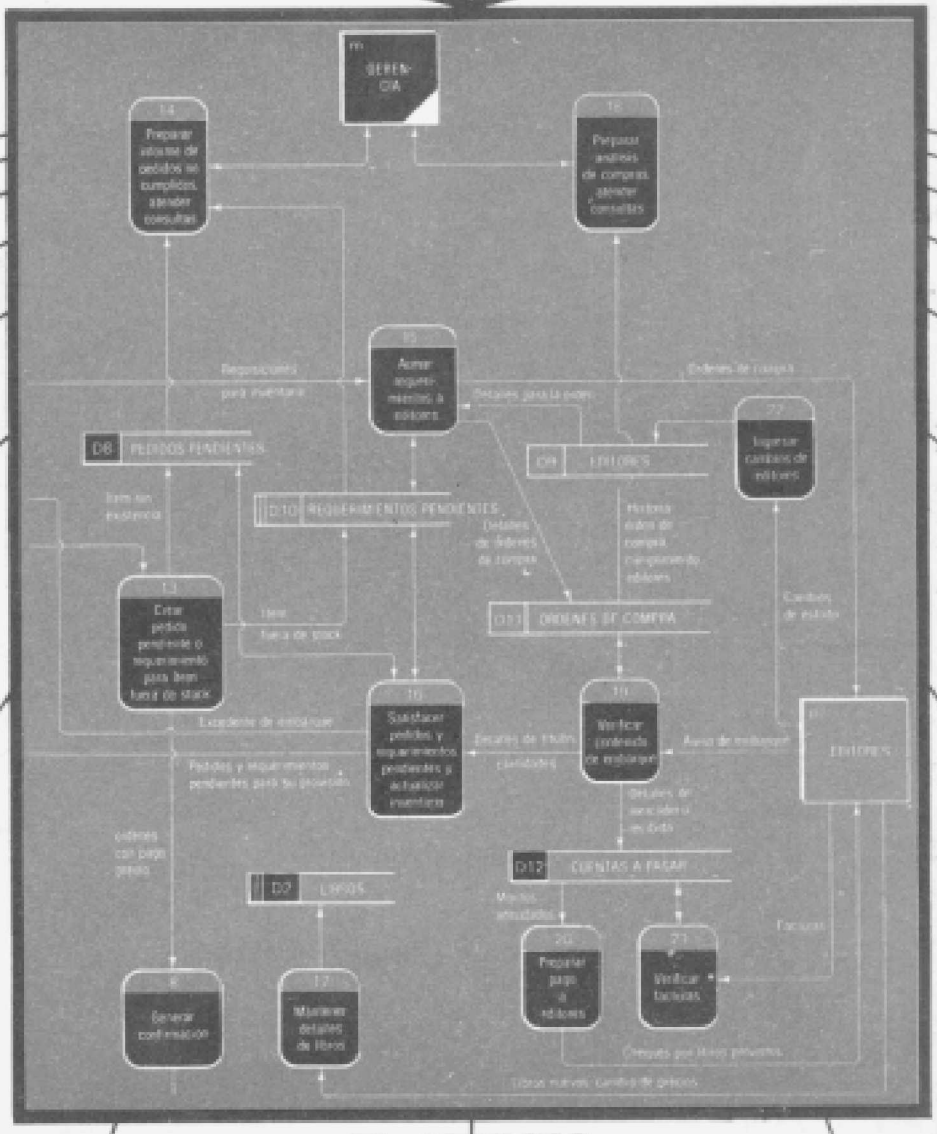
Chris Gane – Trish Sarson

ANALISIS ESTRUCTURADO DE SISTEMAS



EL ATENEO

SISTEMAS DE PROCESAMIENTO ELECTRONICO

Alkins, A. C. — **Computadores y procesamiento de datos**
Bellavoine, C. — **¿Qué es una computadora?**
Benice, D. D. — **Temas de computación electrónica**
Brady, A. H. - Richardson, J. T. — **Lenguaje de programación BASIC (SEPA)**
Clark, F. J. — **Procesamiento de información**
Couger, J. D. - Shannon, L. E. — **Introducción al lenguaje FORTRAN (SEPA)**
Christie, L. G. - Curry, J. W. — **El ABC de los microcomputadores**
Donovan, J. J. — **Programación de sistemas**
Ekman, T. - Nilsson, K. — **COBOL**
Elliot, C. O. — **Introducción al procesamiento de datos (SEPA)**
Flores, I. — **Arquitectura de bases de datos**
Flores, I. - Terry, Ch. — **Sistemas de microcomputación**
Gane, Ch. - Sarson, T. — **Análisis estructurado de sistemas**
García Poquet, J. — **Computación para todos**
Gardner, A. — **Programación estructurada: LCP práctico**
Jensen, K. - Wirth, N. — **Pascal**
Kallin, S. — **FORTRAN**
Langefors, B. — **Teoría de los sistemas de información**
Langefors, B. - Samuelson, K. — **Información y datos en los sistemas**
Lyon, J. K. — **Bases de datos**
Morange, P. — **Introducción a la programación**
Myers, G. J. — **El arte de probar el software**
Parker, A. J. - Stewart, J. F. — **Programación BASIC para contadores**
Shipman, C. — **Cómo programar su computador personal. BASIC elemental para PC**
Tomlin, R. — **Introducción de la computadora en la empresa**
Vazsonyi, A. — **Introducción a la computación electrónica**
Wirth, N. — **Introducción a la programación sistemática**

ANALISIS ESTRUCTURADO DE SISTEMAS

Chris Gane

Trish Sarson

Traducción
Ing. Julio C. Abramoff
Sociedad de Computación Argentina

Cuidado de la edición
Dr. Edmundo Said
Universidad Católica Argentina (UCA)

LIBRERIA "EL ATENEO" EDITORIAL
BUENOS AIRES - LIMA - RIO DE JANEIRO - CARACAS
MEXICO - BARCELONA - MADRID - BOGOTA

Serie: Sistemas de procesamiento electrónico
Director: Dr. Federico Frishknecht
Título de la obra original: "Structured Systems Analysis: Tools and Techniques"

Librería, Editorial e Inmobiliaria, Florida 340, Buenos Aires.
Fundada en 1912 por don Pedro García.

ISBN 950-02-5261-2

IMPRESO EN LA ARGENTINA

INDICE

PREFACIO

Tenemos confianza en las técnicas descritas en este libro. Han probado que sirven en un área dificultosa del procesamiento de datos: el análisis y la definición de aquello que debe hacer un sistema nuevo para que tenga mayor valor según el criterio de las personas que pagan por ello.

La disciplina consiste en un conjunto creciente de herramientas y técnicas que han sacado partido del éxito de la programación y el diseño estructurados. El concepto fundamental es la construcción de un modelo lógico *(no físico) de un sistema, empleando técnicas gráficas que permiten a usuarios, analistas y diseñadores obtener un cuadro claro y común del sistema, y, mostrarles cómo se ensamblan sus partes para satisfacer las necesidades de los usuarios. Hasta el desarrollo de las herramientas del análisis estructurado de sistemas, no había forma de mostrar las funciones lógicas básicas y los requerimientos de un sistema; se caía rápidamente en los detalles de la implementación física actual o de la propuesta.*

El libro comienza con una discusión sobre algunos de los problemas que encontramos en el análisis y, luego, se pasa revista a las herramientas gráficas y la forma de combinarlas para obtener un modelo lógico. Luego tomamos una herramienta por vez y las tratamos en detalle en los Capítulos 3 al 7, comenzando con la herramienta clave, el diagrama lógico de flujo de datos. Como empleamos herramientas para construir un modelo lógico, la forma en que desarrollamos el sistema es algo diferente a los enfoques tradicionales; en el Capítulo 8 bosquejamos una metodología para el desarrollo de sistemas estructurados aprovechando las ventajas de las nuevas herramientas. Esta metodología involucra la construcción de un sistema desde arriba hacia abajo mediante refinamientos sucesivos, *produciendo primero un flujo de datos de todo el sistema, luego desarrollando flujos de datos detallados, posteriormente definiendo el detalle de las estructuras de datos y la lógica de los procesos, y por fin para entrar en el diseño de una estructura modular, etc. Analizamos de arriba hacia abajo, diseñamos de arriba hacia abajo, desarrollamos de arriba hacia abajo, probamos de arriba hacia abajo. Incluso reconocemos que el buen desarrollo incluye* iteraciones; *se debe estar preparado para perfeccionar el modelo lógico y el diseño físico a la luz de la información resultante del uso de una versión temprana de este modelo o diseño.*

Distinguimos el trabajo del analista (definiendo "qué" deberá hacer el sistema) del trabajo del diseñador (definiendo "cómo" deberá hacerlo), reconociendo que a menudo el analista hace diseño y los diseñadores también, a menudo, hacen análisis. Parte del valor del análisis estructurado de sistemas radica en proveer al diseñador las entradas necesarias para definir los programas de modo de obtener la máxima cambiabilidad, o facilidad para su cambio, empleando diseño estructurado. En el Capítulo 9, repasamos la importancia de la cambiabilidad y las técnicas y conceptos del diseño estructurado, tomando un sistema real, analizándolo y diseñándolo hasta el nivel de módulo.

Finalmente en el Capítulo 10, discutimos los beneficios que aparecen al cambiar los procedimientos tradicionales por estas nuevas técnicas, con sus implicancias en la administración del control de los proyectos, y los beneficios que se pueden esperar.

Hemos tratado en lo posible de no introducir nuevos términos; pero, como la disciplina se basa en el diseño estructurado (que tiene su propio vocabulario) y en la teoría de la base de datos relacional (que tiene también su propio vocabulario), aparecen algunos términos no familiares. Cada uno de estos términos se explica en su primera aparición y también es definido en el Glosario que se encuentra al final del libro.

Esperamos que usted *encuentre útiles estas herramientas y técnicas, ya sea un analista de sistemas, un diseñador, un gerente o un usuario de los servicios de procesamiento de datos. Desearíamos saber sobre sus experiencias en el empleo del análisis estructurado de sistemas, particularmente si las desea compartir con otros.*

Agradecemos la ayuda de aquellos que nos han autorizado a reproducir su material registrado y las contribuciones realizadas para el desarrollo de estas ideas por parte de nuestros anteriores colegas de Yourdon, Inc., Tom de Marco, Víctor Weinberg y Ed Yourdon.

CHRIS GANE
TRISH SARSON

1

LA NECESIDAD DE MEJORES HERRAMIENTAS

Por muchas razones, el análisis de sistemas es la parte más dificultosa del desarrollo de un sistema de procesamiento de datos. No se trata simplemente de la dificultad técnica del trabajo, si bien muchos proyectos exigen que el analista tenga conocimientos profundos de la tecnología actualizada de procesamiento de datos. No se trata simplemente de las dificultades políticas que se presentan, especialmente en grandes proyectos, donde el nuevo sistema deberá servir a varios grupos de interés, posiblemente en conflicto. Tampoco se trata solamente de los problemas de comunicación que surgen en cualquier situación en la cual personas de diferentes antecedentes, con distintos enfoques del contexto y diferentes vocabularios, deban trabajar juntas. Es la resultante de estas dificultades lo que hace al análisis de sistemas tan duro y exigente: el hecho es que el analista debe hacer de intermediario entre la comunidad de los usuarios —aquellos que tienen la sensibilidad de sus problemas, pero encuentran dificultoso explicarlo y además tienen vagos conocimientos de cómo los puede ayudar el computador— y la comunidad de los programadores —aquellos que están ansiosos de que la organización cuente con una importante sección de procesamiento de datos, pero no poseen la información que permite saber qué es lo mejor para el negocio—. El analista debe hacer un balance entre aquello que es actualmente *posible* en nuestra tecnología en constante progreso (minis, micros, procesamiento distribuido, bases de datos, comunicación de datos) y aquello que *vale la pena hacer* en la empresa, tal como está dirigida por sus administradores.

Si se hace un balance que resulte aceptable por todas las partes y que pueda resistir la prueba del tiempo, se ha logrado la parte más ardua del esfuerzo; si ha sido bien hecho, cualesquiera sean las dificultades de diseño y programación, el sistema que se construya servirá a las necesidades del negocio. Si ha sido hecho pobremente, cualquiera sea la excelencia de la implementación, el sistema no satisfará las necesidades de la organización y el costo excederá a los beneficios. Para lograr dicho balance, necesitaremos toda la ayuda que podamos obtener. Este libro presenta algunas herramientas que han sido probadas satisfactoriamente.

1.1 ¿QUE FRACASA EN EL ANALISIS?

Los problemas que el analista debe enfrentar están todos relacionados; ésta es una de las razones de su dificultad. Podemos distinguir cinco aspectos que merecen comentarse:

Problema 1. El analista encuentra muy difícil aprender lo suficiente del negocio para poder ver los requerimientos del sistema a través de los ojos del usuario. (Cuando empleamos el término *negocio* queremos significar a la empresa de cualquier organización con o sin fines

de lucro.) Una y otra vez oímos decir: "Hemos hecho un sistema técnicamente excelente, pero no era lo que el usuario deseaba." ¿Por qué habrá sido así? ¿Por que no podrá el analista estudiar sencillamente el negocio y reunir los elementos suficientes como para poder especificar el sistema correcto? En el corazón de estos problemas encontramos el hecho de que muchos gerentes usuarios son "ejecutores" más que "explicadores". Conocen y manejan la información que necesitan de una forma intuitiva, sin pensar en términos de flujo de información o de lógica de decisión. Esto es natural; se llega a gerente *tomando* decisiones correctas y *haciendo* tareas superiores y no explicando necesariamente cómo debe hacerse la tarea y cómo deben tomarse las decisiones. Pero esto significa que el analista no tiene derecho a esperar por parte de los usuarios una explicación brillante de los requerimientos del sistema; debe ayudarlos a resolver los problemas. Al mismo tiempo, el analista no posee el don de la telepatía; no puede saber aquello que no le han contado. Este hecho penoso se descubre particularmente en términos de la importancia relativa que los usuarios dan a las distintas partes del sistema. Supongamos que un gerente en particular desee un informe del movimiento diario de caja. ¿Qué es más importante, que lo reciba a las 08.30 horas aunque existan algunos ítem sin completar o bien que lo conozca hasta el detalle de los centavos, aunque ello signifique recibirlo algunos días a las 11.00 horas? El gerente lo sabe muy bien y podrá decir: "¡Vea, cualquier tonto que conozca algo de negocios debería saberlo!" Pero es arduo llegar en el negocio al nivel de sentir intuitivamente la mejor solución de compromiso.

Problema 2. La comunidad usuaria no conoce aún lo suficiente de procesamiento de datos como para saber lo que es factible y lo que no lo es. La publicidad sobre computadoras, en general, no da a las personas una idea específica y real sobre aquello que pueden o no pueden hacer. Muchas personas no tienen idea de las posibilidades que puede brindar una terminal CRT en línea ¿y por qué habrían de tenerla? La tecnología es todavía muy reciente para que la mayoría de las personas tenga un conocimiento básico y una orientación que le permita imaginarse la forma en que podría afectarles un nuevo sistema. Los medios populares no han ayudado; la imagen que tienen de las computadoras es, o bien que son caras y con errores sin sentido, o bien, como en ciencia ficción, de que cajas con decisiones propias se apoderan del mundo.

Comparemos esta situación con las ideas de las personas, digamos por ejemplo, con las vinculadas a la industria de la construcción. Pensemos en un empresario que nunca antes ha encargado la construcción de una fábrica, pero que ha entrado y salido de las fábricas durante toda su vida de trabajo y se ha formado un conocimiento completo que le permite entender todas las cosas de las que su *arquitecto* le habla . Además, una fábrica es muy semejante a las otras; por lo menos tendrán mucho más en común entre sí que, digamos, un sistema en lote con un sistema en línea. Nuestros problemas son mayores en el procesamiento de datos que los de la industria de la construcción, como consecuencia, en gran medida, de que no tenemos manera de hacer un *modelo* de aquello que queremos construir. En un proyecto de ingeniería o de construcción, cualquiera sea su dimensión, el arquitecto discutirá los requerimientos de sus clientes y entonces podrá producir un modelo que represente cómo se verá la estructura terminada. Cualquier interesado podrá mirar el modelo, relacionarlo con sus experiencias anteriores con este tipo de estructuras, y formarse una clara idea de lo que podrá obtener por su dinero. Las herramientas del análisis estructurado de sistemas nos permitirán producir un modelo gráfico de un sistema que podrá jugar casi el mismo rol que el modelo de una construcción o de una refinería de petróleo.

Problema 3. El analista puede rápidamente verse abrumado por los detalles, tanto por los detalles del negocio como por los detalles técnicos del nuevo sistema. La mayor parte del tiempo de la fase de análisis del proyecto se emplea en obtener información detallada de la situación actual, los procedimientos administrativos, los documentos de entrada, los informes producidos y requeridos, las políticas en uso y los miles de hechos que surgen de una cosa tan compleja como es una empresa real. A menos que exista algún esquema o estructura para organizar estos detalles, el analista (y aun todo un equipo de analistas) podrá verse

sobrecargado con hechos y papeles. Los detalles son necesarios y deberán estar disponibles cuando se los requiera, pero el analista deberá tener herramientas para controlar los detalles, o de lo contrario encontrará que el árbol no le dejará ver el bosque. Parte del valor de un enfoque de arriba hacia abajo, tal como veremos más adelante, consiste en que le permitirá mirar el gran cuadro y luego ubicarse en el detalle de cada pieza, cuando y como sea necesario.

Problema 4. El documento donde ubicamos los detalles de un nuevo sistema (el cual puede denominarse de diferentes maneras: especificacion del sistema, diseño general, especificación funcional, o cualquier otro nombre equivalente) constituye un contrato efectivo entre el departamento usuario y el grupo de desarrollo de sistemas, si bien es frecuentemente imposible para dichos usuarios comprenderlo debido a su volumen y a los conceptos técnicos que contiene. De cierta manera recuerda el viejo estilo de las pólizas de seguro; las cosas que realmente son importantes se escriben en letra pequeña. Los usuarios a menudo hacen un esfuerzo valiente para conocer a fondo estos documentos y terminan encogiéndose mentalmente de hombros y los firman, diciéndose a sí mismos "Bueno, creo que esta gente de computación sabe lo que va a hacer." Solo cuando se les entrega el sistema terminado tendrán algo que pueden entender y recién podrán reaccionar; por supuesto, será demasiado tarde.

Problema 5. Si el documento de especificaciones *pudiera* escribirse de manera que tuviera sentido para los usuarios, quizá no sería de mucha utilidad para los diseñadores físicos y programadores que deben construir el sistema. A menudo, deberán hacerse una cantidad considerable de iteraciones en el análisis, duplicando en esencia el trabajo que el analista ha realizado, para redefinir datos y procesos lógicos en términos que los programadores puedan utilizar. Aunque el analista tenga conocimientos técnicos y de esa manera escriba las especificaciones con un ojo puesto en la facilidad de la programación, finalizará limitando la libertad de acción de los programadores para implementar el sistema de la mejor manera. El diseño físico de los archivos, programas y métodos de entrada/salida deberá ser hecho por alguien que tenga un conocimiento técnico actualizado, basado en la comprensión de los requerimientos lógicos completos del sistema. Comenzar a especificar el diseño físico antes que el modelo lógico del sistema esté terminado es ser "físico prematuramente" y a menudo, da como resultado un sistema de inferior calidad.

1.2 ¿PODEMOS CULPAR A NUESTRAS HERRAMIENTAS?

Aun contando con las mejores herramientas analíticas posibles, nos encontraremos con algunos de los problemas que hemos discutido recién. No existen, por ejemplo, herramientas analíticas que permitan al analista saber lo que tiene en su mente el usuario si éste no se lo dice. No obstante, el tema de este libro es que los problemas de análisis pueden ser significativamente facilitados con las herramientas lógicas que describiremos, e identificamos cuatro limitaciones en nuestras actuales herramientas analíticas.

1.2.1 No existe "modelo" en el procesamiento de datos

No tenemos forma de mostrar a los usuarios un *modelo tangible vívido* del sistema. A los usuarios les es muy difícil imaginarse lo que podrá hacer el nuevo sistema hasta que éste se encuentre realmente en operación y para ese entonces, ya será tarde. "¿Cómo puedo saber qué quiero hasta ver lo que tengo?" es el grito oculto de muchos usuarios. Las herramientas gráficas de este libro darán al usuario un mejor "modelo" del sistema de lo que fue posible tener hasta ahora.

1.2.2 La narración en lenguaje corriente es muy vaga y embrollada

Si no tenemos una forma de mostrar un modelo tangible, debemos tener lo mejor que puede reemplazarlo, que es el empleo de la narración en idioma corriente para describir el sistema propuesto. ¿Puede usted imaginarse pagando cinco años de sueldos por una casa construida a su deseo, sobre la base de una exhaustiva descripción narrativa de cómo se construirá la casa? Sin fotografías, sin planos, sin visitas a una casa similar —sólo 150 páginas narrativas. "La sala de estar que mira al sudsudeste tendrá 8 × 5 m de ancho máximo, con su mitad oeste en forma trapezoidal, la pared oeste tendrá 4 m de largo (lindando la parte norte de la pared este con la cocina)..."

¿Habiendo pagado el dinero en base a la descripción narrada y no habiendo visto nada hasta que la casa se ha terminado, estaría usted sorprendido si se desilusionara al mudarse? ¿Nos sorprendería que los usuarios se desilusionaran cuando se les entregan los sistemas?

Si usted utiliza el idioma corriente para describir un sistema complejo (o pará construirlo) el resultado será tan voluminoso que resultaría muy difícil para el lector entender cómo se combinan las partes. Peor que esto, como veremos en el Capítulo 5, el idioma corriente tiene problemas de construcción, lo que hace muy difícil emplearlo donde se requiere precisión.

1.2.3 Los flujogramas hacen más mal que bien

Si no podemos hacer un modelo y el idioma corriente es muy vago y difuso, entonces, ¿qué pasaría con un dibujo? Desafortunadamente hasta ahora el único dibujo que disponemos para un sistema es el flujograma. Aunque un flujograma *puede* valer por mil palabras, atrapa al analista con una obligación; utilizar los símbolos normalizados de flujogramas (ver Fig. 1.1) significa inevitablemente que el analista debe entregar la *implementación física* del nuevo sistema. El solo hecho de dibujar un diagrama de flujo significa que se ha tomado una decisión de cómo será la entrada: en tarjetas o a través de una terminal CRT, qué archivos estarán en cinta y cuáles en disco, qué programas tendrán salidas y cuáles no, etc. Sin embargo, estas decisiones son esenciales para la tarea de los *diseñadores.* Una vez que el analista dibujó un flujograma del sistema ¿qué le queda para hacer al diseñador? Este tiene la opción entre aceptar el diseño físico del analista y continuar con los detalles del programa y de la estructura de los archivos, o (como sucede muy a menudo) retornar a las especificaciones escritas y producir un nuevo diseño a partir de ellas. Ninguno de estos cursos de acción es satisfactorio. Con palabras de Fred Brooks:

> El manual (especificación) debe describir no solo todas las cosas que el usuario ve, incluyendo las interfaces, sino que debe contener aquello que el usuario *no* ve. Esta es la actividad del implementador y aquí su libertad no puede ser restringida. El arquitecto (analista) debe estar siempre preparado para mostrar *una* implementación para cualquier sección o aspecto que describa, pero no debe intentar dictar *la* implementación. [1.1]

Si el analista y el diseñador son la misma persona, el dibujo del flujograma debe tomarse como una acción de diseño, no de análisis. Existe una gran tentación hacia el bosquejo de un diseño físico del nuevo sistema antes de haber comprendido completamente todos los requerimientos lógicos; éste es el significado que tiene la expresión "prematuramente físico."

También es mucho más difícil, una vez iniciado el diseño, considerar situaciones alternativas. Los flujogramas para un sistema en línea y para otro que lleva a cabo las mismas funciones lógicas pero en lote, son muy diferentes. La similitud lógica fundamental es imposible de ver. Para colmo de males, el símbolo de decisión del flujograma alienta al creador a entrar en el detalle en las decisiones —por ejemplo, ¿qué pasa con las transacciones inválidas?— y el diagrama general se ha convertido, antes de ser bien conocido, en el flujograma detallado de un programa. Necesitamos los detalles pero no en el estudio global. El analista necesita desesperadamente tener la posibilidad de ver "un mapa del bosque." No

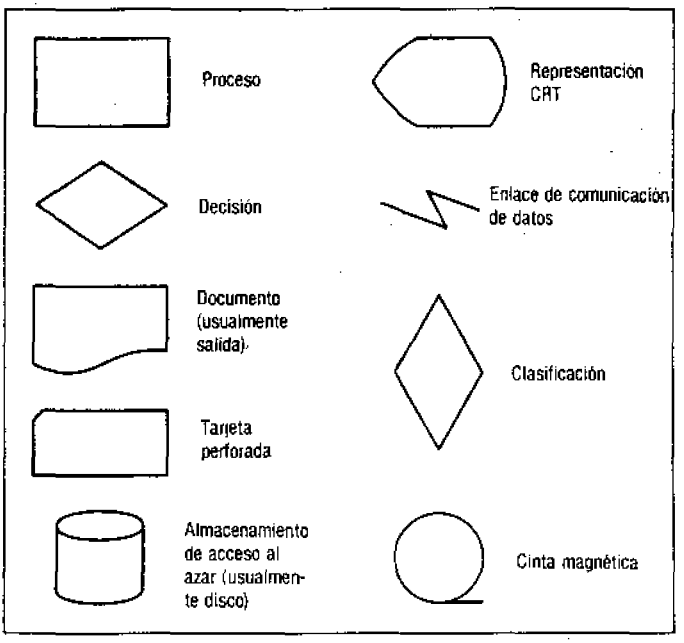


Figura 1.1 Símbolos convencionales de diagramas de flujo (de la plantilla de diagramación IBM X20-8020).

es siempre verdad que el flujograma sirva de ayuda para "modelar" el sistema para los usuarios. Si bien es cierto que algunos usuarios expertos pueden aprender a leer flujogramas, para la mayoría son "jerigonza visual".

1.2.4 No poseemos una forma sistemática para registrar las preferencias y las soluciones de compromiso del usuario, especialmente en términos de acceso inmediato de los datos

Como se ha indicado, el analista necesita detectar las preferencias del usuario (a menudo intuitivas) respecto de los diversos aspectos del nuevo sistema. Para algunos usuarios no importa que los valores tengan el 100% de precisión mientras el informe esté disponible sin falta a las 09.00 horas, mientras que otros están dispuestos a esperar, siempre y cuando los valores que lleguen sean los correctos. Obviamente, podemos dejar contentos a todos pero solo a un precio. Estos datos de las soluciones de compromiso de los usuarios son muy importantes para el diseñador cuando compara la relación costo/eficiencia de varios diseños. Pero, a menos que el analista tenga una herramienta para registrar preferencias explícitamente, la información se perderá. Este es un tema particularmente dificultoso cuando los datos de un archivo deben ser de acceso inmediato en diferentes formas, por ejemplo, en el clásico problema de ventas donde cualquier vendedor suministra varios artículos y cualquier artículo puede ser suministrado por varios vendedores. Si organizamos el archivo por vendedor, podemos fácilmente dar una respuesta inmediata a la pregunta "¿Qué artículos ha suministrado el vendedor A?" Pero nos resta la pregunta "¿Quién suministra el artículo 123456?" Para dar una respuesta inmediata a esta pregunta puede ser necesario construir un índice secundario o alguna otra técnica de base de datos. ¿Pero cuán importante es esta segunda pregunta? ¿Es vital para la forma en que el usuario maneja su negocio? ¿O es solamente "algo lindo", que se mencionó al analista por un gerente al finalizar una entrevista

y se puso dentro de las especificaciones sin mayor análisis? En el Capítulo 7 discutiremos las herramientas para el registro y análisis de las preferencias.

1.3 ¿CUANTO IMPORTAN LAS ESPECIFICACIONES FUNCIONALES?

Podemos utilizar las *herramientas de análisis* estructurado de sistemas descriptas en este libro para preparar una especificación funcional que

1. Sea bien interpretada y cuente con el completo acuerdo de los usuarios,
2. Fije los requerimientos lógicos del sistema sin imponer una implementación física, y
3. Exprese las preferencias y las soluciones de compromiso.

¿Pero qué hacemos con ello? ¿Qué es lo que nos va a proporcionar? ¿Por qué no aceptamos que el análisis es un arte inexacto y que, dado que los usuarios van a cambiar sus puntos de vista de todos modos, podemos trabajar con los métodos de especificación corrientes?

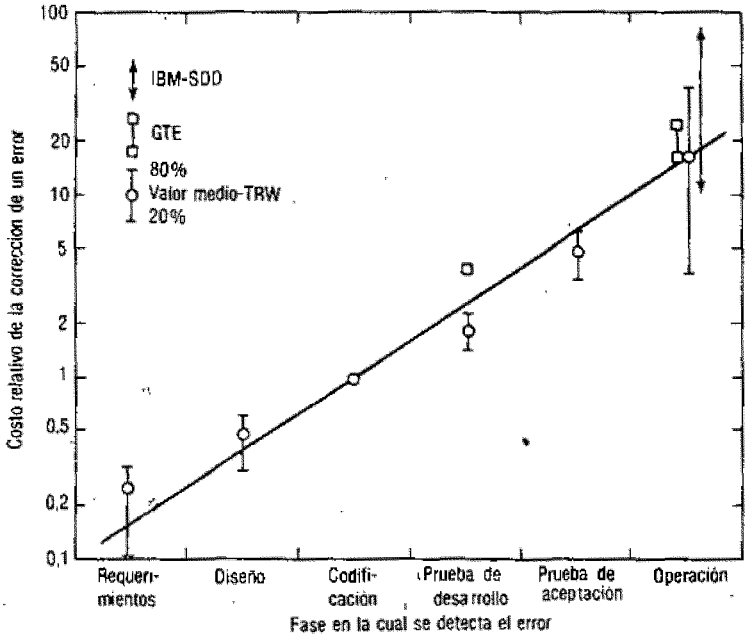


Figura 1.2 Costo relativo de corrección de un error durante el desarrollo del sistema.

Barry Boehm [1.2] ha producido cierta evidencia manifiesta que muestra cómo el costo de corrección de un error crece fuera de toda proporción cuanto más tarde se detecta, en el ciclo de vida del sistema desarrollado (ver Fig. 1.2).

En el gráfico se puede observar el costo relativo de corregir un error —¡nótese la escala logarítmica!— dependiendo de la fase del desarrollo del sistema en que el error es detectado. Así un error que se advierte en la fase de requerimientos —debido tal vez a que el usuario examinó nuestro modelo lógico gráfico— puede costar 0,2 unidades (digamos $ 200), y en cambio, si el mismo error no es detectado hasta que el sistema entra en operación, el costo resulta mayor a 10 unidades ($ 10.000). Así, la construcción de un modelo lógico que comunica claramente al usuario qué es lo que puede y no puede hacer el sistema, es un

ejercicio crucialmente importante en función del costo de corregir los errores más tarde. Realizar los cambios en una hoja de papel es barato; realizar cambios en la codificación es muchas veces más caro, y realizar cambios en un sistema funcionando es muchísimo más caro aún. No es conveniente esperar a que el usuario "vea lo que recibe" antes de que "sepa lo que quiere."

BIBLIOGRAFIA

1.1 F.P. Brooks, *The Mythical Man-Month,* Addison-Wesley, Reading, Mass., 1975.

1.2 B. Boehm, "Software Engineering", *IEEE Transactions on Computers,* Vol. C-25, diciembre de 1976.

2

CUALES SON LAS HERRAMIENTAS Y COMO ENCAJAN ENTRE SI

Antes de examinar en detalle cada una de las herramientas del análisis estructurado, daremos una visión general de cada herramienta mostrando sus relaciones mutuas y viendo cómo se utilizan en un caso de análisis relativamente simple.

La Corporación CBM (Computers Books by Mail - Libros de Computación por Correo) fue adquirida recientemente por una corporación "holding" nacional y es actualmente una división de ella. Establecida hace 12 años, el negocio de la compañía fue actuar como *intermediario de libros* recibiendo pedidos de las librerías sobre libros de computación y pidiendo los libros al editor con su correspondiente descuento para cumplimentar el pedido al recibir los libros de los editores. Las facturas eran confeccionadas por un servicio de computación a partir de los formularios llenados por CBM; el negocio atendía normalmente 100 facturas diarias, cada una con un promedio de 4 títulos de libro con un valor promedio por factura de $ 50. La nueva gerencia planifica expandir considerablemente las operaciones, mejorando los niveles de servicio mediante la conservación en stock de los 100 libros de pedido más frecuente y la admisión de pedidos por parte de todos los profesionales (no solamente libreros) mediante llamada telefónica sin cargo al número 800-372-6657 (800 - DP BOOKS), así como también continuar con los pedidos por correo como en el presente. Todo esto creará problemas de verificación de créditos, por supuesto, y planteará la necesidad de algún tipo de sistema de control de inventario. Los empleados que reciban los pedidos telefónicos necesitarán un acceso rápido a un catálogo de libros para verificar autores y títulos y para poder estar en capacidad de asesorar a quienes consultan, sobre qué libros están disponibles con referencia a determinados temas.

El volumen de transacciones en el nuevo sistema dependerá, por supuesto, de la aceptación de este nuevo método de ordenar libros, que está proyectado para un crecimiento a 1000 facturas diarias, o más, si bien con un promedio menor de libros por factura (teniendo en cuenta que las librerías tienden a pedir más por vez, que los profesionales).

Un analista de sistemas ha sido asignado a la división recién adquirida con la responsabilidad de investigar y especificar el nuevo sistema en representación del Vicepresidente de "Marketing". ¿Cómo puede empezar a construir un modelo lógico del sistema requerido sin caer en las conclusiones de tipo "prematuramente físico", como por ejemplo qué deberá automatizarse, qué deberá ser manual, si deberá el sistema automatizado ser en línea o en lote, si deberá utilizarse el servicio de computación o tener el propio computador, etcétera?

2.1 PRIMERO, DIBUJAR UN DIAGRAMA LOGICO DE FLUJO DE DATOS

En líneas generales podemos decir que al igual que en el sistema actual, el nuevo sistema tomará pedidos de libros, los verificará contra un archivo de libros en existencia, verificará contra algún archivo para determinar si el crédito del cliente es correcto y producirá la orden para que el libro(s) sea despachado con su factura.

Podemos verlo en el diagrama lógico de flujo de datos (DFD; en inglés, data flow diagram) de la Fig. 2.1, donde empleamos los cuatro símbolos que se indican en la Fig. 2.2.

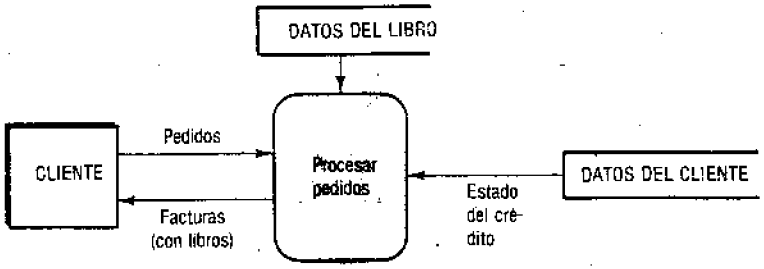


Figura 2.1 Diagrama lógico de flujo de datos.

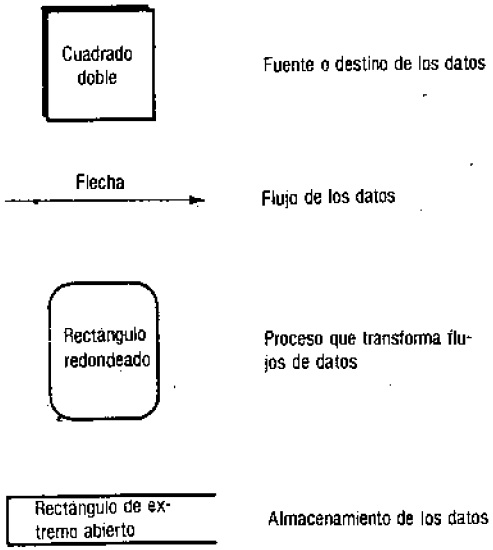


Figura 2.2 Símbolos del DFD.

Estos símbolos y los conceptos que representan, se encuentran en el nivel *lógico;* un flujo de datos puede estar contenido físicamente en una nota, una factura, una llamada telefónica, de programa a programa, —vía enlace de datos por satélite— en cualquier medio por el cual los datos pasan de una entidad o proceso a otro. Un proceso puede ser físicamente una oficina repleta de empleados que calculan descuentos, un procedimiento catalogado en JCL (lenguaje de control de trabajos, en inglés, job control language), o una combinación de actividades manuales y automatizadas. Un almacenamiento de datos puede ser un archivo rotativo de tarjetas, una microficha, un archivo, una tabla en núcleos, o un archivo en cinta o

disco. Empleando los cuatro símbolos disponibles, es posible dibujar un gráfico del sistema sin preocuparnos de cómo deberá ser su implementación.

Por supuesto, el sistema graficado en la Fig. 2.1 es muy general —a tan alto nivel de abstracción que puede resultar bastante inútil—. Sin embargo podemos usar los cuatro símbolos básicos para trazar un "mapa del bosque" a cualquier nivel de detalle que deseemos. Expandamos el "proceso de pedidos" para mostrar las funciones lógicas que realiza el sistema actual. A los novatos podemos indicarles que los pedidos que ingresan deben ser verificados para asegurarnos de que los detalles sean correctos (que el título y el autor se corresponden, por ejemplo). Una vez que se ha validado el pedido, necesitamos despacharlo juntamente con los pedidos de otros libros del mismo editor, de manera que podamos obtener el beneficio del descuento por cantidad.

La Fig. 2.3 muestra cómo ha quedado ahora el diagrama lógico de flujo de datos. Vemos que a medida que cada pedido es verificado se lo coloca en algún almacenamiento de pedidos pendientes hasta que (de acuerdo con cierta lógica que no necesitamos especificar todavía) el despacho de los pedidos pueda ser reunido en una orden masiva.

Hasta ahora todo va muy bien; ¿pero qué pasa con el completamiento de los pedidos y, como esperamos, con su pago? Cada editor remitirá una nota de embarco con cada envío, detallando su contenido; ésta debe compararse con el pedido que le hemos colocado para asegurarnos de que se hayan recibido las cantidades y los títulos correctamente. ¿Dónde encontramos los detalles de los pedidos que colocamos? Evidentemente debe existir otro almacenamiento de datos, llamado tal vez "pedidos a los editores", que pueda ser consultado. Una vez que dispongamos de los libros correctos, podemos armar y despachar los pedidos a nuestros clientes, en forma individual. La Fig. 2.4 muestra el sistema con estos agregados.

Obsérvese que no se ha mostrado el movimiento propio de los libros; para nuestro propósito, los libros no son datos y por ello no se incluyen en el diagrama lógico de flujo de datos. Discutiremos la relación entre un diagrama lógico de flujo de datos y un diagrama de flujo de materiales en el Capítulo 3. Por ahora, nos interesan los ítem, como las notas de embarco, las cuales *son* datos acerca de los libros.

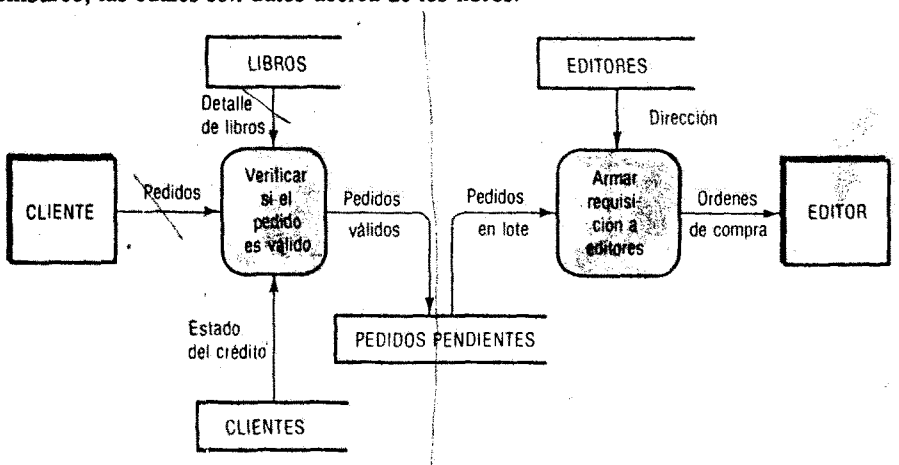


Figura 2.3 Expansión del DFD.

