLIENTE, se válida, y una bandera de control se pasa atrás al módulo EDICTAR NUEVO CLIENTE. EL TIPO DE ERROR (si no hay algunos), uno que contiene un mensaje de error tal como CÓDIGO DE ÁREA INVÁLIDOS O EL NUMERO TELEFONICO NO ES NUMÉRICOS, también se pasa hacia arriba. El mensaje se podría imprimir o desplegar en pantalla.

Aunque dichos módulos son bastante fáciles y crear y modificar cada vez que es necesario editar un número telefónico de un registro fuente diferente, se debe crear un nuevo modulo similar al EDICTAR NUMERO TEEFONICO DEL CLIENTE. Además, si la forma del número telefónico esta di validando los cambios como ocurre cuando se debe agregar un nuevo código de área o un código de país internacional, cada uno de estos módulos del nivel inferior se debe modificar.

Debido a que modulo de nivel o requiere ninguno de los otros elementos en el REGISTRO DEL CLIENTE, la solución es pasar solo el NIMERO TELEFONICO al modulo del nivel inferior. El nombre del modulo en este escenario cambia a EDICTAR NÚMERO TELEFONICO y se podría usar para editar cualquier número telefónico: un

FIGURA16.7

Un diagrama de estructura perfeccionado que muestra el flujo del control hacia arriba.



numero telefónico del cliente o un número telefónico del empleado. Los módulos del lado derecho de la figura ilustran este concepto. Cuando cambia las reglas para validar el número telefónico, solo se necesita EDITAR NIMERO TELEFONICO, sin tener en cuenta cuantos programas utilizan dicho modulo. Con frecuencia estos módulos de uso general se ponen en un programa compilado por separado denominado subprograma, función o procedimiento, dependiendo del lenguaje del programa usado.

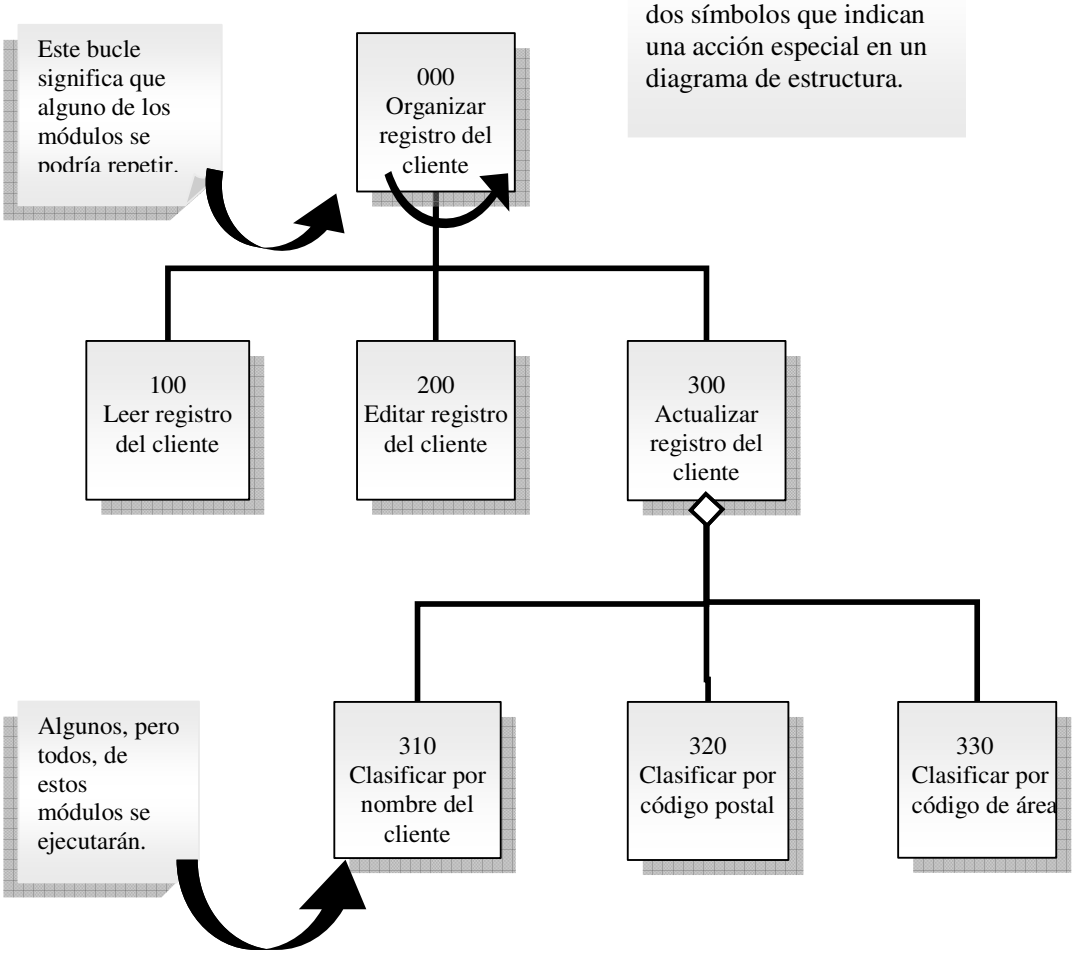
Como se muestra en la figura 16.9, el bucle es otro símbolo usado en los diagramas de estructura. Este símbolo indica que algunos procedimientos encontrados en los módulos 100 y 200 serán respetados hasta terminar. Este ejemplo implica que LLER REGISTRO DEL CLIENTE y EDITAR REGIETRO DEL CLIENTE se repitan hasta que todo el registro del cliente se complete. Después se clasifican en el modulo 300, pero como puede ver, se puede clasificar de tres formas diferentes: por nombre, código postal o código de área. Para mostrar que parte, pero no toda, de las clasificaciones se realizará, se usa otro símbolo, un diamante. Observe que el diamante no indica cual de los tres módulos se desempeñará, ni el bucle indica que modulo se repetirán estos símbolos pretenden deliberadamente ser generales no específicos.

DIBUJO DEL DIAGRAMA DE ESTRUCTURA

Obviamente, los diagramas de estructura se deben dibujar de arriba hacia abajo, pero ¿dónde se empiezan los procesos que serán los módulos? Probablemente, el mejor lugar para buscar esta información es en el diagrama de flujo de datos (véase el capitulo 7).

FIGURA 16.9

El bucle y el diagrama son dos símbolos que indican una acción especial en un diagrama de estructura.



Al trasformar un diagrama de flujos de datos en un diagrama de estructura, se debe tener en cuenta varias consideraciones adicionales. El diagrama de flujos de datos indicara la secuencia de los módulos en un diagrama de estructura. Si un proceso proporciona entrada a otro proceso, los módulos correspondientes se deben acompañar en la misma secuencia. La figura 16.10 es un diagrama de flujos de datos para preparar un informe de clasificación del estudiante.

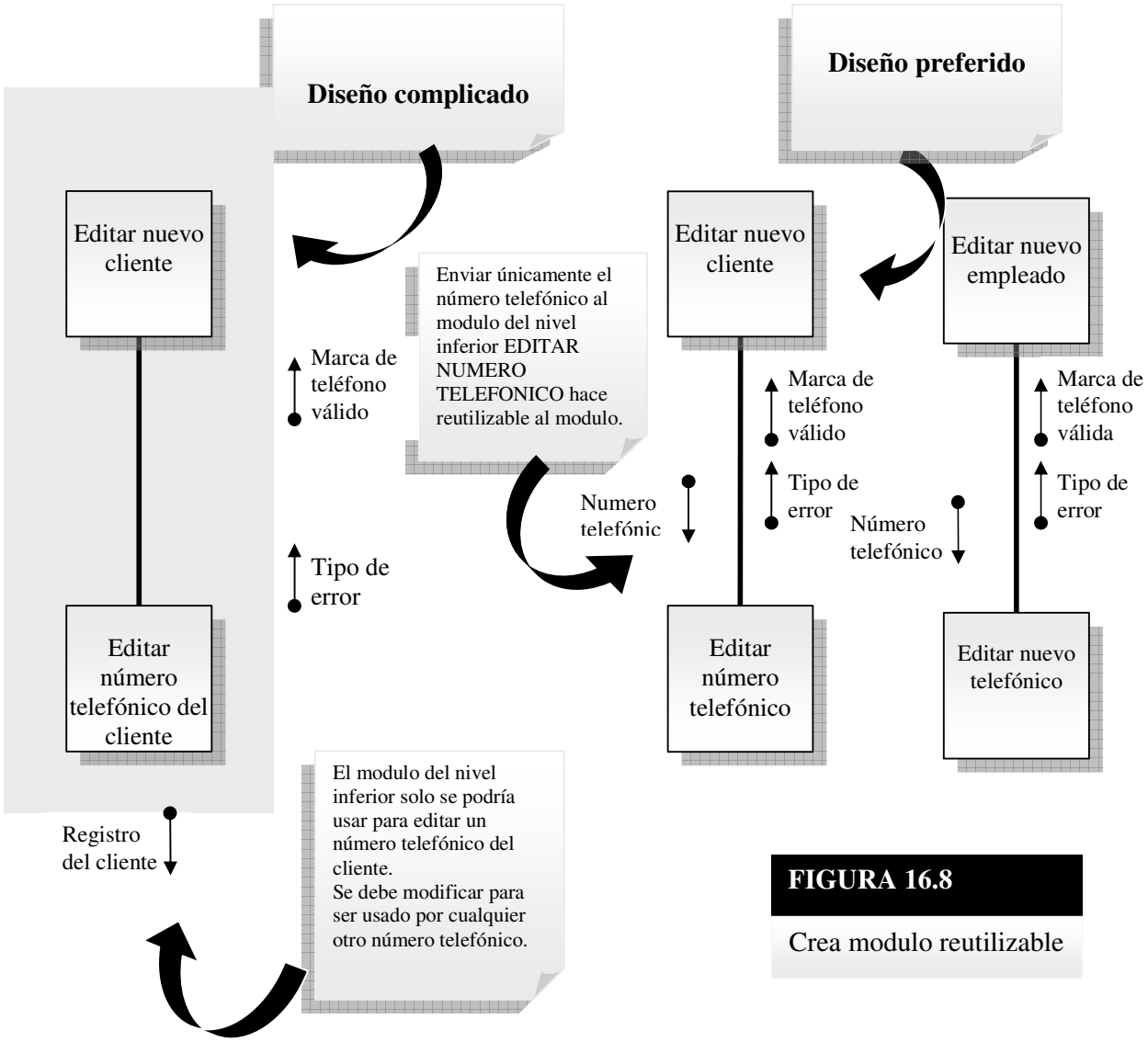


FIGURA 16.8

Crea modulo reutilizable

Observe que el proceso 1, LEER REGISTRO DE CALIFICACION, proporciona entrada para el proceso 2, LEER REGISTRO DEL CURSO, Y PARA EL PROCESO 3, LEER REGISTRO DEL ESTUDIANTE. En la figura 16.11 se ilustra el diagrama de estructura creado para este diagrama. Observe que el modulo 110, LEER REGISTRO DE CALIFICACION, se debe ejecutar primero. Después se deben ejecutar los procesos 2 y 3, pero debido a que no proporcionan entrada para otros, el orden de estos módulos (120 y 130) en el diagrama de estructura no es importante y se podría invertir sin afectar

Los resultados finales. Los procesos 1 y2 proporcionan entrada para el proceso 4, CALCULAR MEDIA DE PUNTO DE CALIDAD (también conocido como modulo 140). El proceso 5, IMPRIMIR INFORME DE CALIFICACION DEL ESTUDIENTE (modulo 150), recibe flujos de datos de todos los demás procesos y debe se el ultimo modulo en ser ejecutado.

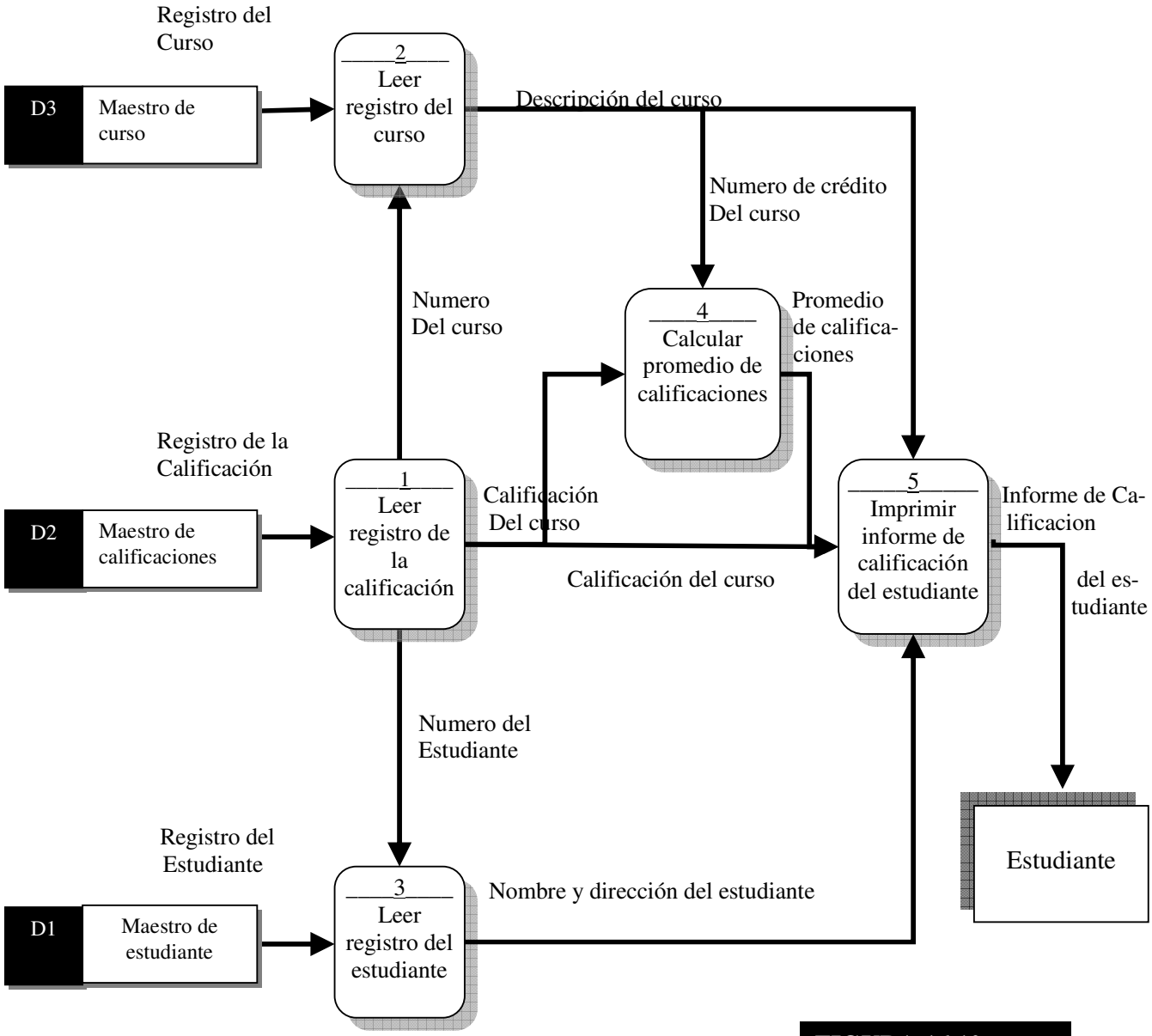
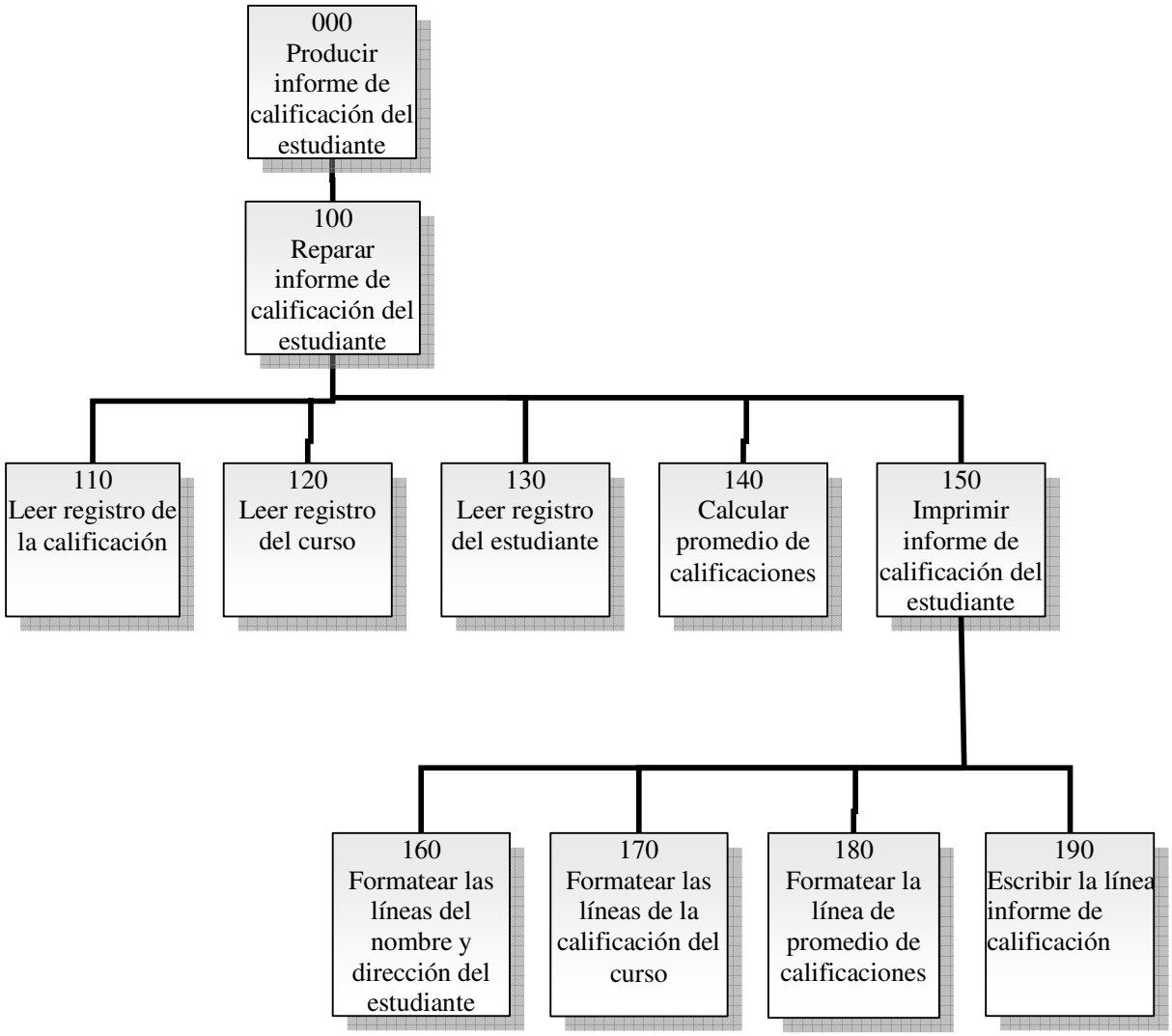