En la Fig. 2.4 hasta ahora nadie ha recibido ningún pago por nada. Necesitamos remitir las facturas a nuestros clientes (hasta ahora lo hace el servicio, pero éste es un detalle físico) y atender el pago de nuestros clientes; como nada es gratis, deberemos a nuestro turno ser facturados por los editores y pagarles. La Fig. 2.5 muestra el agregado de estas funciones financieras con sus almacenamientos de datos asociados, conocidos comúnmente como cuentas a cobrar y cuentas a pagar.

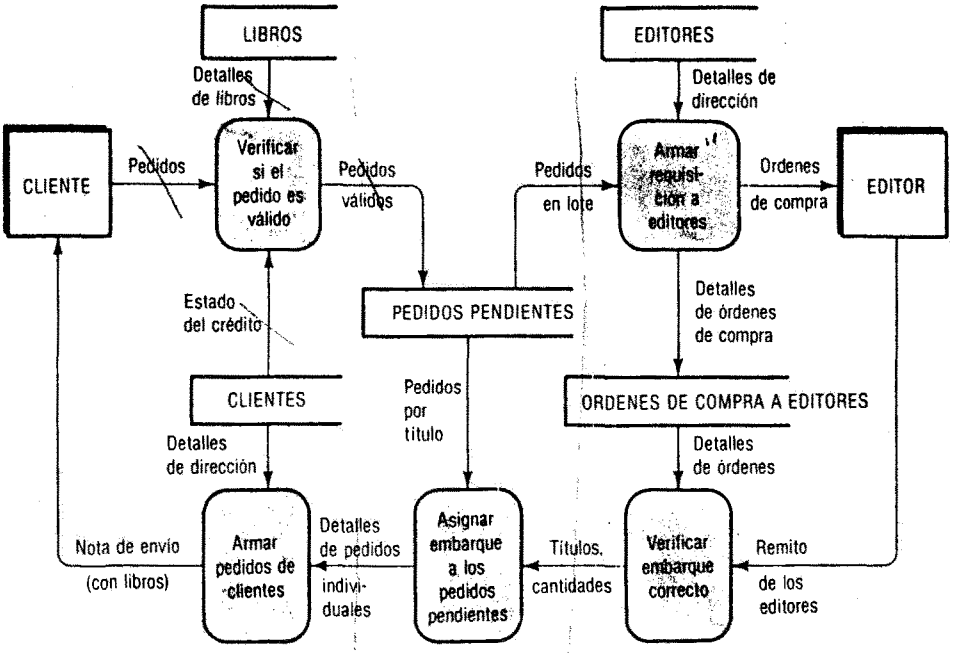


Figura 2.4 Expansión adicional de la DFD.

Por razones de claridad no se indican las funciones lógicas para la creación y mantenimiento del archivo de cliente, del archivo de libros y del archivo de editor y no atendemos consultas. Estas funciones las trataremos en el próximo capítulo.

Por supuesto que cada una de las casillas de proceso que se muestran resumen una cantidad de detalles. Cada casilla de proceso puede *ser explotada* en un nivel inferior más detallado del diagrama lógico de flujo de datos. La Fig. 2.6 muestra la explosión de "reunir requerimientos a los editores."

Si se requiere, cada casilla correspondiente a los procesos componentes puede, a su vez, ser descompuesta a un nivel inferior, a un tercer nivel de detalle. Como veremos en el Capítulo 3, ello a menudo no es necesario. En el Capítulo 3 trataremos en detalle las pautas para dibujar este diagrama lógico de flujo de datos y las convenciones para efectuar las explosiones.

2.1.1 Condiciones de error

Usted se habrá dado cuenta de que no hemos considerado las condiciones de error; no hemos especificado qué pasa con un pedido de un usuario que se encuentra fuera del límite del crédito o qué pasa con una factura de un editor correspondiente a un despacho que nunca recibimos. Por supuesto que debemos considerar estas circunstancias, pero queda su tratamiento para los diagramas de segundo nivel —o inferiores—, de manera que ello no interfiera con el gran "cuadro." Sin esta regla de simplificación es muy fácil empantanarse con el manejo de errores y excepciones, hundiéndose sin dejar rastros.

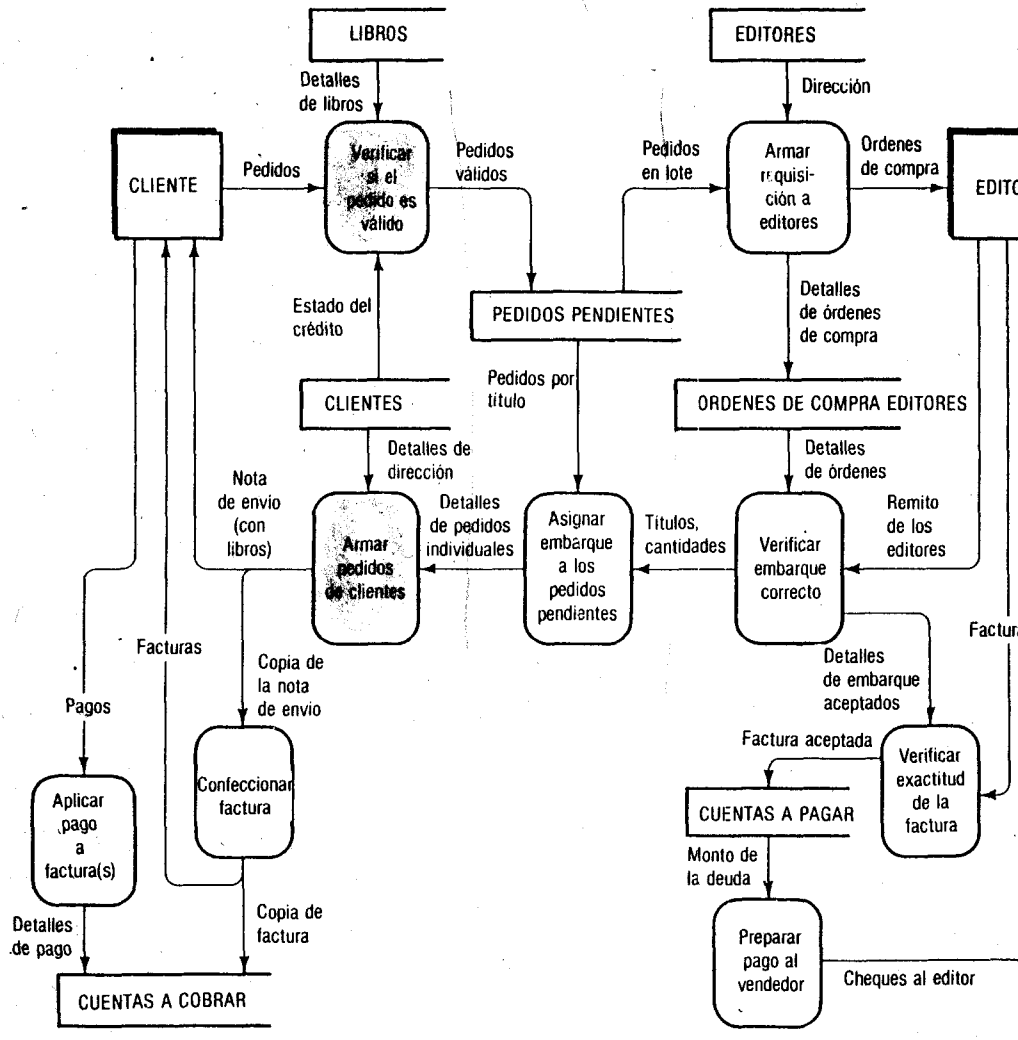


Figura 2.5. DFD completo.

2.1.2 Implementaciones físicas alternativas

Otro punto importante respecto a la Fig. 2.5 es que, dado que es un diagrama *lógico* de flujo de datos, es muy fácil representarse mentalmente diversas implementaciones físicas. Como sabemos, en el sistema presente, "confeccionar factura" y "aplicar pago a factura" son automatizadas por el servicio y el resto de las funciones son manuales. Ahora que tenemos un "mapa del bosque" podemos ver diferentes soluciones alternativas dibujando en el sistema límites alrededor de diferentes procesos y almacenamientos de datos. En la Fig. 2.7 se muestran dos de tales posibilidades.

Límite 1. Automatización de la validación de los pedidos, así como de la facturación y de

las cuentas a cobrar, al tiempo que dejamos en forma manual la acumulación en lotes de los pedidos y todas las funciones de compra. Dentro de esta "región automatizada" podemos representarnos por lo menos dos soluciones físicas.

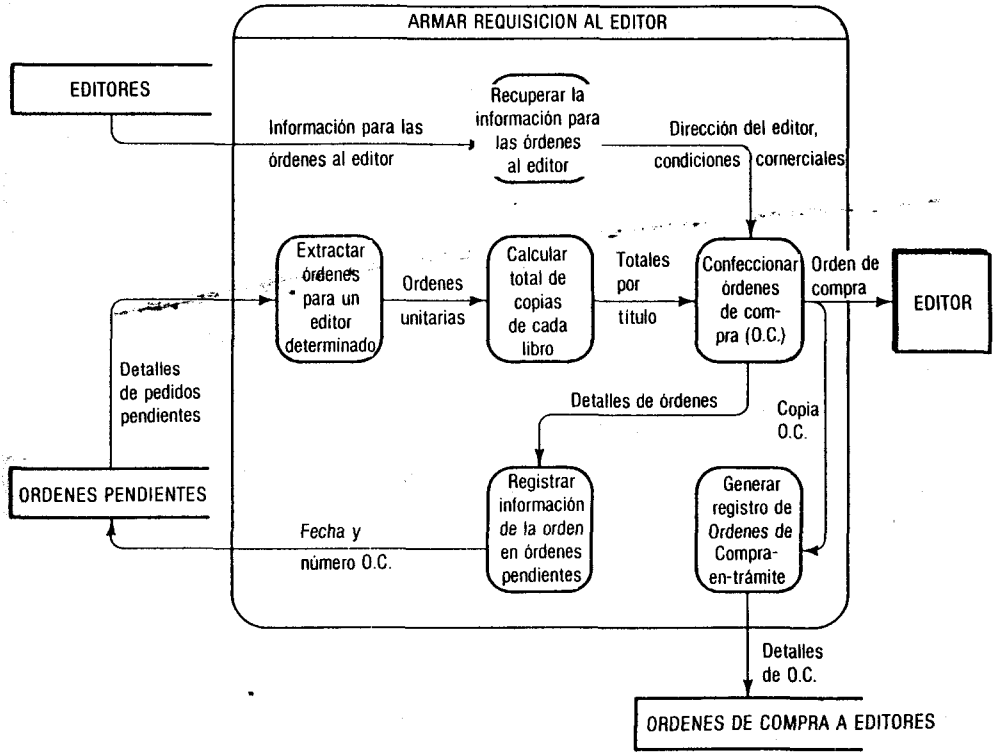


Figura 2.6 Explosión de proceso en otro DFD.

Límite 1: En lote. Los pedidos pueden revisarse para su completamiento y perforación. Después de validarlos contra el archivo de libros y el archivo de clientes, deberán ser agregados al archivo de pedidos pendientes, y debe imprimirse una confirmación impresa de la orden y producirse una salida impresa clasificada por título dentro de editor. Los despachos y los pagos deberán ser perforados y utilizados como entrada de un sistema convencional de cuentas a cobrar.

Límite 1: En línea. A medida que van llegando los pedidos deberán ser introducidos por una terminal CRT con capacidad para llamar a todos los libros por un autor determinado, o por una palabra o una frase del título y con una validación en línea contra el archivo de clientes. Una vez ingresada satisfactoriamente la orden, deberá imprimirse una confirmación en la impresora de la terminal para su control visual contra la orden, actualizándose el archivo de pedidos pendientes.

Límite 2. Automatizar la producción de órdenes de compra y el desglose de los despachos principales en órdenes individuales, así como la entrada de órdenes y las cuentas a cobrar.

Límite 2: En lote. Además del sistema descripto para el Límite 1, este sistema más amplio correrá todas las noches a través de la cinta de pedidos pendientes, extractando aquellos títulos cuyo total de pedidos exceda la cantidad económica de pedido,

clasificando por editor, imprimiendo órdenes de compra y actualizando el archivo de órdenes de editores en trámite. Cuando se reciban los despachos, su contenido deberá perforarse y validarse contra el archivo de pedidos en trámite y luego correrse contra el archivo de pedidos pendientes para confeccionar las notas de embarco por pedidos individuales de los clientes.

Límite 2: En línea. Las funciones adicionales disponibles serán las de controlar de inmediato el contenido del despacho recibido de un editor y generar instrucciones para desglosar en el acto dicho despacho en órdenes individuales.

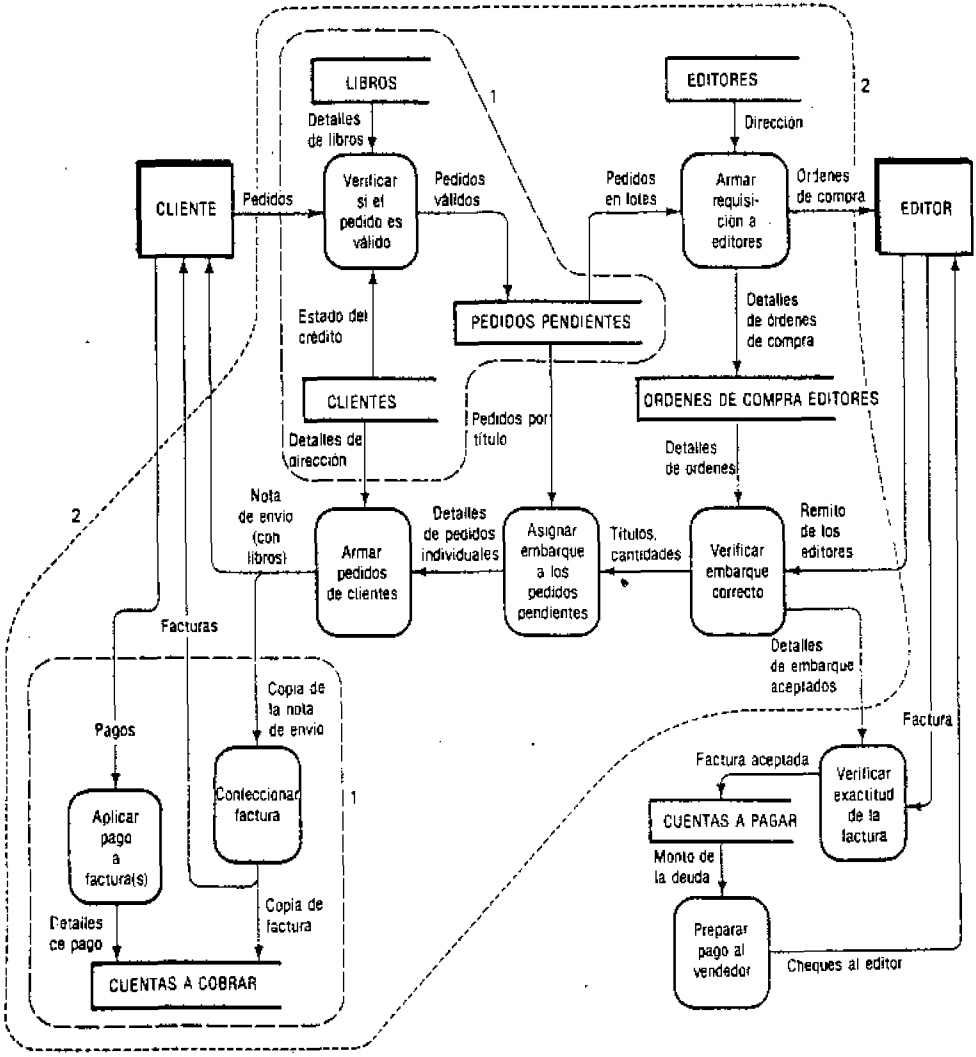


Figura 2.7 Posibles límites de automatización.

Como es de imaginar, resultan factibles varias familias de sistemas. Qué sistema tiene la mejor relación costo/beneficio depende de factores tales como volumen, magnitud de la orden masiva con descuento, valor asignado a la respuesta rápida a los clientes, etc. El punto importante es que cualquiera sea la implementación física que se elija, el resto de las funciones manuales puede ser fácilmente observable, y que tanto las funciones automatizadas como las manuales podrán siempre ser incorporadas al mismo diagrama lógico de flujo de datos.

2.1.3 La clase general del sistema

Deberá quedar claro que la Fig. 2.7, con muy pocos cambios podrá ser aplicada a una empresa distribuidora de autopartes, alimento para ganado, o artículos hospitalarios. Los profesionales de administración de empresas las reconocerán como un miembro de la clase de "operaciones de distribución sin inventario", en otras palabras cualquier empresa que recibe pedidos pequeños los agrupa en pedidos masivos a un gran vendedor al por mayor o fabricante y luego fracciona a éstos para abastecer a los clientes, sin mantener existencias para la provisión en el acto desde los estantes.

En el Capítulo 3 desarrollaremos el diagrama lógico de flujo de datos para el nuevo sistema de CBM, que corresponde a un sistema de distribución *con* inventario.

Se invita al lector a considerar otros sistemas *lógicos* de empresas diferentes. Es un ejercicio muy útil bosquejar un diagrama de flujo de datos para un sistema distinto y marcar los límites de automatización para cada una de las implementaciones que le son familiares.

2.2 A CONTINUACION, COLOCAR EL DETALLE EN UN DICCIONARIO DE DATOS

En los diagramas de flujos de datos de la sección anterior hemos dado a los flujos de datos, almacenamientos de datos y procesos los nombres más significativos y descriptivos posibles, con la condición de ser suficientemente cortos para caber en el diagrama. Tan pronto como empezamos a observar con mayor detalle, por ejemplo, al tratar de responder a la pregunta ¿qué quiere usted significar exactamente con "pedidos"?, nos vemos introducidos inmediatamente en un sinnúmero de detalles, especificando posiblemente el diseño del formulario de pedido, acumulando ejemplos, dibujando disposiciones de registros en tarjeta o en cinta, etc. Lo que deseamos es mantenernos en el nivel lógico, identificar cada uno de los elementos de datos que se encuentran presentes en un flujo de datos, darles nombres significativos, definir cada uno de ellos y organizarlos de manera de localizar fácilmente dicha definición.

¿Qué *queremos* significar con *pedidos*? Como mínimo la orden de pedido para un libro deberá tener cierta identificación de la orden, el nombre y dirección del cliente y los detalles de un libro, por lo menos, (y por lo general, más). Podremos entonces comenzar a dar nombres significativos y mostrar el siguiente desglose,

```
PEDIDO
  PEDIDO-IDENTIFICACION
  CLIENTE-DETALLES
  LIBRO-DETALLES
```

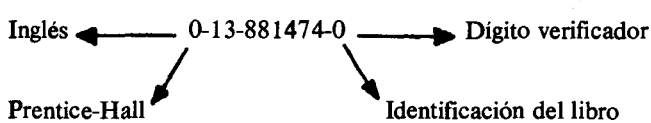
y podemos expandir esto más aún con notas:

```
PEDIDO
  PEDIDO-IDENTIFICACION
    PEDIDO-FECHA
```

```
    CLIENTE-PEDIDO-NUMERO. . . . .Normalmente presente
  CLIENTE-DETALLES
    EMPRESA-NOMBRE
    PERSONA-AUTORIZANTE. . . . . .Opcional
      NOMBRE. . . . . .Puede ser sólo la inicial
      APELLIDO
    TELEFONO
      CODIGO-DE-AREA
      CENTRAL
      NUMERO
      EXTENSION. . . . . .Opcional
    DESPACHAR-A-DIRECCION
      CALLE
      CIUDAD-LOCALIDAD
      CODIGO-POSTAL
    FACTURAR-A-DIRECCION. . . . .Si no se indica, igual a DESPACHAR-A-DIRECCION
      CALLE
      CIUDAD-LOCALIDAD
      CODIGO-POSTAL
  LIBRO-DETALLES. . .Una o más iteraciones de este grupo de elementos de datos
AUTOR-NOMBRE. . .Una o más iteraciones de este elemento de datos
TITULO
ISBN. . . .International Standard Book Number: Número internacional asignado al libro (opcional)
LOCN. . . .Librery of Congress Number: Número de la Biblioteca del Congreso de EE.UU. (opcional)
EDITOR-NOMBRE. . . . . .Opcional
CANTIDAD
```

Obsérvese que esta estructura de datos de una orden de pedido no nos indica nada de su disposición física, tamaño de los campos, si el campo es decimal empaquetado, características de la impresión, etc. Al mismo tiempo que omiten estos detalles físicos es, en cambio, preciso que el analista y el usuario efectúen las revisiones para evitar errores y omisiones. Por ejemplo, el cliente puede decir "Si se da el dato FACTURAR-A-DIRECCION, se debe registrar el nombre de la persona a la cual se debe enviar la factura" y el analista puede agregar otro elemento de datos.

Cada uno de los elementos de datos referido a la estructura de datos de PEDIDO deberá definirse en forma separada. Por ejemplo, ¿qué queremos significar con ISBN, el número internacional asignado al libro? Debemos estar en condiciones de localizar rápidamente una explicación de que el ISBN es un número de diez dígitos, dividido en cuatro grupos. El primer dígito o par de dígitos representa el idioma, el siguiente grupo representa al editor, el tercero al libro en particular y el último dígito es un dígito verificador. Por ejemplo:



Este es el ISBN del libro *Systems Analysis and Design Using Network Techniques* (Análisis y Diseño de Sistemas utilizando técnicas de redes) de Gary E. Whitehouse, editado por Prentice-Hall.

En el Capítulo 4 examinaremos una técnica para documentar las definiciones de los elementos de datos. Por el momento observemos que si tenemos una definición y la explicación de cada flujo de datos, del contenido de cada almacenamiento de datos, y de cada elemento de datos que los componen, de los cuales y si podemos disponer estas definiciones en orden alfabético para referirnos a ellos fácilmente, podremos tener un *diccionario de datos* del sistema. Si alguien desea conocer el significado de AVISO-DE-EMBARCO, deberá

buscar AVISO-DE-EMBARCO en la letra A del diccionario de datos donde encontrará su estructura de datos. Si luego quiere saber qué significa METODO-DE-TRANSPORTE (uno de los elementos de datos de AVISO-DE-EMBARCO) deberá buscar en la letra M donde encontrará su definición y explicación.

Las técnicas, implementaciones y ventajas de los diccionarios de datos se discuten extensamente en el Capítulo 4. El beneficio más importante para los analistas es que se pueden describir flujos de datos y almacenamiento de datos con un solo nombre significativo, sabiendo que todos los detalles correspondientes al nombre estarán rápidamente disponibles.

2.3 DEFINIR LA LOGICA DE LOS PROCESOS

Definido cada elemento de datos del sistema, podemos comenzar a explorar qué está pasando dentro de los procesos. Por ejemplo, ¿qué queremos significar en la Fig. 2.5 con "aplicar pago a la factura"? Hemos visto que cada uno de estos procesos puede ser expandido o explotado en procesos de nivel inferior. Supongamos que uno de tales procesos de nivel inferior, "verificar descuento" implica controlar que se ha efectuado correctamente el descuento. Si el analista pregunta ¿"Cuál es la política de descuentos?" se le podrá mostrar un memorándum, o una página del manual de procedimientos, donde dice más o menos lo siguiente:

> El descuento comercial (a libreros establecidos —al gremio— es del 20%. Para clientes particulares y bibliotecarios se concede el 5% de descuento por 6 o más libros, 10% para pedidos de 20 o más libros y 15% para pedidos de 50 o más. Los pedidos comerciales por 20 o más libros reciben el 10% de descuento sobre el descuento comercial.

Este es un ejemplo muy simple de lo que llamaremos *lógica externa,* externa en el sentido de que tiene relación con la política empresaria, procedimientos o reglas administrativas en oposición a *lógica interna,* la cual especifica la forma en que la computadora la va a implementar.

Aunque este ejemplo es simple, la lógica relativa a políticas puede rápidamente volverse confusa, en especial considerando que las excepciones y cambios de políticas se plantean escribiendo más memorándum, en lugar de corregir los documentos de la política original. El analista necesita herramientas para graficar la estructura de la lógica y expresar las políticas en una forma global y sin ambigüedades.

La primera de estas herramientas es el *árbol de decisión,* que se muestra en la Fig. 2.8. Las ramas del árbol corresponden a cada una de las posibilidades lógicas; es evidente que la forma de obtener la cantidad del descuento depende de la combinación de las posibilidades. Como herramienta para bosquejar una estructura lógica y para permitir al usuario confirmar que la lógica de la política expresada es correcta, el árbol de decisión es excelente. Es posible indicar la combinación de circunstancias que conduce a cada acción en forma clara y directa.

Si bien el árbol de decisión muestra muy claramente el esqueleto de la estructura de la decisión, no nos lleva fácilmente a la incorporación de instrucciones y cálculos. Si necesitamos escribir la lógica como un conjunto de instrucciones paso a paso, incluyendo la estructura de decisión y los cálculos inmediatos o acciones (posiblemente porque deseamos que un empleado sea capaz de seguirlas) quizá prefiramos utilizar una forma lógica estricta del idioma corriente, como se muestra en la Fig. 2.9. Este tipo de idioma corriente se ha denominado *lenguaje estructurado,* porque utiliza construcciones lógicas similares a las de la programación estructurada. Las instrucciones para llevar a cabo las acciones que no incluyen decisión se escriben en forma de frases imperativas ("Sumar el número total de volúmenes..."). Cuando se deba tomar una decisión, se expresará como una combinación de SI, ENTONCES, SI-NO, LUEGO, CON SI y SI-NO alineados apropiadamente para mostrar la estructura de decisión.

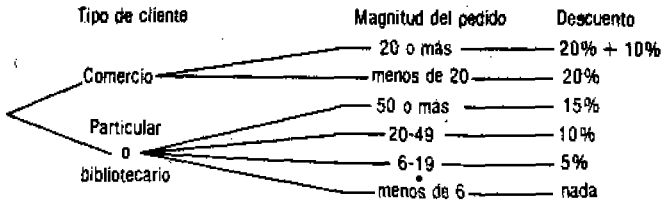


Figura 2.8 Arbol de decisión.

Podemos hacer el lenguaje estructurado (y el árbol de decisión) más preciso y compacto, utilizando, cuando sea pertinente, términos que se han definido en el diccionario de datos. Por ejemplo, podemos definir PEDIDO-MAGNITUD como un elemento de datos que puede tomar hasta cuatro valores:

PEQUEÑA: 5 o menos volúmenes
MEDIANA: 6 a 19 volúmenes
GRANDE: 20 a 49 volúmenes
MASIVA: 50 o más volúmenes

La primer parte de la Fig. 2.9 podrá leerse entonces

```
Política de descuentos

Sumar la cantidad total de volúmenes del pedido (en la columna cantidad)

SI      el pedido es de un cliente comercial
  y-SI      el pedido es por 20 o más volúmenes
          ENTONCES:      el descuento es del 30%
   SI-NO      (el pedido es de menos de 20 volúmenes)
          LUEGO:      el descuento es del 20%

SI-NO     (el pedido es de un cliente particular o bibliotecario)
   SI          el pedido es por 50 o más volúmenes
               el descuento es del 15%
   SI-NO SI    el pedido es por 20 a 49 volúmenes
               el descuento es del 10%
   SI-NO SI    el pedido es por 6 a 19 volúmenes
               el descuento es del 5%
    SI-NO      (el pedido es por menos de 6 volúmenes)
     LUEGO:    no se hará descuento
```

Figura 2.9 Lenguaje estructurado.

SI el pedido es de un comerciante
y-SI PEDIDO-MAGNITUD es GRANDE o MASIVA
ENTONCES: descuento es 30%
SI-NO (PEDIDO-MAGNITUD es MEDIANA o PEQUEÑA)
LUEGO: . . . . . .

Las convenciones y notaciones de los árboles de decisión y del lenguaje estructurado, junto con otras técnicas de representación de políticas y procedimientos, se encuentran detalladas en el Capítulo 5.

2.4 DEFINIR LOS ALMACENAMIENTOS DE DATOS: CONTENIDOS Y ACCESO INMEDIATO

Una vez armado el diagrama de flujo de datos, identificamos los lugares donde deben alojarse los datos entre una transacción y la siguiente o bien almacenarse permanentemente debido a que describen algún aspecto del mundo exterior al sistema (por ejemplo, la dirección del cliente). El analista debe obviamente especificar los elementos de datos que se deben conservar en cada almacenamiento de datos en forma similar a las especificaciones de cada flujo de datos de la Sección 2.2. Claro está, como los datos solo pueden llegar a un almacenamiento de datos a través de algún flujo de datos y no pueden salir si antes no fueron colocados, el contenido de un almacenamiento de datos puede ser obtenido a partir de las especificaciones de los flujos de datos entrantes y salientes, como podrá verse en el Capítulo 6. El contenido lógico de cada almacenamiento de datos se encuentra en el diccionario de datos bajo el nombre del almacenamiento de datos. Tal como ocurre con el flujo de datos, los elementos individuales de datos del almacenamiento de datos se definen en otra parte del diccionario de datos. Por ejemplo, en la Fig. 2.5 identificamos un almacenamiento de datos denominado LIBROS, el cual es utilizado en la verificación del pedido, entrando con un título o autor y recibiendo detalles del libro. ¿Qué queremos significar con *detalles del libro*? Si buscamos LIBRO-DETALLES en el diccionario de datos, encontraremos lo siguiente:

LIBRO-DETALLES
AUTOR-NOMBRE. . . . . .Una o más iteraciones
ORGANIZACION-ASOCIACION. . . . .Universidad, corporación
TITULO
ISBN. . . . . .Número Internacional asignado al libro (opcional)
LOCN. . . . . .Número de la Biblioteca del Congreso de EE.UU (opcional)
EDITOR-NOMBRE. . . . .Opcional; ver archivo de abreviaturas
PRECIO-ENCUADERNADO
PRECIO-RUSTICA
PUBLICACION-FECHA. . . . .Puede haber iteraciones si existen ediciones

Evidentemente, si todos estos elementos de datos se encuentran presentes en el flujo de datos, también deberán estarlo en el almacenamiento de datos. El analista puede tomar esto como punto de partida y a medida que desarrolla el flujo del sistema puede preguntarse "¿Existirá cualquier otro elemento de datos que describa libros y que debería estar almacenado aquí?" ¿Qué pasa, por ejemplo, con PESO-PARA-CORRESPONDENCIA? Si el peso de un pedido debe ser calculado a partir del peso de los libros componentes, el franqueo deberá calcularse en base a una tabla de tarifas postales e ingresado como parte de la factura. ¿Deberá almacenarse este elemento de datos? Si es así, ¿cómo debe (a nivel lógico) obtenerse el valor del elemento de datos para cada nuevo libro?

Una vez que se ha establecido el contenido de cada almacenamiento de datos propuesto a nivel lógico en el diagrama de flujo de datos, el analista tiene que resolver dos problemas:

1. ¿Son estos almacenamientos lógicos de datos los más simples posibles? ¿Pueden combinarse? ¿Deben combinarse?
2. ¿Qué accesos inmediatos necesitaremos para el almacenamiento de datos, y qué valor implica cada tipo de acceso?

2.4.1 ¿Son los almacenamientos lógicos de datos los más simples posibles?

Una vez identificado el contenido de cada almacenamiento de datos, los mismos deben compararse para ver si existen similitudes o superposiciones. Supongamos que posteriormente identificamos un almacenamiento de datos de inventario: ¿Qué deberá contener? Los

elementos de datos deberán incluir el título del libro, probablemente el autor(es), y el ISBN como única identificación, y también elementos que describan

Cantidad-disponible
Cantidad-ordenada
Pedidos-cumplidos-en-los-últimos-30-días
Pedidos-pendientes-durante-los-últimos-30-días
Fecha-última-orden-colocada
Tiempo-promedio-de-entrega

y cualquier otra información necesaria para la gestión del inventario. Pero este almacenamiento de datos contiene información sobre cada libro, exactamente igual que el almacenamiento de datos LIBROS. ¿Por qué no combinamos los dos y mantenemos la información de inventario en LIBROS? De acuerdo con la implementación elegida existirán argumentos a favor de combinar estos archivos y argumentos a favor de mantenerlos separados, lo cual será discutido en el Capítulo 9. Por ejemplo, si el analista y el diseñador deciden que el almacenamiento de datos LIBROS deberá ser un catálogo de libros impreso con secciones organizadas por autor, por título y por materia, se ve claramente que es inapropiado incluir en el mismo datos de inventario que cambian rápidamente. Por otra parte, si LIBROS llega a ser una base de datos en línea accesible a los empleados que toman pedidos telefónicos, puede tener sentido disponer de un inventario actualizado e información de fechas de entrega dentro del archivo LIBROS, de modo que puedan ser recuperados al mismo tiempo que los datos necesarios para verificar el pedido.

Independientemente de consideraciones físicas como las vistas, es importante en las primeras etapas de análisis que el analista sea capaz de definir los detalles del contenido de cada almacenamiento de datos, esté seguro que la definición es completa, y sea capaz de identificar similitudes y diferencias existentes en los diferentes almacenamientos de datos del sistema. Es muy útil aplicar en este caso el enfoque *relacional* —expresar un archivo complejo en forma de varias *relaciones normalizadas* simples— que se discutirá en detalle en el Capítulo 6.

2.4.2 ¿Qué accesos inmediatos se necesitarán?

Más difícil y aun más crítica que la tarea de especificar contenidos, es la tarea de decidir qué accesos inmediatos a cada almacenamiento de datos necesitará el usuario. Por ejemplo, supongamos que hemos decidido tener el archivo de LIBROS disponible en pantalla, en una terminal en línea. El elemento de datos que identifica unívocamente cada libro es el ISBN. Pero si lo hacemos clave principal del archivo y solo tenemos acceso inmediato por esta clave, ello significa que tendremos que conocer el ISBN de un libro para poder desplegar en la pantalla su título, código del tema, editor, precio, etc. Ello es comparable a tener que conocer el número de la póliza de seguro de una persona para poder localizar la póliza o tener que conocer el número de parte de un artículo para poder saber su precio.

Mucho más frecuentemente queremos tener acceso inmediato por el *nombre* de algo, y esto significa que el analista deberá especificar accesos *múltiples* del archivo. En este caso, el usuario puede decidir en forma estricta que desea desplegar los detalles del libro en su pantalla como resultado de digitar, o bien el nombre de un autor (en cuyo caso podrían aparecer varios libros), o bien un encabezamiento en base al tema, (en cuyo caso se encontrará seguramente con varias pantallas completas), o bien el título real, o el ISBN. Podrá especificar posteriormente que desea tener la respuesta *de inmediato,* digamos lo suficientemente rápido como para poder contestar una consulta telefónica. Por *acceso inmediato* queremos significar "más rápido que lo que insumiría leer de punta a punta el archivo o clasificarlo" Obviamente el significado exacto de esto será variable, dependiendo del tamaño del archivo y de la potencia de la computadora; en una 370/168 será posible leer

un archivo de 100.000 caracteres en menos de un segundo, y en cambio en una máquina más lenta algunas bases de datos grandes de clientes tomarán todo un fin de semana para ser procesados de punta a punta.

Es importante conocer el significado de *inmediatamente* en cada contexto, debido a que si el usuario quiere saber la respuesta de una consulta *y puede esperar mientras pasamos o clasificamos el archivo completo,* el diseñador no tiene necesidad de construir ninguna estructura en el archivo. Si el usuario insiste en una respuesta inmediata, tan rápida que no dé tiempo a pasar o clasificar el archivo, el diseñador deberá crear algún tipo de índice o alguna estructura y proveer algún software que le permita al usuario tener acceso al archivo con argumento de búsqueda que desea. Por ejemplo, el diseñador podría crear un archivo subsidiario por autor, conteniendo por cada autor el ISBN de todos los libros que ha escrito. Conociendo el ISBN, el software de recuperación deberá utilizarlo como acceso al archivo principal y desplegar en la pantalla del usuario todos los libros escritos por el autor cuyo nombre fue tecleado.

Hay diversas técnicas para lograr accesos múltiples, y las mismas se verán brevemente en el Capítulo 7. No interesa la forma de realizarlo, el acceso inmediato cuesta dinero en función del mayor tiempo necesario para su desarrollo y mantenimiento, de la necesidad de mayor software para manejar los accesos y de la utilización de más espacio en disco. Si bien es difícil obtener costos exactos, como regla aproximada se puede esperar que un acceso inmediato cueste hasta cinco veces más que la obtención de la misma información mediante clasificación del archivo en los momentos en que no se encuentra en uso.

En consecuencia, es muy importante para el analista establecer exactamente qué accesos se requiere que sean inmediatos y, además, cuáles de esos accesos inmediatos son los más importantes para el negocio. Para el diseño de los accesos de la base de datos, el analista y el diseñador realizarán una serie de iteraciones: primero, tratando de satisfacer todo aquello que fue solicitado por los usuarios y luego, calculando el costo del sistema y, si resulta mayor que el presupuesto, suprimiendo las facilidades más caras o difíciles, realizando un nuevo diseño tentativo, etc. En esta última proposición hacemos un llamado de atención a una de las cosas que llevan a error, "suprimiendo las facilidades más caras o difíciles". Si fuera muy caro proveer *todas* las facilidades solicitadas, la primera economía deberá ser la supresión de la facilidad de *menor valor para el usuario,* no la más cara o difícil. ¿Pero, cómo podrá hacerse si el analista no conoce la importancia relativa de los diferentes accesos?

En el simple caso del archivo LIBROS, se supone que el público o bien conoce al autor pero tiene una vaga idea del título, o bien conoce el tema pero tiene una vaga idea tanto del título como del autor. La capacidad de localizar un libro por su ISBN (que es fácil de proveer dado que el ISBN es la clave primaria) es de muy poco valor. Ubicar un libro solamente por su título es útil, pero de relativo valor. Podemos crear una imagen de estos accesos inmediatos en un diagrama de datos de acceso inmediato (DIAD, en inglés, data inmediate-access diagram); ver la Fig. 2.10. El diagrama indica que existe un archivo que describe libros con un ISBN como clave primaria y elementos de datos adicionales que describen los atributos de cada libro, tales como autor, precio, etc. El hecho de señalar al ISBN como la clave primaria (dentro de un rectángulo de líneas llenas), implica que la pregunta "Dado un ISBN, muéstreme el autor(es) y el título" puede ser contestada inmediatamente.

Una pregunta tal como "Dado un editor, muéstreme todas sus publicaciones" *no puede* ser contestada inmediatamente con los accesos definidos en el DIAD anterior. Sin embargo, el editor debe *aparecer* en cada registro de LIBROS, de manera que siempre es posible clasificar el archivo por el campo de editor o escribir un programa que sea capaz de leer todo el archivo, extrayendo los registros de libros para un editor determinado. (Es cierto que un grupo del ISBN es el código del editor, pero el que consulta no siempre conoce ese código.)

Por otra parte, el DIAD muestra que las preguntas "Dado el nombre de un autor, muéstreme los libros escritos por él" y "Dado un título, muéstreme todos los libros con ese título" *pueden* ser respondidas inmediatamente, aun cuando "autor" y "título" sean atributos de LIBROS y campos dentro de los registros de LIBROS. El hecho de indicar rectángulos

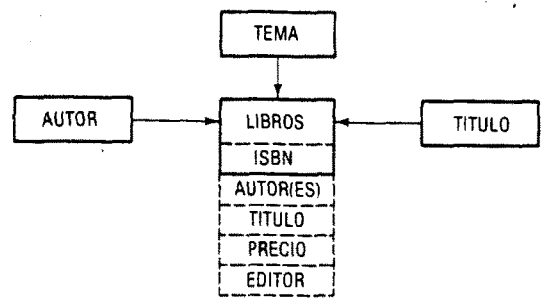


Figura 2.10 Diagrama de datos de acceso inmediato.

separados en el DIAD con flechas que apuntan a LIBROS, indica que se tendrá acceso inmediato al archivo mediante índices de "autor" y "título" o mediante algún mecanismo de búsqueda rápida.

El acceso "tema" es completamente claro; TEMA no es un atributo de LIBROS, es un índice totalmente separado dentro del archivo y por haberlo incluido en el diagrama de acceso inmediato estamos indicándole al diseñador de la base de datos que de alguna manera esta facilidad deberá ser incluida en el diseño.

Podemos utilizar el diagrama de accesos inmediatos para registrar la información que necesitamos sobre las preferencias de los usuarios y sobre el valor de cada acceso para la empresa. La Fig. 2.11 muestra los resultados de una consulta realizada a tres gerentes sobre el valor relativo que tienen los diferentes accesos para ellos y su personal. La pregunta formulada fue "Considerando que usted puede tener la respuesta de esta pregunta mañana por la mañana al costo de $ 1, ¿cuánto pagaría por tener la respuesta inmediatamente (mientras el cliente está al teléfono)?" Las preferencias de los tres gerentes se han codificado en la Fig. 2.11 como M1, M2, y M3.

¿Si usted fuera el analista que está en consulta con el diseñador, y recibe la información de que solamente dos de los tres accesos son económicamente posibles, cuál eliminaría?

Las convenciones para la representación de un DIAD y las técnicas de entrevistas se encuentran más detalladas en el Capítulo 7. También deseamos proveerle al diseñador la información sobre el volumen y la frecuencia de estos accesos inmediatos. Además, debemos reunir información de seguridad (por ejemplo, ¿quién está autorizado a tener acceso a la historia de los salarios del personal?)

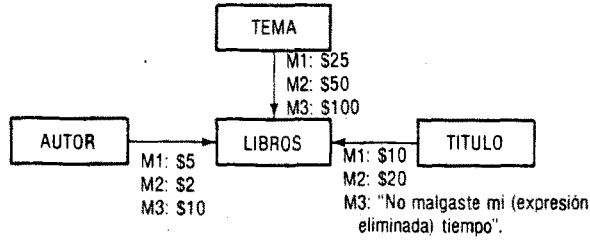


Figura 2.11 Valores relativos de varios accesos.

El lector puede encontrar de utilidad dibujar un DIAD para un sistema de acceso inmediato que le sea familiar, luego dibujar el diagrama que podría, en su opinión, satisfacer todos los requerimientos de todos los usuarios y a continuación comparar ambos.

2.5 UTILIZACION DE LAS HERRAMIENTAS PARA CREAR UNA ESPECIFICACION FUNCIONAL

Para resumir este capítulo de visión global diremos que el diagrama lógico de flujo de datos muestra las fuentes y destinos de los datos (y en consecuencia los límites del sistema), identifica y asigna nombre a las funciones lógicas, identifica y da nombre a los grupos de elementos de datos que conectan una función con otra, e identifica los almacenamientos de datos a los cuales tienen acceso. Se analiza cada flujo de datos, y sus estructuras y las definiciones de sus elementos de datos componentes se almacenan en el diccionario de datos Cada función lógica puede ser fraccionada en diagramas de flujo de datos más detallados. Cuando ya no resulta de utilidad subdividir las funciones lógicas, se expresa su lógica *externa* empresaria utilizando árboles de decisión, lenguaje estructurado, u otras herramientas. Los contenidos de cada almacenamiento de datos son analizados y almacenados en el diccionario de datos. El valor relativo, para los usuarios, de los diferentes caminos de acceso inmediato a los datos en los almacenamientos de datos, es analizado y expresado en un diagrama de datos de acceso inmediato. La Fig. 2.12 muestra esas relaciones.

Estos documentos configuran una descripción completa del sistema, ya sea el sistema existente en una empresa o el "nuevo sistema" que se va a construir. Cuando se ha preparado el paquete completo para el nuevo sistema tenemos una *especificación funcional lógica,* una presentación detallada de lo que hará el sistema, la cual es, en lo posible, independiente de las consideraciones físicas de cómo será implementado. Este tipo de especificación funcional brindará al diseñador una idea clara del resultado final que debe alcanzar y una evidencia escrita de las preferencias del usuario para los casos en que haya que adoptar soluciones de compromiso. En este punto el analista transfiere la responsabilidad primaria del trabajo técnico a un diseñador, y continúa con los otros aspectos del proyecto (tales como planificar la prueba y aceptación del sistema). Si el analista y el diseñador son una misma persona, ésta ahora "cambia de sombrero" y mira la especificación funcional lógica con los ojos de un diseñador, confiado en el conocimiento de que la misma representa una base firme de requerimientos a partir de los cuales debe diseñar el sistema.

A menudo, como ya lo hemos indicado, el analista y el diseñador deben realizar diversas iteraciones de ajuste de diseños físicos de prueba, trabajando sobre sus implicancias técnicas y económicas, cambiando algunos aspectos, y volviendo a probar. Discutiremos este proceso de *diseño tentativo* y la intervención del usuario en el Capítulo 8.

Ejercicios y puntos de discusión

1. Liste tantos distintos métodos físicos para implementr un flujo de datos determinado como le sea posible.
2. Liste tantos distintos métodos físicos para crear un almacenamiento de datos, como le sea posible.
3. Explote cada uno de los procesos expuestos en la Fig. 2.5 al mismo nivel de detalle de la Fig. 2.6. Incluya cualquier condición de error y excepción que le parezca posible.
4. Adapte el diagrama de flujo de la Fig. 2.5 para representar la Compañía Amiga de Alimentos, que toma pedidos de los agricultores para alimento de ganado y cerdos, coloca pedidos masivos en fábricas y luego los fracciona para su entrega a los agricultores en forma individual.
5. ¿Cuántos tipos de empresas diferentes desde el punto de vista lógico (tales como distribuidoras sin inventario, distribuidoras con inventario, fabricantes a pedido, fabricantes con inventario) puede definir?
6. Elija una declaración de política compleja y dibuje un árbol de decisión para indicar su estructura lógica.

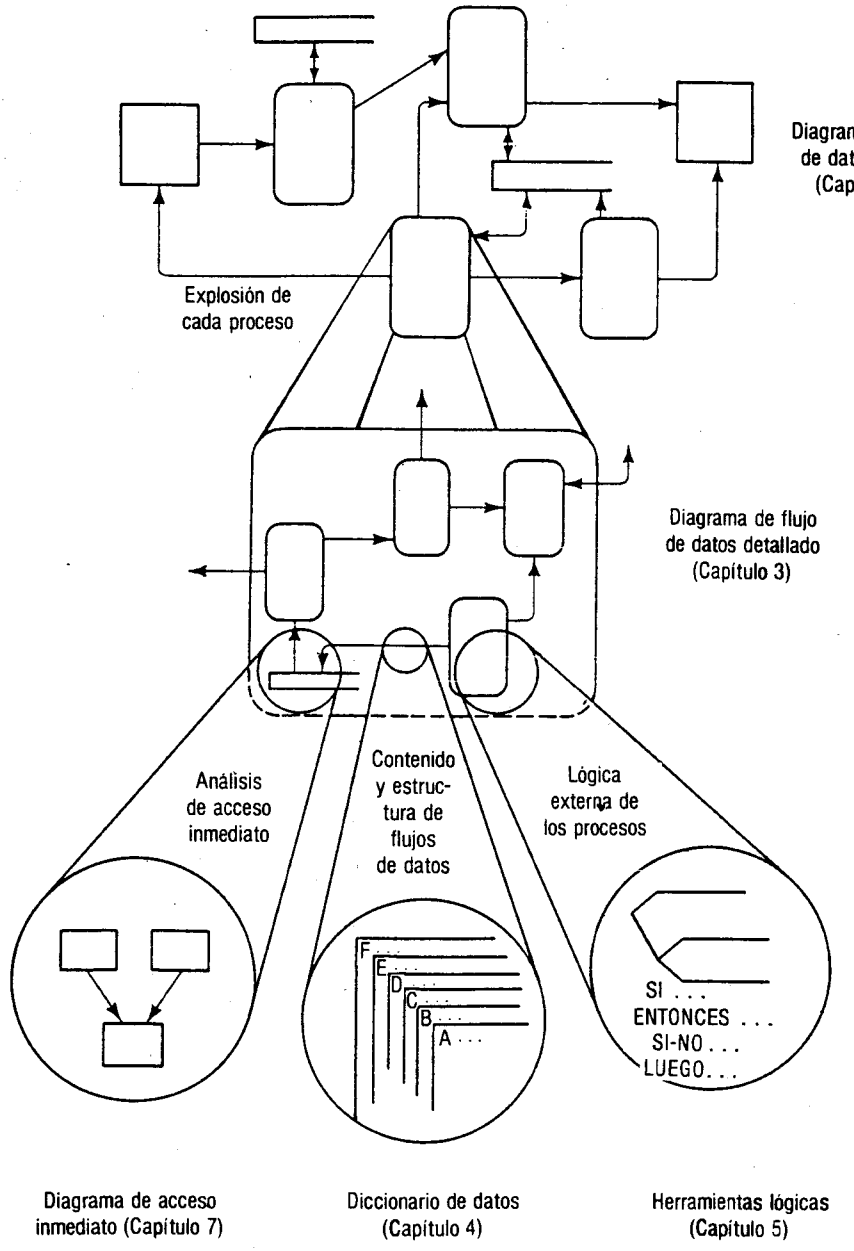


Figura 2.12 Componentes de un modelo lógico.

7. ¿Cuál es el mayor archivo legible por máquina que conoce personalmente? ¿Cuánto tarda
   (a) para buscar?
   (b) para clasificar?
8. ¿Cuál es, según su punto de vista, la diferencia entre análisis y diseño? ¿Los analistas que usted conoce, hacen solamente análisis?

3

DIBUJO DE LOS FLUJOGRAMAS DE DATOS

La idea de un gráfico para representar el flujo de datos a través de un sistema, no es nueva; Martin y Estrin [3.1] propusieron un *grafo de programa* en 1967, donde los flujos de datos se representaban mediante flechas y las transformaciones, con círculos. En el enfoque del flujo de datos se ha utilizado una notación similar al diseño del programa, que forma parte de la disciplina de diseño estructurado [3.2]. Whitehouse [3.3] describe una notación gráfica de flujo similar para modelizar sistemas matemáticos. Ross [3.4] ha descripto un enfoque gráfico muy general para el análisis de sistemas, el cual comprende el flujo de datos como uno de sus aspectos. Lo que *es* nuevo, es el reconocimiento de que el diagrama de flujo de datos en el nivel lógico es la herramienta clave para comprender y trabajar con un sistema de cualquier complejidad, junto con el refinamiento de la notación para su uso en el análisis.

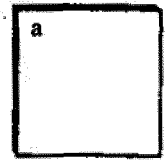
Como vimos en el Capítulo 2, en el análisis necesitamos reconocer entidades externas y almacenamientos de datos, así como también flujos de datos y transformaciones o procesos. Para poder representar completamente nuestro sistema lógico, necesitaremos agregar símbolos al gráfico simple de programa. También, como necesitamos describir nuestras transformaciones o procesos claramente y es difícil escribir claramente dentro de un círculo, adoptaremos un retángulo con ángulos redondeados como símbolo de proceso.

3.1 CONVENCIONES SOBRE SIMBOLOS

Vamos a examinar en detalle las convenciones de símbolos.

3.1.1 Entidad externa

Las entidades externas son generalmente clases lógicas de cosas o de personas, las cuales representan una fuente o destino de transacciones, por ejemplo, clientes, empleados,



aviones, unidades tácticas, proveedores, contribuyentes, tenedores de pólizas. También pueden ser una fuente o destino específico, por ejemplo, Departamento de Contabilidad, DGI, Oficina del Presidente, depósito. Como el sistema que estamos considerando acepta datos de otro sistema o bien se los provee, este otro sistema es una entidad externa.

Una entidad externa puede simbolizarse con un cuadrado de *líneas llenas,* con los lados superior e izquierdo de doble ancho para hacer resaltar el símbolo del resto del diagrama. La entidad puede identificarse con una letra minúscula en el ángulo superior izquierdo, como referencia.

Para evitar el cruzamiento de las líneas de flujo de datos, la misma entidad se puede dibujar mas de una vez en el mismo diagrama; las dos (o más) casillas por entidad pueden identificarse con una linea inclinada en el ángulo inferior derecho, como se observa en la Fig. 3.1.

Cuando otra entidad deba ser duplicada, se dibujará con dos líneas en el ángulo inferior derecho, y así sucesivamente; ver Fig. 3.2.

Mediante la designación de alguna cosa o algún sistema como una entidad externa, estamos estableciendo implícitamente que se encuentra fuera de los límites del sistema que estamos considerando.

A medida que avancemos en el análisis y aprendamos más sobre los objetivos del usuario, tomaremos algunas entidades externas y las introduciremos en nuestro diagrama de flujo de datos del sistema, o, por el contrario, dejaremos de considerar alguna parte de las funciones de nuestro sistema designándola como una entidad externa con datos que fluyan desde y hacia ella.

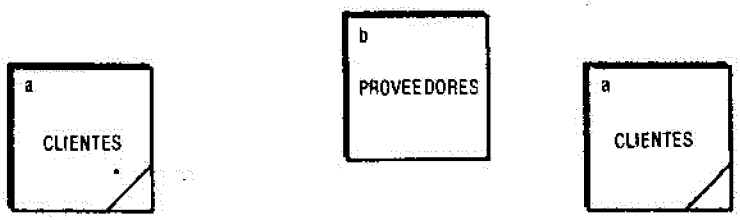


Figura 3.1 Símbolo de duplicación para entidades externas.

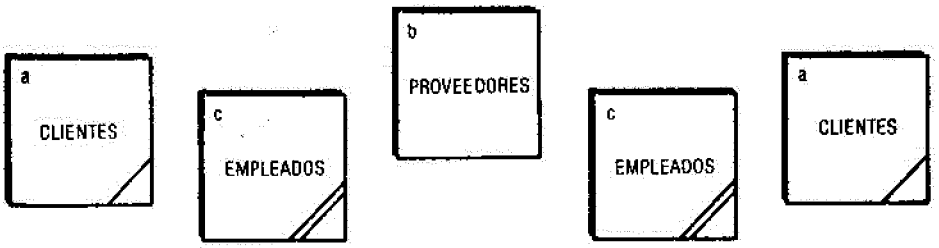


Figura 3.2 Duplicación múltiple para entidades externas.

3.1.2 Flujo de datos

El flujo de datos se simboliza mediante una flecha, preferentemente horizontal y/o vertical, con la punta indicando la dirección del flujo. Para mayor claridad, y especialmente en los primeros borradores del diagrama se utilizará una flecha con doble punta en lugar de dos flechas, cuando los flujos de datos estén apareados. Ver Fig. 3.3.

Cada flujo de datos se asemeja a una tubería por donde se envían paquetes de datos. Se puede hacer referencia a la tubería del flujo de datos dando los procesos, entidades o almacenamiento de datos en su punta y cola, pero cada flujo de datos deberá tener a lo largo una descripción escrita de su contenido. La descripción deberá elegirse de manera que sea lo más útil posible a los usuarios que deseen revisar el diagrama de flujo de datos (consistente con el nivel de una descripción lógica). En las primeras versiones, la descripción del flujo de datos se escribirá en el diagrama con letras mayúsculas y minúsculas. Ver Fig. 3.4.

En una etapa posterior del análisis, cuando han sido definidos los contenidos del diccionario de datos, la descripción puede cambiarse por letras mayúsculas únicamente, para indicar que se ha ingresado en el diccionario de datos. Encontramos frecuentemente que se trasladan varios diferentes "paquetes" de datos a lo largo del mismo flujo de datos. Por ejemplo, si observamos en el diccionario de datos en VENTAS-INFORMES, encontraremos el listado de sus componentes como:

VENTAS-INFORMES-POR-DIA
VENTAS-TENDENCIA-ANALISIS
VENDEDOR-RENDIMIENTO-ANALISIS
VENTAS-POR-PRODUCTO-ANALISIS

A su vez, cada uno de estos componentes se describe en el diccionario de datos; cada uno contendrá alguna disposición diferente de elementos de datos.

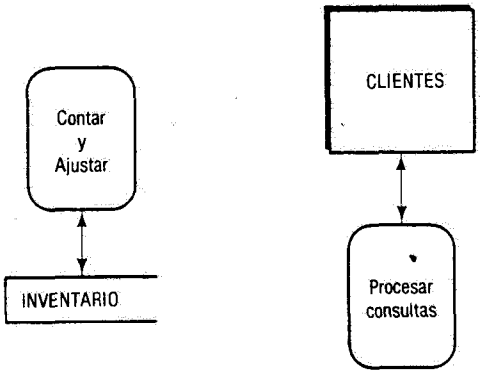


Figura 3.3 Flujos de datos apareados.

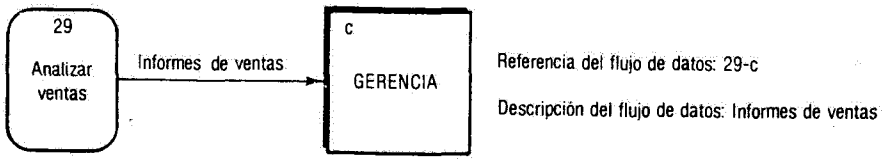


Figura 3.4 Descripción de flujo de datos.

Particularmente cuando se dibuja un flujograma de datos, es aceptable no colocar la descripción si ella resulta evidente para quien la va a revisar, pero el creador del diagrama deberá estar en condiciones, en cualquier momento, de suministrar la descripción de todos los ítem. La Fig. 3.5 nos muestra un flujo de datos que no necesita descripción.

En algunas ocasiones es difícil encontrar una descripción que caracterice adecuadamen-

te el contenido de un flujo de datos. Por ejemplo, los clientes pueden enviar pedidos, pagos, devolución de mercadería deteriorada, consultas, reclamos, etc. Es muy pesado dibujar el flujo de datos como se muestra en la Fig. 3.6. Hay dos formas de resolver este tipo de situación. Si el hecho lógico más importante es que solo existe un único flujo de datos (tal vez hacia una oficina de ventas) y la función "encaminar transacciones" es muy importante, entonces la solución podría ser agrupar el contenido bajo un término lo suficientemente vago, tal como "transacciones de los clientes" o "informes gerenciales". El contenido de este flujo de datos se puede hallar tanto en el diccionario de datos como por medio de un examen de la salida de la función de encaminamiento. La Fig. 3.7 muestra un ejemplo de esta solución.

La segunda solución se puede utilizar cuando la función de encaminamiento es trivial y cada transacción se procesa en forma diferente (y por supuesto consta de elementos diferentes). En este caso se puede dibujar una flecha diferente por cada tipo de transacción distinta, cada una de ellas dirigida a una casilla diferente de proceso; ver Fig. 3.8.

La descripción y contenido de cada flujo de datos en el diccionario de datos, se trata en detalle en el Capítulo 4.

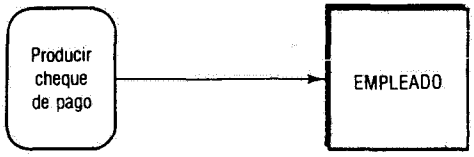


Figura 3.5 Flujo de datos que no necesita descripción.

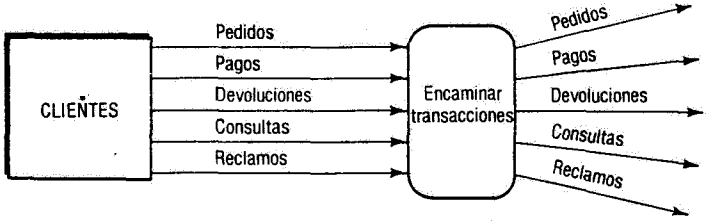


Figura 3.6 Flujo de datos de tratamiento difícil.

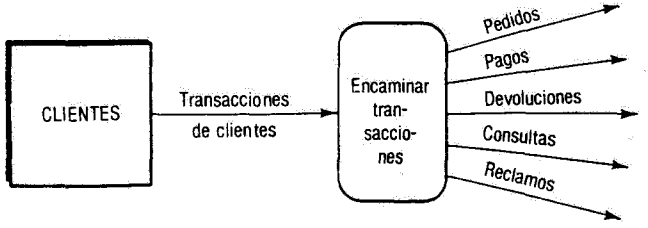


Figura 3.7 Una solución para el flujo de datos de tratamiento difícil.

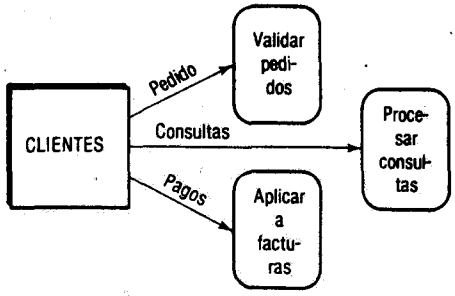


Figura 3.8 Otra solución para el flujo de datos de tratamiento difícil.

3.1.3 Proceso

Necesitamos describir la función de cada proceso y para una fácil referencia, dar a cada proceso una única identificación vinculada, en lo posible, a un sistema físico. Los procesos pueden simbolizarse con un rectángulo vertical, con los ángulos redondeados, y dividido opcionalmente en tres áreas; ver Fig. 3.9.

La *identificación* puede ser un número atravesado de izquierda a derecha en el diagrama de flujo de datos, con el solo objeto de identificar el proceso. No hay dificultad en asignar números a los procesos si tenemos en cuenta que algunos procesos se abrirán a su vez en dos o más procesos (lo que implica la asignación de nuevos números) o bien algunos procesos se unificarán (lo que implica la desaparición de números) durante el trabajo de análisis. Una vez asignada, la identificación del proceso no deberá cambiarse, excepto para aperturas o unificaciones, dado que se utiliza como referencia para los flujos de datos y para la descomposición del proceso en niveles inferiores, como se describió en la Sec. 3.2. Para mayor claridad, las líneas finas divisorias de la identificación y la descripción podrán omitirse, en especial, si el diagrama ha de mostrarse a los usuarios.

La *descripción de la función* deberá hacerse con una frase imperativa, que consistirá idealmente en un verbo activo (extractar, calcular, verificar) seguido por una cláusula objeto, cuanto más simple, mejor. Por ejemplo:

Extractar-ventas mensuales
Ingresar-nuevos detalles de los clientes
Verificar-si el crédito del cliente es bueno

En caso de dudas, una ayuda sería pensar que la descripción de la función es "una orden a un empleado tonto". Si la descripción no resulta ambigua para un empleado, y si usted puede representarse mentalmente que la función podrá llevarse a cabo en 5-30 minutos en un ambiente administrativo simple, posiblemente tenga una buena descripción de la función.

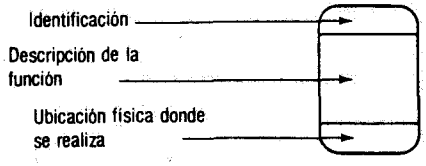


Figura 3.9 Símbolo de proceso.

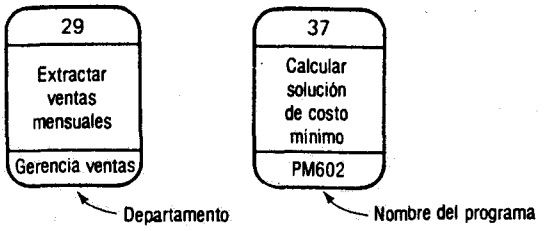


Figura 3.10 Casillas de proceso con referencias físicas.

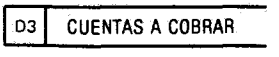
Generalmente hablando, cuando usted se encuentre utilizando los verbos "procesar", "actualizar" o "validar" ello puede significar que aún no ha comprendido mucho acerca de esta función y que aún necesitará posteriores análisis.

"Crear", "producir", "extractar", "recuperar", "almacenar", "computar", "calcular", "determinar" y "verificar" son todos verbos activos, sin ambiguedad. "Controlar" o "chequear", pueden utilizarse, pero puede llevar a confusión con el sustantivo en las áreas de contabilidad o finanzas*. "Clasificar" significa que se ha elegido una solución física, ya que la clasificación es meramente una agrupación física distinta de la secuencia de registros en un archivo y no posee valor *lógico.*

Obsérvese que estas frases imperativas no tienen sujeto; tan pronto como se introduce un sujeto (por ejemplo, "El gerente de ventas extracta mensualmente las ventas") se habrá indicado cómo deberá realizarse físicamente la función. Podrá ser de utilidad cuando se está estudiando un sistema existente, ver cuál departamento o cuál programa lleva a cabo la función. En forma similar, cuando se ha completado el análisis, y se está haciendo el diseño físico del nuevo sistema, es conveniente hacer notar cómo se ejecuta físicamente la función. Este es el propósito de la parte opcional inferior de la casilla de proceso: incluir una *referencia física;* ver Fig. 3.10. De esta manera, la descripción de la función lógica y la implementación física pueden mantenerse separadas.

3.1.4 Almacenamiento de datos

Sin tomar un compromiso físico durante el análisis, encontramos que existen lugares donde necesitamos definir datos para que puedan ser almacenados entre procesos. Los almacenamientos de datos pueden simbolizarse por medio de dos líneas horizontales paralelas cerradas en un extremo, preferentemente del ancho necesario para colocar el nombre (0,6 cm en un diagrama sin reducir). Cada almacenamiento puede identificarse con una letra "D" y un número arbitrario en un recuadro en el extremo izquierdo, para su fácil referencia. Deberá elegirse el nombre que sea más descriptivo para el usuario.



Para evitar complicaciones con el cruce de líneas en un diagrama de flujo de datos, puede dibujarse el mismo almacenamiento de datos más de una vez en el diagrama, identificando los almacenamientos de datos duplicados mediante líneas verticales adicionales colocadas a la izquierda.

* La confusión surge en inglés al utilizar "check" (controlar, verbo, o cheque, sustantivo). (*N. del T.*)

Cuando un proceso *almacena* datos, la flecha del flujo de datos se indica en la dirección al almacenamiento de datos; inversamente, cuando un almacenamiento de datos es *accedido para ser leído únicamente,* es suficiente con mostrar el grupo de elementos de datos recuperados como flujo de datos saliente; ver Fig. 3.11.

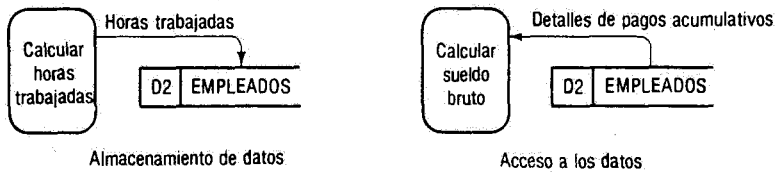


Figura 3.11 Almacenamiento de datos con acceso para lectura solamente.

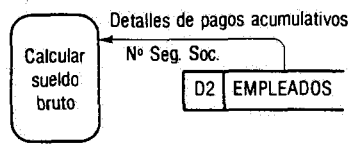


Figura 3.12 Especificación del argumento de búsqueda.

Si fuera necesario especificar el *argumento de búsqueda,* éste podrá mostrarse en el lado opuesto de la descripción del flujo de datos; una punta de flecha indicará que el argumento de búsqueda pasa del proceso al almacenamiento de datos, tal como se muestra en la Fig. 3.12.

3.2. CONVENCIONES SOBRE LA EXPLOSION

Como vimos en el Capítulo 2, cada proceso en el diagrama de flujo de datos de alto nivel de un sistema puede ser "explotado" para convertirse en un diagrama de flujo de datos en sí mismo. Cada proceso en el nivel inferior deberá estar relacionado, inversamente, con el proceso del nivel superior. Esto puede hacerse dándole a la casilla de proceso de nivel inferior un número de identificación que sea una fracción decimal de la casilla de proceso del nivel superior, por ejemplo, el 29 se descompone en 29.1, 29.2, 29.3, etc. y si es necesario llegar a un tercer nivel, el 29.3 se descompone en 29.3.1, 29.3.2, etc.

La representación más clara de la expansión del proceso se obtiene dibujando los diagramas de flujo de datos de nivel inferior dentro de los límites que encuadran la casilla de proceso de nivel superior. Obviamente, todos los flujos de datos hacia adentro y hacia afuera de la casilla de proceso de nivel superior deben entrar y salir a través del límite. Los flujos de datos que se muestran por primera vez en el nivel inferior, tales como los circuitos de errores, también pueden salir fuera de los límites. Cuando ello ocurre, se puede remarcar con una "X" el punto de salida. Los almacenamientos de datos solo se muestran dentro de los límites si son creados y accedidos por este proceso y no por otro. Por ejemplo, la Fig. 3.11 es un extracto de la Fig. 2.5. La explosión del proceso de nivel inferior se muestra en la Fig. 3.14. Esta explosión ilustra un número de detalles gráficos:

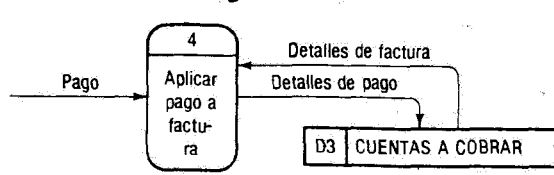
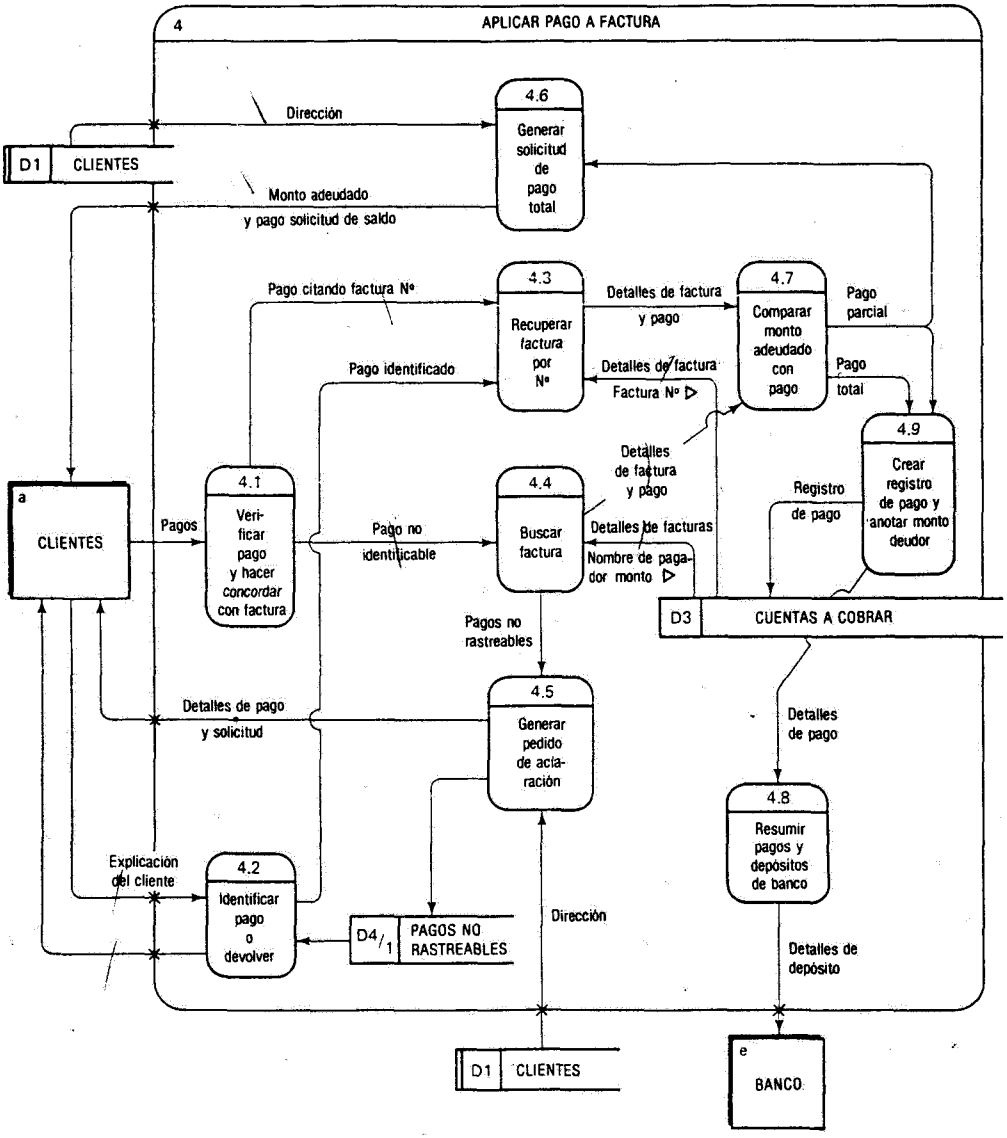


Figura 3.13 Extracto de la Fig. 2.5.



"X": Los flujos de datos que se han visto previamente en el nivel superior, tales como circuito de errores, pueden también salir de los límites. Cuando ello ocurre se marca con "X" en el punto de salida.

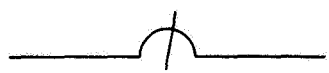
Figura 3.14 Explosión del proceso.

1. Los almacenamientos de datos *externos* al proceso que se está haciendo explotar pueden forzarse dentro de los límites si al dibujarlos así se simplifica el diagrama: "D3: CUENTAS A COBRAR" realmente está fuera de APLICAR PAGO A FACTURA y ha sido dibujado mitad adentro y mitad afuera, para mayor claridad. Los tres flujos de datos "D3-4.4: detalles de factura", "D3-4.3: detalles de factura", y "4.9-D3: registro de pago" cruzan el límite y "concuerdan" con los flujos en la Fig. 3.13.

2. El almacenamiento de datos D4/1, PAGOS NO IDENTIFICADOS, solamente existe para propósitos internos de este proceso y por eso se lo muestra adentro. Si tuviera acceso a él cualquier proceso que no formara parte del 4: APLICAR PAGO A FACTURA, debería dibujarse como externo (como D1: CLIENTES). Hemos utilizado la convención D4/1 para indicar el primer almacenamiento interno de datos del proceso 4. No tiene por qué tener relación con ningún almacenamiento de datos D4 del diagrama de nivel superior correspondiente.

3. Las entidades externas no se muestran dentro de los límites de las explosiones aun cuando, como BANCO en este caso, puedan no estar involucradas en ningún otro proceso.

4. Cuando resulte inevitable que un flujo de datos deba cruzar a otro, utilizaremos en forma convencional un pequeño gancho, como se ejemplifica en "4.4-4.7: factura y detalles de pago".



5. En el caso muy extraño que un flujo de datos necesite cruzar un almacenamiento de datos, como en "4.9-4.8: detalles de pago" también se colocará el gancho, tal como se ilustra.

3.3 TRATAMIENTO DE ERRORES Y EXCEPCIONES

Cuando sea posible, los flujos de datos que resulten de condiciones de error y excepción, deberán manejarse dentro del diagrama de segundo nivel en el cual aparecen. Esto es, las transacciones erróneas rechazadas por "validar" deberán manejarse dentro de la expansión de "validar". Cuando una entidad externa, tal como un supervisor o un empleado de ajustes, deba procesar manualmente el error, la entidad se mostrará en el diagrama de nivel inferior, pero fuera de límite.

Cuando se trate de un proceso completo, comparable en complejidad con el diagrama de nivel superior, se deberá confeccionar un flujo de datos de nivel inferior para el mismo y luego tomar la decisión de si el proceso sumario será o no incluido en el diagrama de nivel superior. Ello podrá ser necesario cuando las excepciones puedan traer consecuencias financieras, tales como mercadería devuelta, cheques sin fondos, etc.

Este proceso surge en APLICAR PAGO A FACTURA respecto al manejo de pagos no identificables, esto es, en el caso en que se recibe un pago sin ninguna indicación de la factura relacionada, y más aún, en que no existen antecedentes de haber enviado ninguna factura al cliente o de haber confeccionado factura alguna por este monto (esto puede suceder, como por ejemplo cuando una casa matriz paga una cuenta de una subsidiaria cuyo nombre es completamente diferente, y el importe es erróneo). Los procesos 4.2 y 4.5 contemplan esta situación, consultando al cliente y manteniendo el pago en D4/1 hasta tanto se reciba una explicación satisfactoria o bien hasta que el cliente que realizó el pago erróneo reclame su devolución.

Una vez que han sido definidos los flujos de datos y los procesos, como se muestra en la Fig. 3.14, el analista deberá decidir si esta función es lo suficientemente importante como para ser incluida en el diagrama de flujo de datos de nivel superior. Aunque ello involucra dinero, la necesidad de la función solo aparecerá en contadas ocasiones y las sumas ingresarán en la empresa, no egresarán. Por estas razones, la función probablemente no será lo suficientemente importante como para ser incluida en el nivel superior.

Para identificar las funciones en niveles inferiores que más bien son agregados que explosiones de procesos de nivel superior, las podemos numerar como procesos X1, X2, etc.

3.4 PAUTAS PARA DIBUJAR LOS DIAGRAMAS DE FLUJO DE DATOS

1. Identificar las entidades externas involucradas. Como ya dijimos, ello implica decidir sobre un límite preliminar del sistema. Si tiene dudas, incluya dentro de los límites de su sistema la primera "capa exterior" de los sistemas manuales o automatizados que tenga como interfaces. Recuerde que los flujos de datos se crean cuando pasa algo en el mundo exterior; una persona decide comprar algo, u ocurre un accidente o un camión llega a la playa de cargas. Si puede, vaya directamente a la última fuente de sus datos y dibuje el flujo desde allí.
2. Identificar las entradas y salidas programadas que se espera puedan producirse en el curso normal del negocio. Si la lista crece, trate de descubrir agrupamientos lógicos de entradas y salidas. Marcar las entradas y las salidas que estén relacionadas solo con condiciones de error y de excepción.
3. Identificar las consultas y los pedidos de información que podrían surgir. Especificar un flujo de datos que defina la información "dada" al sistema y un segundo flujo de datos que indique qué se "requiere" del sistema.
4. Tomar una hoja grande de papel (el reverso de una hoja usada de impresora, es bueno) y, comenzando por el costado izquierdo con la entidad externa que le parezca la principal fuente de entradas (por ejemplo, clientes), dibujar los flujos de datos que surjan, los procesos que son lógicamente necesarios y los almacenamientos de datos que cree que se requerirán. No preste atención a las condiciones de tiempo, excepto a las naturales precedencias lógicas y a los almacenamientos de datos necesarios desde el punto de vista lógico. Dibujar un sistema que nunca comience ni pare. Algunas veces es muy útil seguir una buena transacción típica de entrada a través de todo el sistema y preguntarse, "¿Qué es lo próximo que debe sucederle a esta transacción?" Seguir las reglas de notación indicadas en la Sec. 3.1, pero no numerar los procesos hasta el borrador final.
5. Dibujar el primer borrador a mano alzada y concentrarse en incluir todo, excepto errores, excepciones y decisiones. Si se encuentra con que está dibujando un rombo de decisión, péguese en la mano —las decisiones se toman dentro de los procesos de nivel inferior y no aparecen en los diagramas de flujo de datos.
6. Aceptar que se van a necesitar como mínimo tres borradores del flujo de datos del nivel superior. No debe importar que el primer borrador resulte una maraña infructuosa. Se lo podrá ordenar (ver el ejemplo de la sección siguiente).
7. Cuando tenga listo el primer borrador, controle nuevamente con su lista de entradas y salidas para asegurarse que están todas incluidas con excepción de aquéllas que operan con errores y excepciones. Anote en el borrador cualquier entrada o salida normal que no pueda ubicar. Recordar que cada dato que describe algún hecho exterior del mundo real debe ser creado y mantenido.
8. Confeccionar ahora un segundo borrador más claro utilizando una plantilla para dibujar los símbolos. Usted está pretendiendo realizar un diagrama con procesos únicos y con un mínimo de cruzamiento de flujos de datos. Para minimizar los cruzamientos:

- Primero duplique las entidades externas, si fuera necesario;
- Luego duplique los almacenamientos de datos, si fuera necesario;
- Admita entonces que se crucen los flujos de datos si no puede encontrar la disposición que reduzca el cruzamiento.