FIGURA 16.10

Diagrama de flujos de datos para imprimir un informe de calificación de estudiante

Si un proceso se divide en un diagrama de flujo de datos de hijo, el modulo correspondiente para el proceso padre tendrá módulos subordínales que correspondan a los procesos encontrado en el diagrama de flujo. El proceso 5, IMPRIMIR INFORME DE CALIFICACION DEL ESTUDIANTE, tiene cuatros flujos de datos de entrada y uno de salida y por ello es buen candidato para un diagrama hijo. En la figura 16.12 se

ilustra en el diagrama 5, los detalles del proceso 5. Los procesos en el diagrama 5, traduce los módulos subordinados al modulo 150, IMPRIMIR INFORME DE CALIFICACION DEL ESTUDIANTE.



TIPOS DE MÓDULOS

Los módulos del diagrama de estructura entran en una de las tres categoría generales: (1) control, (2) transformacional (a veces denominado trabajador) o (3) funcional. Al producir un diagrama de estructura que es fácil de desarrollar y modificar se debe tener cuidado, de no mezclar los diferentes tipos de módulos.

Los módulos de control normalmente se encuentran siempre en la parte superior del diagrama de estructura y contienen las lógicas para decenpeñan los módulos del nivel inferior. Los módulos de control podrían estar, o no estar representado en el día-

grama flujo de datos. Los tipos de instrucciones que normalmente están en los módulos de control son IF, PERFORM DO. Las instrucciones detalladas tal como ADD y MOVE normalmente se mantienen al mínimo. Con frecuencia la lógica de control es la mas difícil de enseñar por lo tanto, los módulos de control no deben de ser muy grande. Si un modulo de control tiene mas de nueve módulos subordinado, se deben crear nuevos módulos de control que sean subordinado de modulo de control original. La lógica de un modulo de control se podría determinar desde un árbol de disición o una tabla de decisión. Una tabla de decisión con demasiada reglas se deben dividir en varias tablas de decisión, con la primera tabla llamando a ejecución a la segunda tabla. Cada tabla de decisión produciría un modulo de control (véase el capitulo 9 para mas información de los árboles y la tabla de decisión).

Los módulos transformacionales son aquello creados de un diagrama de flujos de datos. Normalmente desempeñan una sola tarea aunque varias tareas secundarias se podrían asocial con la principal. Por ejemplo, un modulo denominado IMPRIMIR LINEA TOTAL DEL CLIENTE podría formatear toda la línea, imprimir la línea, agregarla a los totales finales y establecer los totales del cliente a cero en la preparación para acumular las cantidades del siguiente cliente. Los módulos transformacionales normalmente incluyen una mezcla de intrusiones, unas cuantas instrucciones IF y PERFORM o DO y muchas instrucciones detalladas tales como MOVE y ADD. Estos módulos son inferiores en la estructura que los módulos de control.

Los módulos funcionales son los más bajos en la estructura, rara vez tiene un módulo subordinado bajos ellos. Solo desempeñan una tarea, tal como formatear, leer, calcular o escribir. Algunos de estos módulos se encuentran en un diagrama de flujo de datos, pero otros se tendrían que agregar, tal como leer un registro o imprimir una línea de error.

La figura 16.13 representa el diagrama de estructura para agregar las reservaciones para los huéspedes de un hotel. Los módulos 000, AGREGAR RESERVACIÓN DEL HUÉSPED y 100, AGREGAR RESERVACIÓN DEL CUARTO, son módulos de control que representan el programa entero (modulo 000) y proporcional el control necesario para hacer una reservación de cuarto (modulo100). El modulo 110, DESPLEGAR PANTALLA DE RESERVACIÓN, es un modulo funcional responsable de mostrar la pantalla de reservación inicial. Los módulos 120, OBTENER RESERVACIÓN DE CUARTO VÁLIDA, y 160, CONFIRMAR CONSERVACIÓN DEL CUARTO, son los módulos de control de nivel inferior

El modulo 120, OBTENER RESERVACIÓN DE CUARTO VÁLIDA, se desempeña relativamente hasta que los datos de la reservación sean válida o hasta que el operador de la reservación cancele la transacción. Este tipo de modulo OBTENER RESERVACIÓN… desahoga el modulo 100 de una cantidad considerable de código completo. Los módulos subordinados para OBTENER RESERVACIÓN DE CUATRO VÁLIDAD son módulos funcionales responsable de recibir la pantalla de reservación, editar o validar la reservación del cuarto y mostrad una pantalla de error si los datos de entrada no válidos. Debido a que estos módulos están en un bucle, el control permanece en esta parte de la estructura hasta que los datos de la pantalla sean válidos.

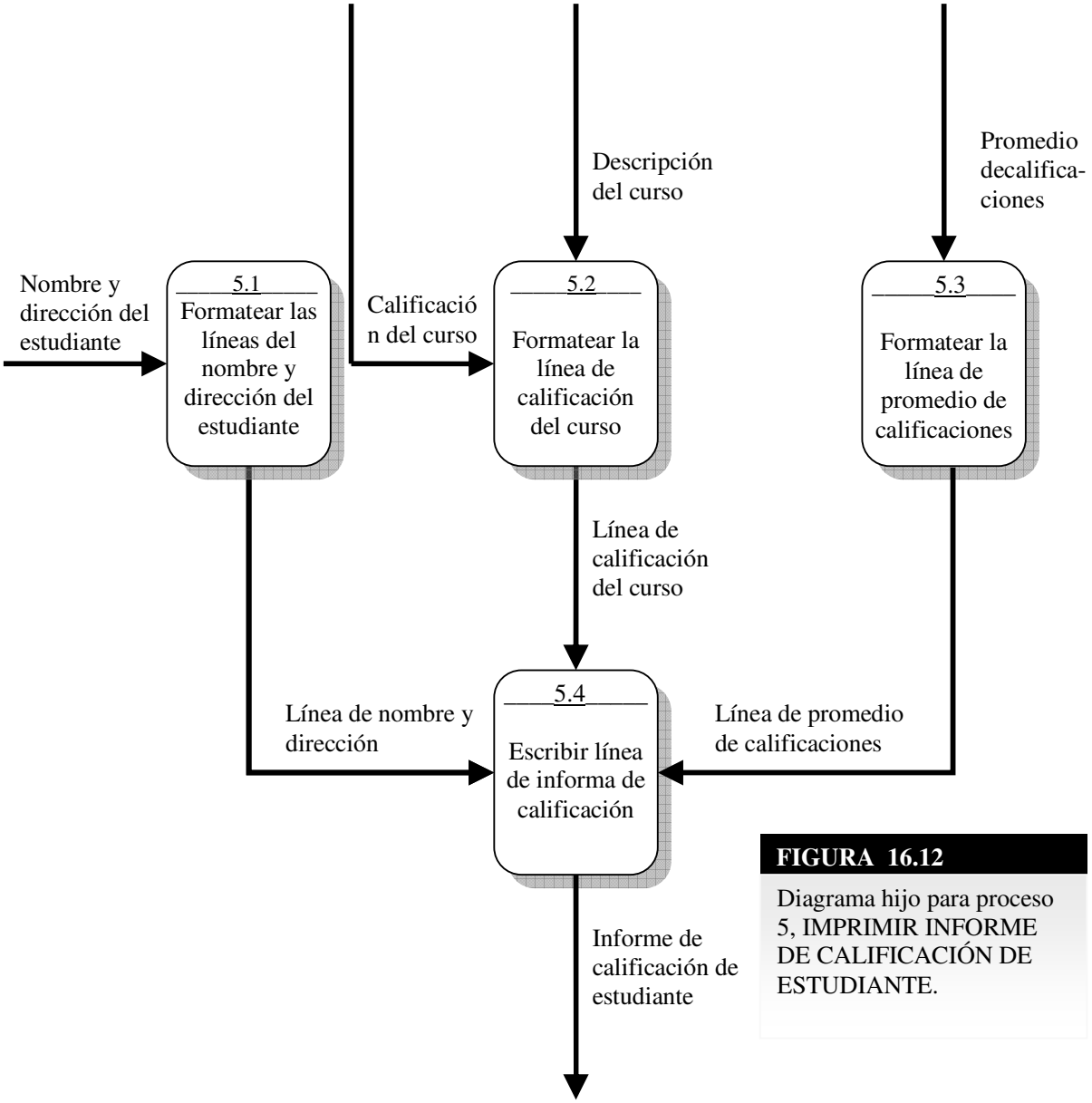


FIGURA 16.12

Diagrama hijo para proceso 5, IMPRIMIR INFORME DE CALIFICACIÓN DE ESTUDIANTE.

El modulo 160, CONFIRMARRESERVACION DEL CUARTO, también se desempeña rápidamente y permite al operador verificar que se haya introducido la información correcta. En esta situación, el operador inspeccionará la pantalla y presionará una tecla especificada, tal como **Enter**, si los datos son correctos, o una tecla diferente para modificar o cancelar la transacción. De nuevo, el programa permanecerá en estos módulos, repitiéndose hasta que el operador acepte o cancele la reservación.

El modulo 190 es un modulo transformacional que formatea el REGISTRO DE RESERVACION y desempeña el modulo 200 para escribir el REGISTRO DE RESERVACION. Los módulos 130, 140, 150, 160, 170, 180, y 200 son módulos funcionales que desempeñan una sola tarea: aceptar una pantalla, desplegar una pantalla o editar o escribir un registro. Estos módulos son los más fáciles de codificar, depurar y mantener.

SUBORDINACIÓN DE MODULO

Un modulo subordinado es un inferior en el diagrama de estructura llamado por otro modulo superior en le estructura. Cada modulo subordinado debe representar una tarea que es una parte de la función del modulo del nivel superior. Permitir que el modulo del nivel inferior desempeñe una tarea que no es requerida para el modulo que lo llama se denomina subordinación inadecuada. En tal caso, el modulo inferior se debe mover al nivel superior de la estructura.

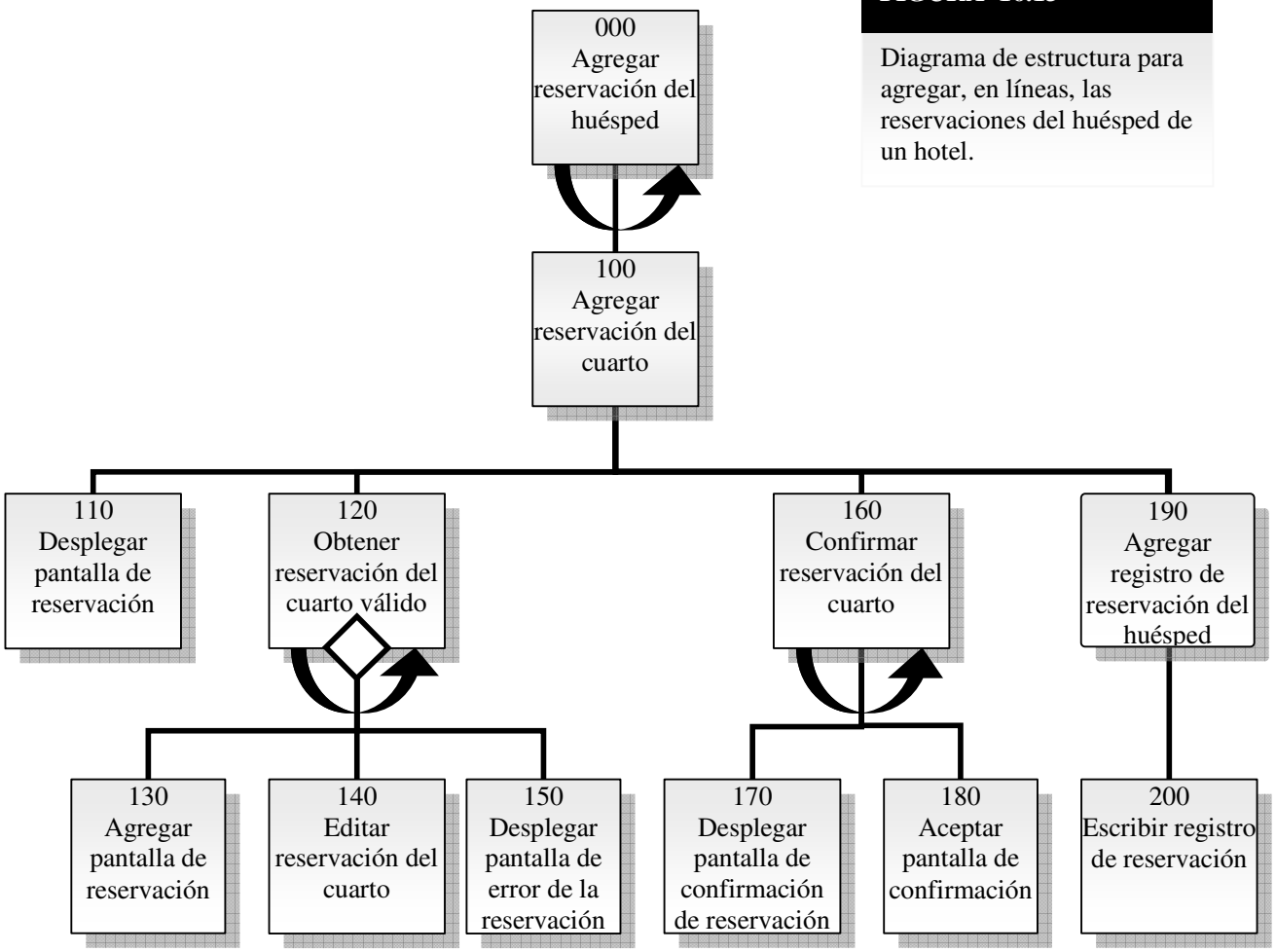


FIGURA 16.13

Diagrama de estructura para agregar, en líneas, las reservaciones del huésped de un hotel.