Su segundo borrador lucirá más claro, pero aún contendrá algunos cruces innecesarios, y si los revisa, verá que se puede mejorar el diseño y la relación de los símbolos de proceso. Controle nuevamente su lista de entradas y salidas y anote en el segundo borrador cuáles no ha podido ubicar.

9. Si tiene un representante simpático del usuario, o alguien que conozca la aplicación, revise con él el segundo borrador, explicando que solo es un borrador y tome nota de cualquier cambio que pueda resultar de la revisión.

10. Producir una explosión de nivel inferior de cada proceso definido en el segundo borrador, de acuerdo con las convenciones especificadas en la Sec. 3.2. Resolver el manejo de los errores y excepciones y si es necesario incorporar modificaciones en el diagrama de nivel superior. Ahora puede completarse la versión tercera y final del diagrama del nivel superior.

11. Si cree que vale la pena la inversión, pídale a un dibujante que realice una excelente copia de su diagrama de nivel superior, ya finalizado en un tamaño aproximado de 0,90x1,20 Servirá de invalorable ayuda para la presentación a los usuarios, implementadores, revisores y todos aquellos que tengan que ver con el sistema.

3.5 EJEMPLO: DISTRIBUCION CON INVENTARIO

Como ejemplo de la construcción de un diagrama de flujo de datos a escala completa, consideraremos la Corporación CBM (Computer Books by Mail) que hemos descripto al comienzo del Capítulo 2. Como ejemplo de un diagrama de flujo de datos, mostraremos el sistema actual de CBM, que resulta un clásico caso de distribución sin inventario. El sistema requerido por la nueva gerencia para la comercialización a particulares y a bibliotecarios, así como la provisión de la mayor parte de los libros desde un stock, será mucho más complejo. Se ha extraído de las especificaciones del sistema del viejo estilo la siguiente descripción. Numeramos las líneas para facilitar la referencia.

1. Los pedidos serán recibidos por correo, o tomados telefónicamen-
2. te a través de la línea interna WATS. Los pedidos telefónicos se
3. tomarán en un formulario standard, o serán ingresados directamente en
4. una CRT usando un formato standard. Cada pedido será revisado
5. para verificar que contiene toda la información importante,
6. que existe el título (o puede ser identificado), que el autor
7. es el correcto (o puede ser identificado) y que se dispone del
8. Libro (por ej. no está fuera de impresión). Si el pedido fuera defectuoso se lo encaminará a un supervisor para
9. verificar si, por ejemplo "La programación de la gerencia" por Does Jane es
10. realmente "La gerencia de programación" por Jane Doe. Cuando se in-
11. cluye el pago, se controlará que el importe sea correcto (en su de-
12. fecto se confeccionará una nota de débito o de crédito). Las
13. pequeñas diferencias se podrán ignorar.
14. Cuando el pago no acompaña al pedido, deberá controlarse el
15. archivo del cliente para verificar si proviene de una persona u
16. organización que dispone de crédito; de no ser así se
17. remitirá a la persona una confirmación del pedido solicitándole
18. el pago anticipado. Si el cliente es nuevo será incluido en el
19. archivo de clientes. Para los pedidos con pago incluido o con
20. crédito disponible, se deberá controlar el inventario para verificar si se
21. puede cumplir. Si se puede, se preparará una nota de embar-
22. que con una factura (sellada "pagado" para las facturas pre-
23. pagas) y se remitirá con los libros. Si el pedido solo puede
24. cumplirse parcialmente, se preparará una nota de embarque y una
25. factura por la parte remitida con una confirmación de la parte
26. no satisfecha (y la factura pagada cuando el pago fue enviado
27. con el pedido) creándose un registro de entrega pendiente. Los
28. pedidos pendientes se satisfarán tan pronto se reciban del editor. Cuando el pedido corresponda a
29. un libro sin existencia en el inventario, los pedidos se agruparán en
30. lotes para efectuar una requisición de compra al editor, al alcanzarse el tamaño

con asignación de descuento. Los libros devueltos se examinarán para verificar si están deteriorados y se ingresarán nuevamente al stock, emitiendo un crédito o un reintegro al cliente, según corresponda. Cuando el libro devuelto no fuera un ítem del inventario, y el editor admite devoluciones, será devuelto al editor.

Cuando se recibe el despacho de libros de un editor, su contenido se controlará contra la orden de compra original y las diferencias serán consultadas. Los títulos del despacho se controlarán con los pedidos pendientes para su prioridad de envío y el resto se ingresará al inventario. La política de control de inventario requiere un punto de pedido para cada título igual al (promedio de los pedidos recibidos en las 4 semanas previas) X (plazo de entrega de los editores) más un factor de seguridad del 50%. De este modo, si la venta promedio de un título es de 10 semanales y el plazo estimado de entrega es de 3 semanas, deberá colocarse una orden al editor cuando el total de ejemplares disponibles (y en pedidos) haya llegado a 45 (3x10x150%). El factor de seguridad podrá modificarse cada tanto por la gerencia, siendo incrementado para los títulos con ventas en aumento y viceversa. La cantidad para cada orden se determinará tomando el producto de la tasa relación de orden promedio por el tiempo de entrega, como se indicó anteriormente, y multiplicando el resultado por un factor de acopio (normalmente 3), y redondeando hasta el próximo punto de descuento superior, a menos que esto aumente la orden más del 25%. Así, en el caso anterior la orden normal será de 3x10x3 (factor de acopio) o sea 90 ejemplares. Si el editor ofrece un descuento adicional por órdenes de 100 ejemplares o más, se deberá ordenar 100. Si el descuento se ofrece solo para 120 o más ejemplares, se ordenarán 90, puesto que al ordenar 120 habría un excesivo incremento del 33%. El factor de acopio puede ser modificado por la gerencia para cada título, de tiempo en tiempo.

El cálculo de la tasa de la orden promedio incluye las órdenes ya cumplidas y los pedidos no cumplidos, tales como órdenes devueltas, órdenes sin pagos y consultas que no se transformaron en órdenes debido a que el libro no podía suministrarse desde el stock.

Cuando se recibe el pago por los libros provistos, deberá compararse con la correspondiente factura. Cuando varias facturas están pendientes en una cuenta y el pago no coincide exactamente con ninguna de ellas, se aplicará en primer término a la factura más antigua. Con frecuencia un cliente enviará un pago para cancelar varias facturas. Cuando una factura cualquiera esté atrasada en su pago más de 30 días, se le enviará al cliente un estado de cuenta con toda las facturas impagas. Cuando una factura cualquiera esté atrasada en su pago más de 60 días se confeccionará una nota enérgica con la firma del Vicepresidente.

Cuando se reciben facturas de los editores, se controlarán contra los registros de recepción de embarques y se ingresarán en cuentas a pagar. Si el descuento por pronto pago dado por el editor excede, en una base anualizada, el costo marginal de fondos (tal como la gerencia lo especifica de tiempo en tiempo), el sistema deberá confeccionar un cheque de pago el último día disponible para el descuento. Por ej. si se ofrece el 2 1/2% por pago a 30 días, equivalente al 30% anual. El sistema deberá confeccionar un cheque el día 29.

Regularmente deberán producirse informes de facturas remitidas por día, semana, mes, pagos recibidos por día, semana, mes, importes impagos por distintos períodos, agotamientos de stock, pedido pen-

89. dientes y compras a los editores. Los análisis de demanda de ven-
90. tas por título, tema, editor, con información sobre tendencias,
91. deberán estar disponibles en una base inmediata, junto con los
92. tiempos de entrega de los editores y tendencia de compras.
93. El acceso inmediato a los valores de inventario, tales como,
94. cantidad disponible, cantidad en pedido y fechas esperadas de
95. entrega son muy convenientes, así como la posibilidad de suministrar a un cliente
96. información inmediata sobre el estado de su
97. pedido en particular. Si un cliente llama y dice, "Yo le envié un cheque
98. por $10 hace cinco semanas por un libro de Bloggs", nos gustaría
99. poder decirle qué día le despachamos
100. el libro, o qué día estaremos en condiciones de despa-
101. chárselo.

Esta declaración descriptiva de los requerimientos del nuevo sistema, como toda exposición, contiene información correspondiente a distintos niveles de detalle —procedimientos combinados con estructuras, políticas importantes mezcladas con cosas triviales. ¿Cómo podemos extractar la información para confeccionar un diagrama de flujo de datos de nivel superior?

Primero, identifiquemos cuántas entidades externas tenemos, mediante un examen de la declaración, antes de identificar y clasificar las entradas y salidas:

ENTIDADES EXTERNAS

NOMBRE	REFERIDAS EN LA LINEA Nº
Clientes	1-2 (por implicación), 15-19, 70, 73
Supervisor de Entrada de Pedidos	8-9
Editor	28, 30, 34-35, 36, 43, 46, 55, 76, 79
Gerencia	48, 60, 81, 86-89 (por implicación)
Vicepresidente	75, 89-95?

Podrá resultar que "gerencia" y "Vicepresidente" sean la misma entidad, o podremos encontrar que "gerencia" realmente está compuesta por "gerente de ventas", "gerente de abastecimiento" y "gerente de compras". Por lo menos tenemos una lista en borrador como resultado de la exposición. En cuanto a las entradas y salidas, véase la tabla de la página siguiente. Allí el análisis de entradas/salidas identifica todos los flujos de datos que podemos encontrar en la exposición, con excepción de las consultas que puedan realizarse. Conviene tratarlas por separado (ver más adelante), puesto que comúnmente consisten en entradas y salidas apareadas.

En varios lugares hemos colocado un asterisco en una entrada o salida que corresponde a un flujo de materiales (envíos, devoluciones) sin un flujo de datos especificado. Normalmente, por supuesto, cada movimiento de materiales va acompañado de un comprobante (nota de embarque, reclamo de crédito por devolución) que contiene los datos de interés. Hacemos notar que estos flujos de datos deben ser determinados y analizados.

Vale la pena recalcar también que este listado de entradas/salidas se ha confeccionado a partir de una exposición que nos fue provista. También podría hacerse a partir de las entrevistas del analista, de un conjunto de memorando, o de un conjunto de ambos. En cualquier caso, la referencia debe indicarse en el listado de entradas/salidas.

LISTADO DE ENTRADAS/SALIDAS

ENTRADAS	DESDE (EE)	SALIDAS	HACIA (EE)	LINEA (REF)
Pedidos-Correo	Cliente			1
Pedidos-telefónicos	Cliente			1
		Ordenes defectuosas		8
			Supervisor de ingreso de pedidos	
Pago	Cliente			11
		Confirmación	Cliente	17
		Pedido de prepago	Cliente	18
		Nota de envío	Cliente	21
		Factura	Cliente	22
		Libros *	Cliente	23
		Requisición de compra	Editor	30
Libros devueltos *	Cliente			31
		Crédito	Cliente	32
		Reintegro	Cliente	32
		Libros devueltos *	Editor	34
Embarque de libros *	Editor			36
		Consulta por diferencias	Editor	38
		Orden	Editor	45
		(¿En qué se diferencia esto de la "Requisición de Compra", 30?)		
Modificación factor de seguridad	Gerencia			47
Modificación factor de acopio	Gerencia			59
Pedido no cumplido *	Cliente			64
		Informe	Cliente	72
		Nota enérgica	Vicepresidente, luego Cliente	74
Facturas	Editor			76
		Cheque de pago	Editor	81
		Facturas-informe diario	Gerencia	86
		Facturas-informe semanal	Gerencia	86
		Facturas-informe mensual	Gerencia	86
		Ingresos-informe diario	Gerencia	87
		Ingresos-informe semanal	Gerencia	87
		Ingresos-informe mensual	Gerencia	87
		Importes impagos-informe	Gerencia	87
		Falta de stock-informe	Gerencia	88
		Pedidos pendientes-informe	Gerencia	88
		Compras a editores-informe	Gerencia	88

* Movimiento de materiales: ¿qué datos están asociados con él?

Consultas			
Desde/Hacia	Dado	Representar	Línea (Ref.)
Gerencia	Título del libro	Análisis de ventas	90
Gerencia	Tema	Análisis de ventas	90
Gerencia	Editor	Análisis de ventas	90
Gerencia	Editor	Análisis tiempo de entrega	92
Gerencia	Editor	Análisis de compras	93
Gerencia	Título	Cantidad disponible	94
Gerencia	Título	Cantidad pedida	94
		Fecha de entrega	94
Cliente	Nombre y detalle del pedido	Estado del pedido	96-101

Por el momento no nos interesa mayormente la inmediatez o el valor de estas consultas; en este contexto "representar" podría significar "producir un informe impreso el lunes próximo por la mañana". Las consultas se analizarán en detalle en el Capítulo 7. Lo que nos preocupa es la existencia de los flujos de los datos necesarios (y el almacenamiento de datos).

Para mantenernos dentro de la regla de simplificación de los diagramas de flujo de datos (pág. 35), queremos eliminar de nuestro primer borrador aquellas entradas y salidas relacionadas con las condiciones de error o de excepción y tratar como un solo flujo a aquellas que sean similares lógicamente. Por eso trataremos "pedidos por correo" y "pedidos telefónicos" simplemente como "pedidos"; lo lógicamente importante es conocer si están acompañadas o no del pago. Omitiremos en nuestro flujo de datos de nivel superior a "órdenes defectuosas", "libros devueltos", "crédito", "reembolsos", "consultas por diferencias", "pedidos sin completar", "informes" y "notas enérgicas".

Hecho esto, realicemos un primer borrador del diagrama de flujo de datos partiendo del hecho saludable de que los clientes originan los "pedidos".

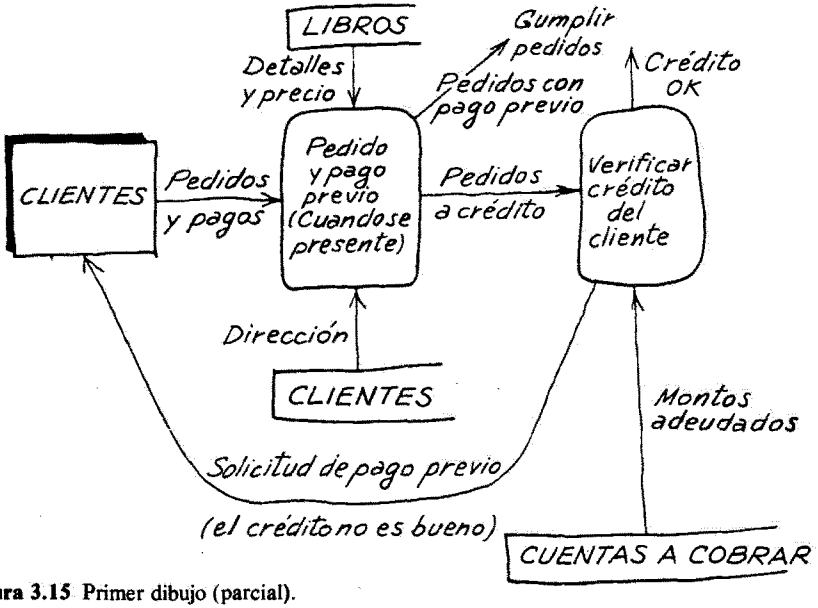


Figura 3.15 Primer dibujo (parcial).

La Fig. 3.15 muestra un borrador a mano alzada de los primeros pocos pasos del procesamiento de pedidos. En este primer borrador reunimos todos los procesos que examinan el pedido para comprobar si está completo, para controlar el archivo de libros a fin de verificar si existe el libro, para controlar que el pago (si el cliente lo ha enviado) tiene el importe correcto y para consultar en el archivo de clientes si ya hemos tenido relaciones anteriores con él. Confeccionamos luego el proceso simple "validar pedido y prepago (cuando lo haya)", y observamos que necesitaremos expandirlo cuando comencemos a dibujar el flujo de datos de nivel inferior.

Nuestra próxima tarea será explicar qué queremos decir con "cumplir pedido". Vamos a ampliar un poco nuestro borrador, hasta el punto que se ve en la Fig. 3.16. Aquí tomamos los pedidos y obtenemos acceso al inventario existente para verificar si los podemos satisfacer o no. Para aquellas que podemos satisfacer, generaremos una nota de envío (la que puede ser utilizada por el depósito para tomar los libros del stock) y una factura (la que deberá sellarse "pagado" en el caso de haberse recibido el pago juntamente con el pedido). El borrador se está haciendo algo complejo, pero podemos ver cómo está creciendo el sistema. Continuamos por este camino hasta finalizar el primer borrador del sistema completo, tal como se muestra en la Fig. 3.17.

Como se puede ver, el primer borrador puede resultar algo desordenado y confuso; sólo cuando se identifican los flujos de datos y los procesos se puede ir analizando cómo se conectan entre sí, de manera que es casi imposible obtener de entrada una buena distribución. Por lo menos ahora tenemos los aspectos más importantes del sistema en una hoja de papel. El próximo paso es volver a dibujar el primer borrador con una plantilla. La Fig. 3.18 muestra el resultado.

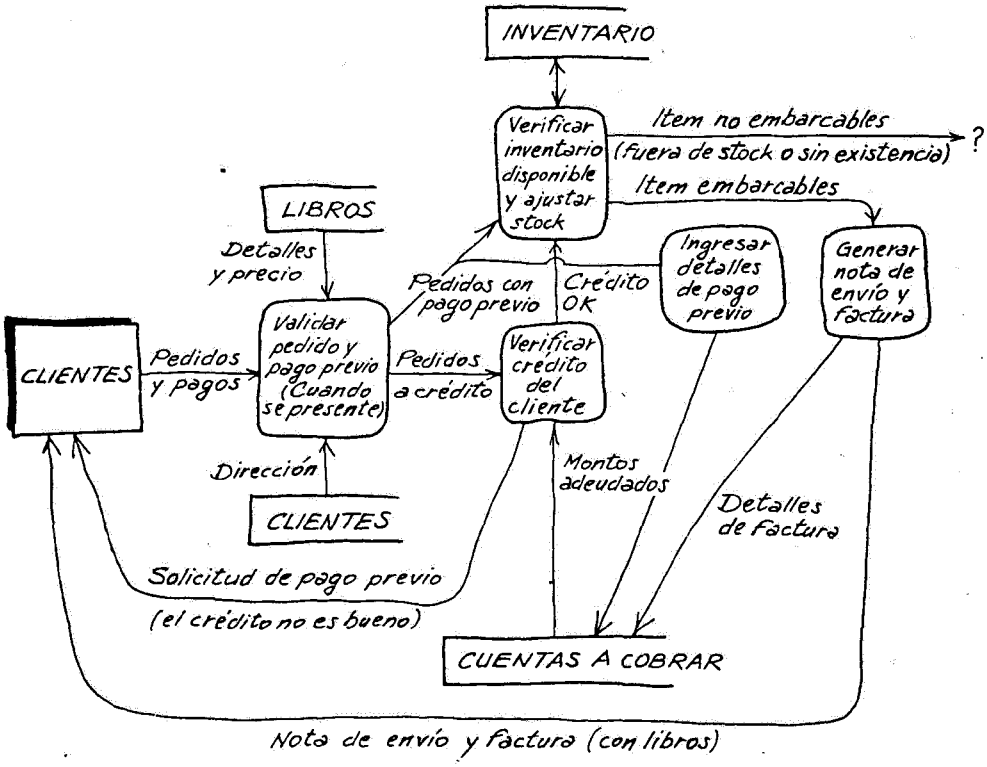


Figura 3.16 Primer dibujo (extendido).

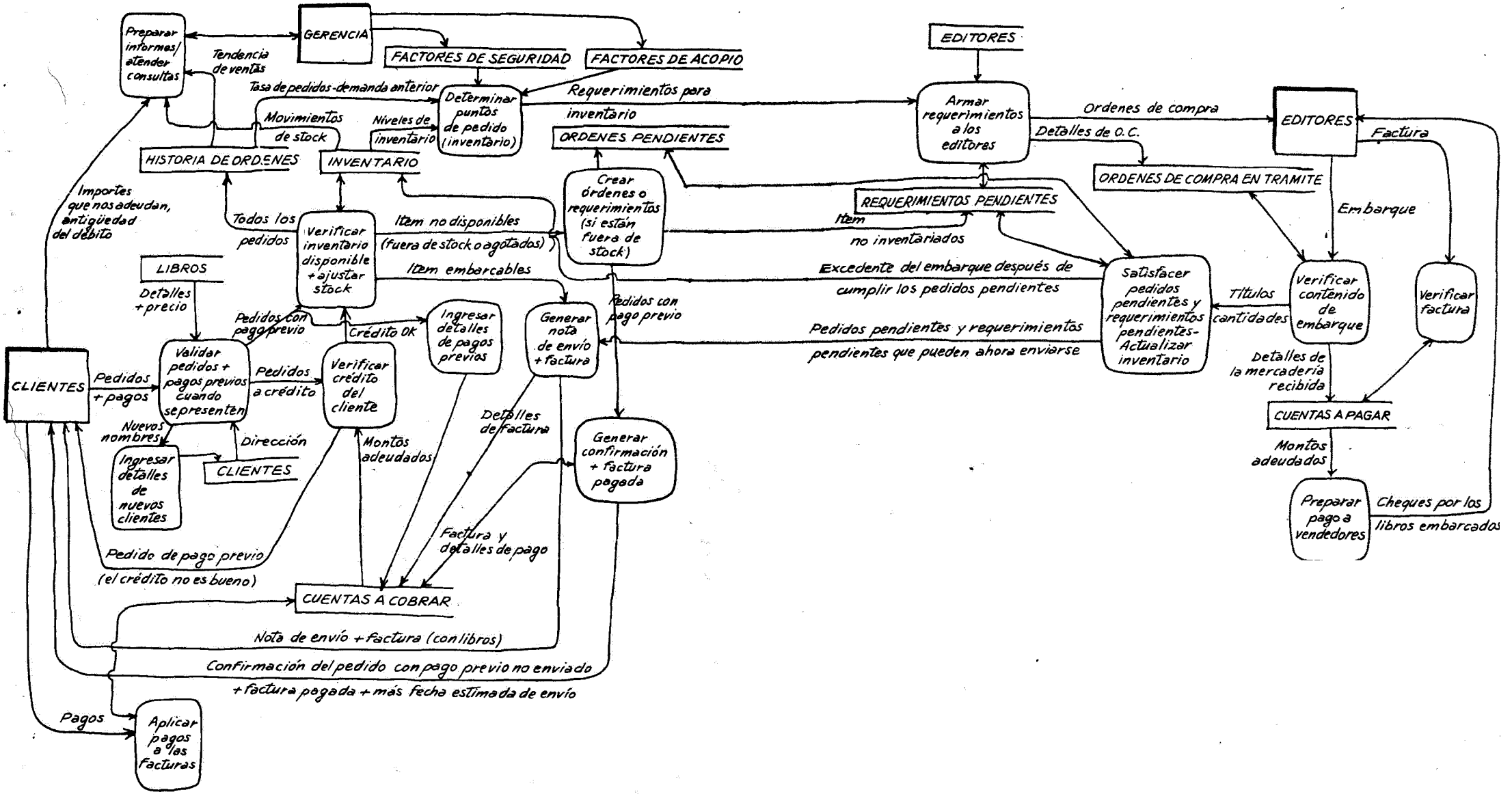
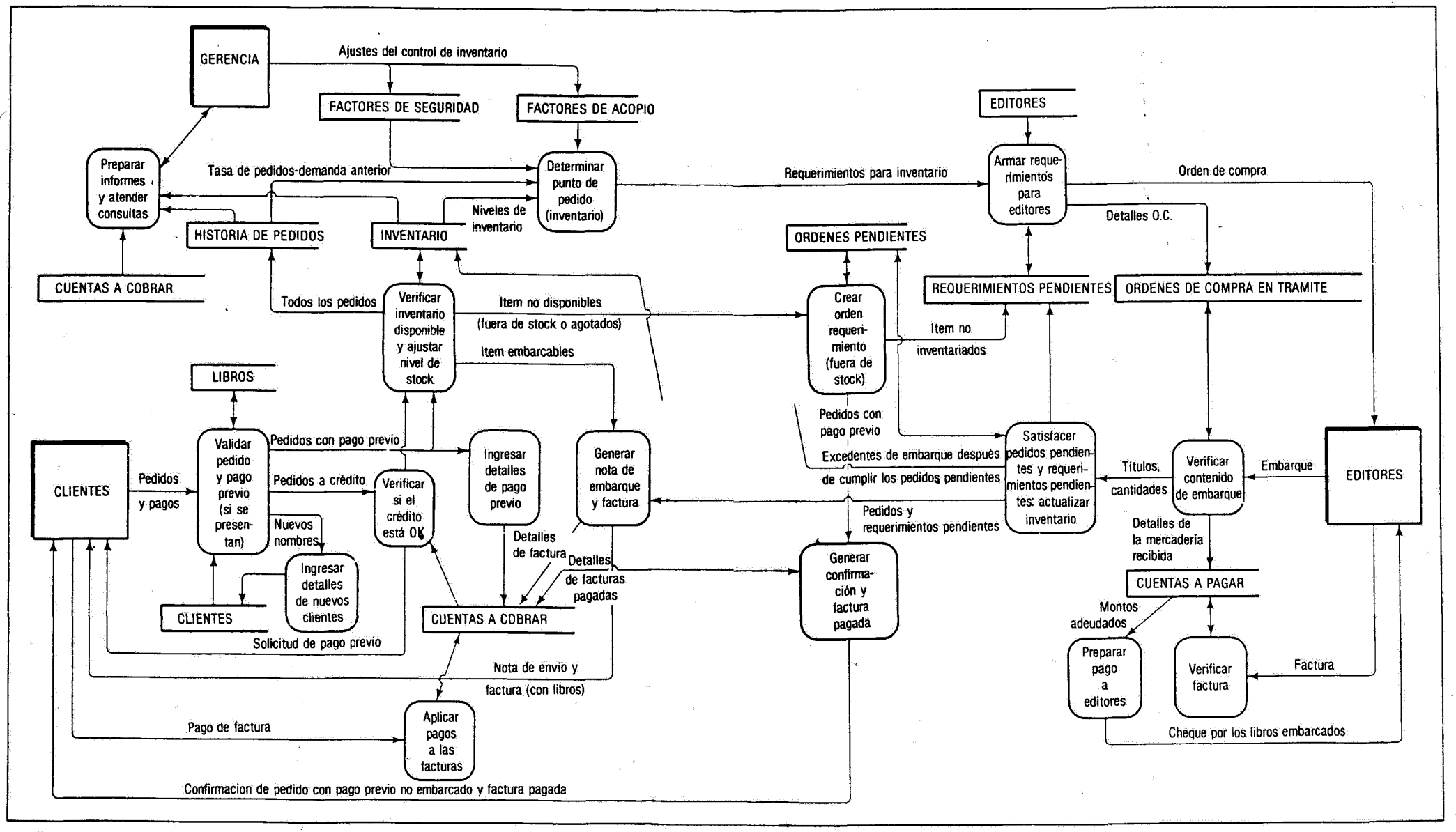


Figura 3.17 Primer dibujo (final).



No se indica: Consulta de clientes, informes a gerencia sobre compras y pedidos pendientes, creación/mantenimiento de archivos para libros y editores.

Figura 3.18 Segundo dibujo.

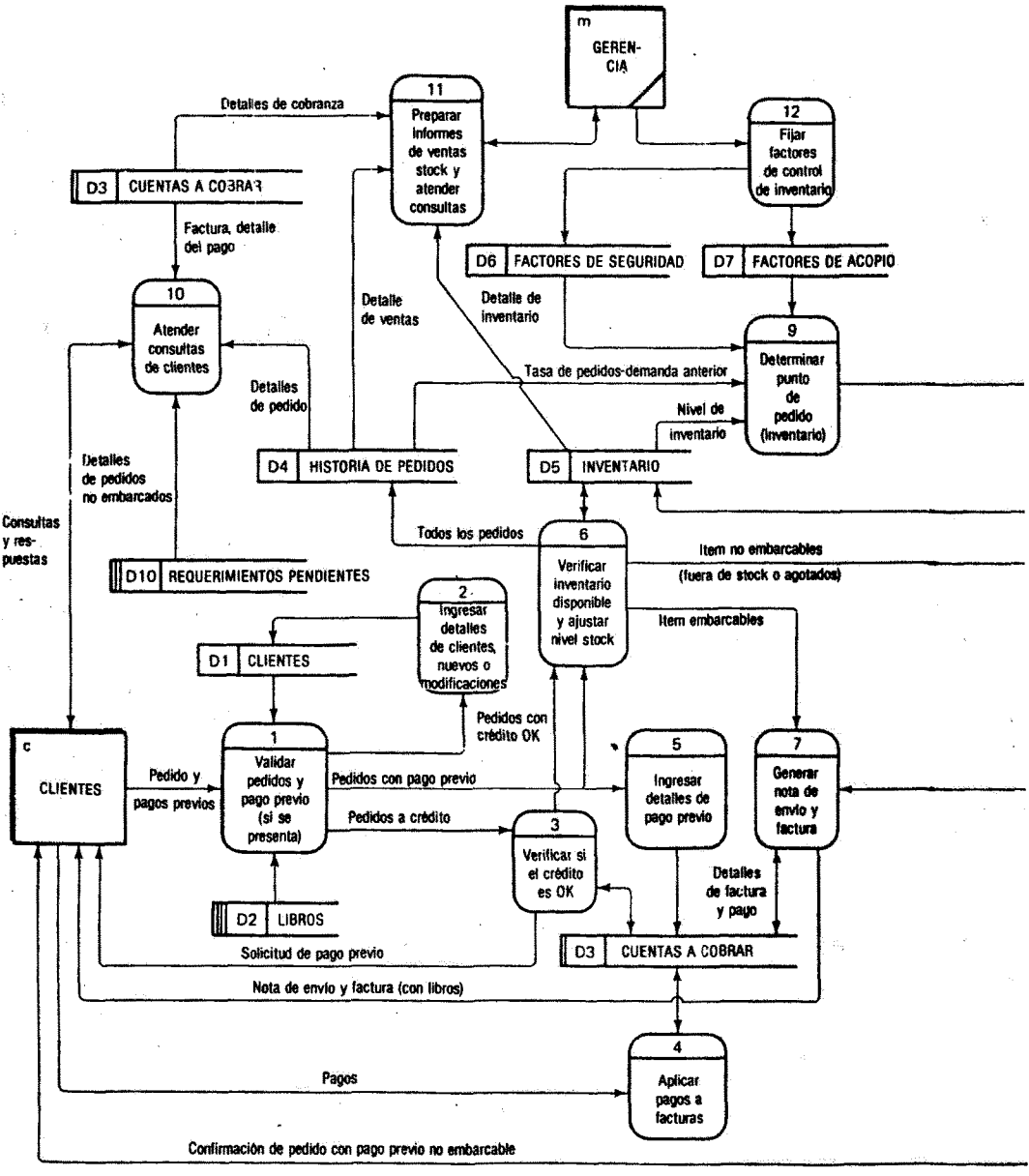
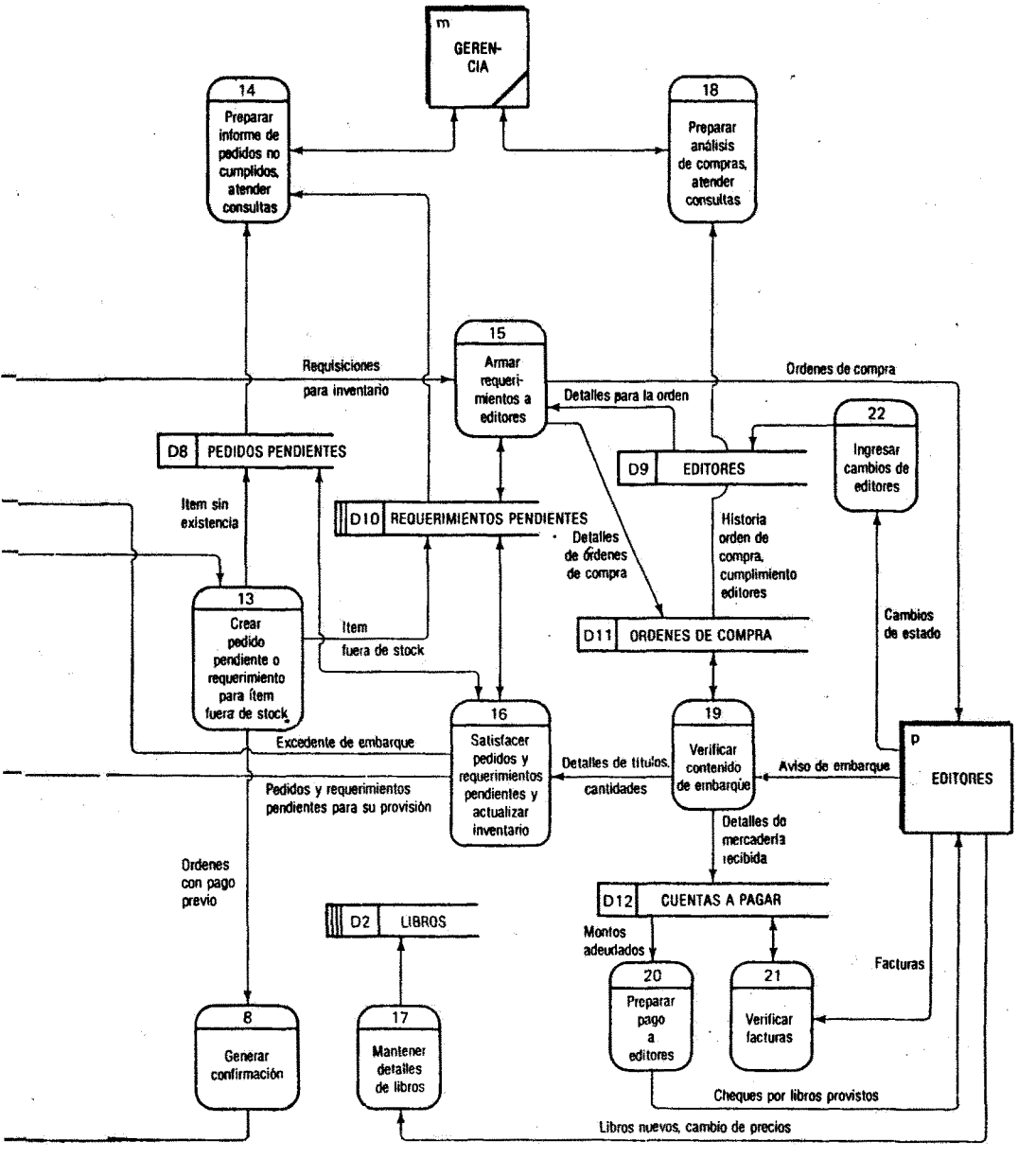


Figura 3.19 Dibujo final.


m
GEREN-
CIA
14
Preparar informe de pedidos no cumplidos, atender consultas
18
Preparar análisis de compras, atender consultas
15
Armar requeri-mientos a editores
Requisiciones para inventario
Ordenes de compra
Detalles para la orden
22
Ingresar cambios de editores
D8 PEDIDOS PENDIENTES
D9 EDITORES
Item sin existencia
D10 REQUERIMIENTOS PENDIENTES
Detalles de órdenes de compra
Historia orden de compra, cumplimiento editores
13
Crear pedido pendiente o requerimiento para ítem fuera de stock
Item fuera de stock
D11 ORDENES DE COMPRA
Cambios de estado
Excedente de embarque
16
Satisfacer pedidos y requerimientos pendientes y actualizar inventario
19
Verificar contenido de embarque
Detalles de títulos, cantidades
Aviso de embarque
P
EDITORES
Pedidos y requerimientos pendientes para su provisión
Detalles de mercadería recibida
Ordenes con pago previo
D12 CUENTAS A PAGAR
D2 LIBROS
Montos adeudados
Facturas
20
Preparar pago a editores
21
Verificar facturas
8
Generar confirmación
17
Mantener detalles de libros
Cheques por libros provistos
Libros nuevos, cambio de precios

El segundo borrador es más claro y prolijo, ya que una vez que se pueden ver todos los procesos y conexiones, es factible mejorar la distribución de los mismos en la hoja. Obsérvese que un flujo de datos confuso se ha aclarado mediante la duplicación del almacenamiento de datos de CUENTAS A COBRAR. El gráfico todavía está muy lleno en su parte superior, alrededor de la entidad GERENCIA y se puede observar en la parte inferior que algunas funciones significativas (aunque no complejas) han sido omitidas para no restarle claridad al resto del diagrama. Como se ha dicho, cada almacenamiento de datos que contiene información "básica" sobre el mundo exterior, debe ser creado y mantenido a medida que se modifique el mundo exterior. El almacenamiento de datos CLIENTES se mantiene en el segundo borrador en la medida en que podamos detectar y agregar nuevos nombres. También necesitamos las funciones de modificación (por ejemplo, de la dirección o de la persona responsable de la facturación) y de eliminación o baja de clientes. El almacenamiento de datos LIBROS deberá actualizarse con bastante frecuencia, ya que todos los días se publican nuevos libros y los editores ocasionalmente modifican los precios. El almacenamiento de datos EDITORES, una vez establecido, podrá tener unos pocos cambios durante el año, pero generalmente es muy estable.

A costa de complicar el diagrama podemos agregar los diferentes flujos de datos y las funciones asociadas con las consultas gerenciales sobre ventas, inventarios, compras y devoluciones como se muestra en el diagrama de flujo de datos final de la Fig. 3.19.

Se invita al lector a recorrer los flujos de datos y funciones y verificar que las pautas especificadas en las páginas 35 y 36 han sido satisfechas, excepción hecha de aquéllas relacionadas con condiciones de error y excepción que específicamente hemos excluido. Hacemos notar que aun con un gráfico de esta complejidad solo ha sido necesario duplicar una entidad externa (m: GERENCIA) y tres almacenamientos de datos (D3, D10 y D2).

3.6 FLUJO DE MATERIALES Y FLUJO DE DATOS

En varios puntos anteriores cuando describíamos el sistema que acabamos de graficar, comentábamos que estábamos tentados a describir el flujo de materiales en lugar del flujo de datos asociado, que es lo correcto. Es importante mantener separados estos dos conceptos, especialmente en industrias manufactureras y de distribución, donde la mayor parte del negocio gira alrededor del movimiento de materiales. Aun en bancos, negocio que pensamos se encuentra más próximo al dato puro, existen serios problemas vinculados con el movimiento físico de cheques y comprobantes.

En consecuencia, necesitamos disponer de una forma para graficar el flujo de materiales cuando sea necesario, y vincular el diagrama de flujo de materiales al diagrama de flujo de datos. El flujo de materiales es de naturaleza esencialmente física, pero deseamos en la medida de lo posible, describir en términos lógicos las operaciones realizadas con los materiales. Por ello no estamos satisfechos con el gráfico de flujo de materiales del tipo de la Fig. 3.20 pero necesitamos alguna forma de describir las funciones lógicas que transforman el material en cada punto con la indicación de los flujos de datos asociados. La Fig. 3.21 muestra ambos flujos —de datos y materiales— con referencias al diagrama de flujo de datos original (Fig. 3.19).

Obsérvese que solo dos de las transformaciones de materiales corresponden a procesos del diagrama de flujo de datos (16 y 19) y que algunos flujos de materiales tienen lugar sin un flujo de datos (por ejemplo, agregados al stock).

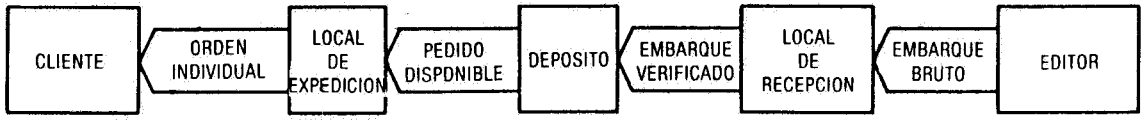


Figura 3.20 Gráfico de flujo de materiales.

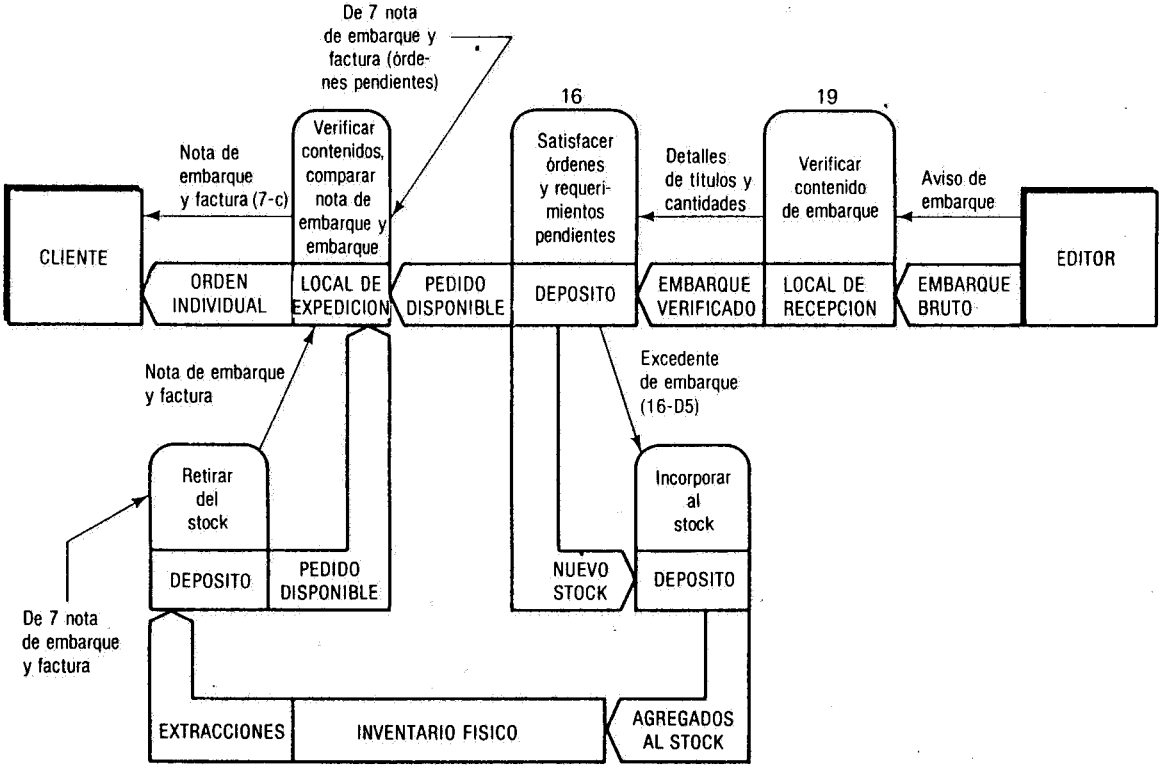


Figura 3.21 Flujo de datos y flujo de materiales.

BIBLIOGRAFIA

3.1 D. Martin y G. Estrin, "Models of Computations and System-Evaluations of Vertex Probabilities in Graph Models of Computations", *Journal of the ACM*, Vol. 14, N° 2, abril de 1967.

3.2 E. Yourdon y L.L. Constantine, *Structured Design*, Prentice-Hall, Englewood Cliffs, N.J., 1979.

3.3 G.E. Whitehouse, *Systems Analysis and Design Using Network Techniques*, Prentice-Hall, Englewood Cliffs, N. J., 1973.

3.4 D. Ross, "Structured Analysis (SA): A Language for Communicating Ideas", *IEEE Transactions on Software Engineering*, Vol. 3, Número 1, enero de 1977.

Ejercicios y puntos de discusión

1. En la Fig. 3.19, ¿por qué "requerimientos para inventario" (9-15) va directamente a "15: Reunir requerimientos para editores", mientras que los ítem que no están almacenados en el inventario son acumulados en D10: REQUERIMIENTOS PENDIENTES?
2. ¿Por qué el proceso 9 "determinar puntos de pedido para ítem de inventario", no requiere los datos de D8: ORDENES DEVUELTAS?
3. ¿Es razonable el algoritmo de pedido utilizado por CBM (líneas 44-49 de la exposición)? ¿Puede mejorarse?
4. Suponiendo que el único negocio de CBM es la venta de pedidos por correo, ¿qué otros sistemas de información y procesamiento de datos podría suponer Ud. que posee, además del sistema indicado en la Fig. 3.19?
5. Adapte la Fig. 3.19 para indicar el flujo de datos de una empresa que almacena autopartes para automóviles nacionales e importados (slogan: "Si no lo tengo, lo conseguiré") para su posterior venta de pedidos por correo. ¿Cómo deberá modificarse el flujo de datos si se crea un departamento de ventas en el mostrador?
6. Haciendo algunas suposiciones razonables, explotar los procesos "19: Verificar contenidos de embarques", "20: Preparar pago a vendedores", y "21: Verificar facturas" (en la Fig. 3.19) a los efectos de realizar un diagrama de flujo de datos en detalle del subsistema de cuentas a pagar, que incluya las condiciones probables de error y excepción.
7. Liste todas aquellas empresas en que considere que el negocio consiste en el mantenimiento de inventarios y la distribución, sea por correo o por mostrador. Identifique las similitudes y diferencias de sus flujos de datos.
8. Dibuje los diagramas de flujo de datos completos de los siguientes tipos de empresas:
   (a) Fabricación a pedido
   (b) Fabricación, mantenimiento en stock y distribución
   (c) Concesión de préstamos al consumidor
   (d) Mantenimiento de cuentas de ahorro
   (e) Emisión y mantenimiento de pólizas de seguros contra accidentes y atención de reclamos
   (f) Venta por horas de servicios profesionales y pago de honorarios a profesionales (por ejemplo, práctica legal).
   (g) Agencia de registros de reservas para viajes.

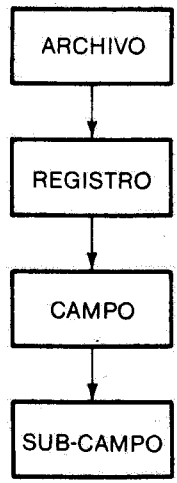
4

CONSTRUCCION Y USO DE UN DICCIONARIO DE DATOS

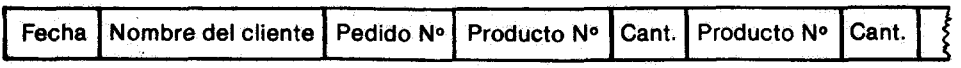
En el Capítulo 2, vimos que si quisiéramos profundizar los detalles de los contenidos de los flujos de datos, almacenamientos de datos y procesos, necesitaríamos algún lugar estructurado para guardar todos estos detalles. El diccionario de datos provee este lugar. El nombre *diccionario de datos* comienza a extenderse cuando empezamos a incluir los detalles de los procesos, que estrictamente hablando son lógica más bien que datos. Posiblemente el diccionario de datos debería llamarse realmente *guía de proyecto.* Sin embargo, el nombre de diccionario de datos es de un uso muy amplio y lo adoptaremos sin inhibiciones para todo lo que necesitemos describir.

4.1 EL PROBLEMA DE DESCRIBIR DATOS

En los buenos tiempos del registro unitario, los términos utilizados para describir los datos, eran muy simples. Una tarjeta se dividía en campos; la tarjeta en sí misma era un registro, y una cantidad de registros constituían un archivo. No era *tan* simple como esto; un campo de datos de la forma MMDDYY podía considerarse compuesto por tres sub-campos, de manera que la descripción jerárquica del dato abarcaba cuatro niveles, como se muestra a continuación:



Esta jerarquía se tomó para el procesamiento de cinta y luego para el procesamiento de disco. IBM denomina a un archivo como *conjunto de datos,* y encontramos registros de longitud variable en donde un encabezamiento era seguido por un número de registros de cola o grupos repetitivos, tal como en un pedido:



Todavía podemos utilizar las mismas palabras para describir nuestros datos.

Con el desarrollo de la tecnología de base de datos, el vocabulario simple ya no es suficiente. Desde que parte de la técnica de la base de datos se basa en tomar los archivos para cada aplicación e integrarlos en una base de datos que pueden utilizar todas las aplicaciones, el término *archivo* se hace cuestionable. El Information Management System (IMS) (Sistema de Información Gerencial) [4.1] de IBM describe a los datos como *campos,* que se combinan en *segmentos,* que se combinan en *bases de datos.* El Date Base Task Group (DBTG) (Grupo de Tareas de Base de Datos) de la Conference on Date Systems Languages CODASYL (Conferencia sobre lenguajes de Sistemas de Datos) [4.2] describe a los datos como *ítem de dato,* los cuales se combinan en *agregaciones de datos,* los cuales a su vez se combinan en *registros,* donde sus relaciones se expresan como *conjuntos.* La División de Datos de COBOL estipula ítem elementales, los cuales se combinan en *ítem de grupo,* los cuales por su parte se combinan en *registros,* que conforman *archivos.* Discutiremos otras terminologías en el Capítulo 6; algunos términos diferentes se resumen en la Tabla 4.1.

Dado este cúmulo de conceptos y términos, necesitamos elegir algunas palabras simples que no deben entrar en conflicto indebidamente con estos vocabularios comunes y que deberán permitirnos describir tanto los flujos de datos como los almacenamientos de datos a nivel lógico.

La experiencia nos enseña que podemos describir en tres niveles todo lo que nos resulte de interés como analistas.

1. *Elementos de datos:* Son partes de datos que no resulta significativo descomponer más aún para nuestro propósito. Por ejemplo, "fecha" es un elemento de datos a los fines de la mayor parte de nuestros propósitos durante el análisis, aunque sea necesario al codificar una rutina de conversión de fecha, tenerla en cuenta como una estructura formada por "mes, día, año".

2. *Estructuras de datos:* está formada por elementos de datos, o por otras estructuras de datos, o por una combinación de ambas. Consideremos el contenido del flujo de datos "pedidos" que vimos en el Capítulo 2:

```
PEDIDO
  PEDIDO-IDENTIFICACION
    CLIENTE-DETALLES
    LIBRO-DETALLES
```

y podemos expandir esto más aún con notas:

```
PEDIDO
  PEDIDO-IDENTIFICACION
    PEDIDO-FECHA
    CLIENTE-PEDIDO-NUMERO. . . . .Normalmente presente
  CLIENTE-DETALLES
    EMPRESA-NOMBRE
    PERSONA-AUTORIZANTE. . . . . .Opcional
      NOMBRE. . . . . .Puede ser sólo la inicial
      APELLIDO
   TELEFONO
```

TABLA 4.1 Ejemplos de definiciones de datos

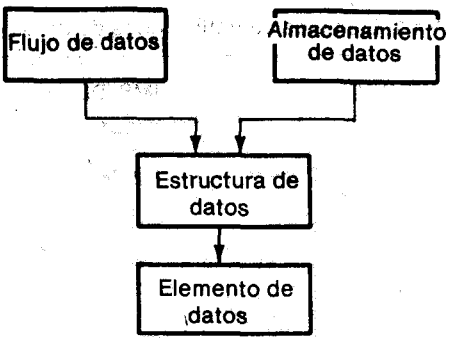
	COBOL	PL/1	IMS (DL/1)	CODASYL
Unidad más pequeña de dato	Item elemental	Elemento	Campo	Item de datos
Grupo de las unidades más pequeñas	Item de grupo	Estructura	Segmento	Agregación de datos
Unidades repetitivas	Tabla/grupo de repetición	Matriz	Segmento	Vector/grupo de repetición
Entidad que se procesa de una vez	Registro	Registro	Registro lógico	Registro
Mayor agrupamiento reconocido	Archivo	Archivo	Base de datos	Conjunto/esquema

CODIGO-DE-AREA
CENTRAL
NUMERO
EXTENSION. . . . . .Opcional
DESPACHAR-A-DIRECCION
CALLE
CIUDAD-LOCALIDAD
CODIGO-POSTAL
FACTURAR-A-DIRECCION. . .Si no se indica, igual a DESPACHAR-A-DIRECCION
CALLE
CIUDAD-LOCALIDAD
CODIGO-POSTAL
LIBRO-DETALLES. . .Una o más iteraciones de este grupo de elementos de datos
AUTOR-NOMBRE. . .Una o más iteraciones de este elemento de datos
TITULO
ISBN....International Standard Book Number: Número internacional asignado al libro (opcional)
LOCN....Librery of Congress Number: Número de la Biblioteca del Congreso de EE.UU. (opcional)
EDITOR-NOMBRE. . . . . .Opcional
CANTIDAD

"TELEFONO" es una estructura de datos compuesta por cuatro elementos de datos CODIGO-DE-AREA, CENTRAL, NUMERO y EXTENSIÓN. "CLIENTE-DETALLES" es una estructura de datos compuesta por el elemento de datos "EMPRESA-NOMBRE" más la estructura de datos "PERSONA-AUTORIZANTE" más la estructura de datos "TELEFONO", etc. "PEDIDO" es en sí mismo una gran estructura de datos.

3. *Flujos de datos y almacenamiento de datos:* Ya hemos discutido este nivel de la descripción de datos. Los flujos de datos son circuitos o "tuberías" a través de las cuales se trasladan las estructuras de datos; los almacenamientos de datos son lugares donde las estructuras de datos se almacenan hasta que son utilizadas. *Los flujos de datos son estructuras de datos en movimiento; los almacenamientos de datos son estructuras de datos en reposo.*

Nuestra descripción jerárquica de datos es entonces la siguiente:



Una vez que tenemos la descripción lógica de datos en este nivel, es fácil volcarla al vocabulario del lenguaje particular o sistema de base de datos que se utilizará en la implementación física.

4.2 QUE DESEARIAMOS QUE CONTENGA UN DICCIONARIO DE DATOS

Previo a discutir las implementaciones específicas de los diccionarios de datos manuales y automatizados, examinaremos los elementos de datos y estructuras de datos para ver qué podemos desear registrar acerca de ellas en las diferentes circunstancias. Veremos también los atributos de los flujos de datos, almacenamientos de datos, procesos, entidades externas y demás elementos que encontramos en el análisis, para analizar qué es lo que tiene valor registrar. Ello nos dará un panorama para evaluar cualquier sistema diccionario de datos para programar uno propio.

4.2.1 Descripción de un elemento de datos

La información mínima necesaria para definir un elemento de datos es su *nombre* y su *descripción.* Por ejemplo,

"CALLE": -número de la casa
-nombre
-calle

también puede incluir

-número del departamento
-posta restante
-estafeta

El nombre deberá elegirse de manera tal que tenga la mayor significación posible para el usuario; es conveniente, aunque no necesario, seguir las reglas de definición de datos de la programación en COBOL, especialmente si el sistema será implementado en COBOL. Los nombres de los datos en COBOL pueden tener hasta 30 caracteres de largo, utilizando solamente los caracteres desde "A" hasta "Z" y desde 0 hasta 9 con guiones para separar las palabras.

A menudo, se deberá necesitar el mismo nombre en diferentes contextos, de manera que también es conveniente poder utilizar la convención de calificación de COBOL para que los nombres sean únicos. Luego, podemos definir FECHA como parte de FACTURA y también como parte de PEDIDO y CHEQUE-PARA-EL-PAGO. Podemos establecer una

FECHA única, escribiendo FECHA-DE-FACTURA, FECHA-DE-PEDIDO, FECHA-DE-CHEQUE-PARA-EL-PAGO, etc.

La descripción podrá ser un breve esbozo del significado del elemento de datos y puede incluir un típico ejemplo.

Además del nombre y la descripción deseamos poder registrar, entre otras cosas, lo siguiente:

1. Alias del elemento de datos
2. Elementos de datos relacionados
3. Rango de los valores y significado de los valores
4. Longitud
5. Codificación
6. Otras informaciones de validación

1. *Alias.* Los alias pueden aparecer debido a que diferentes departamentos usuarios han denominado a una misma cosa con nombres diferentes, por ejemplo el personal del depósito la denomina "número de requisición" y el personal de compras, "número de pedido". Los alias también pueden aparecer debido a que una misma cosa se define en programas escritos en diferentes lenguajes o por distintos programadores. De esta manera, NUMERO-DE-REQUISICION puede encontrarse en el sistema corriente como REQUIS-NUM, REQNO, ORDNO, ORDNUM, POSEQNO, PUR7061 y otros alias internos. Deseamos registrar todos ellos e incluirlos no solo bajo la definición de NUMERO-DE-REQUISICION sino separadamente, con una entrada propia. De manera que si vamos a alguna documentación vieja y encontramos PUR7061 la podemos hallar en el diccionario de datos en una entrada que diga

"PUR7061 alias de NUMERO-DE-REQUISICION

y luego en NUMERO-DE-REQUISICION encontraremos los detalles completos.

También podemos en forma deliberada crear alias para nuestro nuevo sistema. Por ejemplo, en IMS todos los nombres de datos tienen un máximo de ocho caracteres. En consecuencia podemos elegir un nombre del elemento de datos REQUISNO para IMS y podemos aparearlo con su alias "externo" NUMERO-DE-REQUISICION cuando se requiera.

2. *Elementos de datos relacionados.* Algunas veces queremos poder puntualizar los elementos de datos que tienen nombres relacionados, aunque no sean alias. Por ejemplo, en una declaración la fecha puede aparecer como 15JUL1977, pero la fecha en cada factura o pago aparece como 07/15/77. El primer elemento de datos se denomina FECHA-COMPLETA y el segundo simplemente FECHA. En la entrada de cada una de ellas en el diccionario de datos desearíamos encontrar "también ver. . .". Por supuesto, debemos tener cuidado de iniciar los nombres de ambas con las mismas cuatro letras, de manera que en la lista alfabética de los elementos de datos se encuentren muy próximas.

3. *Rango de los valores y significado de los valores.* Tan pronto como comenzamos a examinar los valores que puede tomar un elemento de datos, nos damos cuenta de que existen dos tipos de elementos de datos:

— Aquellos que en todos los casos prácticos pueden tomar cualquier valor dentro de un determinado rango, por ejemplo, un importe en dólares desde cero hasta $ 999.999,99 con exactitud al centavo o una temperatura desde 0° hasta 300°.
— Aquellos que solo pueden tomar determinados valores, por ejemplo, el número de departamento que puede ser 36, 08, 29 o 71, pero no tener otros valores. Otro ejemplo de este segundo caso puede ser el estado civil, que puede ser, soltero, casado, viudo o divorciado. Generalmente estos valores forman un *código*, con *significados* convencionales. Así podemos tener

Número de Departamento

Valor	Significado
36	Ventas
08	Contaduría
29	Depósito
71	Publicidad

Estado civil

Valor	Significado
C	Casado
S	Soltero
D	Divorciado
V	Viudo

Al primer tipo de elemento de datos lo denominaremos *continuo* porque sus valores son prácticamente continuos dentro de un rango; al segundo tipo lo denominaremos *discreto,* porque toma valores discretos.

Para los elementos de datos *continuos* debemos indicar el rango de los valores que pueden tomar, un valor típico y alguna información sobre el tratamiento de los valores límites. Por ejemplo, PAGO-IMPORTE en un cheque puede tener un rango desde 1 centavo hasta diez millones de pesos. Debemos observar que un valor cero es sospechoso, y que un cheque de menos de un peso, aunque es legal, deberá ser señalado para su análisis por ser inusual. Los cheques cuyos importes superen los cien mil pesos también deberán ser señalados para su control por un funcionario. También podemos hacer notar que los reclamos por pagos inferiores a cinco pesos no se deben enviar, aunque la información pertenece en rigor a la lógica del proceso que genera los reclamos.

Para los elementos de datos *discretos* debemos hacer notar los valores y el significado que se da a cada valor. Algunos elementos de datos discretos comunes son ESTADO-CIVIL como hemos visto anteriormente, SEXO, PROPIA o ALQUILADA (vivienda), CUENTAS-CORRIENTES -o- CAJA-DE-AHORRO (tipo de cuenta) y DEPARTAMENTO-CODIGO. A medida que una empresa va creciendo, la cantidad de los valores de DEPARTAMENTO-CODIGO también crecerá hasta llegar posiblemente a centenares. Un ejemplo familiar de elemento de datos discreto con varios valores es el código de tres letras utilizado para identificar aeropuertos. La Fig. 4.1 muestra los primeros veinticinco valores y significados del AEROPUERTO-CODIGO de entre más de mil listados en la Guía Oficial De Aerolíneas de Norteamérica.

Resulta claro que el analista deberá apreciar en qué punto deja de tener sentido considerar al elemento de datos como discreto y tratarlo como si fuera un elemento continuo, el cual será utilizado como clave para el acceso a un valor en un almacenamiento de datos. Como regla, si la tabla de valores y significados puede ser mecanografiada en una o dos páginas, valdrá la pena la tabulación de los valores en el diccionario de datos; si fuera mayor

Código	Nombre del Aeropuerto	Código	Nombre del Aeropuerto
ABE	Allentown, PA	AGN	Angoon, Alaska
ABI	Abilene, TX	AGS	Augusta, GA
ABL	Ambler, Alaska	AHN	Athens, GA
ABQ	Albuquerque, NM	AIA	Alliance, NE
ABR	Aberdeen, SD	AIN	Wainwright, Alaska
ABY	Albany, GA	AIY	Atlantic City, NJ
ACA	Acapulco, Méjico	AIZ	Lake of the Ozarks, MO
ACK	Nantucket, MA	AKI	Akiak, Alaska
ACT	Waco, TX	AKK	Akhiok, Alaska
ACV	Eureka/Arcata, CA	AKN	King Salmon, Alaska
ADK	Adak Is., Alaska	AKP	Anaktuvuk Pass, Alaska
ADQ	Kodiak, Alaska	ALB	Albany, NY
AET	Allakaket, Alaska		

Figura 4.1 Ejemplo de un elemento de datos discreto con muchos valores.

será más conveniente definir un almacenamiento de datos para contener los significados.

Por ejemplo, PARTE-NUMERO puede considerarse como un elemento de datos discreto. Cada valor tiene un significado particular:

7A4601B significa "Vástago, ranurado 7 1/4 pulgada"

etc. . . Pero en una empresa fabril con 40.000 partes, no podemos esperar que se coloquen estos valores y significados en un diccionario de datos. Aquí el factor crítico es que casi seguramente desearemos almacenar otros atributos de cada parte: su peso, precio, el proveedor, etc. La tabla de valores de un elemento de datos discreto debe servir simplemente para vincular el valor con el significado — y nada más.

Un interesante caso límite lo tenemos en la especificación de un número telefónico*. El código de área es un número de tres dígitos en que el primer dígito nunca es 0 ó 1 y el dígito del medio es siempre 0 ó 1 (por lo menos en aquellos códigos disponibles para el público). Esto da un total de 144 códigos de áreas posibles, 112 de las cuales están asignadas. Cada valor tiene un significado en el sentido que especifica un área:

212: New York City
201: Northern New Jersey
415: San Francisco Bay
etc.

Como analistas tenemos una elección: o bien que tratemos AREA-CODIGO (central y número) como un elemento de datos continuo con restricciones en su rango, o bien que consideremos como un ítem a cada valor que pueda tomar y le demos un significado. Los significados pueden utilizarse para validar el número telefónico por comparación del código de área con el Estado o utilizarse como base regional para el análisis de ventas. La decisión crítica es:

¿Nos interesan los significados?

Si nos interesan, y usaremos los significados en alguna forma, entonces deberemos definir el elemento de datos como discreto. Si no nos interesan, ahorraremos algunos problemas haciéndolo continuo.

* Este ejemplo se aplica a los EE.UU. (*N. del T.*)

Otro uso de los elementos de datos discretos es para la definición de *adjetivos*. Como hemos dicho, la temperatura es un elemento de datos continuo, con un rango apropiado para el propósito a considerar. Pero queremos describir a las temperaturas mayores a 30° como "ALTA", a las temperaturas entre 30° y 13° como "NORMAL" y a las temperaturas menores a 13° como "FRIA".

Debemos definir "ALTA", "NORMAL" y "FRIA" como valores de un elemento de datos discreto "TEMP-RANGO", con significados descriptos en función del elemento de datos continuo "TEMPERATURA".

4. *Longitud.* En algún punto, el analista deberá especificar la longitud (o posible rango de longitudes) de cada elemento de datos. Deberá especificar la longitud tal como se encuentra en el mundo real y no la longitud que deberá tomar, por ejemplo, codificada en binario o decimal empaquetado.

El analista podrá dejar de especificar longitudes en un primer paso de la creación de un diccionario de datos, pero debe estar en libertad para poder agregarlas en etapas posteriores.

En el caso de un importe de dinero, podría ser conveniente especificar en forma implícita la longitud estableciendo simplemente el importe máximo redondeado, donde se pueden o no mantener los centavos. Luego, "Máximo $ 1 millón, sin centavos" implica una longitud de seis dígitos (999.999) y "Máximo $ 100 millones con centavos" implica una longitud de diez dígitos (99.999.999,99).

5. *Codificación.* El diseñador y el programador necesitan un lugar para registrar en el diccionario de datos, la forma en que el elemento de datos será codificado físicamente en el sistema, por ejemplo, si un número deberá ser almacenado en disco como decimal empaquetado o si la transmisión por una línea de comunicación será ASCII o EBCDIC. Realmente, deberán ser capaces de representar diferentes codificaciones ya que el mismo elemento de datos puede transmitirse por una línea de comunicaciones con caracteres ASCII, ser procesado por varios programas en EBCDID y/o binario, y almacenarse en disco como decimal empaquetado. Estas decisiones físicas no deberán ser tomadas por el analista ya que no forman parte de la especificación funcional *lógica*. Sin embargo, en algunas circunstancias no habrá decisión alguna que tomar ya que el sistema propuesto tendrá que trabajar en interfaz con otro sistema (digamos que tiene que aceptar un flujo de entrada de mensajes télex). En este caso, no tenemos control sobre el formato físico, y el analista deberá consignar la codificación de este flujo en el diccionario de datos. En la mayoría de los casos, el analista no necesita ocuparse de este nivel de detalles y puede limitarse a especificar el tipo de dato externo (si es numérico, puramente alfabético o alfanumérico).

6. *Otras informaciones de edición.* En esta sección debemos incluir información que nos ayude a validar el elemento de datos, especialmente donde sea parte de una entrada al sistema. Ya hemos registrado rango, longitud y tipo de información, de modo que bajo este encabezamiento comúnmente anotaremos las referencias externas para otros elementos de datos o almacenamientos de datos. Debemos registrar el hecho que el usuario desea verificar AREA-CODIGO con ESTADO-CODIGO o CUENTA-NUMERO contra la lista de cuentas morosas o que las PARTES-NUMERO 7000-7999 solamente se suministren a comerciantes autorizados y no al público en general.

Una forma simple de registrar elementos de datos. Como vimos en la Sec. 4.3, es posible construir en forma manual un diccionario de datos utilizando un lote de tarjetas, o construir un diccionario de datos automático utilizando paquetes o software escrito por el usuario. No importa el método que se utilice, es conveniente tener un formato de formulario que sirva ya sea para el archivo manual o para la entrada de datos. La disposición de la Fig. 4.2 es conveniente para tarjetas índice de 12 x 7,5 cm.

4.2.2. Descripción de estructuras de datos

Como vimos en los "pedidos" en el Capítulo 2, las estructuras de datos se construyen en base a los elementos de datos y a otras estructuras de datos, de manera tal que en principio

podemos describir cualquier estructura de datos que deseemos mediante la especificación de los nombres de las estructuras y elementos que la componen, asegurándose que dichos componentes estén definidos en alguna otra parte del diccionario de datos.

Necesitamos saber más acerca de una estructura que la sola relación de sus componentes. Consideremos una situación en la cual una empresa envía un aviso de pago al abonar a sus proveedores. El aviso de pago lista las facturas cubiertas por este pago y da la cantidad total del pago. Algunas veces se incluye un cheque con el aviso de pago; otras el pago se hace directamente sobre una cuenta bancaria especificada previamente y solo se recibe el aviso de pago. Por cada factura se cita el número de factura, con la fecha original de emisión y el importe, y (si fuera necesario) alguna descripción narrativa. ¿Cómo podemos representar la estructura de AVISO-DE-PAGO? Listemos los componentes con notas:

E|S|T|A|D|O|-|P|R|O|V|I|N|C|I|A|-|C|O|D|I|G|O| | | | | | Elemento de datos

Breve descripción *Código de dos letras, para cada Estado/territorio de EE.UU. o provincia de Canadá.* Tipo (A) AN N ___

Alias (contextos) *C Estado (BAL) ESTADO-CODIGO (SISTEMA VENTAS) BREVE-ESTADO (estafeta)*

Si es discreto		Si es continuo
Valor	Significado	Rango de valores ___
AK	Alaska	
AL	Alabama	Valor típico ___
AR	Arkansas	Longitud *2 caracteres*
AS	American Samoa	Representación interna
AZ	Arizona	*Aún sin asignar*

(Si son más de 5 valores, continúe a la vuelta o indique referencia a hoja separada) *Ver arriba: total 63*

Otra información de edición *Puede requerirse para ajustar el código postal.*

Estructuras de datos/elementos relacionados *Dirección del cliente, dirección del proveedor.*

Figura 4.2 Ejemplar de un formulario para registrar elementos de datos.

Componente	Notas
AVISO-DE-PAGO	
FECHA	
NOMBRE DEL PROVEEDOR	
DIRECCION DEL PROVEEDOR	
CHEQUE-NUMERO, BANCO-DETALLES-DE-PAGO	Cualquiera de ellos, depende del método de pago
BANCO-NOMBRE DEL PROVEEDOR	
BANCO-CUENTA-NO-DEL-PROVEEDOR	
FACTURA	Una o más
NUMERO	
FECHA	
IMPORTE	
DESCRIPCION	Opcional
PAGO-TOTAL	

Vemos que algunos componentes de la estructura son obligatorios, algunos son alternativos, algunos son opcionales, y algunos son repetitivos (iterativos) una o más veces.

Necesitamos una convención para especificar estas características de una estructura de datos; una forma es adaptar la notación empleada en los manuales de lenguajes de programación para mostrar la estructura de las instrucciones del lenguaje, de la siguiente manera:

1. *Estructuras opcionales:* Una estructura de datos o elemento de datos, entre corchetes,

[DESCRIPCION]

significa que éste es un componente opcional de la estructura.

2. *Estructuras alternativas:* Dos o más nombres de estructuras de datos o de elementos de datos, dentro de llaves,

CHEQUE-NUMERO
BANCO-DETALLES-DE-PAGO

Significan que solo uno de los componentes estará presente en cada caso, en la estructura.

3. *Iteraciones de estructuras.* Los manuales de lenguajes de programación muestran iteraciones mediante la colocación de puntos suspensivos después del ítem. Esto no basta para nuestros propósitos, de modo que tomaremos la notación de estructura de datos de Jackson [4.3] marcando las estructuras iterativas con un asterisco,

FACTURA*

significa que puede no haber facturas, o puede haber una, o varias. Si conocemos el rango de posibilidades, podemos mostrar la estructura como

FACTURA* (0-10)

lo cual significa que podrá no haber facturas, o bien cualquier número de ellas, hasta un máximo de 10.

Aquí vemos cómo queda la estructura de aviso de pago:

```
AVISO-DE-PAGO
    FECHA
    NOMBRE DEL PROVEEDOR
    DIRECCION DEL PROVEEDOR
     { CHEQUE-NUMERO                }
     { BANCO-DETALLES-DE-PAGO       }
          BANCO-NOMBRE-DEL-PROVEEDOR
          BANCO-CUENTA-NO-DEL-PROVEEDOR
    FACTURA* (1-)
          NUMERO
          FECHA
          IMPORTE
          [DESCRIPCION]
    PAGO-TOTAL
```

Un formato simple para registrar estructuras de datos. Una vez que hemos definido los elementos de datos de interés, necesitamos registrar la composición de la estructura de datos empleando la notación ya descripta. Cuando la estructura de datos se relaciona con algo físico (por ejemplo, un formulario de factura o un informe de existencias de inventario), podemos referirnos a ella en una *breve descripción.* A menudo es conveniente dar un ejemplo medianamente complejo de la estructura de datos en el reverso de la tarjeta (si es manual) o remitirse a un lugar donde pueda encontrarse este ejemplo. Ver Fig. 4.3.

Es una cuestión discutible si es mejor almacenar información con volúmenes promedio o de pico para entradas y salidas, a nivel de estructura de datos o a nivel de flujo de datos (ver Sec. 4.2.3). Cuando una estructura de datos pasa a través de un número de procesos sin

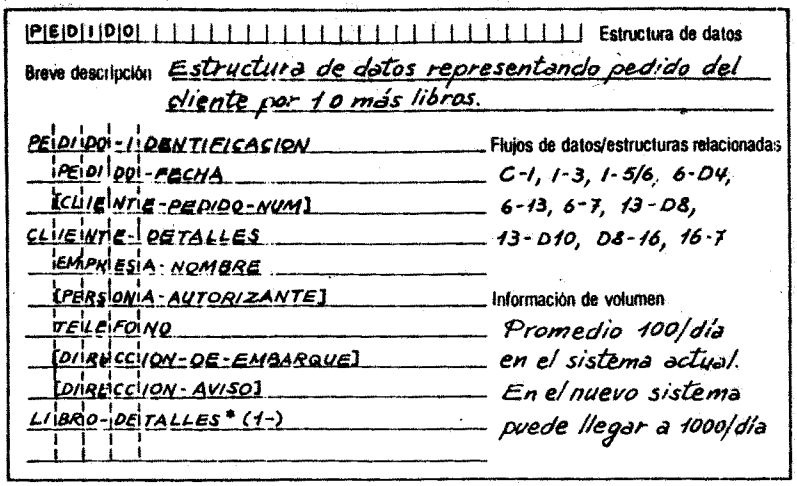


Figura 4.3 Ejemplar de un formulario para registrar una estructura de datos.

mayores cambios (como "pedidos" en el diagrama de flujo de datos del Capítulo 3), tiene sentido registrar los volúmenes a nivel de estructura de datos. Cuando la estructura de datos sólo toma parte en uno o dos flujos de datos, probablemente sea mejor registrar los volúmenes a nivel de flujo de datos.

4.2.3 Descripción de los flujos de datos

Ahora podemos expresar el *contenido* de un flujo de datos mediante la definición de los nombres de las estructuras de datos que circulan a través del mismo.(Para ser estrictos, los contenidos de un flujo de datos constituyen específicamente una gran estructura de datos, por ejemplo, PEDIDOS, o DEVOLUCIONES o PAGOS, pero ello significa gastar una entrada en el diccionario de datos para expresarlo de este modo). También queremos hacer notar

- La fuente del flujo de datos
- El destino
- El volumen de cada estructura de datos o transacción (quizás con su distribución a lo largo del día o del mes)
- La actual implementación física del flujo de datos (cuando estamos describiendo un sistema existente)

En la Fig. 4.4 se muestra un formulario simple.

Una breve acotación. Como vimos, muchos flujos de datos incluyen una sola estructura de datos. Si tomamos el flujo de datos de CBM, en el Capítulo 3, tendríamos ejemplos como:

21-p: Control de libros provistos
6-D4: Todos los pedidos
D12-20: Importes adeudados

Podemos ahorrar entradas omitiendo estas definiciones de flujos de datos del diccionario de datos y anotando simplemente las referencias de la fuente y destino en la entrada de la única estructura de datos involucrada. Así podemos registrar el flujo de datos 21-p en el diccionario de datos anotando en la entrada de la estructura de datos CHEQUE-AL-PROVEEDOR que es el único contenido de 21-p.

ITEMS-NO-EMBARCABLES — FLUJO DE DATOS

Fuente Ref: 6 Descripción: Verificar inventario existente

Destino ref: 13 Descripción: Crear pedido pendiente o requisición

Descripción expandida: Detalles de cada item por los cuales un pedido aceptable ha sido recibido, pero no puede despacharse porque está sin stock o porque no está incluido en el inventario.

Estructuras de datos incluidas:	Información de volumen
Pedido	Sin stock - aproximadamente 5 por semana (esto es aceptable para la Gerencia)
Pedido-identificación	
Cliente-detalles	
Libro-detalles*	Items no inventariados - aproximadamente 30 por semana
Causa de no embarco	
Cuando el pedido original es para múltiples libros, solo algunos pueden aparecer en el flujo de datos.	No crecen los datos

Figura 4.4 Ejemplar de un formulario para la registración de flujo de datos.

4.2.4 Descripción de los almacenamientos de datos

Dado que un almacenamiento de datos es una estructura de datos en reposo, describimos el contenido de cada almacenamiento de datos en función de las estructuras de datos que encontramos en él. También queremos registrar los flujos de datos que alimentan los almacenamientos de datos y aquellos que se extraen del mismo. Si solamente se harán unas pocas consultas definidas, ellas también podrán describirse en el diccionario de datos, pero si las consultas tienen cierta complejidad, deberán ser atendidas en la sección de acceso inmediato de la especificación funcional (descrita en el Capítulo 7). La entrada de un almacenamiento de datos se verá como lo muestra la Fig. 4.5.

Cuando se deba tomar una decisión acerca de la organización física y de la implementación del almacenamiento, pueden agregarse los detalles de la clave primaria, índices secundarios, ISAM, BDAM, residencia en dispositivos, etc. Podemos apreciar a través del diagrama de flujo de datos que existen relativamente pocos almacenamientos de datos en comparación con los elementos de datos y las estructuras de datos; no es agobiador encarar sus detalles.

4.2.5 Descripción de los procesos

Se indicó en el Capítulo 2 que la lógica de los procesos puede documentarse mediante diferentes herramientas tales como árboles de decisión, tablas de decisión y lenguaje estructurado. Un diccionario de datos, especialmente uno manual, no tiene muchas veces espacio suficiente para contener la descripción lógica completa. En estos casos debemos especificar las entradas y salidas del proceso, resumir la lógica e ingresar una referencia del lugar de la documentación de la especificación funcional donde se puede localizar dicha lógica.

Por supuesto, si podemos introducir de un modo conveniente la lógica completa, mucho mejor. Un formulario simple de una tarjeta índice de 12 x 7,5 cm, se muestra en la Fig. 4.6.

Se debe observar que la lógica se describe en esta tarjeta en tres niveles:

1. El nombre del proceso en el diagrama de flujo de datos "VERIFICAR-CREDITO-OK" (cuando la entrada del diagrama de flujo de datos excede los 30 caracteres, se debe abreviar adecuadamente).

P E D I D O - H I S T O R I A — Almacén de Datos ref: D4

Descripción: Todos los pedidos aceptados para su cumplimiento - últimos 6 meses.

Flujo de datos de entrada:
6 - D4 Todos los pedidos

Flujo de datos de salida (buscar argumentos)
D4 - 10 Detalles de pedidos (nombre del cliente, fecha del pedido)
D4 - 11 Detalle de ventas (ISBN, nombre del editor)
D4 - 9 Demanda anterior (ISBN)

Contenidos:
Pedido
Pedido - identificación
Cliente - detalles
Libro - detalles * (1 -)

Análisis de acceso inmediato se podrá encontrar en:
Especificación funcional, Sección 8.17

Organización física:
Aún sin especificar

Figura 4.5 Ejemplar de un formulario para la registración de almacenamiento de datos.