La figura 16.14 ilustra este concepto mediante un diagrama de estructura para cambiar un archivo MAESTRO DE CLIENTES. Examine el modulo 120, LEER MAESTRO DE CLIENTES. Este tiene la tarea de usar el NUMERO DEL CLIENTE de CAMBIAR REGISTRO DE TRANSACION para obtener directamente la coincidencia del REGIDTRO DEL CLIENTE. Si no se encuentra el registro, se imprime una línea de error. Por otra parte, el MAESTRO DE CLIENTES se cambia y se vuelve a escribir el registro. Este modulo debe ser un modulo funcional, que simplemente lee un registro, pero en cambio tiene tres módulos subordinados. La pregunta debe ser, º ¿se debe imprimir una línea de error para lograr la lectura del MAESTRO DE CLIENTES?º Además, º¿se debe formatear y volver a escribir el MAESTRO DE NUEVOS CLIENTES par leer el MAESTRO DE CLIENTES?º Debido a que las repuestas no son para ambas preguntas, los módulos 130, 140 y 150 no deben ser subordinados de LEER MAESTRO DE CLIENTES.

La figura 16.15 muestra el diagrama de estructura modificado. Las instrucciones de control se sacaron del registro LEER MAESTRO DE CLIENTES Y se pusieron en los módulos de control principal, CAMBIAR REGISTRO DEL CLIENTE. LEER MAESTRO DE CLIENTES se vuelve un modulo funcional (módulo 120).

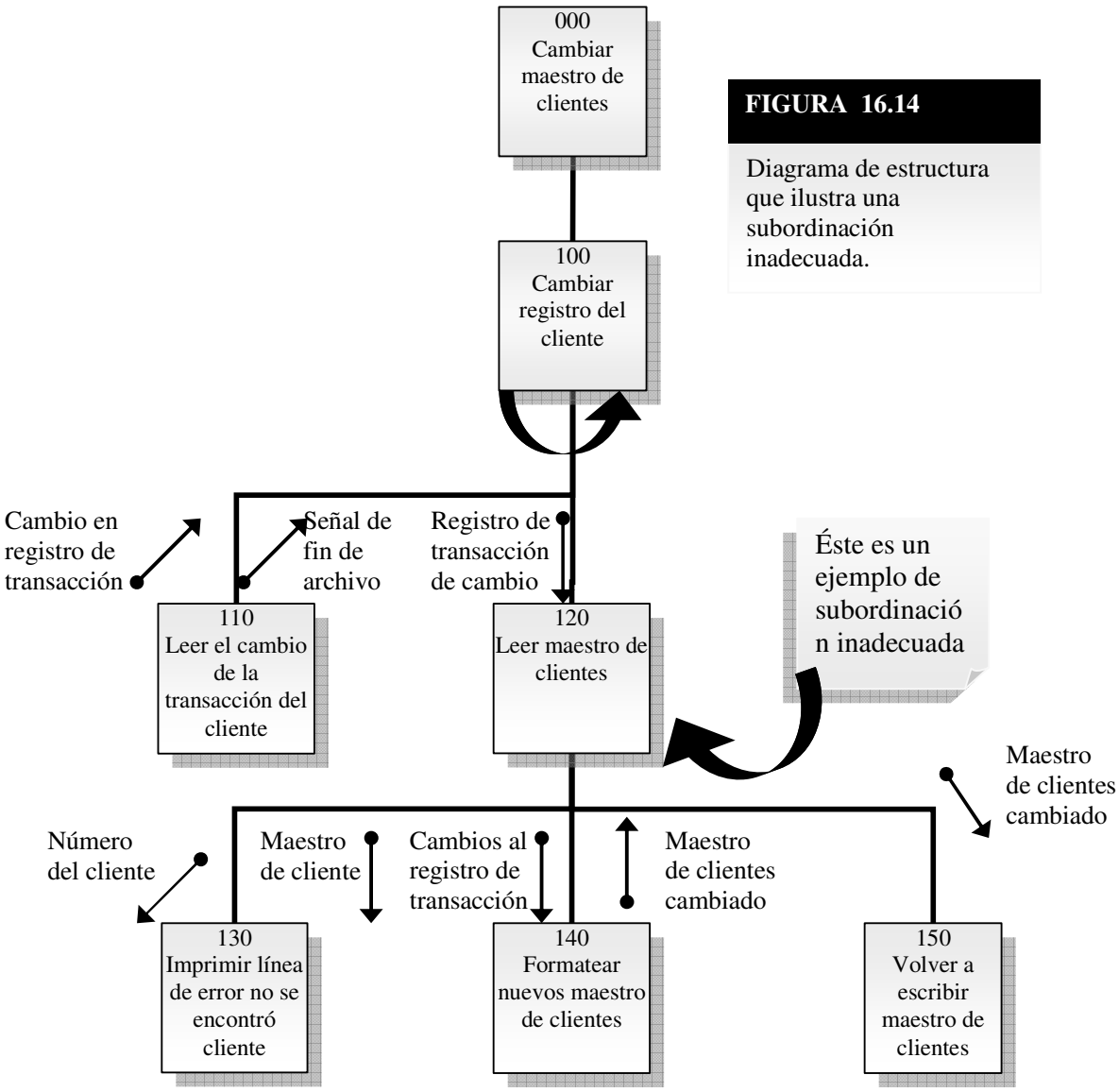


FIGURA 16.14

Diagrama de estructura que ilustra una subordinación inadecuada.

Aun cuando un diagrama de estructura logra todo los propósitos para los cuales fue diseñado, no puede ser la única técnica usada para diseño y documentación. Primero, no muestra el orden en que se deben ejecutar los módulos (un diagrama de flujos de datos lo hará). Segundo, no muestra suficiente detalles (un pseudocodigo lo hará). El resto de este capitulo discute estas técnicas mas detalladas usando como ejemplo el problema de suscripción a un periódico pensando anteriormente, el cual se describe ahora con mas detalle.

INGENIERÍA DE FOFTWARE Y DOCUMENTACION

La plantación y control son elementos fundaménteles en todo sistema exitoso. En el desarrollo de software para el sistema, el analista de sistema debe saber que la plantación tiene lugar en el diseño, incluso antes de que empiece la programación. Necesitamos técnicas que nos ayuden a establecer los adjetivos del programa, de manera que nuestros programas estén completos. También necesitamos técnicas de diseños que nos ayuden a separar el esfuerzo de programación de módulos manejables.

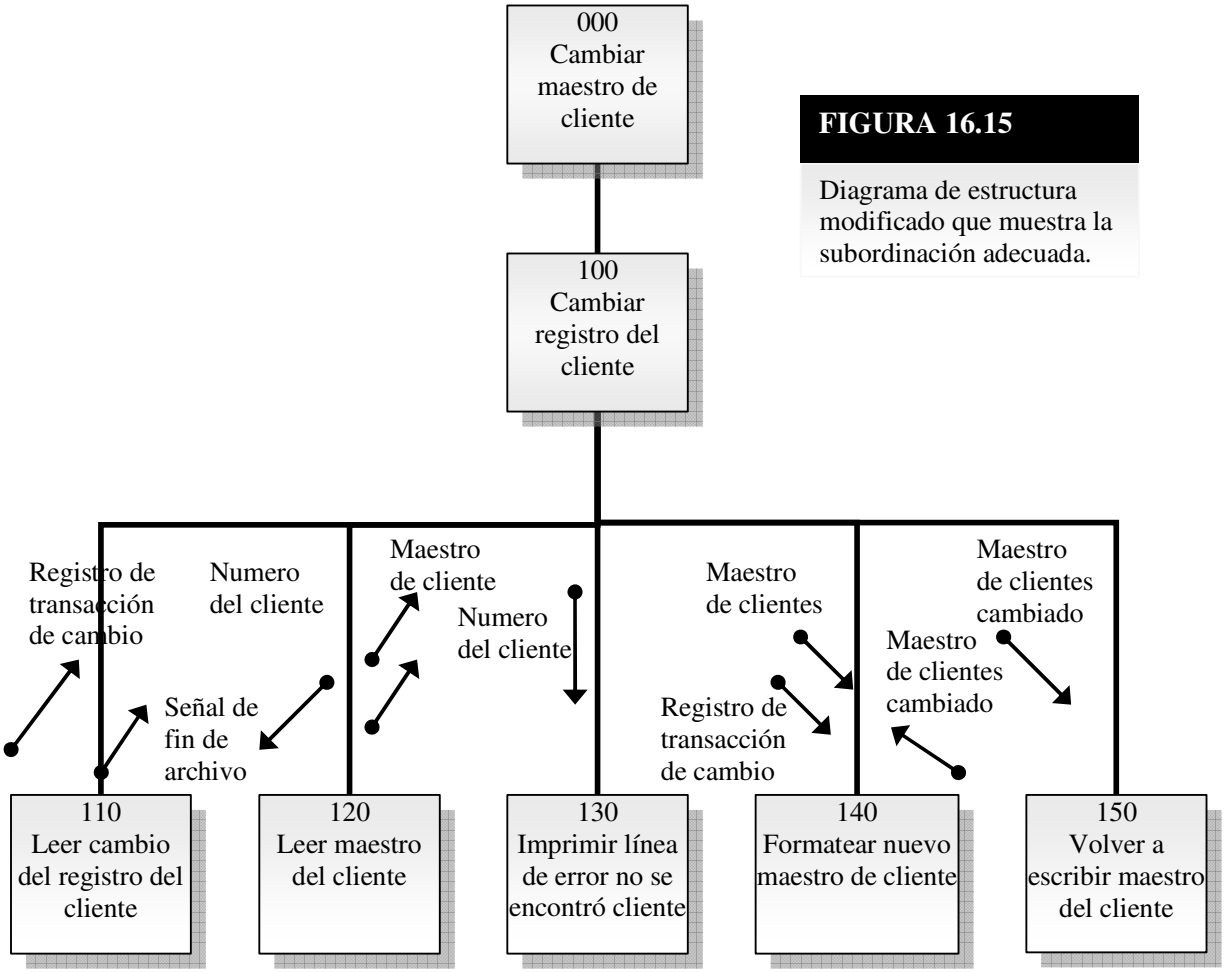


FIGURA 16.15

Diagrama de estructura modificado que muestra la subordinación adecuada.

Sin embargo, no es satisfactorio insertar tener éxito tan solo con las etapas de la planeación. Después que se completa los programas, se deben mantener y los esfuerzos de mantenimientos normalmente son mayores que el esfuerzo empleado en el diseño y la programación originales.

Las técnicas descritas en la siguiente sección no solo están hechas para usarse inicialmente en el diseño de software, sino también en un mantenimiento. Debido a que la mayorías de los sistemas no se consideran desechables, muy probablemente necesitarán ser mantenidos. El esfuerzo de aseguramiento de la calidad total requiere que los programas se documenten adecuadamente.

El software y los procedimientos se documentan de manera que se codifiquen en un formato que pueda acceder fácilmente. El acceso a esta documentación es necesario para las nuevas personas que aprenden el sistema y como un recordatorio para aquellos que no usan el programa con frecuencia. La documentación permite a usuario, programadores y analista ªverª el sistema, su software y procedimientos sin tener que interactuar con él.

Cierta documentación proporciona una aparición global del propio sistema, mienta que la documentación de procedimiento detalla lo que se debe hacer para ejecutar el software en el sistema y la documentación del programa detalla el código del programa que se usa.

La rotación del personal de servicio de información tradicionalmente ha sido alta en la comparación con otros departamentos, de manera que probablemente las personas que concibieron e instalaron el sistema original no serán mismas que las que lo mantienen.

La documentación consistente y bien actualizada acortará el número de horas requerido para que las nuevas personas aprendan el sistema antes de realizar el mantenimiento.

Hay muchas razones por las cuales los sistemas y programas no están documentados o presentan subdocumentacion. Algunos de los problemas residen en los sistemas y programas, otros en los analistas de sistemas y programadores.

Algunos sistemas heredados fueron escritos antes de que un negocio estandarizara sus técnicas de documentación, pero todavía se usan (sin documentación). Muchos otros sistemas han sufrido modificaciones mayores o menores y se han actualizados durantes los años, pero su documentación no se ha modificado para reflejar esto. Incluso se puede dar el caso que se hayan comprados algunos sistemas especificados para aplicaciones importantes a pesar de su falta de documentación.

Los analistas de sistemas podrían no documentar adecuadamente los sistemas debido a que no tiene tiempo usado en la documentación. Algunos analistas no documentan porque tienen miedo de hacerlo o piensan que no es su trabajo. Además, muchos analistas son reservados sobre documentar sistemas que no son suyo, quizás temen a las represalias si incluyen material incorrecto en el sistema de alguien mas la documentación lograda por medio de una herramienta CASE durante las fases del análisis puede resolver mochos de estos problemas.

Actualmente no se usa una sola técnica estándar de diseño y documentación. En las siguientes secciones, discutimos varias técnicas diferentes que actualmente se usan. Cada técnica tiene sus propias ventajas y desventajas, debido a que cada una tiene propiedades únicas.

PSEUDOCÓDIGO

En el capitulo 9, introducimos el concepto de español estructurado como una técnica de analizar decisiones. El pseudoscódigo es similar al español estructurado porque no es un tipo particular de programar códigos, pero se puede usar como un paso intermedio para desarrollar el código de programa.

La figura 16.16 es un ejemplo de pseudocódigo para chenoweth enterprises, un conglomerado del periodo que publica el *charlie brown's* journal, the steel pier observer wicked, el siempre popular periódico orientado a los adolescente. El conglomerado del periódico pasa por un proceso de actualizar, imprimir y proporcionar informe de administración para cada una de sus periódicos. El pseudocódigo para este proceso involucra un proceso de actualizar cada lista de suscritores al periódico. Esta estructura se puede ver fácilmente en el pseudocódigo.

En la industria es común el uso del pseudocódigo, pero falta la estandarización evitará que sea aceptado por todo. Debido a que su pseudocódigo esta tan cerca del código de programa, naturalmente es favorecido por programadores y por consiguientes no es favorecido por analista de negocio. El pseudocódigo con frecuencia se usa para representar la lógica de cada modulo en un diagrama de estructura.

El diagrama de flujos de datos se podría usar para escribir la lógica del pseudocódigo. Al usar un nivel del programa en un lugar de un nivel de sistema, el diagrama de flujo de datos podría agregar varios símbolos adicionales. El asterisco (*), que significa º y º, se usa para indicar que deben estar presente los dos flujos de datos nombrado. Consulte la parte de datos de un diagrama de flujos de datos que se ilustra en la figura 16.17 si los flujos de datos de entrada son de procesos diferentes, la presencia del conector º y º significa que el proceso que recibe cuencial, leer todo los registro de ambos archivos o una lectura indexada de un segundo archivo usando un campo importante obtenido del primer archivo.

El signo de suma encerrado en un circulo (+) representa un º o º exclusivo e indica que uno u otros flujos de datos esta presente en cualquier momento dado. Usar este símbolo implica que el proceso que recibe o produce el flujo de datos debe tener una instrucción correspondiente IF… THEN…ELSE…

FIGURA 16.16

Usar pseudocódigo para describir un servicio de actualización de suscripción para una compañía editorial especializadas en periódicos

```
Abrir archivos
Leer el primer nombre periódico
DO WHILE hay mas nombre (s).periódico
     PRINT fecha
     PRINT NOMBRE.PERIODICO
     Leer el primer registro.subcriptor
     DO WHILE hay mas registro(s).suscriptor
       IF transacción = modificar.renovacion
          THEN durecion.suscrip. = duracion.suscrip. + Números.semanas
       ELSE IF transacion = nueva
          THEN PERFORM agregar.suscriptor
          THEN duracion.suscrip = numero.semanas
       ELSE IF transacion = modicar.direccion
          THEN PERFORM cambiar.direccion
       ELSE IF transacion = eliminar.suscriptor
          THEN PERFORM preparer.reembolsos.actualizar
          PERFORM imprimir.reembolso.cheque
          Duracion.suscrip. = 0
       ELSE PERFORM error.en.transacion
       ENDIF
       PERFORM preperar.suscriptor.lista
       Leer otro registro.suscriptor
   ENDO
   PERFORM imprimir. Suscriptor.lista
   Obtener siguiente nombre.periodico
ENDO
Cerrar archivos
```

.

MANUALES DE PROCEDIMIENTOS

Los materiales de procedimiento son documentos organicionales comunes que la mayoría de las personas ha visto. Son el componente en español de la documentación, aunque también podrían contener códigos de programas, diagramas de flujos, etc. se pretende que los manuales comuniquen a aquellos que lo usan. Podrían contener comentarios de fondos, los pasos requeridos para lograr diferentes transacciones, instrucciones de como recuperarse de los problemas y qué hacer si algo no funciona (solucionar problema). Actualmente muchos manuales están disponible en línea, con capacidad de hipertexto que facilita el uso.

Se desea un enfoque directo y estandarizado para crear documentación de apoyo de usuario. Para ser útil, la documentación del usuario se debe mantener actualizada. El uso de Web ha revolucionado la velocidad con lo que los usuario pueden obtener asistencia. Muchos diseñadores de software están desplazando el soporte de usuario- con la lista de preguntas frecuentes (FAG), escritorio de ayuda, soporte técnico y servicio de fax- para Web. Además, muchos vendedores de software COTS incluyen archivos ºLeameº con descarga o envió de nuevos software. Estos archivos sirven para varios propósitos: documentan cambios, agostan o reparan fallas recientemente descubiertas en la aplicación que han ocurrido demasiado tarde en su desarrollo para poder ser incluida en el manual del usuario.

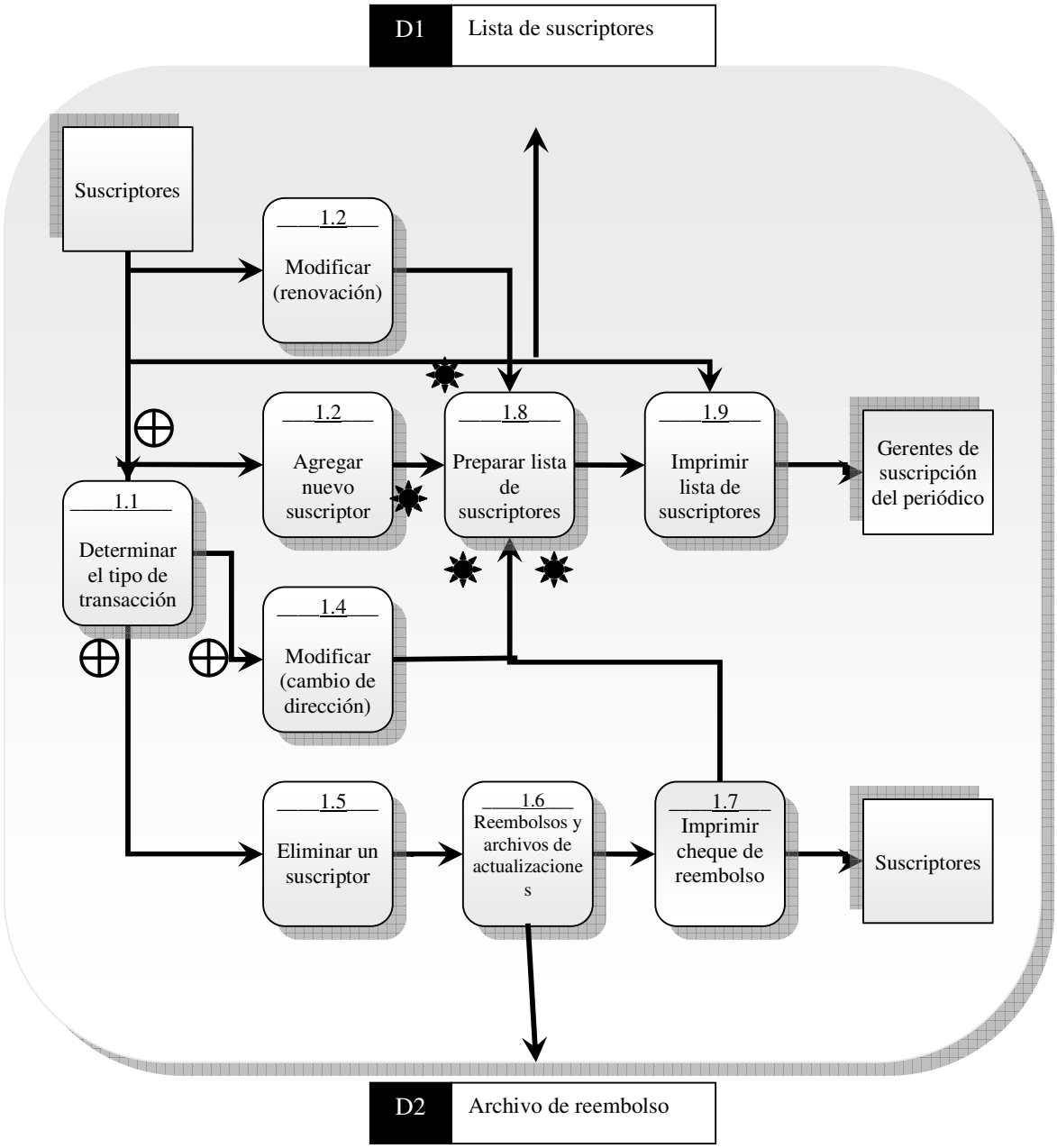
Las selecciones importante de un Manuel deben incluir una introducción, como usar el software, que hacer si las cosas salen mal, una sección de referencia técnica, un índice e información de cómo contactar al fabricante. Los manuales en línea en los sitios Web también deben incluir información sobre descargar actualizaciones y una pagina de FAQ. Los problemas con los manuales de procedimiento son que (1) están mal organizados,(2) es difícil encontrar la información necesaria en ellos,(3) el caso especifico en cuestión no aparece en el manual y (4) el manual no esta inscrito en español. Mas adelante, en la selección donde se habla de pruebas, discutimos la importancia de tener usuario que ºpruebaº los manuales de sistemas y prototipos de los sitios Web antes de que se terminen.

EL METODO DE FOLKLORE

EL FOLKLORE: es una técnica de documentación de subte mas creada para complementar algunas de las técnicas ya tratada. Aun con la abundancia de técnicas disponible, muchos sistemas se documentan inadecuadamente o no se documentan en absoluto. EL FOLKLORE recopila información que normalmente se comparte entre los usuarios pero raramente se pone por escrito.

EL FOLKLORE es una técnica sistemática, basada en métodos tradicionales usado para recopilar el folklore sobre las personas y leyenda. Este enfoque para la documentación de sistemas requiere que el analista entreviste a los usuarios, investigue la documentación existente en los archivos y observe el procedimiento de información. El objetivo es recopilar la información correspondiente a una de cuatros categoría: costumbre, anedota, proverbios forma artística. La figura 16.18 siguiere como se relaciona cada categoría a la documentación de sistemas de información.

Al documentar las costumbres, el analista (u otros folkloristas) intenta capturar por escrito lo que los usuarios hacen para conseguir que los programas puedan ejecutar sin problemas. Por ejemplo de unas costumbres es: º normalmente nos toma dos días



actualizar los registros mensuales por que la tarea es bastante grande. Ejecutamos cuentas comerciales en un día y guardamos las otras para el siguiente día.

Las anécdotas son historias que los usuarios dicen respecto a como funcionó el sistema. Por supuesto, la exactitud de la anécdotas depende de la memoria del usuario y es, en el mejor de los casos, una opinión sobre como funcionó el programa. Lo siguiente es un ejemplo de una anécdota:

> El problema ocurrió de nuevo en 1995. Esa vez, el trabajo LIB 409 (actualización mensual) solo se ejecutó con los registro tipo º6 °. Debido a este error no había registrado financiero en el archivo LIBFIN. Cuando intentamos leer el archivo vació este inmediatamente se serraba y por consiguiente los totales se reportaron como cero. Pudimos corregir este problema agregando un registro tipo 7 ° y ejecutando el trabajo.

Las anécdotas normalmente tienen un principio, un cuerpo y un fin. En este caso tenemos una historia sobre un problema (el principio), una descripción de los efectos (el cuerpo) y la solución (el fin).

Los proverbios son declaraciones breves que representan generalizaciones o consejos. Tenemos muchos proverbios en la vida cotidiana, tal como mas vale pájaro en manos que cinto volando ºoº centavos ahorrado, centavos ganados º. En la documentación de sistemas, tenemos muchos proverbios, tal como Omita éstas sección de código y el programa fallarán o º Haga frecuentemente copia de seguridad. A los usuarios le gusta dar consejos y el analista debe intentar captura dicho consejos e incluirlo en la documentación del FOLKLORE.

Recopilar formas artísticas es otra actividad importante del folklorista tradicional, y también el analista de sistemas debe entender su importancia. Los diagramas de flujos, diagrama sí tablas que los usuario diseñan algunas veces podrían ser mejores o mas útil que los diseñados por el autor de sistema original. Los analistas con frecuencias entrarán tal arte en los carteles de anuncios o podrían pedir a los usuarios vaciar sus archivos y recuperar cualquier diagrama útil

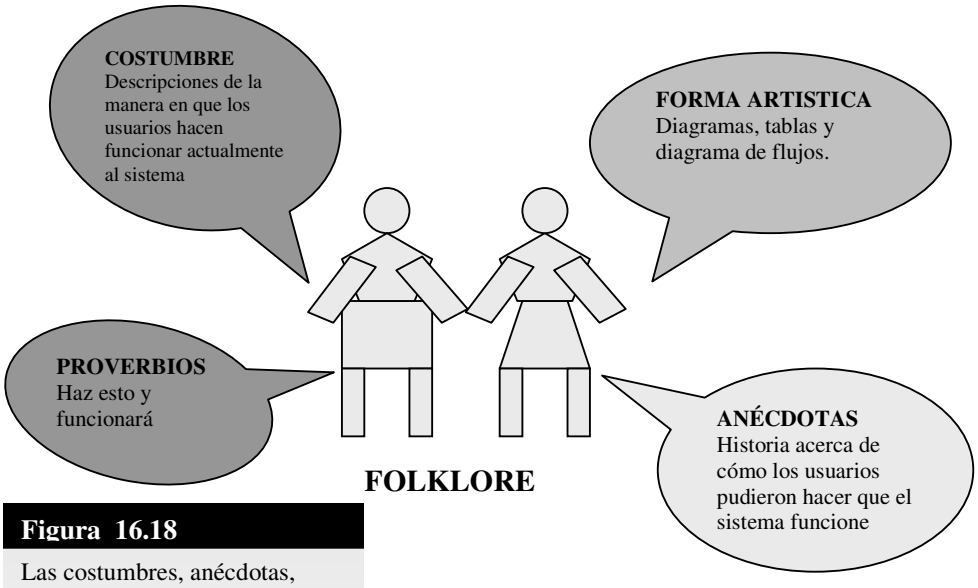
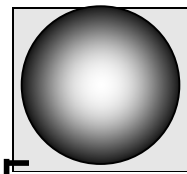


Figura 16.18

Las costumbres, anécdotas, proverbios, y formas artísticas que se usan en el método FOLKLORE de documentación se aplican a los sistemas de información

El enfoque FOLKLORE funciona debido a que puede ayudar a reparar la falta de conocimiento cuando un autor de programa se retira. Los construyen tes al documento FOLKLORE no tiene que documentar el sistema entero, sólo las partes que conozcan. Por ultimo, es divertido para los usuarios contribuir, quitando así algo de cargas de los analistas. Observe que la clase de sistema recomendación que se discutió anteriormente en el libro está muy cerca de la conceptualizacion de FOLKLORE. Estos sistemas amplían la idea de FOLKLORE para incluir todos los tipos de recomendaciones, tales como las calificaciones de restaurantes y películas. Usando métodos económicos o gratuito como el correo electrónico, se han superados algunas barreras iniciales para recopilar y compartir la información informal.

El peligro de confiar en el FOLKLORE en que la información recopilada de los usuarios podrían ser correcta, particularmente correctas incorrecta. Sin embargo, a menos que alguien se tome el tiempo de rehacer completamente la documentación de programa, la descripción de costumbres, anécdotas proverbios y formas artísticas podrían ser la única información escrita acerca de cómo funciona un grupo de programas.



OPORTUNIDADES DE CONSUTARIA 16:2

Escribir es correcto

º Es fácil de entender. Creo que si todos usamos pseudocódigos, no tendremos problemas, es decir, con cosas que no se estén esTandarizandoº, dice al gorithm, un nuevo programados que trabajará con su equipo de analista de sistemas. Al participar en una reunion informal con tres miembros del equipo de analista de sistemas, un grupo de trabajo de MIS compuesto por seis miembros del despartamento de publicidad, y dos programadores, quienes trabajan en equipo para desarrollar un sistema de información para el personal de publicidad.

Philip, un ejecutivo de cuentas del área de Publicidad y miembro del grupo de trabajo MIS, Pregunta sorprendido: º¿Cómo se llama este meTodo?º. los dos programadores contestan al mismo tiempo: ºpseudocódigoº. Philip contestan, Imperturbable: ºeso no me dice nadas.

Neeva phail, una de las analista de sistema, Empieza a explicar, talvez no importe tanto lo Que utilicemos, si…

Flo chart, otra analista de sistema, la inteRrumpe: yo odio el pseudocódigoº. Flo mira a los programadores en busca de apoyo. º estoy segura de que podemos ponernos deacuerdo en una mejor técnicas.

David, un ejecutivo de publicidad mas experimentado, parece un poco disgustado y dice: º yo aprendí algo de los diagramas de flujos con los primero analistas de sistemas que tuvimos hace años. ¿ustedes ya no los utilizan? Creo que son una mejor occionº.