2. Una breve descripción del proceso, que deberá estar en "una línea", suficiente para permitir a una persona familiarizada con la empresa entender el significado del proceso.
3. El resumen lógico en la columna central, que deberá contener las funciones principales a un nivel que permita a una persona familiarizada con la empresa llevar a cabo la función (o programarla). El resumen lógico no es una presentación exhaustiva, aunque deberá escribirse sin ambigüedades en la medida en que el espacio lo permita.

En el Capítulo 5 se discutirán las relaciones entre estos tres niveles y la presentación exhaustiva y formal de la lógica en lenguaje estructurado.

V E R I F I C A R - C R E D I T O - O K — Proceso ref: 3

Descripción: Decidir adónde se embarcan los pedidos sin previo pago, o si debe requerirse al cliente pago previo.

Entradas	Resumen de lógica	Salidas
1-3 PEDIDOS	Recuperar historia de pago	3-C Pedido de pago previo [Recordatorio de balance]
D3-3 Historia de pago	Si el cliente es nuevo, enviar pedido de pago previo.	3-D3 Nuevo balance en orden
FECHA-APERTURA-CUENTA	Si es cliente corriente (promedio de dos pedidos mensuales)	3-6 Pedidos con crédito OK
FACTURA *	OK el pedido, a menos que el balance esté vencido con más de 2 meses.	
PAGO *	Para clientes anteriores que no sean corrientes, OK los pedidos, a menos que tengan cualquier balance vencido.	
BALANCE-EN-ORDEN		

Ref. física: Porte de la entrada del pedido en línea, OE 707

Detalles completos de eta lógica se pueden encontrar: Especificación funcional, Sección 7.2

Figura 4.6 Ejemplar de un formulario para la registración de procesos.

4.2.6 Descripción de las entidades externas

Como se ha podido ver en los diagramas de flujo de datos que hemos dibujado, por lo general hay relativamente pocas entidades externas y puede no justificarse incluirlas en un diccionario de datos. Sin embargo, si deseamos hacerlo, es de interés la siguiente información:

Nombre	como se describe en el diagrama de flujo de datos:
Flujos de datos asociados	referencias y descripciones

Cuando la entidad externa es una persona o un grupo de personas:

Números	por ej., cuántos clientes, tasa de crecimiento, si son de tipos diferentes
Identificación	por ej., nombres de la gerencia, supervisor de contaduría

Cuando la entidad externa es otro sistema de procesamiento de datos:

Lenguaje	por ej., Autocoder 1401
"Hardware"	por ej., 360/25 con emulador, Sistema/34
Fuente de información	donde se puede obtener más información sobre las interfaces (persona o documentación)

4.2.7 Descripción de las entradas al glosario

En una cantidad de aplicaciones, los usuarios poseen un vocabulario propio que puede desorientar a los analistas y programadores, a menos que éstos tengan experiencia sustancial de la empresa. ¿Cuando un banquero dice "flotante", usted sabe exactamente qué significa? ¿Cuando un funcionario municipal le habla de "heredables", usted titubea?

El diccionario de datos es un lugar conveniente para guardar estos términos de *glosario*. Una vez definidos pueden ingresarse utilizando los formularios de elementos de datos, con la salvedad de que solo se necesitan el nombre, una breve descripción y las secciones de alias, tal como se ve en la Fig. 4.7.

V|N,P| Glossary Item

Short description *El monto que hoy debería invertirse para producir un flujo de efectivo fijo.* Type A AN N

Aliases (contexts) *Valor Neto Presente*

IF Discrete		IF Continuous
Value	Meaning	Range of values
		Typical value
		Length
		Internal representation
(If more than 5 values, continue on reverse or give reference to separate sheet)		

Other editing information

Related data structures/elements

4.3 DICCIONARIOS DE DATOS MANUALES Y AUTOMATIZADOS

Gran parte del valor de un diccionario de datos proviene del hecho de que existe un almacenamiento de datos *central* para todos los analistas, diseñadores y programadores que están trabajando en un determinado proyecto, o en un área de aplicación dada o, como veremos en la Sec. 4.6, para toda la empresa. Si el diccionario es un almacenamiento central, deberá ser controlado por una persona o un grupo. A nivel de proyecto, dicha persona podrá llamarse administrador de datos o administrador de elementos de datos. (A nivel de la empresa el diccionario de datos es controlado por la función administración de base de datos). El administrador de datos es responsable de ejercer el control sobre las entradas y cambios del diccionario de datos facilitando el acceso de todos los participantes en el proyecto a las versiones actualizadas, y ayudando a evitar "la reinvención de la rueda" que es lo que ocurre cuando un equipo de analistas trabaja sin coordinación en las definiciones de los datos. En las organizaciones que poseen un bibliotecario como integrante del equipo de proyecto, las funciones de administración de los datos recaen naturalmente en el mismo.

Como se señaló en nuestros ejemplos, el propio diccionario de datos puede almacenarse en tarjetas de 12x7,5 cm que contienen las entradas o en una tarjeta grande que sirve para contener todos los tipos de entradas. Las tarjetas pueden agruparse alfabéticamente dentro del tipo de entrada (esto es, todos los nombres de las estructuras de datos por orden alfabético y luego todos los nombres de los elementos de datos por orden alfabético) o bien en un riguroso orden alfabético general, sin tener en cuenta el tipo de entrada. Pueden usarse tarjetas de diferentes colores para cada tipo de entrada. Es conveniente tener a la mano una máquina fotocopiadora, ¡ya que sacar una tarjeta y olvidarse de volverla a colocar en el lugar correcto será un crimen castigado con la pena de muerte! El administrador de los datos necesitará copiar regularmente todas las tarjetas y entregar a cada miembro del proyecto una copia de referencia actualizada para los usos que se discuten en la próxima sección.

Depende del ingenio del administrador y de la cooperación del equipo, que algunos sistemas bastante respetables con más de 100 elementos de datos puedan manejarse con un archivo de tarjetas. Veremos luego, sin embargo, que el diccionario de datos tiene varios usos que requieren ser leídos por la máquina. Un primer nivel de automatización puede lograrse ingresando los datos del archivo de tarjetas en un sistema de tiempo compartido con capacidad de edición, tal como TSO, y permitiendo a los miembros del proyecto leer e imprimir el archivo a voluntad (aunque solamente el administrador de los datos tiene autoridad para actualizarlo). Los sistemas de tiempo compartido simples tienen dificultad para manejar todas las relaciones de las diversas entradas (las cuales pueden ser algo complejas) y para obtener una máxima utilidad muchas intalaciones están siendo equipadas con paquetes de "software" de diccionarios de datos, los que están disponibles cada vez en mayor cantidad. Una lista de algunos paquetes automatizados de amplio uso y sus vendedores se encuentra al final del capítulo.

En general, los paquetes automatizados suministran una capacidad amplia de edición de entradas y construyen una base de datos con punteros e índices para permitir a los usuarios de los diccionarios rastrear las relaciones, normalmente con acceso en línea. Como ejemplo, en la Secc. 4.5, se discuten las facilidades ofrecidas por DATAMANAGER, uno de dichos paquetes.

4.4 QUE PODEMOS DESEAR EXTRAER DE UN DICCIONARIO DE DATOS

Una vez creados todos los "datos sobre los datos" como se describió en la sección anterior, podemos utilizarlos de muchas maneras en el análisis y posteriormente en el diseño y la programación. Podemos identificar siete tipos diferentes de salidas deseables como facilidades de los diccionarios de datos; como veremos, algunas de ellas solo son practicables con "software" (y la lista provee un buen criterio para la evaluación de los paquetes de

"software" de los diccionarios de datos). Sin embargo, es posible utilizar un diccionario de datos manual, especialmente en proyectos pequeños y medianos.

4.4.1 Listados ordenados de todas las entradas o de varias clases de entradas con un detalle total o parcial

Listado completo. El listado completo da todos los detalles de todas las entradas (flujos de datos, estructuras de datos, elementos de datos, almacenamiento de datos, procesos, alias, términos de glosario) listados en secuencia alfabética por nombre. Como es de imaginar, este listado puede llegar a ser bastante grande cuando la cantidad de elementos de datos crece por encima de 100. En consecuencia, es conveniente proveer un listado resumen.

Listado resumen. El listado resumen da todas las entradas en secuencia alfabética por nombre acompañado solo de una breve descripción. La Fig. 4.8 muestra un extracto de un listado resumen de un sistema de personal.

Si el diccionario está en un dispositivo de acceso al azar, será preferible permitir formular las consultas por nombre, o por tipo de entrada, en vez de imprimir listados extensos, debido a que al avanzar el análisis se definen y modifican rápidamente nuevos elementos y estructuras, de manera que cualquier listado impreso o en microficha se hace anticuado donde no se dispone de un acceso en línea. El administrador de los datos puede contrarrestar este problema de envejecimiento haciendo circular un listado actualizado a cada uno de los participantes del proyecto, digamos, cada lunes, con un listado de "adiciones" y "modificaciones" los miércoles por la tarde.

4.4.2 Informes compuestos

Cuando llegamos al análisis detallado de la lógica de un proceso o a la estructura de un almacenamiento de datos, necesitamos conocer todo lo que se ha definido acerca de la estructura de los datos de interés. Deseamos conocer no solo el contenido de una estructura de datos, sino también los detalles de cada elemento de datos componente. Por supuesto, podemos reunir esta información a partir del listado completo, pero es conveniente contar con una reseña que integre la información en forma práctica.

4.4.3 Capacidad de referencia cruzada

Luego de realizado el trabajo inicial de definición del flujo de datos del sistema, y creada la mayoría de las entradas en el diccionario de datos, tiene lugar, durante el análisis y el diseño, la ejecución de varias etapas de revisión y perfeccionamiento. A menudo se descubre, por ejemplo, que en la etapa inicial se omitieron algunos elementos de datos que necesitan ser capturados, o en el diseño se decide aumentar la longitud de un elemento de datos para permitir valores mayores en ciertas circunstancias. Se necesita pues *modificar* alguna entrada en el diccionario. Todo cambio en cualquier entrada, por supuesto, puede ejercer un efecto sobre una cantidad de otras entradas. Si se agrega "FAMILIAR-PROXIMO" a un registro de empleado se deberán modificar las estructuras de datos y procesos que han creado este registro, se debe tener cuidado con los cambios a introducir, y se deben revisar todas las salidas relacionadas con los datos del empleado para ver si se debe incluir en las mismas "FAMILIAR-PROXIMO". De manera similar, si se ha llegado a la etapa de definición de programas y de archivos y se decide modificar la longitud del elemento de datos "IMPORTE-DEL-PEDIDO" de cinco a siete caracteres se necesitará conocer todos los programas y archivos que contengan o utilicen "IMPORTE-DEL-PEDIDO".

Esto se conoce como el problema *dónde se usa;* para cualquier elemento o estructura, es

necesario buscar en todo el diccionario y encontrar todas aquellas entradas que lo utilizan. Con un diccionario de datos manual esto representa obviamente una cantidad considerable de trabajo, acompañado de un cuidadoso estudio del diagrama de flujo de datos. Un sistema automatizado puede producir informes "dónde se usa" ya sea mediante clasificación o manteniendo un índice en el cual se muestra dónde se usa cada elemento, dónde se menciona cada estructura de datos, etc. . .

4.4.4 Encontrar el nombre a partir de una descripción

Supongamos que sabemos que cada cliente ha calculado una rotación promedio de tres meses para sus compras. No tenemos seguridad si este elemento de datos ha sido definido en el diccionario de datos, y en caso afirmativo, no estamos seguros de su nombre. ¿Cómo podemos extraer esta información? Necesitamos tener un "supercompetente" administrador de los datos, que conozca cada elemento de datos como a un amigo, o bien tener la capacidad de buscarlo empleando palabras claves, o encadenamientos a partir de palabras claves parciales. Por ejemplo, si pudiéramos representar todos los elementos de datos que tuvieran las cadenas de caracteres CLIENT o COMPR o AV en cualquier lugar de sus nombres, hallaríamos probablemente el elemento de datos buscado.

REMOCION-REGISTRO Detalles de reubicación pagada por Compañía	Estructura de datos
SALARIO Salario anual de empleados a jornal y a sueldo	Elemento de datos
SALARIO-CAMBIO Entrada desde la gerencia para modificar salario	Flujo de datos
MOSTRAR-ESTADO Representar el estado actual para un empleado determinado	Proceso
NUM-SEG-SOC Número de Seguridad Social	Elemento de datos
SSNO Alias de NUM-SEG-SOC	Elemento de datos
COMPARACION STANDARD Relación entre el salario del empleado y el salario de todos aquellos de la misma categoría, expresada como una desviación standard	Glosario de entrada
ESTADO-CODIGO-POSTAL Combinación de ESTADO-PROVINCIA-PAIS-CODIGO y CODIGO-POSTAL. Cubre todos los países	Estructura de datos
ESTADO Estructura de datos Detalles de todas las informaciones actuales acerca de un empleado, fuera del salario.	Flujo de datos
ESTADO-HISTORIA Contiene datos y cambios del ESTADO y SALARIO del empleado durante toda su carrera en la compañía	Almacenamiento de datos

Figura 4.8 Extracto del listado resumen.

4.4.5 Control de consistencia e integridad

Una vez que hayamos hecho todo lo posible para completar el diccionario de datos partiendo del diagrama de flujo de datos, sería muy conveniente utilizar algún "software" para contestar las siguientes preguntas:

1. ¿Existe algún flujo de datos especificado sin una fuente o sin un destino?
2. ¿Existe algún elemento de datos especificado en algún almacenamiento de datos, que no tenga forma de ingresar al mismo, debido a que no existe en los flujos de datos entrantes?
3 ¿Hay alguna parte del proceso lógico en el cual se use un elemento de datos que no exista en ninguna de las entradas a dicho proceso?
4. ¿Existe algún elemento de datos en cualquier flujo de datos entrante en un proceso que no se utilice en el mismo y/o no aparezca en la salida?

Especialmente cuando el sistema crece, se hace más dificultoso controlar manualmente el pasaje de cada elemento de datos a través de cada flujo y cada proceso. Como consecuencia, los elementos faltantes no se descubrirán hasta llegar a la etapa de programación. La pregunta 4 anterior, para elementos de datos redundantes puede que *nunca* sea hecha y ¡posiblemente la redundancia no se conozca a través de toda la vida del sistema!

4.4.6 Generación de definiciones de datos legibles por máquina

Cuando se completa el diseño físico del sistema, los programadores querrán utilizar las definiciones de los datos del diccionario de datos para crear sentencias de definición para sus programas: 01 para COBOL, DSECT para lenguaje assembler, DBD para IMS DL/I, etc. Antes de recorrer el proceso larguísimo y propenso a errores, de copiar definiciones de un listado del diccionario de datos para luego proceder a su perforación, lo sensato es tener un "software" que transforme las definiciones de datos del diccionario al formato requerido por el lenguaje de aplicación en cada caso y los almacene en una biblioteca o los perfore en tarjetas.

Así, si tenemos las siguientes entradas en el diccionario de datos,

1. Estructura de datos:	CLIENTE-DIRECCION	
Componentes:	ORGANIZACION CALLE CIUDAD ESTADO-CODIGO-POSTAL [TELEFONO]	
2. Estructura de datos:	TELEFONO	
Componentes:	AREA-CODIGO CENTRAL NUMERO [EXTENSION]	
3. Estructura de datos:	ESTADO-CODIGO-POSTAL	
Componentes:	ESTADO-CODIGO-POSTAL CODIGO-POSTAL	
4. Elemento de datos:	AREA-CODIGO	Longitud: 3 Numérico
5. Elemento de datos:	CIUDAD	Longitud: 15 Alfabético
6. Elementos de datos:	CENTRAL	Longitud: 3. Alfanumérico
7. Elemento de datos:	EXTENSION	Longitud: 4 Numérico
8. Elemento de datos:	NUMERO	Longitud: 4 Numérico
9. Elemento de datos:	ORGANIZACION	Longitud: 25 Alfanumérico

10. Elemento de datos:	ESTADO-CODIGO	Longitud: 2 Alfabético
11. Elemento de datos:	CALLE	Longitud: 30 Alfanumérico
12. Elemento de datos:	CODIGO-POSTAL	Longitud: 5 Numérico

quisiéramos emitir el comando "Crear entrada 01 de la División de Datos COBOL llamada CLIENTDIRECT empleando la estructura de datos CLIENTE-DIRECCION, y perforarla en tarjetas" y lograr que el "software", en base a las premisas adecuadas, genere las siguientes sentencias:

```
01  CLIENTDIREC
    05  ORGANIZACION              PICTURE X (25)
    05  CALLE                     PICTURE X (30)
    05  CIUDAD                    PICTURE X (15).
    05  ESTADO-CODIGO-POSTAL
        10  ESTADO-CODIGO         PICTURE X (2).
        10  CODIGO-POSTAL         PICTURE 9 (5).
    05  TELEFONO
        10  AREA-CODIGO           PICTURE 9 (3).
        10  CENTRAL               PICTURE X (3).
        10  NUMERO                PICTURE 9 (4).
        10  EXTENSION             PICTURE 9 (4)
```

4.4.7 Extracción de las entradas del diccionario de datos desde programas existentes

Este proceso es justamente el inverso al que hemos descrito. Si estamos abocados al mantenimiento de un sistema o tenemos que desarrollar un sistema que se comunique con una cierta cantidad de sistemas existentes, querríamos tener la capacidad necesaria para alimentar a todos los programas de interés existentes dentro de nuestro "software" y generar entradas del diccionario de datos, cualquiera sea nuestro formato standard. Esto nos proporcionará un almacenamiento central de información del sistema que tenemos que mantener, o con el que debemos estar en comunicación, almacenamiento que podremos explorar utilizando las otras facilidades de búsqueda del diccionario de datos. Por ejemplo, podríamos hallar rápidamente que uno de los programas existentes contiene la dirección de la calle de un cliente en un campo denominado CLIENT-CAL, que tiene una longitud de 22 caracteres, mientras que otro contiene dicha dirección en un campo llamado CAL-LIN, que tiene solo 20 caracteres. Estos *sinónimos* (nombres distintos para una misma cosa) y definiciones incompatibles deberán ser resueltas previamente para que podamos construir el nuevo sistema en forma confiable.

4.5 EJEMPLO DE UN DICCIONARIO DE DATOS AUTOMATIZADO

Algunos paquetes de "software" de diccionarios de datos automatizados se indican en el apéndice de este capítulo. Como ejemplo de las facilidades ofrecidas por estos paquetes, veremos el DATAMANAGER*

* Los gráficos y ejemplos de las salidas del DATAMANAGER de esta sección están reproducidos con autorización de MSP Inc. *(N. del E.)*

DATAMANAGER es un paquete independiente; esto es, no requiere el uso concurrente de ningún sistema administrador de bases de datos (a diferencia, por ejemplo, del UCC-TEN, que requiere IMS). El "software" consiste en un solo programa, escrito en lenguaje ensamblador, que corre en computadores IBM 360/370 bajo cualquier versión de OS o de DOS. El tamaño del programa varía entre 50K y 80K bytes, dependiente de los recursos opcionales incluidos. Cada entrada (denominada *miembro*) puede tener un nombre de hasta 32 caracteres de largo, con hasta 16 alias.

Las palabras que utiliza DATAMANAGER para describir procesos y datos son algo físicas en comparación con la terminología que hemos utilizado. Los procesos se describen en tres niveles. Un SISTEMA puede definirse como constituido por uno o más SISTEMAS, cada uno de los cuales está compuesto por PROGRAMAS y cada uno de estos últimos a su vez está compuesto por MODULOS. Los datos pueden describirse como una BASE DE DATOS, que puede contener ARCHIVOS, que a su vez pueden contener GRUPOS. Los GRUPOS pueden estar compuestos ellos mismos por una cantidad de GRUPOS, siendo cada GRUPO compuesto por ITEM. Estas relaciones se indican en la Fig. 4.9.

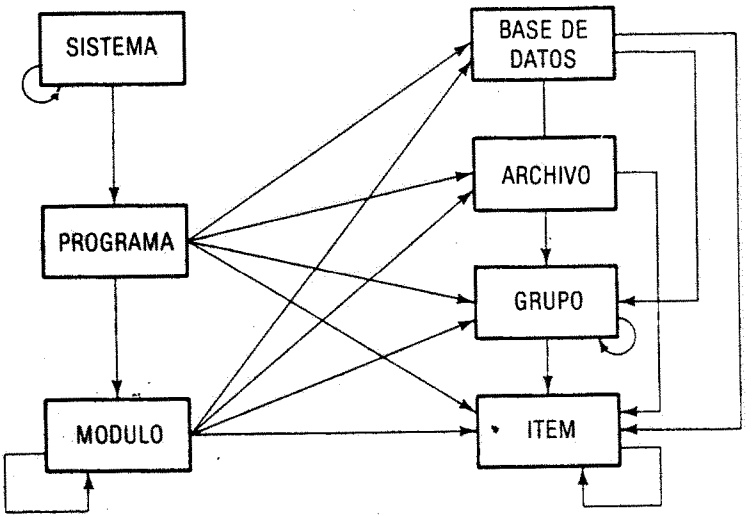


Figura 4.9 Relaciones del DATAMANAGER.

Todos los procesos han sido concebidos con archivos o bases de datos en sus entradas y salidas. Por lo tanto DATAMANAGER requiere que los flujos de datos sean descriptos como archivos o grupos, como se muestra en la Fig. 4.10.

DATAMANAGER provee un conjunto de comandos parecidos al inglés, para almacenar y recuperar las entradas. Supongamos que queremos comenzar definiendo un sistema que hemos decidido llamar "MANTENER-DATOS-EMPLEADO"; este podría corresponder a una casilla de proceso de un diagrama de flujo de datos global de un sistema de personal complejo. (En realidad MANTENER-DATOS-EMPLEADO es un subsistema, pero en DATAMANAGER, como dijimos, los SISTEMAS pueden contener SISTEMAS.) Damos en DATAMANAGER el comando "ADD-MANTENER-DATOS-EMPLEADO" ya sea a través de una terminal o de una tarjeta y cuando el comando es aceptado, especificamos las características de la entrada.

La Fig. 4.11 muestra el inicio de este proceso. Lo que ingresamos se encuentra recuadrado; las respuestas de DATAMANAGER se observan en las líneas que comienzan con el prefijo DM. En la línea 100 indicamos a DATAMANAGER que la entrada es un

SISTEMA; en las líneas 200-400 indicamos sus procesos componentes e ingresamos el nombre corto del sistema "MEDS" como un alias en la línea 500. La facilidad de CATALOG nos permite introducir las palabras claves que queremos utilizar posteriormente; en este caso prevemos la necesidad de recuperar todos los sistemas que serán corridos mensualmente o todos aquellos que están escritos en COBOL.

Descripción de análisis estructurado	Descripcion DATAMANAGER
Elemento de datos	ITEM
Estructura de datos	GRUPO
Flujo de datos	ARCHIVO/GRUPO
Almacenamiento de datos	ARCHIVO/BASE DE DATOS
Proceso	SISTEMA/PROGRAMA/MODULO

Figura 4.10 Descripciones en análisis estructurado y DATAMANAGER.

Si se ingresa nuestra corta descripción como una NOTA, una vez completada la entrada, DATAMANAGER controlará las sintaxis de las sentencias y creará miembros simulados para estos nuevos procesos que hemos enumerado pero que aún no hemos descrito. A continuación completamos los detalles de, digamos, EMPLEADO-VET y definimos los archivos empleados con sus grupos componentes e ítem de datos.

Las entradas se pueden efectuar en cualquier orden, de modo que se pueden definir los elementos de datos como ITEM antes de que tengamos la seguridad de cuáles serán los ARCHIVOS o estructuras de datos que los contendrán.

Cuando se haya descrito completamente el sistema podríamos desear el tipo de listado alfabético que contiene todas las entradas, que hemos descrito en la Sec. 4.4. La Fig. 4.12 muestra el tipo de salidas que podemos obtener del DATAMANAGER, listado de nombres, el tipo de miembro (ITEM, GRUPO, etc.), cuántos otros miembros hacen referencia a cada miembro y el estado (tanto si es un registro fuente como un registro codificado).

```
        00005    AGREGAR MANTENER-DATOS EMPLEADO
DM011311         MANTENER-DATOS-EMPLEADO BIEN INTERCALADO
DM012961         CODIFICACION DE MANTENER-DATOS-EMPLEADO
        00100    SISTEMA CONTIENE
        00200    EMPLEADO-VET
        00300    ARCHIVO-MAESTRO-ACTUALIZADO
        00400    INFORME-EMPLEADO
        00500    ALIAS 'MEDS'
        00600    CATALOGO "MENSUAL" "COBOL"
        00700    NOTA: "ESTA APLICACION MANTIENE EL REGISTRO MAESTRO DE
        00800    EMPLEADOS Y EMITE INFORMES DEL MISMO"
DM012801         MANTENER-DATOS-EMPLEADO BIEN CODIFICADO

        00015    SKIP.
```

Figura 4.11 Agregado de un sistema al diccionario de datos.

Si fuera más conveniente podemos listar los miembros de cada tipo (SISTEMAS, PROGRAMAS, ARCHIVOS, etc.). Por ejemplo, si deseamos examinar los elementos de datos, podemos emitir el comando "LISTAR ITEM" y la salida se verá como en la Fig. 4.13.

DATAMANAGER atiende referencias cruzadas mediante la colocación de punteros para cada relación posible. Por ejemplo, si el grupo TRANSACCION-REGISTRO (que se muestra en la Fig. 4.12) contiene DIRECCION-ACTUALIZADA, que contiene DIRECCION, sabemos que hay dos relaciones directas, que se verán como en la Fig. 4.14. Pero estas dos relaciones directas implican otras cuatro relaciones *indirectas,* haciendo un total de seis entre los tres miembros, como se indica en líneas punteadas en la Fig. 4.15.

LIST OF MEMBERS (Lista de Miembros) MEMBER NAME (Nombre de Miembro)	TYPE (Tipo)	USAGE (Uso)	CONDITION (Condición)	AC	ALT	REM	OWNER (Dueño)
ACTION-CODE (Código de Acción)	ITEM	5	SCE ENC	0	0	0	
ADDRESS (Dirección)	ITEM	2	SCE ENC	0	0	0	
ADDRESS-UPDATE (Dirección-Actualizada)	(Grupo)	1	SCE ENC	0	0	0	
BASIC-UPDATE (Actualización)	(Grupo)	1	SCE ENC	0	0	0	
DEDUCT-CODE (Salida - Código)	ITEM	2	SCE ENC	0	0	0	
DELETE (Eliminar)	(Grupo)	1	SCE ENC	0	0	0	
DEPARTMENT (Departamento)	ITEM	5	SCE ENC	0	0	0	
EMPLOYEE-HISTORY-LIST (Empleado-Historia-Lista)	(Archivo)	1	SCE ENC	0	0	0	
EMPLOYEE-HISTORY-MASTER (Empleado-Historia-Maestro)	(Archivo)	3	SCE ENC	0	0	0	
EMPLOYEE-HISTORY-REPORT (Empleado-Historia-Informe)	(Programa)	1	SCE ENC	0	0	0	
EMPLOYEE-HISTORY-UPDATE (Empleado-Historia-Actualiz.)	(Programa)	1	SCE ENC	0	0	0	
EMPLOYEE-LIST (Empleado-Lista)	(Archivo)	1	SCE ENC	0	0	0	
EMPLOYEE-MASTER (Empleado-Maestro)	(Archivo)	4	SCE ENC	0	0	0	
EMPLOYEE-MASTER-UPDATE (Empleado-Maestro-Actualiz.)	(Programa)	1	SCE ENC	0	0	0	
EMPLOYEE-NUMBER (Empleado-Número)	ITEM	5	SCE ENC	0	0	0	
EMPLOYEE-RECORD (Empleado-Registro)	(Grupo)	1	SCE ENC	0	0	0	
EMPLOYEE-REPORT (Empleado-Informe)	(Programa)	1	SCE ENC	0	0	0	
EMPLOYEE-TRANSACTIONS (Empleado-Transacciones)	(Archivo)	1	SCE ENC	0	0	0	
EMPLOYEE-TRANSACTIONS-SORTED (Empleado-Transac.-Clasificado)	(Archivo)	3	SCE ENC	0	0	0	
EMPLOYEE-VET (Empleado-Vet)	(Programa)	1	SCE ENC	0	0	0	
FILLER00002	ITEM	13	SCE ENC	0	0	0	
HISTORY-RECORD (Historia-Registro)	(Grupo)	1	SCE ENC	0	0	0	
HISTORY-REPORT-RECORD (Historia-Informe-Registro)	(Grupo)	1	SCE ENC	0	0	0	
JOB-COUNT (Trabajo-Conteo)	ITEM	2	SCE ENC	0	0	0	
JOB-ENTRY (Trabajo-Entrada)	(Grupo)	1	SCE ENC	0	0	0	
JOB-STATUS (Trabajo-Estado)	ITEM	5	SCE ENC	0	0	0	
JOB-TITLE (Trabajo-Título)	ITEM	5	SCE ENC	0	0	0	
MAINTAIN-EMPLOYEE-DATA (Mantener-Empleado-Datos)	(Sistema)	0	SCE ENC	0	0	0	
MAINTAIN-EMPLOYEE-HISTORY (Mantener-Empleado-Historia)	(Sistema)	0	SCE ENC	0	0	0	
NAME (Nombre)	ITEM	4	SCE ENC	0	0	0	
REPORT-COUNT (Informe-Conteo)	(Grupo)	1	SCE ENC	0	0	0	
REPORT-RECORD (Informe-Registro)	(Grupo)	1	SCE ENC	0	0	0	
SALARY (Salario)	ITEM	5	SCE ENC	0	0	0	
SOCIAL-SECURITY-NUMBER (Social-Seguridad-Número)	ITEM	3	SCE ENC	0	0	0	
TAXCODE (Código Impositivo)	ITEM	2	SCE ENC	0	0	0	
TRANSACTION-RECORD (Transacción-Registro)	(Grupo)	2	SCE ENC	0	0	0	
TYPE (Tipo)	ITEM	6	SCE ENC	0	0	0	

LIST CONTAINS (Lista contiene)

14 ITEMS
10 GROUPS (Grupos)
6 FILES (Archivo)
5 PROGRAMS (Programas)
2 SYSTEMS (Sistemas)
37 MEMBERS IN TOTAL (Miembros en total)

Figura 4.12 Salida DATAMANAGER.

LIST OF ITEMS (lista de item) MEMBER NAME (Nombre de Miembro)	TYPE (Tipo)	USAGE (Uso)	CONDITION (Condición)	AC	ALT	REM	OWNER (Dueño)
ACTION-CODE (Código Acción)	ITEM	5	SCE ENC	0	0	0	
ADDRESS (Dirección)	ITEM	2	SCE ENC	0	0	0	
DEDUCT-CODE (Deduc.-Código)	ITEM	2	SCE ENC	0	0	0	
DEPARTMENT (Departamento)	ITEM	5	SCE ENC	0	0	0	
EMPLOYEE-NUMBER (Empleado-Número)	ITEM	5	SCE ENC	0	0	0	
FILLER00002	ITEM	13	SCE ENC	0	0	0	
JOB-COUNT (Trabajo-Conteo)	ITEM	2	SCE ENC	0	0	0	
JOB-STATUS (Trabajo-Estado)	ITEM	5	SCE ENC	0	0	0	
JOB-TITLE (Trabajo-Título)	ITEM	5	SCE ENC	0	0	0	
NAME (Nombre)	ITEM	4	SCE ENC	0	0	0	
SALARY (Salario)	ITEM	5	SCE ENC	0	0	0	
SOCIAL-SECURITY-NUMBER (Seguridad Social-Número)	ITEM	3	SCE ENC	0	0	0	
TAXCODE (Código Impositivo)	ITEM	2	SCE ENC	0	0	0	
TYPE (Tipo)	ITEM	6	SCE ENC	0	0	0	

LIST CONTAINS (Lista contiene) 14 ITEM

Figura 4.13 Salida LIST ITEM (lista de Item).

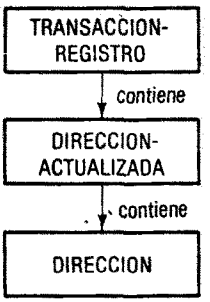


Figura 4.14 Dos relaciones directas (explícitas).

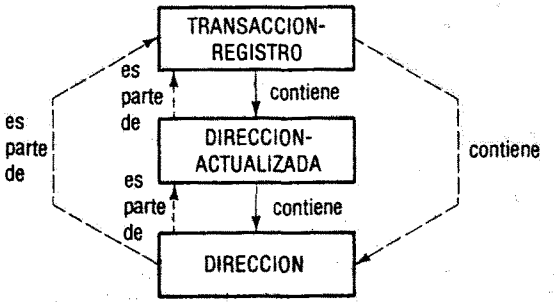


Figura 4.15 Dos relaciones directas implican otras cuatro relaciones indirectas (implícitas).

DATAMANAGER produce y mantiene referencias cruzadas bidireccionales completas, basadas en las relaciones directas, introducidas por el usuario. Esto hace relativamente simple obtener información donde se usa. Por ejemplo, supongamos que tenemos interés en el ítem DEPARTAMENTO y queremos rastrear a través del sistema para ver en qué flujos de datos, procesos, y almacenamientos de datos está incluido. Podemos utilizar el comando "WHAT USES"... (QUE USA)... y obtener el rastro de todas las relaciones, tal como se muestra en la Fig. 4.16. (Por razones de brevedad se han omitido las relaciones de PROGRAMA y SISTEMA).

El comando "WHICH... USE" (CUALES... USAN) permite preguntar más específicamente

```
00018        QUE ARCHIVOS USAN DEPARTAMENTO

LOS SIGUIENTES USAN ITEM DEPARTAMENTO
     ARCHIVOS                          MAESTRO-EMPLEADO
                                       MAESTRO-HISTORIA-EMPLEADO
                                       LISTA-HISTORIA-EMPLEADO
                                       LISTA-EMPLEADO
                                       TRANSACCIONES-EMPLEADO
                                       TRANSACCIONES-CLASIFICADAS
                                       -EMPLEADO

00019        QUE PROGRAMAS USAN DEPARTAMENTO

LOS SIGUIENTES USAN ITEM DEPARTAMENTO
     PROGRAMAS                         INFORME-HISTORIA-EMPLEADO
                                       ACTUALIZAR-MAESTRO-EMPLEADO
                                       INFORME-EMPLEADO
                                       ACTUALIZAR-HISTORIA-EMPLEADO
                                       EMPLEADO-VET
```

Otros comandos DATAMANAGER permiten explorar a requerimiento referencias cruzadas entre miembros y palabras clave.

Al mismo tiempo que escribe, DATAMANAGER puede leer definiciones de datos desde programas existentes en COBOL y PL/1. Puede generar automáticamente definiciones de datos para COBOL, PL/1 y "assembler" IBM 370 y ponerlos en bibliotecas indicadas previamente, listos para ser usados por los programadores. Puede generar definiciones de datos para TOTAL, ADABAS e IMS.

Se pueden mantener y recuperar versiones múltiples de entradas, de manera que es posible llevar a cabo pruebas con una versión de una estructura de datos y retener mientras tanto la versión previa para trabajos de producción.

Existen varios niveles de protección de seguridad. Se provee un sistema de palabras clave, y un determinado usuario puede estar limitado a tener acceso a ciertas entradas y a otras no (por ejemplo, SALARIO no es accesible a nadie que no sea el gerente de personal) o estar limitado a ciertos comandos (por ejemplo, MODIFY (MODIFICAR) solo puede ser utilizado por el administrador de los datos.)

Es posible tener disponibles el DATAMANAGER y el diccionario de datos para otros programas en línea en "modo activo". De esta manera, ya que el rango admisible de un elemento de datos está contenido en el diccionario de datos, un programa de validación puede recuperar ese rango admisible durante la validación de una transacción. Si deseamos cambiar el rango admisible, modificaremos el valor contenido en DATAMANAGER y *no hace falta* re-compilar el programa (o programas) de validación.

DATAMANAGER nos permite realizar una limitada cantidad de controles de

```
00021        WHAT USES DEPARTMENT (QUE USA DEPARTAMENTO)

ITEM                        DEPARTMENT IS USED BY (Departamento es usado por)
  GROUP (Grupo)               EMPLOYEE-RECORD (Empleado-Registro)
  GROUP (Grupo)               HISTORY-RECORD (Historia-Registro)
  GROUP (Grupo)               HISTORY-REPORT-RECORD (Historia-Informe-Registro)
  GROUP (Grupo)               REPORT-RECORD (Informe-Registro)
  GROUP (Grupo)               TRANSACTION-RECORD (Transacción-Registro)

GROUP (Grupo)               EMPLOYEE-RECORD IS USED BY (Empleado-Registro es usado por)
  FILE (Archivo)              EMPLOYEE-MASTER (Empleado-Maestro)

GROUP (Grupo)               HISTORY-RECORD IS USED BY (Historia-Registro es usado por)
  FILE (Archivo)              EMPLOYEE-HISTORY-MASTER (Empleado-Historia-Maestro)

GROUP (Grupo)               HISTORY-REPORT-RECORD IS USED BY (Historia-Informe-Registro es usado por)
  FILE (Archivo)              EMPLOYEE-HISTORY-LIST (Empleado-Historia-Lista)

GROUP (Grupo)               REPORT-RECORD IS USED BY (Informe-Registro es usado por)
  FILE (Archivo)              EMPLOYEE-LIST (Empleado-Lista)

GROUP (Grupo)               TRANSACTION-RECORD IS USED BY (Transacción-Registro es usado por)
  FILE (Archivo)              EMPLOYEE-TRANSACTIONS (Empleado-Transacciones)
  FILE (Archivo)              EMPLOYEE-TRANSACTIONS-SORTED (Empleado-Transacciones-Clasificado)

FILE (Archivo)              EMPLOYEE-MASTER IS USED BY (Empleado-Maestro es usado por)
  PROGRAM (Programa)          EMPLOYEE-HISTORY-REPORT (Empleado-Historia-Informe)
  PROGRAM (Programa)          EMPLOYEE-MASTER-UPDATE (Empleado-Maestro-Actualiz.)
  PROGRAM (Programa)          EMPLOYEE-MASTER-UPDATE (Empleado-Maestro-Actualiz.)
  PROGRAM (programa)          EMPLOYEE-REPORT (Empleado-Informe)

FILE (Archivo)              EMPLOYEE-HISTORY-MASTER IS USED BY (Empleado-Historia-Maestro es usado por)
  PROGRAM (Programa)          EMPLOYEE-HISTORY-REPORT (Empleado-Historia-Informe)
  PROGRAMA (Programa)         EMPLOYEE-HISTORY-UPDATE (Empleado-Historia-Actualiz.)
  PROGRAM (Programa)          EMPLOYEE-HISTORY-UPDATE (Empleado-Historia-Actualiz.)

FILE (Archivo)              EMPLOYEE-HISTORY-LIST IS USED BY (Empleado-Historia-Lista es usado por)
  PROGRAM (Programa)          EMPLOYEE-HISTORY-REPORT (Empleado-Historia-Informe)

FILE (Archivo)              EMPLOYEE-LIST IS USED BY (Empleado-Lista es usado por)
  PROGRAM (Programa)          EMPLOYEE-REPORT (Empleado-Informe)

FILE (Archivo)              EMPLOYEE-TRANSACTIONS IS USED BY (Empleado-Transacciones usadas por)
  PROGRAM (Programa)          EMPLOYEE-VET (Empleado-Vet)

FILE (Archivo)              EMPLOYEE-TRANSACTIONS-SORTED IS USED BY (Empleado-Transac.-Clasificacion es usado por)
  PROGRAM (Programa)          EMPLOYEE-HISTORY-UPDATE (Empleado-Historia-Actualiz.)
  PROGRAM (Programa)          EMPLOYEE-MASTER-UPDATE (Empleado-Maestro-Actualiz.)
  PROGRAM (Programa)          EMPLOYEE-VET (Empleado-Vet)

PROGRAM (Programa)          EMPLOYEE-HISTORY-REPORT IS USED BY (Empleado-Historia-Informe es usado por)
  SYSTEM (Sistema)            MAINTAIN-EMPLOYEE-HISTORY (Mantener-Empleado-Historia)

PROGRAM (Programa)          EMPLOYEE-MASTER-UPDATE IS USED BY (Empleado-Maestro-Actualiz. es usado por)
  SYSTEM (Sistema)            MAINTAIN-EMPLOYEE-DATA (Mantener-Empleado-Datos)
```

Figura 4.16 Salida WHAT USES (QUE USA).

consistencia. Puesto que los flujos de datos deben ser representados como archivos, no hay forma de asegurarnos de que cada flujo de datos tenga tanto una fuente como un destino; debemos suministrarlos nosotros mismos. Si bien es posible mantener la lógica de los procesos como NOTE (NOTA) en la entrada del programa o del modulo, no es factible manipular esta lógica de ninguna forma ni comparar los elementos de datos en los flujos de entrada de datos.