Lo que al principio era una reunión amistosa repentinamente parece haber llegado a un callejón sin salida. Los participantes se miran uno a otro con recelo. como un analista de sistema que ha trabajado en muchos proyectos diferentes con muchos tipos distintos de persona, usted comprende que el grupo espera que usted haga algunas sugerencias razonables.

Con base en lo que usted conoce sobre las diversas Tecn.cas de documentación, ¿Qué técnica (s) propondría a los miembros del grupo? ¿De que manera la (s) técnica(s) usted proponga solucionará (n) algunas de las preocupaciones que han expresados los miembros del grupo? ¿Qué Proceso utilizará para elegir las técnicas apropiadas.

SELECCIÓN DE UNA TÉCNICA DE DISEÑO Y DOCUMENTACIÓN

Las técnicas discutidas en este capitulo son sumamente valiosas como herramientas de diseños, ayudas de memoria, herramienta de productividad y como medio de reducir las dependencias en los miembros de personal claves. Sin embargo, el analista de sistema se encuentra con una decisión difícil con respecto a qué modulo adoptar. Lo siguiente es un grupo de lineamiento para ayudar al analista a seleccionar la técnica adecuada.

Escoja una técnica que:

1. Es compatible con la documentación existente.
2. Se entiende por otro en la organización.
3. Le permite regresar a trabajar en el sistema después que ha estado fuera de él por un periodo.
4. Sea conveniente para el tamaño del sistema en que está trabajando.

5. permita un enfoque de diseño estructurado si se considera como más importante que otros factores.
6. permita fácil modificación.

CÓMO PROBAR, MANTENER Y AUDIATR

Una vez que el analista ha diseñado y modificado el sistema, probar, mantener y auditar son las primeras consideraciones.

EL PROCESO DE PROBAR

Todos los programas de aplicación del sistema recién escrito o modificado ─así como también nuevos manuales de procedimiento, nuevo hardware y todas las interfaces del sistema─ se deben probar completamente. Probar al azar y por tanteo no será suficiente.

Las pruebas se hacen durante todo el proceso de desarrollo de sistemas, no solo al final. Se busca descubrir errores desconocidos hasta ahora, no demostrar la perfección de programas, manuales o equipos.

Aunque probar es tedioso, es una serie esencial de pasos que ayuda a asegurar la calidad eventual sistema. Es mucho menos inquietante probar de antemano que tener un sistema probado deficientemente que falle después de la instalación. Las pruebas se realizan en subsistemas o módulos del programa enfoque avance su desarrollo. Las pruebas se hacen en muchos niveles diferentes a varios intervalos. Antes de que el sistema se ponga en producción, todos los programas se deben verificar en el escritorio, verificar con datos de prueba y verificar para ver si los módulos trabajan entre sí como se planeó.

El sistema también se debe probar como todo en funcionamiento. Incluso hay que probar las interfaces entre los subsistemas; la exactitud de salida; y la utilidad y entendimiento de la documentación y la salida de sistemas. Como se muestra en la figura 16.19, los programadores, analista, operadores y usuarios cumplen un papel diferente en los varios aspectos a probar. Las pruebas d hardware normalmente se proporcionan como un servicio por los vendedores de equipo quienes ejecutarán sus propias pruebas en el equipo cuando se libere en el sitio.

Pruebas de programas con datos de prueba Mucha de la responsabilidad para probar el programa radica en el autor(es) original de cada programa. El analista de sistema sirve como consejo y coordinador para las pruebas del programa. En esta capacidad, el analista trabaja para asegurar que los programadores implementen las técnicas de pruebas correctas pero probablemente no desempeñe personalmente este nivel de verificación.

En esta FACE, los programadores deben hacer pruebas de escritorio de sus programas para verificar las formas en que funcionará el sistema. En la verificación de escritorio, el programador sigue cada paso en el programa impreso para verificar si la rutina funciona como se espera.

Luego, los programadores deben crear datos de pruebas válidos e inválidos. Estos datos se ejecutan después para ver si la rutinas de base trabajan y también para descubrir errores.

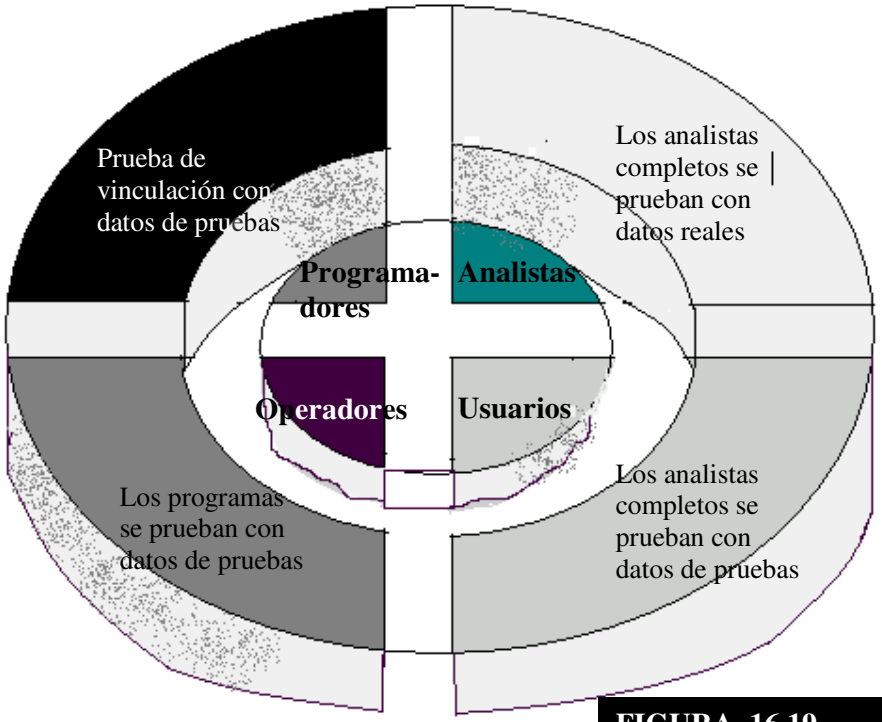


FIGURA 16.19

Los programadores, analistas, operadores y usuarios desempeñan un papel diferente en probar el software y sistemas.

Si las salidas de los módulos principales es satisfactoria, puede agregar más datos de pruebas para verificar otros módulos. Los datos de pruebas creados deben probar posibles valores mínimos y máximos así como también todas las variaciones posibles en el formato y códigos. Los archivos de salidas de los datos de pruebas de deben verificar cuidadosamente. Nunca se fue creado y accesado.

A lo largo de este proceso, el analista de sistemas verifica la salida de búsqueda de errores, abusando al programador de cualesquiera correcciones necesarias. El analista normalmente no recomendará o creará datos de pruebas para las pruebas de programas pero podría señalar al programador las omisiones de tipos de datos a ser agregados en pruebas posteriores.

Prueba de vínculos con datos de pruebas Cuando los programas pasan la verificación de escritorio y la verificación de datos de pruebas, se debe pasar por las pruebas de vínculos, que también se conocen como prueba de cadena. Estas pruebas verifican si los programas que realmente son interdependientes trabajan juntos como se planeo.

Una pequeña cantidad de datos de prueba, normalmente diseñado por el analista de sistema para probar las especificaciones del sistema así como también los programas, se usan para las pruebas de vinculación. Podría tomar varios pasos a través del analista para probar todas las combinaciones, debido a que es inmediatamente difícil resolver los problemas si intenta probar todo a la vez.

El analista crea datos de pruebas especiales que cubren una variedad de situaciones de procedimientos para la pruebas de vinculación. Primero se procesan datos de pruebas típicos para ver si el sistema puede manejar transacciones normales, se agregan las variaciones, incluyendo los datos inválidos para asegura que el sistema puede detectar adecuadamente los errores.

Prueba completa de sistemas de datos de pruebas Cuando las pruebas de vinculación se concluyen sastifactoriamente, se debe probar el sistema como una entidad completa. En esta fase, los operadores y usuarios finales se involucran activamente en la prueba. Los datos de prueba, creados por el equipo de análisis de sistemas para el propósito expreso de probar los objetivos de analista, se usan.

Como se puede esperar, hay varios factores a considerar cuando se prueban los sistemas de datos de pruebas:

1. examinar si los operadores tienen la documentación adecuadas en los manuales de procedimientos (en papel o en línea) para asegurar un funcionamiento correcto y eficaz.
2. verificar si los manuales de procedimientos son lo bastante claro como para comunicar cómo se deben preparar los datos para la entrada.
3. determinar si los flujos de trabajos necesarios para el sistema nuevo o modificado realmente Fluyen.
4. determinar si las salidas es correcta y si los usuarios entienden que esta salida es como se verá en su formulario final.

Recuerde fijar el tiempo adecuado para la prueba del sistema. Desafortunadamente, con frecuencia este paso se elimina si la instalación del sistema se retrasa de la fecha indicada.

La prueba de sistema incluye reafirmar los estándares de calidad para el desempeño del sistema que se estableció cuando se hicieron las especificaciones iniciales del sistema. Todos los involucrados deben estar de acuerdo una ve mas en cómo determinar si el sistema está haciendo lo que se supone que hace. Este paso incluirá medidas de error, oportunidades, facilidad de uso, clasificación apropiada de transacciones, tiempo fuera de servicio aceptable y manuales de procedimiento entendible.

Prueba completa de sistemas con datos reales Cuando las pruebas de sistemas con datos de prueba se realizan de manera satisfactoria, es bastante recomendable probar el nuevo sistema repetidas con lo que se conoce como datos reales, datos que se han procesados de manera exitosa con el sistema existente. Este paso permite una comparación precisa de la salida del nuevo sistema con la salida que sabe ha sido procesada correctamente, y también es una buena opción para probar como se manejarán los datos reales. Obviamente, este paso no es posible al crear salida completamente nueva (por ejemplo, salida de una transacción de comercio electrónico de un nuevo sitio Web corporativo).al igual que con los datos de prueba. Sólo se usa pequeñas cantidades de datos reales en este tipo de pruebas de sistemas.

Las pruebas constituyen un periodo importante para evaluar la manera en que los usuarios finales y los operadores interactúan realmente con el sistema. Aunque se da mucha importancia a la interacción de usuario-sistema (véase el capitulo 14), nunca podrá producir totalmente el amplio rango de diferencias en la forme en que los usuarios interactuarán realmente con el sistema. No basta con entrevistar a los usuarios sobre cómo interactúan con el sistema, debe observarlos directamente.

OPORTUNIDAD DE CONSULTARÍA 16.3

ESTUDIANDO PARA SU PRUEVA DE SISTEMAS

ªTenemos el tiempo encima. Tan solo mira esta proyección, dice lou ecuntroll, el miembro mas nuevo de su equipo de analisis de sistemas, mientras le muestra el diagrama PERT que el equipo ha estado usando para proyectar la fecha en que el nuevo Sistema quedaría listo. Quizá no podemos cumplir la fecha prevista de julio para reaLizar las pruebas con datos reales. Estamos atrasados tres semanas debido a que el equiPo se embarcó tardes.

Como uno de los analistas de sistemas que han vivido la presión de fijar y reprogramar fechas limite en otro proyectos, usted intenta permanecer tranquilo y evaluar cuidadosamenTe la situación antes de hablar. Lentamente usted planeas a lou posibilidad de postergar las pruebas.

Lou contesta: sí tratamos de retrasar las pruebas hasta las primeras semanas de agosto, en esas fechas dos personas importante de contaValidad estarán de vacaciones. Lou está visiblemente molesto con la posibilidad de retraSar la fecha limite.

Stan Dards, otro miembro reciente de su equipo de análisis de sistemas, entra en la oFicinas de lou. Ambos tienen un aspectos peSimo. Todo está bien, ¿no es cierto? No me reasignaron a programar una aplicación de Nómina, ¿o sí?

A lou no le hace gracia el sentido del humor De Stan ni su aparente preocupación por sí mismo. Todo estaba bien hasta antes de que lle- garas. Estábamos apunto de tomar algunas dediciones importante sobre el calendario. Lou le pasa a Stan el diagrama PERT para que revise.

Observa la fecha de prueba de junio. Como podrás darte cuenta, no hay manera de cumplirla. ¿Algunas brillantes ideas?

Stan completa unos instantes el diagrama, luego señala. Algo ha pasado. Veamos aquí… quizá si movemos el modulo de contabilidad a…

Lou lo interrumpe bruscamente: ¡no!, ya pensamos en eso, pero estanford y bidet, de contabilidad, esta de vacaciones en agosto. Quizás podríamos omitir esas partes de las pruebas. Ellos han sido muy cooperativos. No creo que ponga ninguna objeción si sacamos los sistemas y hacemos las pruebas ya que estamos en las fases de producción.

Creo que esa es una buena idea, lou, coincide Stan, tratando de congraciarse con Lou después de su bromas anteriores. No hemos tenidos ningún problema real con eso, y los programadores son confiables. Así podríamos mantener el calendario con todos los demás. Yo voto por no probar la parte de contabilidad, y solo darle un vistazo cuando se ponga en marcha.

Como el miembro con mas eperiensia del equipo, ¿Qué puede usted argumentar para convencer a Lou y Stan de la importancia de probar el modulo de contabilidad con datos reales? ¿Qué pueden hacer los analistas de sistemas para planificar sus actividades de tal manera que le dediquen un tiempo razonable a realizar las pruebas con datos reales? ¿Cuáles son algunos de los posibles problemas que los miembros de equipos podrían encontrar si no prueban el sistema completamente con datos reales antes de poner el sistema en producción? ¿Hay en el proceso de análisis y diseños de sistemas pasos que pueden omitirse con el propósito de poner al dia un proyecto retrasado?

Los aspectos que debe vigilar con la facilidad con que un usuario aprende el sistema y sus reacciones con su retroalimentación del sistema, incluyendo lo que hace cuando recibe un mensaje de error y cuando se le informa que el sistemas esta ejecutando sus comandos. Ponga especial atención a las maneras en que racionan los usuarios al tiempo de respuesta de sistema y a la redacción de las respuestas. También escuche lo que dicen los usuarios acerca del sistema cuando trabajan con él. Es necesario resolver todos los problemas reales antes de que el sistema se ponga en producción, no solo considerarlo como ajustes al sistema que los usuarios y operadores deben hacer por sí mismos.

Como se mencionó anteriormente, también los manuales de procedimientos necesitan ser probados. Aunque los manuales se puedan corregir por el personal de apoyo, y verificar su exactitud técnica por el equipo de análisis de sistemas, la única forma real de probarlo es tener usuarios y operadores que lo prueben, de preferencia durante la pruebe de sistemas completos con datos reales. Haga que usen versiones precisas, pero no necesariamente versiones finales de los manuales.

Es difícil comunicar con precisión los procedimientos. Demasiado información será con un oct aculo para el uso del sistema. El uso de documento basado en Web puede ayudar en esta consideración. Los usuraos pueden pasar a los temas de interés y recargar e imprimir lo que quieren conservar. Considere las sugerencias de los usuarios e incorpórela en las versiones finales de la página Web, manuales impresos y otras documentaciones.

PRACTICAS DE MANTENIMIENTO

Su objetivo como analistas de sistemas debe ser instalar o modificar sistemas que tienen una vida bastante útil. Quiere crear un sistema cuyo diseño es bastante comprensivo y previsivo para atender las necesidades actuales y proyectadas del usuario durante varios años. Debe usar parte de su experiencia para proyectar lo que podría ser esas necesidades y después construir flexibilidad y adaptabilidad en el sistema. Lo mejor y más fácil del diseño de sistemas será asegurar que el negocio tendrá que gastar menos dineros en el mantenimiento.