Las entidades externas, aunque no estén representadas como un tipo de miembros separados, pueden ser introducidas como SISTEMAS (o como ARCHIVOS o GRUPOS).

Esta descripción no agota las facilidades de DATAMANAGER, el cual, como otros paquetes de diccionarios de datos, es un producto en evolución. Sin embargo, creemos haber dado una buena idea de la potencia, flexibilidad y utilidad de los paquetes automatizados como apoyo para el análisis de sistemas y posteriormente para su diseño y programación.

4.6 PROYECTOS CRUZADOS O DICCIONARIOS DE DATOS GLOBALES PARA LA EMPRESA

Muchas discusiones de este capítulo asumen tácitamente que cada proyecto está confeccionando un diccionario de datos para su propio uso. Una vez que tenemos la posibilidad de un diccionario de datos automatizado, resulta atractivo considerar que puede contener *todos* los elementos de datos, estructuras de datos y otras entradas para las aplicaciones de un área completa (digamos entrada de pedidos y control de inventarios) o de toda la empresa.

La construcción de un diccionario de datos para toda la empresa involucra un compromiso mayor, puesto que los cientos de archivos y aplicaciones que existen en una organización han sido creados por separado, existirán muchos alias, definiciones incompatibles de un mismo elemento, códigos en conflicto, ítem redundantes y casos donde el mismo nombre se utiliza para dos cosas diferentes. La creación de un diccionario de datos integrado exige un gran volumen de trabajo para resolver las incompatibilidades entre los datos: aunque el "software" de los diccionarios de datos puede manejar alias, la existencia de definiciones incompatibles y duplicaciones significa que los programas deberán ser modificados antes de compartir los datos. Las personas necesitan modificar los hábitos que han practicado por años; supongamos que el Departamento de Ventas ha dividido a los EE.UU. en regiones a los fines de producir informes, definiendo por ejemplo a Nueva Inglaterra integrada por Maine, Vermont y New Hampshire, mientras que el Departamento de Expedición siempre ha definido a Nueva Inglaterra formada por Maine, Vermont, New Hampshire, Massachusetts y Rhode Island. Si ahora deben moverse en la dirección de una base de datos integrada, alguna de las definiciones de REGION deberá modificarse, o de lo contrario parecerá que hemos despachado a los comerciantes yankees ¡más mercadería que la que hemos vendido! ¿Quién deseará cambiar su descripción? Ningún departamento se beneficia directamente, aunque la gerencia de la corporación se beneficia como consecuencia de que ahora se podrán generar informes que comparen las ventas y los embarques. El administrador de la base de datos deberá emplear tacto y persuasión para ir compatibilizando lentamente las definiciones de datos.

Una vez confeccionado el diccionario de datos de toda una empresa, los beneficios son importantes:

- Los usuarios se benefician porque pueden tener una lista uniforme de los datos que tienen disponibles. Esto es particularmente valioso cuando los usuarios tienen acceso en línea a las bases de datos y pueden emplear lenguajes de consulta que le permiten plantear cuestiones acerca de los datos, tales como "¿Qué PRODUCTO ha tenido la semana pasada VENTAS en cualquier REGION que fuera un 10% mayor que la VENTA de la semana anterior en esa REGION? Para que el "software" del lenguaje de consulta (explicado con mayor detalle en el Capítulo 7) sea capaz de manejar esta pregunta debe existir una

definición en el diccionario de datos para PRODUCTO,VENTAS y REGION. Si el diccionario de datos cubre toda la empresa, los usuarios pueden formular preguntas sobre todos los aspectos. Si, como sucede con muchos sistemas de consulta actualmente instalados, el diccionario de datos solo cubre un área de aplicación, las posibilidades son mucho más limitadas.

- Los analistas se benefician porque pueden iniciar un nuevo proyecto con definiciones rápidamente disponibles de los datos a proveer al nuevo sistema. Gran parte del trabajo preliminar para la construcción de un diccionario de datos ya habrá sido ejecutado para ellos.
- Los diseñadores y programadores se benefician debido a que las características físicas de muchos de los datos que necesitan en sus programas están ya disponibles en formularios legibles por la máquina.

4.7 DICCIONARIOS DE DATOS Y PROCESAMIENTO DISTRIBUIDO

En los años 60 existían importantes economías de escala en computación; salía más barato tener una computadora grande que dos computadoras de la mitad de su capacidad. Por diversas razones, esto ya no es cierto; a fines de la década del 70 hemos llegado a una etapa donde las computadoras pequeñas se han vuelto más baratas, capacidad por capacidad, que las grandes. En consecuencia, es muy atractivo proveer a cada departamento, sucursal, fábrica y depósito su propia computadora pequeña y tener una sola computadora en la casa matriz con la finalidad de integrar la información gerencial desde todas estas computadoras pequeñas distribuídas.

Por ejemplo, la sucursal de Nueva York de una firma establecida en Chicago podría tener una computadora pequeña, digamos un Sistema 34 o una PDP-11 con unas pocas terminales con CRT y emplearla para la entrada de pedidos y la generación de facturas. Todas las noches la computadora de Nueva York disca automáticamente a la computadora principal de Chicago y transmite los detalles de los pedidos y los importes a Chicago, que recibe todos los pagos para la corporación; esta información se utiliza para actualizar los archivos principales de la corporación. A su turno, la computadora principal de Chicago transmite a Nueva York los detalles de los pagos recibidos de los clientes de Nueva York durante el día anterior e información, por ejemplo, sobre plazo de entrega de todos los productos. Al inicio del nuevo día la computadora de Nueva York ha actualizado sus archivos locales de manera que puede saber qué clientes están atrasados con sus pagos, qué productos no tienen stock y en qué depósitos, etc. El hecho es que Nueva York ahora tiene suficiente información para su trabajo del día y solo necesitará comunicarse con Chicago para atender condiciones de excepción.

Si, como se ha visto en muchas corporaciones a mediados de la década del 70, todos los procesamientos se hacían en una gran computadora en Chicago, la única forma de dar un servicio comparable a Nueva York era teniendo una costosa (y propensa a faltas) línea dedicada entre cada sucursal y la casa matriz.

Las ventajas del procesamiento distribuido sobre el procesamiento centralizado son:

1. *Mayor confiabilidad:* Las sucursales no se paralizan por fallas de la computadora central o de la línea telefónica; puesto que cada una tiene su propio procesador, pueden continuar sus actividades aun cuando no puedan intercambiar información con la casa central.

2. *Menor costo:* El costo total del equipamiento para el sistema distribuido es a menudo más barato que el del sistema centralizado más las líneas telefónicas.

3. *Mayor flexibilidad:* Cada gerente de sucursal tiene la posibilidad de programar su propia computadora para tomar en cuenta las variaciones locales del negocio o los diferentes informes que requiera. Verdaderamente, ha sido esta característica del procesamiento distribuido, más que ninguna otra, lo que realmente ha contribuido a su rápido desarrollo. Muchos gerentes usuarios no están satisfechos con el nivel de servicio que reciben de los

recursos de computación centralizados y de la calidad de la respuesta dada a sus necesidades por los servicios de computación centralizados así que aprovechan la oportunidad de tener un servicio de computación con máquinas baratas, fáciles de instalar y que parecen fáciles de programar. Una minicomputadora ha sido definida como "¡cualquier computadora tan barata que se puede comprar sin conocimiento de la casa matriz!"

Sin embargo, los peligros de esta tendencia hacen que, a menos que cada empresa sea muy cuidadosa, los sistemas distribuidos evolucionen cada uno por su propio camino, con definiciones de datos y programas incompatibles, de manera que será imposible a los componentes de la empresa compartir los datos. Aquí es donde entra en acción el diccionario de datos. Con la condición de que cada sistema local se atenga rígidamente a los elementos de datos y estructuras definidas en el diccionario de datos, siempre le será posible al sistema central reunir y utilizar los datos de los procesadores locales y viceversa, no importa qué programas sean escritos en cada punto local. Nadie puede cumplir con las definiciones de un diccionario de datos a menos que disponga del diccionario; de allí se desprende que es probable que asistamos al resurgimiento de la actividad de creación y difusión del diccionario de datos en toda la organización.

APENDICE

Paquetes de "software" de diccionario de datos disponibles comercialmente:

DATA CATALOGUE-Synergetics Corporation-1 De Angelo Drive-Bedford, MA 01730
DATAMANAGER-MSP Inc.-594 Marrett Road-Lexington, MA 02173
DATA DICTIONARY-Cincom Systems Inc.-307 Maples Ave. West-Vienna, VA 22180
IMB DB/DC DICTIONARY SISTEMS-See IBM GH20-9104
LEXICON-Arthur Anderson and Co.-69 West Washington Street-Chicago, IL 60602
UCC TEN-University Computing Company-8303 Elmbrook Drive-Dallas TX 75247

BIBLIOGRAFIA

4.1 *IMS/VS General Information Manual, GH20-1260,* IBM Corporation, White Plains, N.Y., 1974.
4.2 National Bureau of Standars Handbook 113, *CODASYL Data Description Language Journal of Development,* Government Printing Office, Washington, D.C., 1974.
4.3 M. Jackson, *Principles of Program Design,* Academic Press, New York, 1974.
4.4 H. Lefkovitz, *Data Dictionary Systems,* QED Information Sciencies, Wellesley, Mass., 1977.
4.5 Leong-Hong and B. Marron, *Technical Profile of Seven Data Element Dictionary/Directory Systems,* National Bureau of Standards Special Pub. 500-3, Government Printing Office, Washington, D.C., 1977.

Ejercicios y puntos de discusión

1. Confeccionar un listado exhaustivo de todos los elementos de datos (solamente nombre y una concisa descripción) de un sistema simple con el que usted esté familiarizado (o utilice el sistema CBM sin inventario descripto en el Capítulo 2) ¿Con cuántos elementos de datos terminó usted? ¿Cuánto tiempo le llevó por elemento? Si usted tiene acceso a un sistema de tiempo compartido, úselo para clasificar los elementos de datos por nombre.
2. Liste algunos ejemplos de incompatibilidades de datos en su empresa.
3. ¿Qué elemento de datos entre los que usted conoce tiene la mayor cantidad de alias?
4. ¿Cuantos formatos diferentes de fecha calendaria se utilizan en su empresa?
5. Describa algunas estructuras de datos complejas usando la notación de opción, alternación e iteración.
6. Diseñe un formulario de propósitos múltiples que podría usarse para efectuar cualquier entrada en un diccionario de datos.
7. Familiarícese con las facilidades y comandos del "software" del diccionario de datos de su instalación (o de la instalación más próxima que tuviera dicho "software").

8. Construya una tabla relacionando las abreviaciones standard de dos letras para los Estados Unidos con los rangos válidos del código postal de cada estado. ¿En qué circunstancias valdría la pena validar una dirección usando esta tabla?
9. En un sistema de procesamiento distribuido ¿en qué lugar debería estar el diccionario de datos?
10. ¿Piensa usted que vale la pena el esfuerzo que demanda la creación de un diccionario de datos de proyectos? En caso negativo indique por qué.

5

ANALISIS Y PRESENTACION DE LA LOGICA DEL PROCESO

5.1 LOS PROBLEMAS PARA EXPRESAR LA LOGICA

Al definir un proceso en el diccionario de datos, especificamos las entradas y las salidas y escribimos un resumen de la lógica tan claramente como fue posible en lenguaje corriente. Como se indicó en el Capítulo 2, existe una cantidad de peligros al expresar la lógica en lenguaje descriptivo; en este capítulo consideraremos estos problemas y luego examinaremos las herramientas que nos permiten expresar la lógica del proceso con claridad y sin ambigüedades.

5.1.1 No solo pero no obstante, y/o a menos que...

¿Cuál es la diferencia entre las siguientes cinco oraciones?

1. "Sumar *A* a *B* a menos que *A* sea menor que *B*, en cuyo caso restar *A* de *B*."
2. "Sumar *A* a *B*. Sin embargo, si *A* es menor que *B*, la respuesta es la diferencia de *A* y *B*."
3. "Sumar *A* a *B*, pero restar *A* de *B* cuando *A* es menor que *B*".
4. "El total se encuentra sumando *B* a *A*. A pesar de la expresión previa, en caso que *B* sea mayor que *A* el resultado será la diferencia entre *B* y *A*."
5. "El total es la suma de *A* y *B*. Solo cuando *A* sea menor que *B* deberá usarse la diferencia como total".

La respuesta, por supuesto, es que *no* existe diferencia lógica. Las formas de las cinco oraciones en lenguaje narrativo o descriptivo oscurecen en realidad su similitud, ya que cada una puede ser transformada en una oración de la forma standard SI-LUEGO-SI NO-ENTONCES o SI-ENTONCES-SI NO-LUEGO.

SI	*A* es menor que *B*
LUEGO	restar *A* de *B*
SI NO	*(A* no es menor que *B)*
ENTONCES	sumar *A* a *B*

Al tratar de comprender la descripción narrada de documentos, memorandos y especificaciones de política, continuaremos tropezando con la variedad de posibles formas que el lenguaje corriente permite. Como analistas, necesitamos ser capaces de reducir estas variaciones a simples declaraciones lógicas para usarlas como especificaciones de programas

y de sistemas. Nuestra aspiración es reducir todos los documentos a frases imperativas y a condiciones. Usaremos el término *acción* para referirnos a alguna frase imperativa tal como "Sumar *A* a *B*" o "Remitir requerimiento de prepago" y *condición* para referirnos a algún hecho que determine cuál es la acción a tomar (por ejemplo, "*A* es menor que *B*" y "El cliente es regular").

Declaración 1.

"Sumar *A* a *B* a menos que *A* sea menor que *B*, en cuyo caso restar *A* de *B*"

se reduce a

"Acción-1, *salvo* Condición-1, en *cuyo caso* Acción-2

la cual en nuestra forma standard SI-LUEGO-SI NO-ENTONCES, es

SI	Condición-1
LUEGO	Acción-2
SI NO	(no Condición-1)
ENTONCES	Acción-1

Obsérvese que el orden de las acciones en la estructura SI-LUEGO-SI NO-ENTONCES es inverso al orden en la declaración en lenguaje corriente, a menos que volvamos a escribir la condición para hacer primero una pregunta negativa:

SI	no Condición-1	SI	*A no* es menor que *B*
LUEGO	Acción-1	LUEGO	sumar *A* a *B*
SI NO	(Condición-1)	SI NO	(*A es* menor que *B*)
ENTONCES	Acción-2	ENTONCES	restar *A* de *B*

Declaración 2.

"Sumar *A* a *B*: Sin embargo, si *A* es menor que *B*, la respuesta es la diferencia entre *A* y *B*".

se reduce a

Acción-1; *sin embargo, si* Condición-1, Acción-2

(La frase "la respuesta es la diferencia entre *A* y *B*" es un imperativo implícito y por lo tanto se la considera una acción.)

Declaración 3.

"Sumar *A* a *B*, pero restar *A* de *B* cuando *A* sea menor que *B*"

se reduce a

Acción-1, *pero* Acción-2, *donde* Condición-1

Las declaraciones 4 y 5 pueden reducirse de igual forma; se las muestra en la Fig. 5.1 con algunas otras declaraciones o sentencias comunes. Ello indica la variedad de estructuras de declaraciones en el lenguaje común con las que el pobre analista debe trabajar para obtener su información sobre las políticas y procedimientos que han de configurarse en el sistema.

El principio rector es

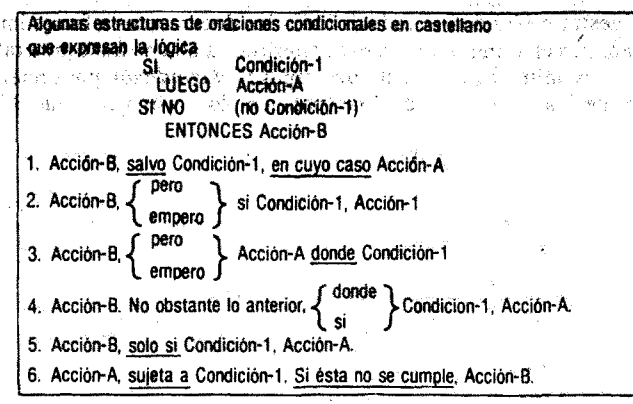


Figura 5.1 Lógica SI-LUEGO-SI NO-ENTONCES.

Identificar las acciones y las condiciones y utilizar la estructura de declaraciones comunes que se encuentran en la Fig. 5.1 para volcar la oración al modelo SI-LUEGO-SI NO-ENTONCES.

5.1.2 Mayor que, menor que

Como si la variedad de estructuras no fuera ya una situación bastante confusa, el lenguaje común a menudo nos presenta problemas cuando se necesita expresar un rango de valores como parte de una condición. Supongamos que decimos "Hasta 20 unidades sin descuento. Más de 20 unidades, 5% de descuento." ¿Qué puede ser más claro que esto? ¿Pero qué descuento se hace para exactamente 20 unidades? El problema es que el lenguaje necesita la delicada frase "inclusive" o "hasta e inclusive". Podríamos decir "1-19 unidades" o "cantidad menor o igual que 20" y ya estamos de nuevo ante el problema de expresar de varias maneras la misma cosa, como se muestra en la Fig. 5.2.

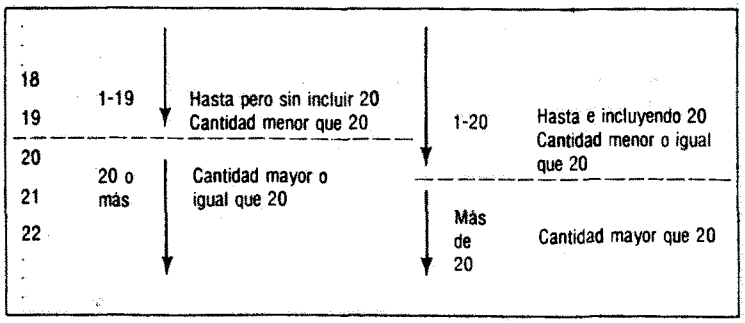


Figura 5.2 Expresión de rangos.

Tenemos tendencia a ser descuidados respecto de la forma de especificar rangos, porque *nosotros* sabemos qué posibilidades incluimos, pero los usuarios a menudo se sorprenden cuando la estricta interpretación de sus palabras resulta ser diferente de aquella que habían pensado. Para verificar que se hayan contemplado todos los posibles rangos, podemos buscar a través de las descripciones las palabras "menor", "hasta" "mayor que" y "más" y así sustituirlas adecuadamente por los términos del programa MQ, MI, mI y mQ, donde

MQ significa "mayor que."
MI significa "mayor o igual que."
mI significa "menor o igual que."
mQ significa "menor que."

Está claro que si la condición-1 es mQ 20 y la condición-2 es MQ 20, aún tenemos el problema de saber qué debemos hacer cuando la cantidad es exactamente 20. Debemos asegurarnos de que todos los usuarios estén familiarizados con estas distinciones, aunque las abreviaturas son una jerga fácilmente aceptada.

5.1.3 Ambigüedad y/o

Consideremos la siguiente declaración de política:

> Los clientes que nos compran más de $ 10.000 por año y tienen una buena historia de pagos o que han comerciado con nosotros por más de 20 años deberán recibir trato preferencial.

¿Es suficiente comprar más de $ 10.000 por año para tener trato preferencial? Es claro que no, como consecuencia de la frase "*y* tengan una buena historia de pagos." ¿Pero será suficiente con haber sido cliente por más de 20 años? Bueno, todo dependerá del tono de voz con que uno lea la política. Si usted la lee así

("más de $ 10.000 por año")

y

("buena historia de pagos" o "más de 20 años")

diría "No; también deberá comprar más de $ 10.000 por año." Pero si usted lo lee así

("más de $ 10.000 por año" y "buena historia de pagos")

o

("más de 20 años")

diría "Sí, porque siendo un cliente con más de 20 años tiene prioridad de trato, no importando lo pequeñas que sean las compras que haga, o lo mal que pague."

La persona que escribió (o más probablemente dictó) la política original sabía lo que quería decir; desgraciadamente cualquier oración en lenguaje corriente que incluya cualquiera de las condiciones conjuntivas "y" y "o" es clara cuando se la dice, debido al énfasis dado, pero no es clara cuando se ve escrita. Peor aún, si el analista no hace cuestión por la ambigüedad y/o escribe la política en la especificación, el programador puede no plantearse problema alguno y codificar, digamos en COBOL tal como está escrito. El compilador COBOL contiene reglas para decidir la precedencia de "y" (and) y "o" (or) a menos que se le den instrucciones específicas; en este caso ello hará que el programa se comporte como si la política fuera:

("más de $ 10.000 por año" y "buena historia de pagos")

("más de 20 años")

y dará prioridad al viejo Juan que ha comprado mercaderías por valor de $ 100 cada Navidad los pasados 35 años y nos ha pagado ya entrada la época de la cosecha siguiente. Esto puede, o no, ser lo que la gerencia de la empresa desea, pero si la decisión queda en manos del compilador COBOL, ¡solo tenemos el 50% de probabilidad de estar en lo correcto!

La moraleja es que si no podemos evitar escribir las políticas que combinan "y" y "o" en la misma oración, debemos reformular las combinaciones para expresar nuestro real entender, de manera que el programador pueda saber su significado sin ambigüedades. Por ejemplo, podemos escribir

> Los clientes que nos compran más de $ 10.000 por año y, *además, o bien* tienen una buena historia de pagos, *o* han comerciado con nosotros por más de 20 años, deberán recibir trato preferencial.

5.1.4 Adjetivos indefinidos

¿Qué es una buena historia de pagos?
¿Qué es un cliente regular?

Las personas desarrollan en una organización cierta experiencia en sus funciones y como resultado pueden asignar significados a adjetivos como los indicados. Como analistas debemos estar seguros de poder definir los adjetivos y ser capaces de contar con una buena historia a partir de una mala. A menudo, la regla para las definiciones es bastante compleja, y a menos que examinemos cada declaración de política, identifiquemos los adjetivos y estemos seguros de que hemos definido cada uno de ellos, nunca hallaremos la solución.

Como se indicó en la Sec. 4.2, el diccionario de datos nos provee un buen formato para definir los adjetivos que se utilizan en la lógica. La misma frase "buena historia de pagos" implica la posibilidad de una *mala* historia de pagos, y ello implica a su vez la existencia de un elemento de datos "PAGO-HISTORIA-TIPO" que puede tener el valor "BUENO" o "MALO". Volviendo al formato del diccionario de datos, sabemos que tenemos que dar un significado a cada valor de un elemento de datos discreto, como puede verse en la Fig. 5.3.

PAGO-HISTORIA-TIPO Elemento de datos

Breve descripción *Define si el cliente es considerado como un buen pagador o no.* Tipo (A) AN N

Alias (contextos)

Si es discreto		Si es continuo
Valor	Significado	Rango de valores
BUENA	Ningún pago de factura se excedió más de 30 días en los últimos 6 meses.	
MALA	El pago de una o más facturas excedido/s en más de 30 días en los últimos 6 meses.	Valor típico Longitud Representación interna

(Si son más de 5 valores, continuar a la vuelta o hacer referencia a hoja separada)

Otra información de edición

Estructuras de datos/elementos de datos relacionados

Figura 5.3 Elemento discreto de datos expresado por un adjetivo.

5.1.5 Manejo de combinaciones de condiciones

Hasta aquí, hemos visto los problemas lógicos que pueden ocurrir en declaraciones bastante simples y cómo se las puede construir sin ambigüedades. Una clase distinta de problemas aparece cuando, como sucede con frecuencia, las declaraciones de políticas involucran *combinaciones* de condiciones.

La política sobre el trato a los clientes especifica dos conjuntos de condiciones combinadas:

"compras por más de $ 10.000" *y* "buena historia de pagos"

o

"compras por más de $ 10.000 y más de 20 años con nosotros"

La política especifica que estas combinaciones conducen a una acción determinada (tratamiento preferencial) e implica que las demás combinaciones conducen al tratamiento normal. Podemos dibujar un gráfico de esta política como un *árbol de decisión*, como en la Fig. 5.4 o volcarlo en el formato SI-LUEGO-SI NO-ENTONCES como en la Fig. 5.5.

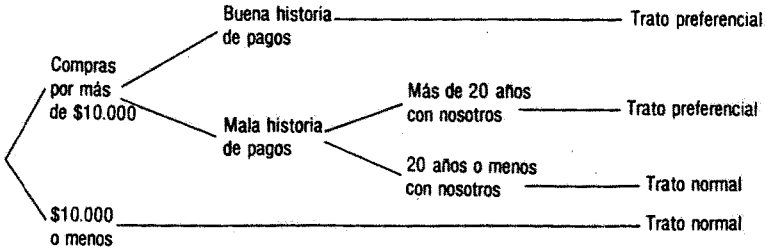


Figura 5.4 Arbol de decisión.

```
SI      cliente compra más de $10.000
  y-SI          cliente tiene buena historia de pagos
      LUEGO     trato preferencial
    SI NO       (mala historia de pagos)
     entonces SI      cliente tiene más de 20 años con nosotros
              LUEGO   trato preferencial
            SI NO     (20 años o menos)
              ENTONCES    trato normal
SI NO (cliente compra $10.000 o menos)
  ENTONCES trato normal
```

Figura 5.5 SI-LUEGO-SI NO-ENTONCES.

Hacemos notar que las combinaciones de las condiciones nos llevan a "incluir" o "anidar" SIs dentro de SIs y SI NOs. Hemos "endentado" o hecho sangrías con los SIs y SI NOs que están incluidos o "anidados" dentro de otros SIs y SI NOs y hemos alineado cada SI NO con su correspondiente SI.

El formato SI-LUEGO-SI NO-ENTONCES, también llamado "lenguaje estructurado", es más fácil de mecanografiar para una buena presentación que el árbol de decisión, y permite escribir completamente las condiciones y las acciones, pero no muestra la estructura de las políticas tan vívidamente como el árbol de decisión. Posteriormente discutiremos los pros y contras de estas técnicas; para una decisión de este simple tipo *ambas* son menos fáciles de manejar que la declaración sin ambigüedades que hemos construido:

Los clientes que nos compran más de $ 10.000 por año y *además, o bien* tienen una buena historia de pagos *o* han comerciado con nosotros por más de 20 años, deberán recibir trato preferencial.

¿Pero qué sucede si la decisión es realmente más compleja que esto? Supongamos clientes que han comerciado por menos de $ 10.000 por año y tienen una buena historia de pagos ¿pueden tener también trato preferencial?

El árbol de decisión sería entonces de la forma de la Fig. 5.6

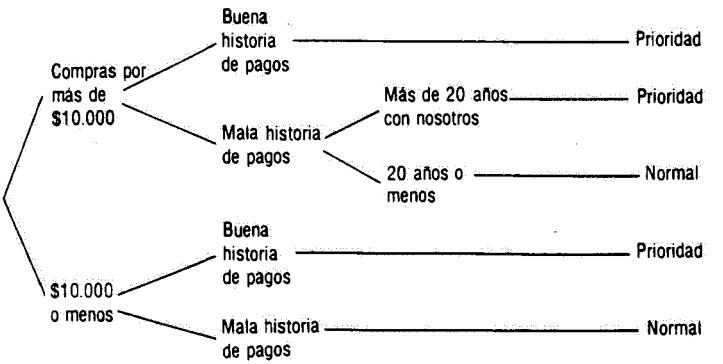


Figura 5.6 Arbol de decisión.

(a) Con condiciones y acciones incluidas:

Condiciones	c1: ¿más de $10.000 por año?	
	c2: ¿buena historia de pagos?	
	c3: ¿con nosotros más de 20 años?	
Acciones	a1: trato prioritario	
	a2: trato normal	

8 filas: 3 condiciones; cada una tiene 2 posibilidades

$2^3 = 8$

(b) Con todas las posibles combinaciones de condiciones incluidas:

c1: ¿más de $10.000 por año?	S	S	S	S	N	N	N	N
c2: ¿buena historia de pagos?	S	S	N	N	S	S	N	N
c3: ¿con nosotros más de 20 años?	S	N	S	N	S	N	S	N
a1: trato prioritario								
a2: trato normal								

S = SI
N = NO

(c) Con las combinaciones de las condiciones conectadas a la acción:

c1: ¿más de $10.000 por año?	S	S	S	S	N	N	N	N
c2: ¿buena historia de pagos?	S	S	N	N	S	S	N	N
c3: ¿con nosotros más de 20 años?	S	N	S	N	S	N	S	N
a1: trato prioritario	X	X	X		X	X		
a2: trato normal				X			X	X

Figura 5.7 Tabla de decisión para prioridad del cliente.

No hay nada en el formato del árbol de decisión que nos estimule a preguntar si se pueden probar otras combinaciones de condiciones. Aquí aparece la utilidad de la *tabla de decisión*. La Fig. 5.7 muestra una tabla de decisión que representa nuestra nueva política en tres etapas de elaboración.

Como hemos identificado tres condiciones, cada una de las cuales tiene solo dos posibilidades, sabemos que habrá 2^3 u 8 combinaciones posibles. En la etapa (b) listamos completamente todas esas combinaciones como grupos de SI (S) y NO (N). De allí resultará la acción que corresponda a cada combinación de condiciones y la marcamos con una X.

Como se ve, la *tabla* de decisión es más compacta que el *árbol* de decisión; también tendremos la certeza, cuando hayamos completado una tabla de decisión exhaustiva como ésta, de que hemos considerado todas las posibles combinaciones de condiciones que pudieran ocurrir. La desventaja es que no nos dará un cuadro vívido de la estructura. Las tablas de decisión pueden ser incluso desconcertantes si no se las ha visto con anterioridad.

En las próximas tres secciones vamos a realizar el estudio de un caso que corresponde a un proceso algo complejo, viendo a su turno árboles de decisión, tablas de decisión y redacción en lenguaje estructurado, discutiremos las ventajas y desventajas de cada uno, y veremos el rol que cada técnica debe jugar en el análisis y la presentación de la lógica.

5.2 ARBOLES DE DECISION

Para ver el alcance de la técnica del árbol de decisión, vamos a analizar el enunciado de una política medianamente compleja.

La compañía CBM, cuyo flujo de datos analizamos en el Capítulo 3, despacha paquetes de libros a los clientes, y les carga el costo del despacho dentro de la factura (a menos que la factura sea abonada previamente). Los costos de embarque y manipuleo se expresan en unidades cuyo valor en pesos puede modificarse de tiempo en tiempo ajustándose a las tarifas vigentes, a la inflación, etc. El valor actual en pesos de una unidad es de 50 centavos. Lo que sigue se ha extractado de la hoja explicatoria de la empresa sobre tarifas:

> La tarifa por embarque aéreo dependerá del peso del paquete. La tarifa básica es de 3 unidades por libra, reducida a 2 unidades por libra para el exceso sobre las 20 libras, con un mínimo de 6 unidades. El flete terrestre (incluyendo el manipuleo) es de 2 unidades por libra para el despacho expreso; sin embargo, esta tarifa se aplica solamente en el área de despacho local. Si la dirección del destinatario se encuentra fuera del área local y el paquete excede las 20 libras o *no* se requiere el despacho expreso, el flete terrestre es el mismo que para el despacho local expreso. El despacho normal de paquetes hasta 20 libras es de 3 unidades por libra con un recargo de 1 unidad por servicio expreso (por cada libra).
>
> Aun considerando las condiciones del parágrafo anterior, el flete aéreo remitido al oeste del Mississippi se cargará con tarifa doble.

Nuestro primer paso en el análisis de este documento de política debe consistir en identificar

Condiciones
Acciones
Estructuras "a menos que", "sin embargo", "pero"...
Ambigüedades del tipo mayor que/menor que
Ambigüedades del tipo y/o
Adjetivos indefinidos

Como se ha analizado en la sección 5.1.

La Fig. 5.8 muestra el texto de la hoja descriptiva del despacho, acorde a estas

condiciones. Encontramos una confusión del tipo y/o, dos adjetivos indefinidos ("básico", "local") y una posibilidad de confusión de rango (hasta 20 libras, más de 20 libras). En base a este breve examen haremos algunas preguntas al Gerente de Expedición e incorporaremos las respuestas en un texto revisado. Las preguntas realizadas y las respuestas del Gerente de Expedición fueron las siguientes:

P. ¿La tarifa básica indicada en la línea 2 de la hoja se refiere a la vía aérea o a la terrestre?
R. Es para la vía aérea. La terrestre se trata en la declaración siguiente.

P. La hoja algunas veces se refiere al flete, y otras a despacho y manipuleo. ¿Existe alguna diferencia?
R. No; todas las tarifas incluyen flete y manipuleo.

P. ¿Qué quiere expresarse exactamente con área local?
R. Es el área servida por nuestros propios camiones; en la práctica es cualquier lugar que se encuentre dentro de los límites de la ciudad.

P. La hoja menciona "hasta 20 libras" y "más de 20 libras". ¿Cuál se aplicará a un paquete que pesa exactamente 20 libras?
R. Generalmente se interpreta que "hasta 20 libras" significa "hasta e incluso"; no podemos indicar todos los pequeños detalles, usted comprende.

P. La tercera oración de la hoja teóricamente puede tomarse de dos maneras. Puede leerse como "*ambos*-fuera del área local y también más de 20 libras-o *como alternativa*, que no se requiera expreso", o puede ser "fuera del área local y, *además, o bien* que sea de más de 20 libras *o* que no se requiera expreso." ¿Cuál es la correcta?
R. La segunda. El primer significado no puede ser correcto porque se puede terminar cargando la tarifa expreso local cuando no se ha requerido el despacho expreso. Entiendo su punto de vista, el enunciado *es* algo confuso.

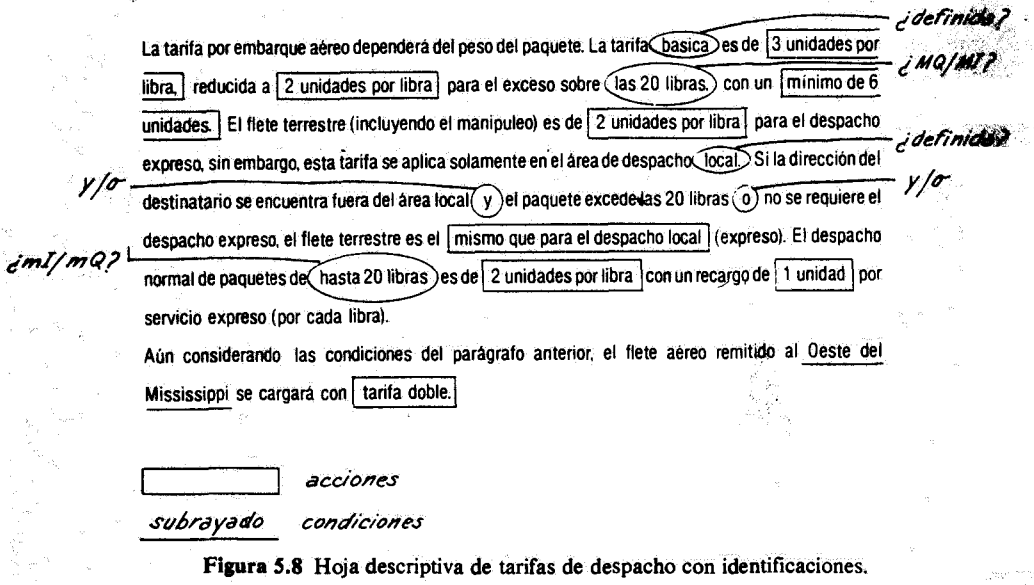


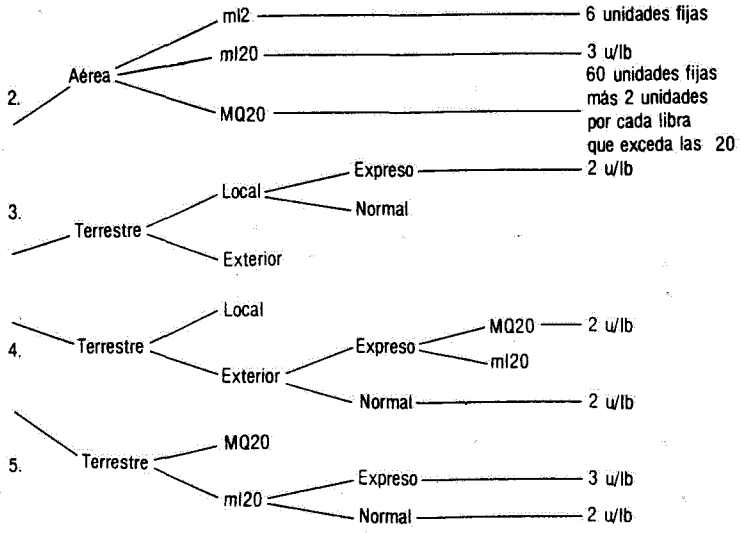
Figura 5.8 Hoja descriptiva de tarifas de despacho con identificaciones.

Ahora podemos volver a escribir la descripción original línea por línea. Usaremos las abreviaturas.

MQ: mayor que
mI: menor o igual que
u/lb: unidades por libra

Original	Revisado
1. El cargo por despacho aéreo dependerá del peso del paquete.	Antecedente útil pero no lógico: omitir
2. La tarifa básica es de 3 unidades por libra reducida a 2 unidades por libra para el exceso sobre 20 libras, con un mínimo de 6 lb.	La tarifa aérea es: Si pesa ml 2 lb: 6 unidades fijas Si pesa MQ 2 : ml 20 } 3 u/lb Si pesa MQ 20: 60 unidades fijas más 2 unidades por cada lb que exceda las 20
3. El flete terrestre, incluyendo el manipuleo, es de 2 u/lb para el despacho expreso; sin embargo, esta tarifa se aplica solo en el área de despacho local.	Tarifa terrestre, si es en el área local, y si es despacho expreso: 2 u/lb.
4. Si la dirección del destinatario se encuentra fuera del área local y el paquete excede las 20 lb o no se requiere el despacho expreso, el flete terrestre es *el mismo* que el despacho local (servicio expreso).	La tarifa terrestre, si está fuera del área local, *y además, o bien* el peso MQ 20 *o bien* el despacho es normal, es de 2 u/lb.
5. El despacho normal de paquetes hasta 20 libras es de 2 u/lb con el recargo de 1 unidad por servicio expreso (por libra).	La tarifa terrestre Si el peso ml20 y despacho *normal*, entonces, la tarifa es de 2 u/lb. Si el peso ml20 y despacho *expreso*, entonces, la tarifa es de 3 u/lb.
6. Aun considerando las condiciones del parágrafo anterior, el flete aéreo remitido al Oeste del Mississippi se cargará con tarifa doble.	Si el despacho es aéreo y el destinatario se encuentra al Oeste del Mississippi, se duplicará la tarifa.