Reducir los costos de mantenimiento es una consideración principal, debido a que el mantenimiento de software aislado puede consumir más de 50% del presupuesto de procedimiento de datos para un negocio. Los costos de mantenimiento excesivo se reflejan directamente en el diseñador de sistemas, debido a que aproximadamente 70% de errores de software se han atribuido al diseño de software inadecuado. Desde una perspectiva de sistemas, tiene sentido que detectar y corregir los errores de diseños de software es menos costoso que permitir que permanezcan advertido hasta que sea necesario el mantenimiento.

Por lo regular el mantenimiento se realiza para mejorar el software existente en un lugar de responder a una crisis o falla del sistema. Al igual con el cambio de requerimiento del usuario, el software y la documentación se deben cambiar como parte del trabajo de mantenimiento. Además, los programas se podrían decodificar para mejorar la eficacia del programa original. Mas de la mitad de todo el mantenimiento está compuesto de dicho trabajo de mejoras.

El mantenimiento también se hace para actualizar el software en respuestas a la organización cambiante. Este trabajo no es tan sustancial como mejorar el software, pero se debe hacer. El mantenimiento de emergencia y de adaptación representa menos de la mitad de todo el mantenimiento del sistema.

Parte del trabajo del analista de sistemas es asegurar que en el lugar haya procedimientos y canales adecuados para permitir retroalimentación sobre ─y respuestas sucecuente para ─las necesidades de mantenimientos. Los usuarios deben poder comunicar fácilmente los problemas y sugerencias aquellos que estarán manteniendo el sistema. Es muy desalentador si el sistema no se mantiene adecuadamente. Las soluciones consisten en proporcionar a los usuarios accesos a

Correo electrónicos para el soporte técnico, así como también permitirle descargar actualizaciones de productos o ajusté de Web.

El analista de sistema también necesita establecer un esquema de clasificación para permitir a los usuarios designar las importancias percibidas durante el mantenimiento sugerido o solicitado. Clasificar las solicitudes permite a programadores de mantenimientos entender como estiman los usuarios de importancia de sus solicitudes. Este punto de vistea, junto con otros factores, se pueden tener en cuentas al establecer el mantenimiento.

CÓMO AUDITAR

Auditar es otras formas de asegurar la calidad de información contenida en el sistema ampliamente definido, auditar se refiere a pedirle a un experto, qué no esté involucrado en crear o usar un sistema, examinar la información para determinar su fiabilidad ya sea que la información se establezca o no para ser fiable, el descubrimiento en su fiabilidad se comunica a otros con el propósito de hacer la información del sistema mas útil para ellos.

Generalmente hay dos tipos de auditores b para los usuarios de información: interno y externo. Determinar si ambos son necesarios para los sistemas que usted diseña., dependerá de que tipo de sistema sea. Los auditores internos trabaja para la misma organización que posee el sistema de información, mientras que los externos (también llamados independientes) se contractas por fuera.

Los auditores externos se usan cuando el sistema de información procesa datos qué influyen en las declaraciones financieras de una compañía. Los auditores externos auditan el sistema para asegurar la veracidad de las declaraciones financieras que se producen. También se podrían traer si ocurre algo fuera de lo normal que involucra a los empleados de la compañía, tal como la sospecha de un fraude electrónico o un defalco.

Los auditados internos estudian los controles usado en sistema de información para estar seguro que son adecuados y que esta naciendo lo que deben hacer. También prueban las suficiencias de controles de seguridad. Aunque trabajan para la misma organización, los auditores internos os informan a las personas responsables del sistemas que está auditándole trabajo de los auditores internos con frecuencia es mas detallado que el de los auditores externos.

°°°°°°°°°°°°°°°°

RESUMEN

El analista de sistema usa tres enfoques amplios, de la administración de calidad total (TQM) para analizar y diseñar sistemas de información: diseñar sistemas y software con un enfoque descendente y modular; diseñar y documentar sistemas y software usando métodos sistemáticos; y probar sistemas y software de manera que se pueda mantener y auditar fácilmente.

Seis sigmas son una cultura, filosofía, metodología y enfoque para la calidad que tiene como meta la eliminación de datos los defectos. Los sietes pasos de enfoque seis sigma son: (1) definir el problema; (2) observar el problema; (3) analizar las causas; (4) actual en las causas; (5) estudiar los resultados; (6) estandizar los cambios, y (7) sacar conclusiones.

Los usuarios son extremadamente importantes para establecer y evaluar, desde varias dimensiones, la calidad de los sistemas de información de administración y de los sistemas de apoyo a la toma de decisiones. Se puede involucrar en la evaluación entera de sistemas a trabas del establecimiento de fuerza de tareas de SI o círculos de calidad.

TQM se puede implementar con éxitos al tomar un enfoque descendente (arriba a abajo) para diseñar. Este enfoque se refiere a observar primero lo objetivos generales de las organización y después dividirlos en requerimientos manejables de subsistemas. El desarrollo modular hace la programación, depuración y mantenimientos más fáciles de lograr. La programación en módulos se complementa bien en un enfoque descendente.

Dos sistemas vinculan programas en el entorno de Windows. Uno e DDE (intercambio dinámico de datos), el cual comparte códigos usando archivos de bibliotecas de vínculos dinámicos (DLL). Al usar DDE, un usuario puede almacenar datos en un programa y después usarlo en otro. Un segundo enfoque para vincular programas en Windows es OLE (vinculación e incrustación de objetos). Debido a su enfoque orientado a objetos, este método de vinculación es superior a DDE para vincular datos y gráficos de la aplicación.

Una herramienta recomendada para diseñar un sistema con enfoque descendente y modular se denomina diagrama de estructura. Se usan dos tipos de flechas para indicar los tipos de parámetros que se pasan entre los módulos. El primero se denomina pareja de datos y el segundo se denomina bandejas de control. Los módulos de un diagrama de estructura entran en una de tres categorías: control, transformacional (a veces denominados trabajador) y funcional o especializados.

“este es un lugar fascinante para trabajar. Estoy seguro de que usted coincide conmigo ahora que ha tenido la oportunidad de observarnos. A veces creo que debe ser divertido ser de fuera… ¿no se siente como un antropólogo que descubre una nueva cultura? Recuerdo cuando llegue aquí por primera vez. Todo era tan nuevo, tan extraño. ¡Vaya!, incluso el idioma era diferente. No era un consumidor; era un cliente. No teníamos departamentos; teníamos unidades. No es una cafetería para en empleados; es la taberna. Estos también se aplican a la manera en que trabajamos. Todos tenemos diferentes maneras de enfocar las casas. Creo que ha logrado entender lo que Snowden quiere, pero también de vez en cuando cometo algún error. Por ejemplo, si le doy trabajo en un disco, igual de censillo es para él verlo así que en un informe impreso. ¡Por eso también tengo computadoras en mi escritorio! Siempre lo veo a usted tomar tantos apuntes… sin embargo, creo que todo esto tiene sentido. Se supones que usted documenta los que nosotros hacemos con nuestros sistemas e información, así como lo que su equipo hace, ¿no es así?”

PREGUNTAS DE HYPERCASE

1. Use el modulo FOLKLORE para completar la documentación del sistema GEMS de la unidad de sistemas de información general. Asegúrese de incluir costumbres, anécdotas, proverbios y formas artísticas.
2. En dos párrafos, sugiera una manera de capturar los elementos de FOLKLORE en una PC de tal manera que no sea necesario usar un registro en papel. Asegúrese de que la solución que surgiera incluya gráficos y textos.
3. Diseñe pantallas de entradas y salidas para FOLKLORE que permitan ingresar datos con facilidad, y proporcione mensaje que recuerden de inmediato los elementos de FOLKLORE.

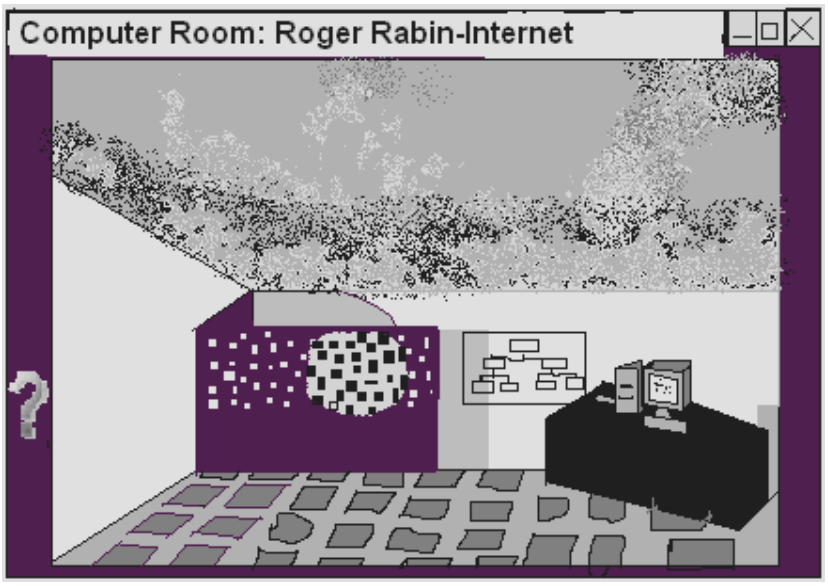


FIGURA 16.HC1

En Hypercase puede utilizar FOLKLORE para documentar formas artísticas que los usuarios hayan creado o corepilado para darle sentido a sus sistemas.

La parte de TQM es para ver que los programas y sistemas se diseñan, documentan y mantienen adecuadamente. Algunas de las técnicas estructuradas que pueden ayudar al analista de sistemas son pseudos códigos, manuales de procedimientos y FOLKLORE. El pseudo código se usa con frecuencia para representar la lógica de cada módulo o diagramas de estructura. El seudo código se usar para repasos estructurados. Los analistas de sistemas deben escoger una técnica que se adapta bien con lo que se usó previamente en la organización y que permita flexicivilidad y fácil modificación.

La prueba de programas específicos, subsistemas y sistemas totales es esencial para la calidad. Las pruebas se hacen para detectar cualquier programa existente con los programas y sus interfaces antes de que el sistema se use realmente. Las pruebas normalmente se hacen en formas ascendente, con códigos de programas que primero se verifican en el escritorio. La prueba del sistema completo con datos reales (datos reales que se han procesados exitosamente con el sistema viejo), se logra siguiendo varios pasos intermedios de pruebas. Esta pruebe proporciona una oportunidad de resolver cualquier problemas que surjan antes de que el sistema se ponga en producción.

El mantenimiento del sistema es una consideración importante. El software bien diseñado puede ayudar a reducir los costos de mantenimientos. Los analistas de sistemas necesitan establecer canales para reducir la retroalimentación del usuario en las necesidades del mantenimiento, debido a que los sistemas que no se mantienen quedarán aspsoleto. Los sistemas Web pueden ayudar al respecto al proporcional acceso electrónico con personal técnico.

Los auditores internos y externos se usan para determinar la falibilidad de la información del sistema. Ellos comunican sus resultados de las auditoras a otros para mejorar la utilidad de la información del sistema.

PALABRASS Y FACES CLAVE

Acoplamiento de sello
Administración de calidad total (TQM)
Auditor interno
Bandeja de control (interruptor)
Biblioteca de vínculos dinámicos (DLL)
Círculos de calidad de SI
Desarrollo modular
Diagrama de estructura
Diseño ascendente
Diseño descendente
Documentación de software
FOLKLORE
Intercambio dinámico de datos (DDE)
Mantenimientos de software
Modulo de control
Modulo funcional
Modulo transformacional
Parejas de datos
Pruebas completas sistemas con datos de pruebas
Pruebas completas sistemas con datos reales
Pruebas de programas con datos de pruebas
Pruebas de vínculos con datos de prueba (prueba de cadenas)
Pseudos código
Repaso estructurado
Seis sigmas
Subordinación inadecuada
Verificación de escritorio
Vinculación e incrustación de objetos (OLE)

PREGUNTAS DE REPASO

1. ¿Cuáles son las tres enfoques amplios disponibles para analista de sistemas para lograr la calidad en los sistemas recientemente desarrollados?
2. ¿Cuál es el factor mas importante para establecer y evaluar la calidad de sistemas de informaron o sistemas de apoyo a la toma de decisiones? ¿por qué?
3. Defina el enfoque de administración de calidad total (TQM) conforme se aplica al análisis y diseño de sistemas de información.
4. ¿Qué significa el término *seis sigmas?*
5. ¿Qué es un circulo de calidad SI?
6. Defina el significado de hacer un repaso estructurado. ¿Quién debe estar involucrado? ¿Cuándo se debe hacer un repaso estructurado?
7. Mencione las ventajas de tomar un enfoque descendente para diseñar.
8. ¿Cuáles son las tres desventajas principales de tomar un enfoque descendente para diseñar?
9. Defina el desarrollo modular.
10. Menciones cuatros lineamientos para la programación modular correcta.
11. ¿Cómo ayudan los diagramas de estructura al analista?
12. Mencione los dos tipos de flechas usados en los diagramas de estructura.
13. ¿Por qué queremos mantener el número de flechas al mínimo al usar los diagramas de estructura?
14. ¿Por qué se deben pasar hacia arriba las banderas de control en los diagramas de estructura?
15. Mencione dos formas en que el diagrama de flujo de datos ayuda a construir diagramas de estructura.
16. Menciones las tres categorías de módulos. ¿por qué se usan en los diagramas de estructura?
17. ¿Cómo puede ayudar un sitio de Web a mantener el sistema y su documentación?
18. Proporcione dos razones que apoyen la necesidad de sistemas bien desarrollado y documentación de software.

19. defina los pseudos códigos.
20. menciones las cuatros quejas principales que los usuarios expresan sobre los manuales de procedimiento.
21. ¿en cuales cuatros categorías el método de documentación de FOLKLORE recopila la información?
22. menciones seis lineamientos para escoger una técnica de diseño y documentación.
23. ¿de quien es la responsabilidad principal para probar los programas de cómputo?
24. ¿cuales es la diferencia entre datos de pruebas y datos reales?
25. ¿Cuáles son los dos tipos de auditores de sistemas?

……..

PROBLEMAS

1. Unos de los miembro de sus equipo de análisis ha estado desalentado los comentarios de los usuarios sobre los estacares de calidad, argumentados que debido a que ustedes son los expertos, realmente son los únicos que hacen lo que constituye un sistema de calidad. En un párrafo, explique a los miembros de su equipo ¿Por qué es importante obtener las opiniones de los usuarios para la calidad de sistemas. Use un ejemplo.

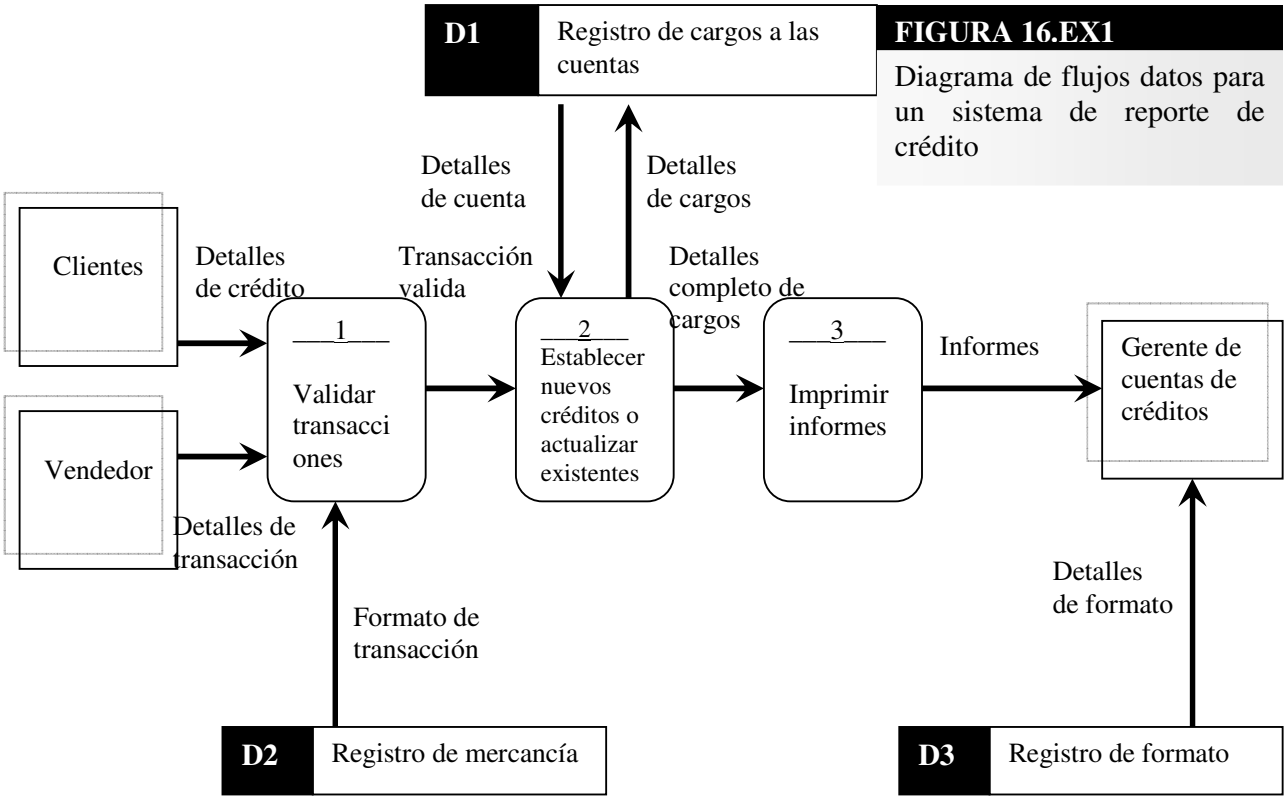


FIGURA 16.EX1
Diagrama de flujos datos para un sistema de reporte de crédito

2. dibuje un diagrama de estructura para el sistemas de reporte crédito en la figura 16.EX1.
3. escribe el pseudocodigo para el problema numero 2.
4. escribe el pseudocodigo para la política de renta de citron car proporcionada en la oportunidad de consultaría 9.3.
5. escribe una tabla de contenido detallada para un manual de procedimientos que explique a los usuarios como conectarse a la red de computo de su escuela, así como también la política de la red (quien es un usuario autorizado, etc.). asegúrese de que el manual se escriba pensando en el usuario.

6. su equipo de análisis de sistema está cerca de completar un sistema para meecham feeds. Roger está bastante seguro de que los programas que ha escrito para el sistema de inventario meecham funcionario como se requiere, debido a que son similares a los programas que han hecho antes. Su equipo ha estado muy ocupado y le gustaría empezar a realizar las pruebas completas de sistemas tan pronto como sea posible.

   Dos miembros jóvenes de su equipo han propuesto lo siguiente:

   a. Omitir la verificación del escritorio de los programas (debido a que programas similares se verifican en otras instalaciones; rober está deacuerdo).
   b. Hacer la prueba de vínculos de cantidades de datos para comprobar que el sistema funcionará.
   c. Hacer la prueba completa del sistema para la prueba de datos reales para comprobar que el sistema está funcionando.

   Responda a cada uno de los tres pasos del programa de prueba propuesto. Explique sus respuestas en un párrafo.

7. proponga un plan de pruebas modificado para meecham feeds (problema 6). Divida su plan en una secuencia de pasos detallados.

……
PROYECTOS DE GRUPO

1. Divida su grupo en dos subgrupos. Un subgrupo debe entrevistar a los miembros de otro subgrupo sobre sus experiencias al registrarse en una clase. Se deben diseñar preguntas que ayudará a documentar el proceso del registro de su escuela.
2. Reúna a sus grupos para desarrollar una página Web para un extracto de un manual de FOLKLORE que documente el proceso de registro para una clase, basado en el FOLKLORE que se utilizó en las entrevistas del proyecto 1. recuerde incluir ejemplos de costumbre, anécdotas, proverbios y formas artísticas.

……
BIBLIOGRAFÍA SELECCIONADA

Dean, j. w., jr y j, r. Evans, *total quality*, st. Paul, mn: west, 1964.

Deming, q. e., *management for quality and productivity*, Cambridge, MA: MIT center for advanced engineering study, 1981.

Juran,j. m., *managerial breakthrough,* new York: McGraw-Hill, 1964.

Kendall, j. e. y p. kerola, ºA Foundation for the use of Hypertext based documentation techniquesº, *journal of END USE Computing,* Vol. 6, num. 1, invierno de 1994, pp. 4-14.

Kendall, K. E. y R. Losee, ºinformation System FOLKLORE: A New Techninque for System Documentationº*, Information and management*, Vol. 10, num. 2, 1986, pp. 103-111.

Kendall, K. E. y S. Yoo, Pseudocodigo-Box Diagrams: An Approach to More Understandable, Productive, And Adaptable Software Desing and Codingº *International Journal on Policy and Information*, vol. 12, num. 1, junio de 1988, pp. 39-51.

Lee, S. M. y M. J. *Schniederjans, Operations Management, Boston*: Houghton-Mifflin, 1994.

CASO DE LA CPU

EPISODIO

ALLEN SCHMIDT, E KENDALL Y KENNETH E. KENDALL

16

DIAGRAMACION DE LA ESTRUCTURA

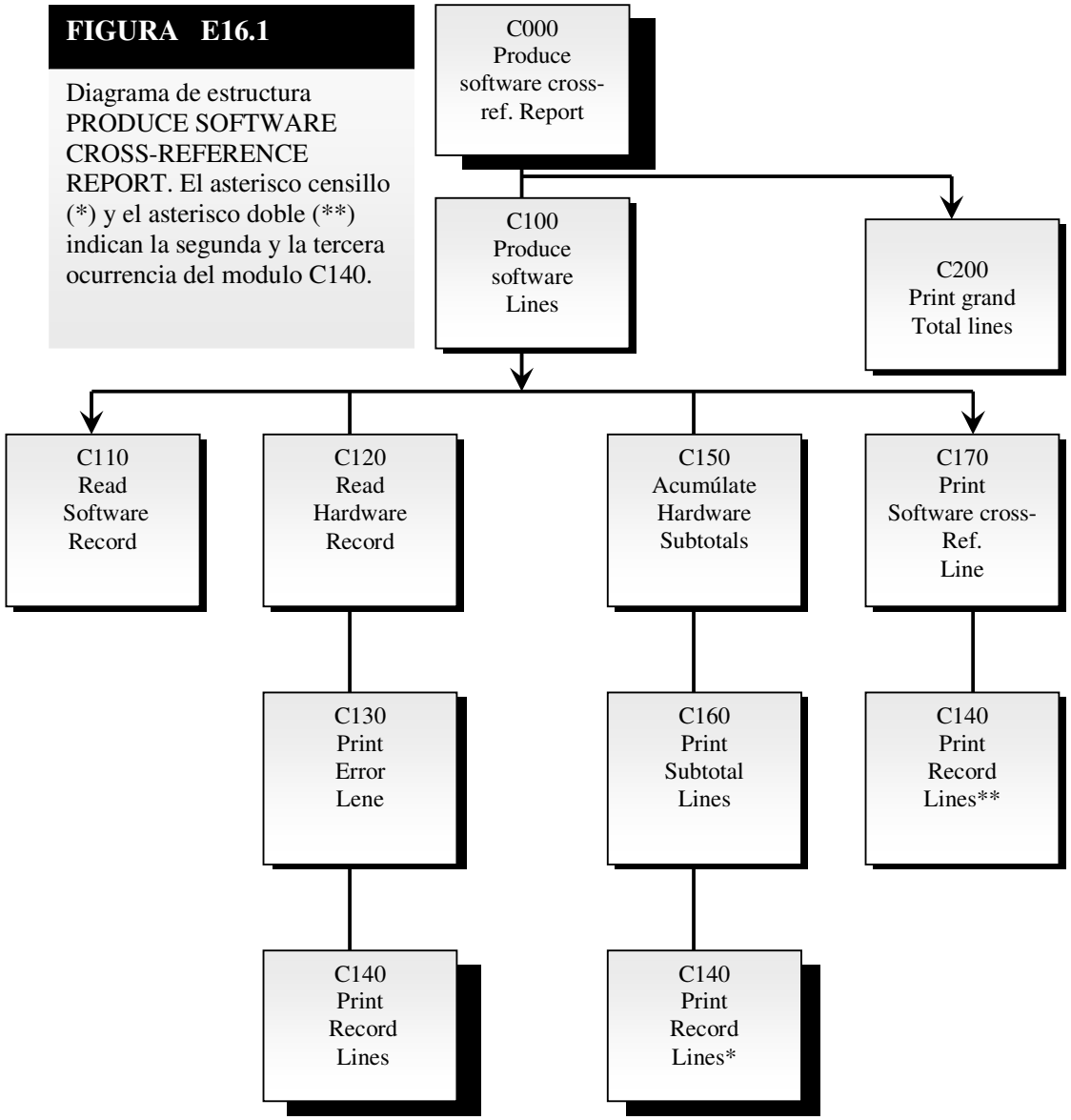
Aquí las tienes, como lo prometimos, dicen chip y anna triunfalmente al entregar sus especificaciones a Mack ROE, EL PROGRAMADOR DEL PROYECTO.

GRACIA º, responde Mack. ºT.tengo mucho trabajo por delante. º

MACK empieza a crear un diagrama de estructura para cada programa y luego pasar al diseño de cada módulo. En la figura e 16.1 se muestra el diagrama de estructura PRODUCE SOFTWARE CROSS-REFERENCE REPORT. La ºcº antes de cada numero de modulo se refiere al CROSS-REFERENCE REPOT(visible Analyst requiere

FIGURA E16.1

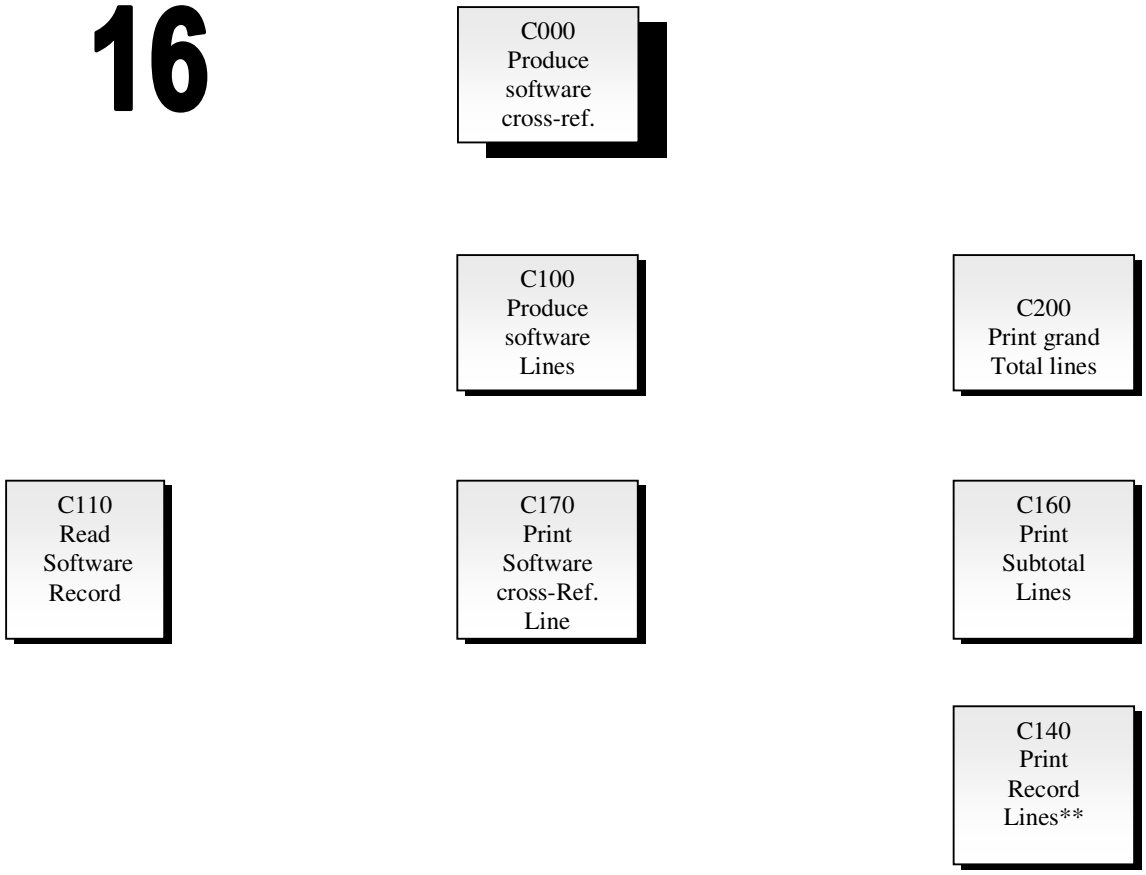
Diagrama de estructura PRODUCE SOFTWARE CROSS-REFERENCE REPORT. El asterisco censillo (*) y el asterisco doble (**) indican la segunda y la tercera ocurrencia del modulo C140.



CASO DE LA CPU

EPISODIO

16



CASO DE LA CPU

EPISODIO 16

una letra como primer carácter en el nombre de un modulo). Este borrador es el primero que Mack utiliza en un repaso estructurado con Dee Ziner, una programada con experiencia.

Dee Ziner tiene varias sugerencias importantes para mejorar la estructura. Ella dice: ºEl modulo C130, PRINT ERROR LINE, está erróneamente subordinado al modulo de llamada C120, READ HARDWARE RECORD. Se podría hacer la pregunta: º ¿El programa debe imprimir una línea de error para leer un HARDWARE RECORD ¿º Puesto que la repuesta es no, el modulo debe colocarse en el mismo nivel que C120, READ HARDWARE RECORDº.

Dee continua analizando la situación con Mack, y dice: “Lo mismo ocurre con el modulo C160, PRINT SUBTOTALS LINES. Esta no es una función de acumulación de subtotales de hardware y por lo tanto no se debe llamar desde el modulo C150, ACUMULATE HARDWARE SUBTOTALS”. Dee continua el repaso y planea la pregunta: “¿Un SOFTWARE RECORD se podría localizar en muchas maquinas?” Mack responde que eso si es válido, y otro modulo de control, PRINT SOFTWARE CROSS-REFERENCE LINE, se incluye en el diagrama de estructura.

Mack procede a incorporar los cambios al diagrama de estructura. Cuando se establece la jerarquía correcta, se agrega al acoplamiento. Se pone especial cuidado a pasar el mínimo de datos y a pasar control sólo hacia arriba del diagrama de estructura. En la figura E16.2 se ilustra la versión final. El modulo C116 es nuevo, y se utiliza el archivo SOFTWARE/HARDTWARE RELATION para enlazar un SOFTWARE RECORD se pasa hacia abajo el modulo y el archivo de relación se lee de manera aleatoria. EL HARTWARE INVENTORY NUMBER y el interruptor de control RELATION NOT FOUND se pasan hacia arriba de la estructura.

El diagrama de estructura final tiene una forma funcional. Unos cuantos módulos de control en la parte superior de la estructura, varios módulos trabajadores en la mitad y unos cuantos módulos especializados en la parte inferior le dan un aspecto general desquilibrado. Todos los nombres de los módulos tiene la estructura verbo-adjetivo-sustantivo, y describen la tarea que se realiza una vez que termina la ejecución del modulo. Por ejemplo, el modulo C150 tiene el verbo “accumulate”, que describe la tarea que desempeña el modulo “subtotals”, un sustantivo, se acumula, y “hardware” describe cuales subtotales se acumulan.

Cada uno de los módulos del diagrama de estructura se describió en el depósito. La figura E16.3 ilustra la pantalla que describe la función del modulo C100, PRODUCE SOFTWARE LINES. Observe que la descripción del modulo contiene pseudocódigo que muestran la lógica del modulo. Dado que PRODUCE SOFTWARE LINES es un modulo de control, sus lógicas deben consistir de bucles y tomas de decisiones, con muy pocas instrucciones relativas a los detalles del procesamiento, como ADD o READ.

CASO DE LA CPU EPISODIO 16

Cada pareja de datos y control del diagrama de estructura se podría describir también en el depósito. El área **related to** ofrece un vinculo a las entradas del deposito que contiene los detalles de los elementos contenidos en las parejas de datos.

"Bueno, creo que estamos apunto de terminar los diagramas para los programadores", comenta Chip.

"De crear los diagramas, sí", responde ana, "pero lo podemos dar un poco mas", "¿Qué quieres decir?", pregunta Chip, un tanto desconcertado.

"Usemos visible analyst para general las tablas de la base de datos para Microsoft Access" responde Anna. "pienso que podríamos comenzar con unas de las entidades principales, como SOFTWARE MASTER, y utilizar la características de generación de códigos de visible Analyst."

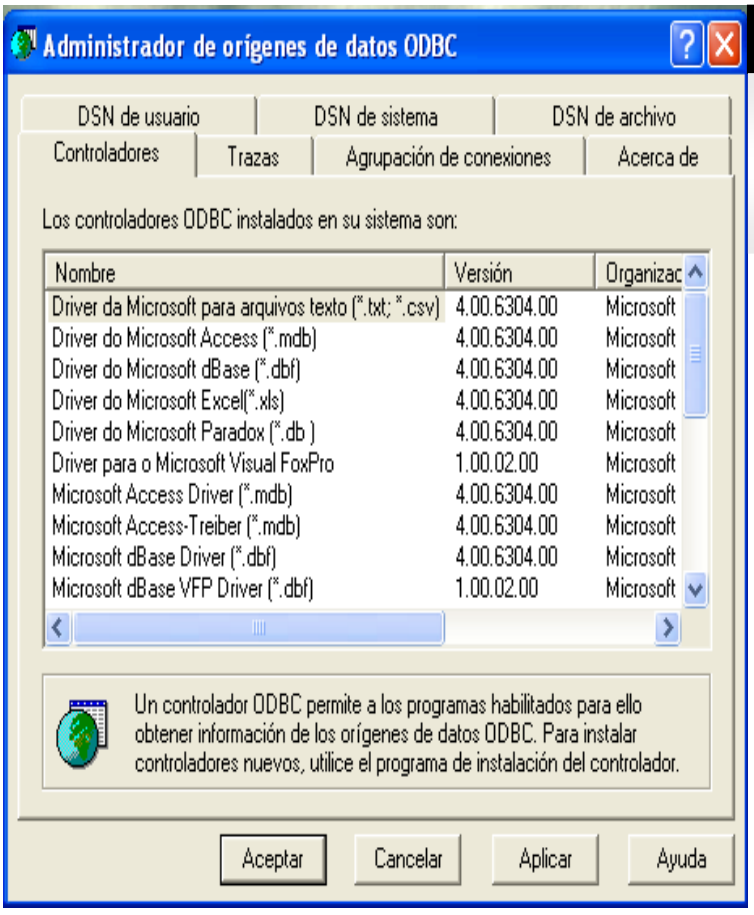


FIGURA E16.3

Pantalla del deposito para el modulo PRODUCE SOFTWARE LINES.

CASO DE LA CPU EPISODIO 16

Anna y Chip proceden a trabajar con visible Analyst para asegurarse de que se hayan definido todo los elementos de SOFTWARE MASTER. Anna hace clic en **Repository** y en **Generaate Database Schema.** Seleccionan el diagrama COMPUTER SYSTEM y le dan el mismo nombre al esquema. Se genera el esquema completo para el sistema de cómputo. E la figura E13.4 se ilustra una parte del código que se generó.

"Voy a copiar una parte de l esquemas para trabajar sólo con el SOFTWARE MASTER", comenta Anna a chip. Anna copia el código SQL generado para el archivo SOFTWARE MASTER. El siguiente paso es crear una consulta en banco en Microsoft Access. Anna ejecuta Microsoft Access y crea la consulta. A continuación hace clic en el botón SQL y pega el archivo SOFTWARE MASTER en la ventana SQL.

"Tengo que cambiar el nombre de la tabla por el de SOFTWARE y cambiar el tipo de consulta a Make Table", continua Anna. Le da el nombre SOFTWARE a la nueva tabla. Anna hace clic en el botón **Run Quero** y sierra la consulta.
"¿Qué ocurrió? ", pregunta Chip DESCONCERTADO. "No veo ninguna salida."

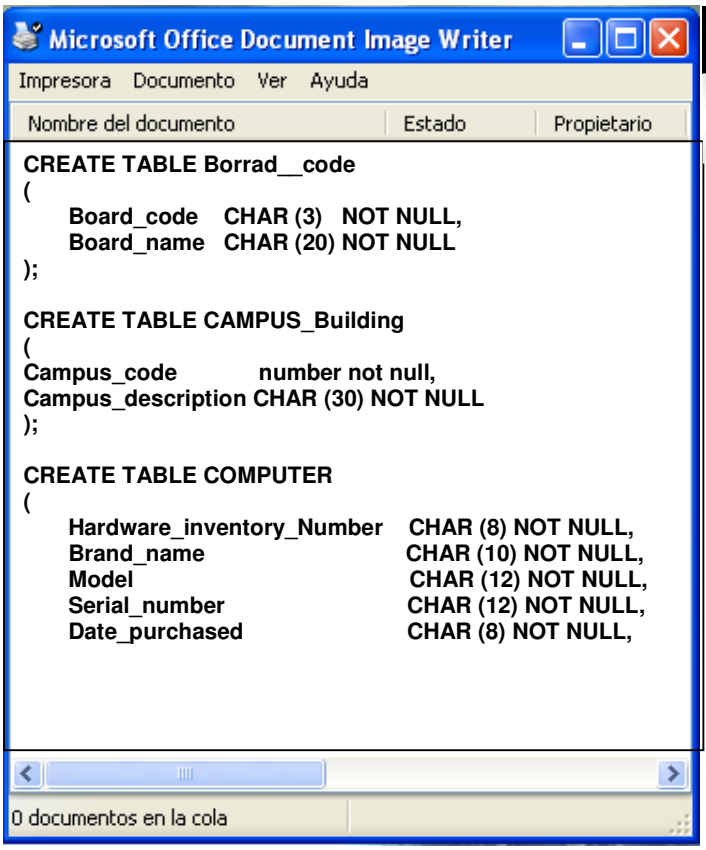


FIGURA E16.4

Ejemplo de generación de código.

CASO DE LA CPU

EPISODIO

16

Anna hace clic en el botón **Tables.** "¡Dale un vistazo a nuestra nueva tabla"¡, exclama Anna. Ella hace clic en la tabla SOFTWARE y en la vista diseño. "Aquí tienes nuestra estructura de visible Analyst, implementada en Microsoft Access."
"Ahora se ve fenomenal", DICE Chip con una amplia sonrisa.
Los siguen generando tablas hasta que finalizan el diseño.
"Creo que podemos dejarle el resto a los programadores", comenta Anna. "Haríamos mejor en comenzar el desarrollo de los planes de pruebas para cada programa."

Los planes de pruebas contienen detalles acerca de cómo determinar si los programas funcionan correctamente, y se lo envía a Mack y Dee, quienes crearán los datos para las pruebas. En cada archivo de prueba que se utiliza en los programas por lotes se incluyen datos validos e inválidos. Lo mismo ocurre con los sistemas interactivos, excepto que los datos de prueba se escriben en formas semejantes al diseño de las pantallas. Una vez que Mack termina de probar sus programas y se convence de que funcionan correctamente, reta a Dee a que encuentre algún error en los programas. A su vez, Dee le pide a Mack que pruebes sus programas en una especie de competencia asmitoza. Ambos están consientes de que los programadores no siempre detectan sus propios errores, porque conocen tanto sus propios programas que podrían ignorar errores sutiles de lógicas.

EJERCICIOS

1. Vea el diagrama de estructura PRODUCE SOFTWARE CROSS-REF REP. Haga dable clic en algunos módulos para ver sus entradas en el depósito.
2. Modifique el diagrama de estructura PRODUCE HARDWARE INVESTMENT RPT. Agregue la función PRINT INVESTMENT LINE en el rectángulo vacío que se proporciona. PRINT HEADING LINES y WRITE REPORT LINE estan subordinados a Este modulo. Describe cada function en el desposito.
3. Modifique el diagrama de estructura del archive CHANGE COMPUTER. Incluya el símbolo de bucle y agregue los siguientes módulos subordinados al modulo 160, CHANGE COMPUTER RECORD (también vea abajo):

    A. SHOW CHANGE DISPLAY
    B. ACCPET COMPUTER CHANGES
    C. VALIDATE CHANGES
    D. DISPLAY ERROR MESSAGE
    E. CONFIRM CHANGES

Los siguientes módulos deben subordinarse al modulo 220, PUT COMPUTER RECORD:

CASO DE LA CPU

EPISODIO

A. FORMAT COMPUTER RECORD
B. REWRITE COMPUTER RECORD

4. Modifique el diagrama de estructura ADD SOFTWARE RECORD agregando un símbolo de bucle y acoplando las conexiónese siguientes acoplamiento se debe colocar en la línea de conexión arriba de cada modulo (también vea abajo):

A. Modulo:	DISPLAY ADD SOFTWARE SCREEN
Pasando hacia arriba:	ADD SOFTWARE SCREEN
B. Modulo:	ACCEPT ADD SOFTWARE SCREEN
Pasando hacia arriba:	EXIT INDICATOR (control)
	ADD SOFTWARE SCREEN DATA
C. Modulo:	VALIDATE ADD SOFTWARE DATA
Pasando hacia abajo:	ADD SOFTWARE SCREEN DATA
Pasando hacia arriba:	CANCEL TRANSACTION (control)
	VALID ADD SOFTWARE DATA
D. Modulo:	READ SOFTWARE RECORD
Pasando hacia abajo:	SOFTWARE INVENTORY NUMBER
Pasando hacia arriba:	RECORD FOUND (control)
E. Modulo:	VALIDATE HARDWARE REQUIREMENTS
Pasando hacia abajo:	ADD SOFTWARE SCREEN DATA
Pasando hacia arriba:	VALID DATA (control)
	ERROR MENSSAGE
F. Modulo:	DISPLAY ERROR MENSSAGE
Pasando hacia abajo:	ERROR MENSSAGE
G. Modulo:	PUT NEW SOFTWARE RECORD
Pasando hacia abajo:	VALID ADD SOFTWARE DATA
H. Modulo:	FORMAT SOFTWARE RECORD
Pasando hacia abajo:	VALID ADD SOFTWARE DATA
Pasando hacia arriba:	FORMATTED SOFTWARE RECORD
I. Modulo:	WRITE SOFTWARE RECORD
Pasando hacia abajo:	FORMATTED SOFTWARE RECORD

5. Elabore el diagrama de estructura PRINT PROBLEM MACHINE REPORT. A continuación se da un esquema de los módulos, con cada módulos subordinados sangrados:

```
PRINT PROBLEM MACHINE REPORT
    PRINT PROBLEM MACHINE LINES
        READ MACHINE RECORD
        DETERMINE PROBLEM MACHINE
        PRINT PROBLEM MACHINE LINES
          PRINT HEADING LINES
          WRITE REPORT LINE
    PRINT FINAL REPORT LINES
        WRITE REPORTLINE
```

CASO DE LA CPU

EPISODIO

6. Elabore el diagrama de estructura CHANGE SOFTWARE RECORD. Los módulos del programa con sus módulos subordinados sangrados:

CHANGE SOFTWARE FILE
- CHANGE SOFTWARE RECORD
  - GET SOFTWARE RECORD
    - DISPLAY SOFTWARE ID SCREEN
    - ACCEPT SOFTWARE ID SCREEN
    - FIND SOFTWARE RECORD
    - DISPLAY ERROR LINE
  - OBTAIN SOFTWARE CHANGES
    - DISPLAY CHANGE SCREEN
    - ACCEPT SOFTWARE CHANGE
    - VALIDATE CHANGES
    - DISPLAY ERROR LINE
  - PUT SOFTWARE RECORD
    - FORMT SOFTWARE RECORD
    - REWRITE SOFTWARE RECORD

7. Elabore el diagrama de estructura SOFTWARE DETAILS INQUIRY. los módulos se enlistan con sus módulos subordinados sangrados:

INQUIRE SOFTWARE DETAILS
- INQUIRE SOFTWARE RECORD
  - GET SOFTWARE RECORD
    - DISPLAY SOFTWARE ID SCREEN
    - ACCEPT SOFTWARE ID SCREEN
    - FIND SOFTWARE RECORD
    - DISPLAY ERROR LINE
  - DISPLAY INQUIRY SCREEN
    - FORMAT SOFTWARE INQUIRY SCREEN
    - DISPLAY SOFTWARE INQUIRY SCREEN

8. Vea el diagrama de flujo del sistema ADD COPMUTER.

9. Modifique el flujo del sistema ADD SOFTWARE. Agregue los siguientes rectángulos del diagrama abajo del proceso manual INSTALL SOFTWARE. Incluye los archivos e informes de entrada y salida especificados para cada programa:

Programa:	UPDATE SOFTWARE RELATIONAL FILE
Entrada:	UPDATE SOFTWARE INSTALLATION LIST, documento
	UPDATE INTALLED SOFTWARE SCREEN, pantalla
Salida:	SOFTWARE RELATIONAL FILE, disco
	INSTALLED SOFTWARE TRANSATION, disco
Programa:	PRINT USER NOTIFICATION REPORT
Entrada:	INSTALLED SOFTWARE TANSATION, disco
Salida:	USER NOTIFICATION REPORT, informe.

CASO DE LA CPU

EPISODIO

10. Elabore el diagrama de flujo del sistema ADD STAFF. Hay dos programas: ADD

11. STAFF y PRINT NEW STAFF LIST. La entrada para el diagrama ADD STAFF es un listado NEW STAFF y una pantalla de entrada ADD NEW STAFF. El archivo STAFF MASTER se actualiza y se produce un nuevo archivo NEW STAFF LOG. Este último archivo es entrada para el programa PRINT NEW STAFF LIDT, que produce el informe NEW STAFF LIST.

12. Diseñe datos de pruebas en papel para probar el programa ADD COMPUTER. Utilice Microsoft Access para probar la pantalla. Anote cualquier inconsistencia.

13. diseñe datos de pruebas y resultados previstos para el programa ADD SOFTWARE. Utilice Microsoft Access para probar la pantalla y anote si los resultados se apagaron a sus predicciones.

14. diseñe datos de pruebas y resultados previstos para el programa ADD TRANINIG CLASS. Utilice Microsoft Access para probar la pantalla y anote si los resultados se apagaron a sus perdiciones.

15. diseñe datos de pruebas en papel para probar el programa CHANGE SOFTWARE EXPERT. Utilice Microsoft Access para probar la pantalla. anote cualquier inconsistencia.

Anderson Méndez
Estudiante de informática
Del politécnico de azua rep. Dom
Trabajo final de procesadores de texto