Estamos en condiciones de comenzar a aclarar el embrollo. Dibujemos una parte del árbol de decisión para cada una de las oraciones revisadas:



Dejaremos la oración 6 correspondiente a despachos aéreos al Oeste, para considerarla en forma separada.

Si observamos las partes del árbol de decisión, vemos que 2, 3 y 4 encajan entre sí con claridad. ¿Qué pasa con 5? Parece que efectuara las preguntas en un orden diferente a 3 y 4 y no está claro si 5 se aplica al área local o a la externa. Sin embargo, como tenemos en 4 una tarifa sin especificar para "externa-expreso-mI20" podemos suponer que 5 se relaciona con esta posibilidad. En este caso, la rama inferior de 5 es redundante; sabemos que la tarifa "externa-normal" es 2. Así (teniendo en mente la necesidad de comprobar nuestra suposición) podemos integrar el árbol, donde cada rama refleja los posibles valores de una condición. Ver. Fig. 5.9.

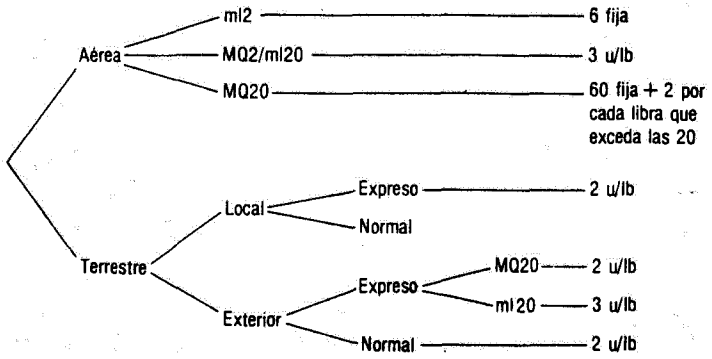


Figura 5.9 Arbol de decisión.

Se ve inmediatamente que una de las posibilidades presentada en el árbol de decisión ("terrestre-local-normal") no tiene especificada la tarifa. Obviamente se deberá volver a ver al Gerente de Expedición y verificar cuál es la tarifa en este caso. Puede ser que se aplique la sobrecarga de 1-unidad por envío expreso mencionada en la oración 5, de manera que la tarifa normal para despacho local sea de 1 unidad. Puede ser que todos los despachos locales sean expresos, de manera que el despacho normal no sea de aplicación. Simplemente no podemos decir nada en base a la información que contiene la hoja.

Si bien no resulta una inconsistencia lógica, en sentido estricto, el árbol destaca algo que parece contrario al sentido común: ¿Cuál será la tarifa para dos paquetes, uno que pesa 19 libras y otro que pesa 21 libras, ambos con envío por vía terrestre, externo al área local y por expreso? Como se ve en el árbol, el paquete de 19 libras cuesta: 3 unidades x 19 libras = 57 unidades, y el paquete de 21 libras cuesta: 2 unidades x 21 libras = 42 unidades. Parece haber algo pintoresco en el corte a través de 20 libras en la tarifa. Posiblemente la hoja debería haber sido redactada para que tuviera el mismo sentido que la oración correspondiente a la tarifa aérea; las 2 unidades por libra se deberían aplicar a la cantidad total. Una vez más anotamos otra pregunta para el Gerente de Expedición.

La oración 6, referida al flete aéreo hacia el Oeste, puede manejarse de una de estas dos maneras:

1. Dividiendo cada uno de los tres precios para flete aéreo en dos.
2. Creando un segundo paso en la decisión, solamente para el flete aéreo.

Es mejor tener todos los factores de la decisión incorporados en un cuadro, ya que así habrá menos posibilidad de omitir alguno, pero el árbol comenzará a "ramificarse" y a ser inmanejable. La Fig. 5.10 muestra el árbol completo.

Si el árbol aparece como confuso o poco familiar a los usuarios, podemos expresarlo como una tabla convencional (Fig. 5.11). Nótese que se ha modificado ligeramente el orden

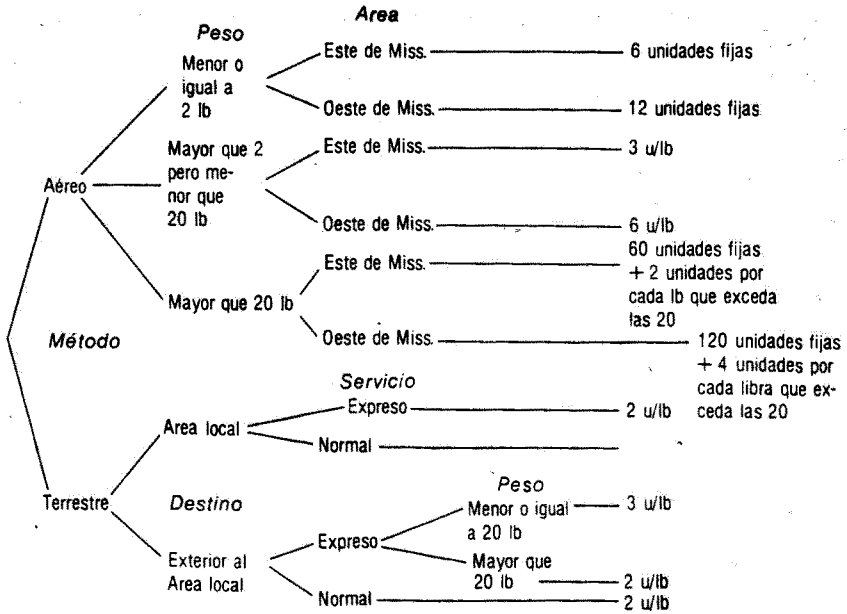


Figura 5.10 Arbol de decisión completo.

Método de embarque	Destinatario	Peso del paquete	Clase de servicio	Tarifas de embarque y manipuleo
Aéreo	Este del Mississippi	Menor o igual que 20 lb	N/A	6 unidades fijas
		Mayor que 2 lb pero menor que 20	N/A	3 u/lb
		Mayor que 20 lb	N/A	60 unidades fijas mas 2 por cada lb que exceda las 20
	Oeste del Mississippi	Menor o igual a 2 lb	N/A	12 unidades fijas
		Mayor que 2 lb pero menor que 20	N A	6 u/lb
		Mayor que 20 lb	N A	120 unidades fijas mas 4 por cada lb que exceda las 20
Terrestre	Area local	N/A	Expreso	2 u/lb
			Normal	?
	Exterior al área local	Menor o igual a 2 lb	Expreso	3 u/lb
			Normal	
		Mayor que 20 lb	Expreso	2 u/lb
			Normal	2 u/lb

Figura 5.11 Tabla convencional.

de las preguntas a los efectos de hacer más uniformes las columnas de la tabla. (N/A = no aplicable).

La tabla muestra otra anomalía que no resultaba obvia en el árbol de decisión. Para un paquete hacia el exterior del área local y de más de 20 libras, ¡no hay diferencia entre las tarifas expreso y normal! Esto confirma la sospecha de que la tarifa para "mayor que 20 libras" tiene realmente una excesiva sobrecarga, pero, como se indicó anteriormente, se deberá verificar con el gerente para asegurarse.

Otro aspecto de la tabla es el número de veces que se ha colocado "N/A". Cada vez que se pregunta al árbol de decisión por la tarifa aérea, no hay indicación del tipo de servicio (expreso o normal).

¿Pero es ésto realmente cierto? ¿Es realmente cierto que el peso de un paquete no tenga importancia en la tarifa local? ¿O es simplemente otra omisión de la hoja? Desarrollar una tabla obliga a considerar un mayor número de combinaciones de posibilidades.

Esta es una virtud de la *tabla de decisión standard;* provee una vía directa para identificar todas las posibles combinaciones de condiciones que puedan aparecer, y verificarlas sistemáticamente, de manera de asegurarnos que hemos llegado al fondo de todas las complicaciones. En la sección próxima se describen las convenciones para desarrollar tablas de decisión y luego se aplicará la tecnica para resolver el problema de la tarifa del flete.

5.3 TABLAS DE DECISION

A fin de ver las convenciones para las tablas de decisión, volvamos a la Fig. 5.7 (c), que reproducimos aquí

	1	2	3	4	5	6	7	8
c1: ¿más de $10.000 por año?	S	S	S	S	N	N	N	N
c2. ¿buena historia de pagos?	S	S	N	N	S	S	N	N
c3: ¿con nosotros más de 20 años?	S	N	S	N	S	N	S	N
a1: trato prioritario	X	X	X		X	X		
a2: trato normal				X			X	X

5.3.1 Condiciones, acciones y reglas

En este formato de tabla, las diversas acciones que deberán tomarse como resultado de una decisión están listadas en la sección inferior izquierda, conocida como *talón de acciones.* En forma similar, las diversas condiciones que afectan las decisiones están listadas en la sección superior izquierda, *el talón de condiciones.* Escribimos las condiciones como preguntas para que puedan ser contestadas por "si" o "no", de manera que podamos colocar todas las combinaciones posibles listando muestras compuestas de S y N en la sección superior derecha, sin repeticiones ni omisiones. Cada combinación se denomina una *regla* en el trabajo de la tabla de decisión; por ejemplo, la regla 3 es:

	3
c1: ¿más de $10.000 por año?	S
c2: ¿buena historia de pagos?	N
c3: ¿con nosotros más de 20 años?	S
a1: trato prioritario	X
a2: trato normal	

lo cual equivale a decir

"Si el cliente nos compra más de $ 10.000 por año

y

si *no* tiene una buena historia de pagos

y

si ha estado con nosotros más de 20 años. . ."

La X es equivalente a completar la oración con

"entonces tiene trato prioritario"

5.3.2 Construcción de la matriz de reglas

Existen diversas maneras de asegurarse la cobertura de cada posibilidad sin producir repeticiones. Una forma simple es :

1. La cantidad de reglas se puede obtener multiplicando entre sí el número de posibilidades de cada condición; por ejemplo,

C1: más de $ 10.000		2 posibilidades
C2: buena historia de pagos	X	2 posibilidades
C3: con nosotros más de 20 años	X	2 posibilidades
total		8

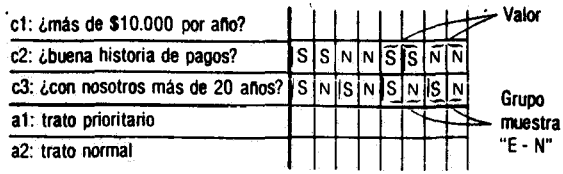
2. Crear los talones de condiciones y acciones y prever suficiente cantidad de columnas para todas las reglas:

c1: ¿más de $10.000 por año?								
c2: ¿buena historia de pagos?								
c3: ¿con nosotros más de 20 años?								
a1: trato prioritario								
a2: trato normal								

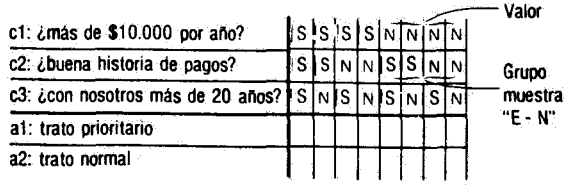
3. Tomar la última condición y alternar sus posibilidades a lo largo de la fila:

c1: ¿más de $10.000 por año?								
c2: ¿buena historia de pagos?								
c3: ¿con nosotros más de 20 años?	S	N	S	N	S	N	S	N
a1: trato prioritario								
a2: trato normal								

4. Observar cuantas veces se repite la muestra. C3 tiene solo dos posibilidades, de manera que la muestra SN se repite cada dos columnas. Si la condición tiene tres posibilidades, se repetirá cada tres columnas, etc. Tomar la condición inmediata superior a la que hemos completado y cubrir cada grupo de la muestra con un valor de la siguiente condición repitiendo hasta que sea necesario:



5. Observar cuántas veces se repite la nueva serie de muestras (cada cuatro columnas en este caso) y cubrir cada serie o grupo de muestras con el valor de la condición superior:



Si se ha calculado correctamente el número de columnas en el primer punto, los valores de la condición superior deben caber exactamente y habrá encontrado así que se han cubierto todas las combinaciones posibles de valores.

Esta técnica para construir la matriz funciona igualmente bien cuando existe más de dos valores por condición, como veremos al tratar el ejemplo de la tarifa de flete.

5.3.3 Indiferencia

Una vez que se ha completado la matriz con todas las combinaciones posibles, podemos colocar las X en la parte inferior derecha de la tabla para indicar las acciones que deben tomarse para cada combinación. En aquellos casos en que no tengamos la seguridad de poseer toda la información sobre la política en cuestión, podemos conversar con el usuario y considerar por turno cada regla para asegurarnos de que nuestra tabla es correcta. Una vez hecho esto, podemos recorrer la tabla para ver si se puede eliminar alguna de las columnas.

Tomemos la tabla completa para decidir el trato prioritario, que volvemos a reproducir aquí

	1	2	3	4	5	6	7	8
c1: ¿más de $10.000 por año?	S	S	S	S	N	N	N	N
c2: ¿buena historia de pagos?	S	S	N	N	S	S	N	N
c3: ¿con nosotros más de 20 años?	S	N	S	N	S	N	S	N
a1: trato prioritario	X	X	X		X	X		
a2: trato normal				X			X	X

Consideremos las reglas 7 y 8. Nos indican, en efecto, que si un cliente comercia menos de $ 10.000 por año y *no* tiene una buena historia de pagos, entonces tendrá trato normal si tiene más de 20 años con nosotros (regla 7) y tendrá trato normal si tiene *menos* de 20 años con nosotros. En otras palabras, si se es un cliente chico y mal pagador, *no importa* cuánto tiempo hace que está con nosotros; nunca tendrá trato prioritario. En la *jerga* de la tabla de decisión, las reglas 7 y 8 son *indiferentes* respecto del valor de C3. Podemos remplazarlas por una única columna, empleando un "-" (símbolo de indiferencia) para C3:

	1	2	3	4	5	6	7/8
c1: ¿más de $10.000 por año?	S	S	S	S	N	N	N
c2: ¿buena historia de pagos?	S	S	N	N	S	S	N
c3: ¿con nosotros más de 20 años?	S	N	S	N	S	N	–
a1: trato prioritario	X	X	X		X	X	
a2: trato normal				X			X

La técnica para compactar de este modo la tabla de decisión es:

1. Encontrar un par de reglas para las cuales
- La acción sea la misma.
- Los valores de la condición sean los mismos excepto para una *y solo una* condición donde sean diferentes.

2. Remplazar aquel par por una sola regla empleando el símbolo de indiferencia para la única condición que es diferente.

3. Repetir para cualquier otro par que cumpla con el criterio.

En los hechos, la tabla de prioridad del cliente se reduce a cinco reglas:

	1/2	3	4	5/6	7/8
c1: ¿más de $10.000 por año?	S	S	S	N	N
c2: ¿buena historia de pagos?	S	N	N	S	N
c3: ¿con nosotros más de 20 años?	–	S	N	–	–
a1: trato prioritario	X	X		X	
a2: trato normal			X		X

¿Por qué no se pueden compactar o condensar las reglas 3 y 4 ya que tienen las mismas entradas, excepto para C3? Porque conducen a diferentes acciones, de modo que C3, lejos de no importar, origina toda la diferencia en la decisión.

El lector podrá encontrar de utilidad dibujar un árbol de decisión que exprese la tabla anterior y luego compararla con la Fig. 5.6. Aun cuando la tabla y el árbol son equivalentes, se ha llegado a obtener la tabla considerando todas las posibilidades; que probablemente el árbol no las tenga.

5.3.4 Extensión de las entradas, el problema de la tarifa del flete

Las tablas del tipo que hemos considerado, donde los valores de la condición están limitados a dos ("si" y "no" en nuestro caso) se conocen como tablas de *entrada limitada.* Cuando la condición puede tener más de dos valores, la tabla se conoce como tabla de *entrada extendida.* Vamos a requerir esta última para resolver el problema de la tarifa del flete, donde el peso de un paquete puede tomar tres valores (mI2, MQ2 pero mI20, MQ20) y como la compañía está instalada en Nueva Jersey, podemos considerar que el destinatario puede tomar tres valores (local, exterior al área local pero todavía al Este del Mississippi, exterior al área local y al Oeste del Mississippi). Veamos los pasos a seguir para desarrollar una tabla de decisión para este caso más complejo.

1. Primero, listemos las condiciones y las acciones que hemos identificado, asignando las correspondientes abreviaturas para ser utilizadas en la tabla.

Condiciones:

C1: Método de despacho	A: Aéreo T: Terrestre
C2: Destinatario:	L: Local E: Exterior al área local pero al Este del Mississippi O: Exterior al área local pero al Oeste del Mississippi
C3: Peso:	L: Liviano, mI2 libras M: Mediano, MQ2 pero mI20 libras P: Pesado, MQ20 libras
C4: Servicio:	E: Expreso N: Normal

(Obsérvese que se han definido solo cuatro elementos de datos discretos y se ha puesto nombres a sus valores).

Acciones:

A1: 6 unidades fijas
A2: 12 unidades fijas
A3: 60 unidades fijas más 2 unidades por libra en exceso sobre 20
A4: 120 unidades fijas más 4 unidades por libra en exceso sobre 20
A5: 2 u/lb
A6: 3 u/lb
A7: 6 u/lb

No hay nada sagrado en el orden de las condiciones y las acciones; el orden de las condiciones es como nos parezca aproximadamente el orden de importancia. Deberemos tener en cuenta que a medida que se vaya desarrollando la tabla el Gerente de Expedición puede mencionarnos otras condiciones y acciones de las que aún no hemos tenido conocimiento.

2. Luego establecemos cuántas columnas deberá tener el árbol de decisión:

C1 tiene 2 posibilidades
C2 tiene 3 posibilidades
C3 tiene 3 posibilidades
C4 tiene 2 posibilidades

lo cual nos da un total de $2 \times 3 \times 3 \times 2$ o sea 36 combinaciones.

3. La Fig. 5.12a muestra las franjas de condiciones y acciones de las primeras 12 columnas y las posibilidades alternativas ocupadas por la primera condición, C4. La muestra se repite cada dos columnas. Como dijimos en la Sec. 5.3.2, el próximo paso es cubrir cada grupo de muestra con el valor de la próxim condición (ver Fig. 5.12.b). Ahora se cubre cada muestra (de 6 columnas) de C3 con un valor de C2, etc. El resultado se ve en la Fig. 5.13.

4. Con la matriz ubicada en la tabla debemos considerar las acciones que corresponden a cada combinación de condiciones. Antes de ello, debemos preguntarnos si existe alguna combinación esencialmente imposible, por ejemplo, si alguna de nuestras reglas es imposible. Encontramos inmediatamente un ejemplo; las reglas 1-6 de la Fig. 5.13 corresponden todas al despacho aéreo a un destino local. Esto es manifiestamente impracticable y, como se ve, las seis reglas han sido eliminadas. Esto nos deja solo 30 reglas para trabajar; hemos completado la sección relacionada con el despacho aéreo, y se invita al lector a completar las otras, antes de ver cuáles pueden condensarse.

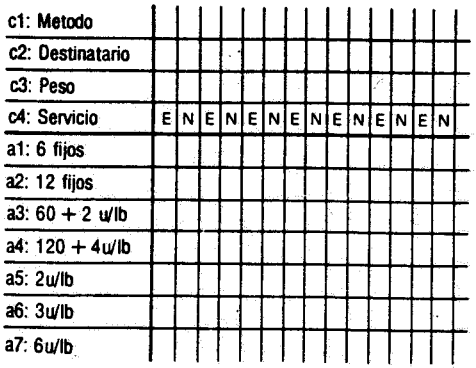


Figura 5.12a Primer paso en la construcción de la matriz.

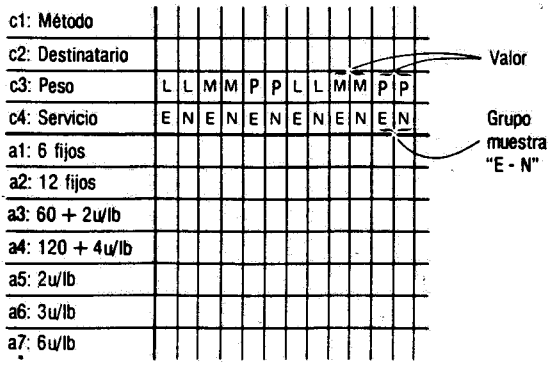


Figura 5.12b Segundo paso en la construcción de la matriz.

5.3.5 Tablas de decisión y árboles de decisión

Supongamos que en vez de las 7 acciones separadas que hemos identificado en el problema de la tarifa del flete, tenemos .30 acciones separadas, una por cada posible combinación de condiciones. La franja de acciones de la tabla de decisión podrá ser más grande pero no esencialmente más compleja. ¿Puede imaginar cómo se vería el árbol de decisión? Esto nos da una guía para

Pauta 1. Utilizar un árbol de decisión cuando el número de acciones es pequeño y no sean posibles todas las combinaciones de condiciones; usar una tabla de decisión cuando el número de acciones es grande y pueden plantearse muchas combinaciones de condiciones.

Si ahondamos más en la hoja de tarifas de flete continuamos encontrando nuevas ambigüedades u omisiones. El criterio de hasta dónde vale la pena el esfuerzo de desarrollar una tabla de decisión depende en parte de la confianza que uno tenga en que el árbol obtenido sea exhaustivo - que se ha cubierto completamente todo. Así tenemos

Pauta 2. Utilizar una tabla de decisión si duda que su árbol de decisión muestre la complejidad total del problema.

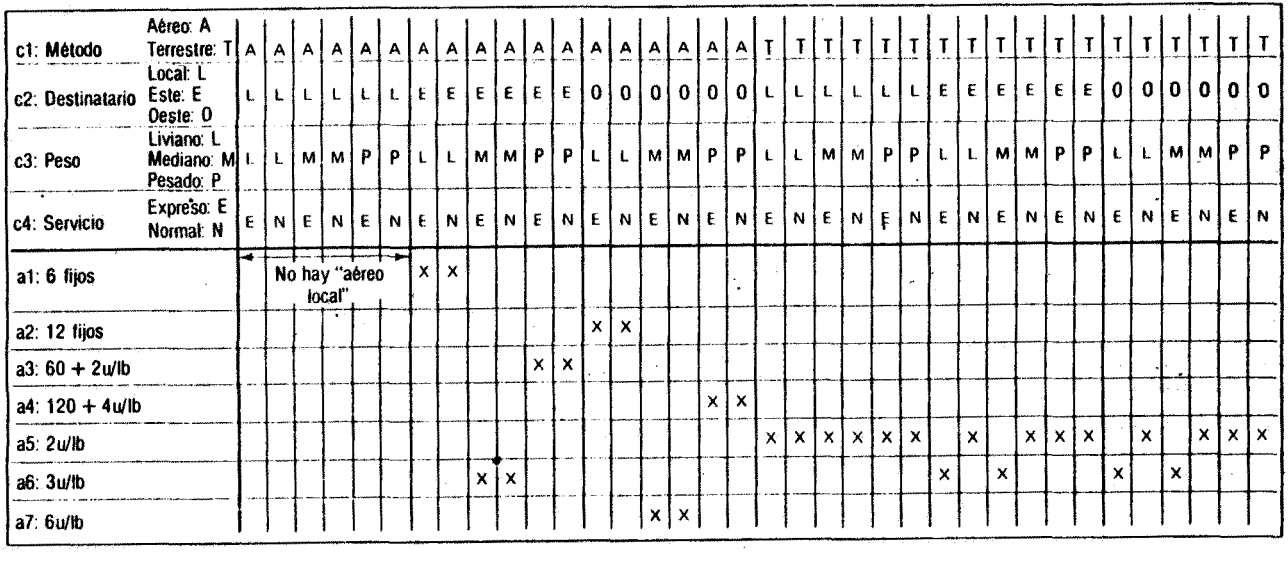


Figura 5.13 Tabla completa del problema de la tarifa del flete.

Posiblemente el factor más importante sea cuán vívida es la impresión que causa el árbol de decisión cuando se utilicen abreviaturas, es posible seguir las combinaciones de las condiciones de cada rama del árbol y "ver cómo encajan todas entre sí". Esto contrasta con la tabla de decisión donde el significado de cada regla debe ensamblarse mentalmente, condición por condición.

Pauta 3. Aun cuando se necesite una tabla de decisión para llegar al final de la lógica, termine presentándola como un árbol, si puede hacerlo sin violar la Pauta 1.

5.4 LENGUAJE ESTRUCTURADO, "PSEUDOCODIGO" Y "LENGUAJE COMPRIMIDO"

Los árboles de decisión y las tablas de decisión son las herramientas preferidas para tratar con los procesos ramificados complejos, tales como los que típicamente se encuentran en los cálculos de descuentos, cálculos de tarifas, comisiones de venta y cálculos de primas de productividad, procedimientos de control de inventarios, etc. Sin embargo, muchos de los procesos que debemos documentar no son tan complejos. Encontramos que incluyen operaciones ("Hacer esto, luego aquello. . ."), algunas decisiones y unos pocos lazos ("Repetir este proceso hasta. . ."). Esto no nos sorprende ya que sabemos que puede probarse [5.4] que cualquier programa se compone de una adecuada combinación de instrucciones paso a paso (como MOVER o SUMAR), decisiones binarias (SI-LUEGO-SI NO-ENTONCES), y lazos. Los procesos lógicos que estamos definiendo en el análisis estructurado son justamente esto, programas para ser ejecutados ya sea por empleados o por computadoras. Estas pocas estructuras proveen las bases para la programación estructurada, parte de cuya efectividad deriva de la simplificación y normalización resultante del uso de unas pocas estructuras, en lugar de la gran variedad permitida por el lenguaje de programación a utilizar. Si pudiéramos escribir nuestras especificaciones empleando un enfoque similar en lenguaje corriente, podríamos obtener algunos de esos mismos beneficios. Vamos a estudiar estas estructuras con un poco más de detalle para ver cómo las podemos incorporar al lenguaje.

5.4.1 Las "estructuras" de la programación estructurada

1. Instrucciones secuenciales. Esta estructura abarca cualquier instrucción o grupos de instrucciones que no tienen repetición o ramificación dentro de ellos, como por ejemplo,

"Multiplicar horas trabajadas por salario horario para obtener el pago bruto."
"Despachar libros a la dirección de destino."
"Sumar importe del flete a la factura."
"Calcular importe del flete. (El importe del flete es de 60 unidades por cada libra que exceda las 20.)"
"Tramitar la cesantía del empleado."
"Conceder el 25% de descuento."

A veces es necesario incluir descripciones o definiciones de los términos empleados, como en el caso precedente de "importe del flete". Lo que no tenemos que hacer es entrar a hurtadillas en cualquier ramificación o repetición escondida, como en

"Despachar libros a la dirección de embarque, o a la de facturación, la que sea de aplicación" (ramificación escondida).
"Continuar asignando espacio, una unidad por vez, y detenerse cuando todas las solicitudes se han satisfecho" (un lazo escondido, con una prueba clara para saber cuándo interrumpir el lazo).

Si tenemos un grupo de declaraciones verdaderamente secuenciales y las juntamos, el *bloque* resultante de declaraciones se verá como una sola declaración secuencial. Si creamos el siguiente grupo de declaraciones y las denominamos "calcular-deducciones,"

- Obtener remuneración bruta
- Obtener detalles de la remuneración acumulada a la fecha
- Calcular retenciones federales
- Calcular retenciones estatales o provinciales
- Calcular seguro social

y también creamos un grupo de declaraciones denominadas "emitir cheques de pago"

- Ingresar bruto y retenciones en el talón del cheque
- Ingresar remuneración neta en el cheque
- Firmar cheque
- Enviar por correo cheque y talón

podemos escribir legítimamente como instrucciones secuenciales

HACER (DO) calcular deducciones
HACER (DO) emitir cheque de pago

sin emplear ninguna decisión o lazo escondido. "HACER" (DO) significa "Llevar a cabo todas las instrucciones listadas bajo el siguiente nombre. . ."

2. *Instrucciones de decisión.* Como vimos, el formato normal para una decisión es la estructura

```
SI              Condición.1
  LUEGO         acción-A
SI NO           (no Condición-1)
  ENTONCES      acción-B
```

Cada acción puede ser un conjunto de instrucciones secuenciales, o un lazo, *u otra decisión.* Supongamos que se reemplaza "acción-A" por otra estructura "SI-LUEGO-SI NO-ENTONCES, obtendremos

```
SI              Condición-1
  LUEGO         SI              Condición-2
                  LUEGO         acción-C
                SI NO           (no Condición-2)
                  ENTONCES      acción-D
SI NO           (no condición-1)
  ENTONCES      acción-B
```

Un ejemplo de este caso podría ser

```
SI              necesita una vacación
  LUEGO         SI              puede pagarse una vacación
                  LUEGO         tómese una vacación
                SI NO           (no puede pagarse una vacación)
                  ENTONCES      pinte la casa
SI NO           no necesita una vacación
  ENTONCES siga trabajando
```

La oración "SI. . . LUEGO SI. . .," si bien es correcta en castellano, se lee un poco extraña. Como en los ejemplos anteriores del capítulo, preferimos escribir "SI. . . y-SI. . .," lo cual significa lo mismo pero se lee un poco más natural, y ahorra el LUEGO para indicar una acción.

Si puede sustituir un SI-LUEGO-SI NO-ENTONCES por una acción en un lugar, lo

puede hacer en cualquier parte. Podemos sustituir otro SI por "tome una vacación"; así

```
SI necesita una vacación
  y-SI puede pagarse una vacación
    y-SI tiene alguien con quien ir
      y-SI tiene algún lugar donde ir
        LUEGO...
        SI NO
```

Así es como aparecen los "SI anidados", a medida que la lógica que estamos representando se hace más compleja. Esta lógica anidada puede ser difícil de seguir, de manera que es importante usar las convenciones de alinear cada SI NO con el SI al que pertenece. Como una regla práctica, si se tiene más de tres SI anidados unos en otros, resulta más claro representar la lógica con un árbol de decisión, al costo de no poder trascribir completamente las condiciones.

2.1 Instrucciones de decisión "caso" (case). Un tipo especial de estructura de decisión aparece cuando hay varias posibilidades de una condición que *nunca se presenta en combinación;* por ejemplo, las mismas representan diferentes casos los cuales son mutuamente excluyentes.

Este tipo de estructura siempre puede representarse en una tabla simple, como por ejemplo,

Condición	Acción
La transacción es un pedido	Sumar a ventas-a-la-fecha
La transacción es una devolución	Restar de ventas-a-la-fecha
La transacción es un pago	Sumar a pagos-recibidos
La transacción es un reclamo	Pasar al Departamento Reclamos
La transacción es una cancelación	Restar de ventas-a-la-fecha

La faceta importante de esta situación es que la transacción debe ser de uno de estos cinco tipos y *solo de uno.* Por ejemplo, no puede ser a la vez un pedido y una cancelación.

Solamente se aplica un caso por vez. Esta estructura especial se conoce como estructura "caso" (case, en inglés) y se representa convencionalmente en lenguaje estructurado de la siguiente manera:

```
SI              caso-1
LUEGO SI        acción-1
SI NO, SI       caso-2
                acción-2
SI NO, SI       caso-3
                acción-3
SI NO...
```

Si aplicamos esta convención a nuestra estructura de transcción tendremos

```
SI              la transacción es un pedido
                sumar a ventas-a-la-fecha
SI NO, SI       la transacción es una devolución
                restar de ventas-a-la-fecha
SI NO, SI       la transacción es un pago
                sumar a contado recibido
SI NO, SI       la transacción es un reclamo
                pasar al Departamento Reclamos
```

SI NO, SI	la transacción es una cancelación restar de ventas-a-la-fecha
SI NO	(no es ninguno de los de arriba)
ENTONCES	dirigirse al supervisor

Obsérvese el último SI NO; se lo requiere a fin de que haya igual cantidad de SI y SI NO. Si se omite este "ninguno de los de arriba" o caso "defecto", existe la posibilidad de que aparezca en la práctica un caso que no se encuentre cubierto por la lógica especificada. El analista debe considerar este caso y proceder en consecuencia. Esta posibilidad puede parecer remota cuando los casos representen divisiones arbitrarias a lo largo de un continuo completo, como por ejemplo.

SI	la persona es menor de 18 años sumar al total de "menores"
SI NO, SI	la persona es de 19 o mayor pero menor de 35 sumar al total de "adultos jóvenes"
SI NO, SI	la persona es de 35 o mayor pero menor de 65 sumar al total de "adultos maduros"
SI NO, SI	la persona es de 65 o mayor sumar al total de "ciudadanos ancianos"
SI NO	(ninguno de los de arriba)

¿Qué puede representar posiblemente "(ninguno de los de arriba)" El analista debe permitir que el último SI NO quede "suspendido" en esta situación; le corresponde considerar al programador cómo debe proceder en un caso donde fallan todas las pruebas de edad (tal vez por que la edad haya sido codificada incorrectamente).

3. Instrucciones repetitivas (lazos). Esta estructura se aplica a cualquier situación en la cual una instrucción o grupo de instrucciones se repite hasta que se obtiene algún resultado deseado.

Para tomar un ejemplo muy simple, en que una factura cubre la venta de una cantidad de ítem, cada uno se listará en un renglón separado con la cantidad vendida y el precio unitario, como se ve en la Fig. 5.14.

Se podrá especificar la confección de la factura escribiendo

> Por cada línea de ítem, multiplicar la cantidad por el precio unitario, para obtener los totales de línea. Cuando se han considerado todas las líneas de ítem, sumar todos los totales de línea para obtener el total de la factura.

Más formalmente, podríamos especificar

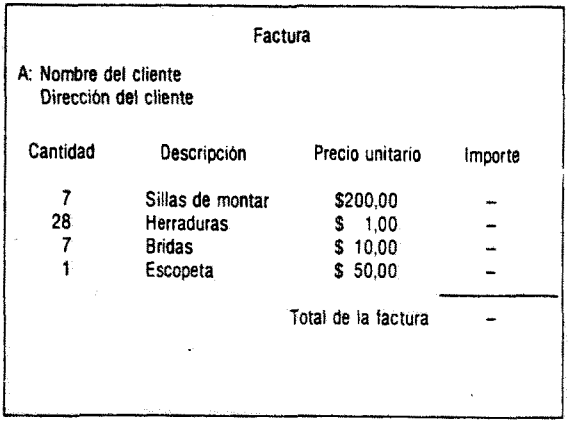
> "REPETIR EXTENDER-LINEA-ITEM HASTA QUE todas las líneas hayan sido resueltas"

y definir EXTENDER-LINEA-ITEM como

> "Multiplicar cantidad por precio unitario para obtener total de la línea"

Con solo una instrucción para construir el cuerpo del lazo, esto es como tomar una maza para partir una nuez, pero puede ser una estructura valiosa para utilizar cuando la situación requiere una cantidad de instrucciones a repetir una y otra vez como un grupo. Lo que es más importante aún, pone de relieve los dos aspectos que debemos especificar para definir una estructura de repetición:

1. Exactamente ¿qué es lo que debe repetirse?
2. ¿Cómo se sabe cuándo hay que parar?



Factura

A: Nombre del cliente
Dirección del cliente

Cantidad	Descripción	Precio unitario	Importe
7	Sillas de montar	$200,00	–
28	Herraduras	$ 1,00	–
7	Bridas	$ 10,00	–
1	Escopeta	$ 50,00	–
		Total de la factura	–

Figura 5.14

Mientras que el analista debe estar preparado para especificar repeticiones cuando ellas sean claramente parte de la función lógica, es más usual que el diseñador y el programador se encarguen de producir lazos a ser ejecutados por la máquina, debido a que el analista está relacionado primariamente con el procesamiento lógico de cada estructura de datos, y el diseñador/programador está vinculado con el procesamiento físico de una serie de estas estructuras. La definición de lazos se hace más importante a medida que nos acercamos más a la implementación física, como veremos en la Sec. 5.4.3.

5.4.2 Convenciones para el lenguaje estructurado

Como se indicó anteriormente, es posible escribir cualquier especificación lógica empleando las cuatro estructuras básicas que hemos expuesto. Cuando la lógica está escrita en idioma corriente, utilizando letras mayúsculas y sangrado, se la conoce como *lenguaje estructurado.* Podemos resumir las convenciones para el lenguaje estructurado de la siguiente forma:

1. La lógica de todos los procesos en un sistema se expresa como una combinación de estructuras secuenciales, de decisión, de caso y de repetición.
2. Deben observarse las reglas del castellano sin ambigüedades, tal como se indicó en la Sec. 5.1.1.
3. Las palabras clave SI, LUEGO, SI NO, ENTONCES, REPETIR y HASTA deberán escribirse con mayúscula y las estructuras deben sangrarse para mostrar su jerarquía lógica.
4. Los bloques de instrucciones pueden agruparse entre sí y debe dárseles un nombre significativo que describa su función, que también deberá escribirse con mayúsculas.
5. Cuando se utilice una palabra o frase que se encuentre definida en el diccionario de datos, dicha palabra o frase deberá subrayarse.

Para ilustrar la fortaleza y la debilidad del lenguaje estructurado tomaremos como ejemplo parte de la función 7, "Generar nota de embarque y factura" del diagrama de flujo de datos del Capítulo 3. La generación de la factura comprende tres pasos:

- El cálculo del total de cada ítem y de la factura
- El cálculo del descuento (si existe alguno)
- El cálculo del cargo por despacho y manipuleo (empleando una versión simplificada de la política que hemos examinado con el árbol de decisión y la tabla de decisión).

La Fig. 5.15 muestra una representación, en lenguaje estructurado, de la mayor parte de la lógica. Se efectuarán varias aclaraciones sobre la base de este ejemplo de lenguaje estructurado:

```
GENERAR FACTURA
    HACER (DO) CALCULAR-TOTAL-FACTURA
    HACER CALCULAR-DESCUENTO
    HACER CALCULAR-DESPACHO-MANIPULEO
    Restar descuento de total-factura para tener neto-factura
    Sumar cargo-despacho-manipuleo a neto-factura para tener total-a-pagar
    Escribir factura
CALCULAR-TOTAL-FACTURA
    REPETIR EXTENDER-LINEA-ITEM-HASTA que todas las líneas-ítem hayan sido
    extendidas
    Sumar todos los totales líneas-ítem para obtener total-factura

EXTENDER-LINEA-ITEM
    Multiplicar cantidad por costo-unitario para obtener total-línea-ítem

CALCULAR-DESCUENTO
    SI              factura total es MI$10.000
                    descuento es 5% de total-factura
    SI NO SI        factura-total es MI$250 pero mI $1.000
                    descuento es 2 1/2% de total-factura
    SI NO SI        factura-total es MI$100 pero mI $250
                    descuento es 1% de total-factura
    SI              (factura-total es mQ$100)
        ENTONCES    descuento es cero

CALCULAR-DESPACHO-MANIPULEO
    SI              pedido especifica despacho aéreo
      LUEGO HACER CALCULAR-FLETE-AEREO
    SI NO (pedido especifica despacho terrestre o no especifica el método)
      ENTONCES HACER CALCULAR-FLETE-TERRESTRE
    Multiplicar tarifa por valor-unitario-corriente para tener cargo-despacho-manipuleo

CALCULAR-FLETE-AEREO
    SI              peso es mI2
                    flete es 6 unidades
    SI NO SI        peso es mQ2 pero mI20
                    Multiplicar cada libra de peso por 3 unidades para obtener flete
    SI NO           (peso es MQ20)
        ENTONCES    Restar 20 del peso para obtener exceso
                    Multiplicar exceso por 2 unidades por libra y sumar 60 (20 libras a
                    3 unidades por libra) para obtener tarifa

CALCULAR-FLETE-TERRESTRE
    SI              destino es local
      y-SI            código-servicio es expreso
        LUEGO           Multiplicar cada libra de peso por 2 unidades para obtener
                        tarifa
    . . .
    . . .
    etc.
```

Figura 5.15 Lenguaje estructurado.

1. Tiene mucho de la precisión de un programa de computación, pero *no* es un programa de computación. No existe especificación para la lectura y grabación de los archivos físicos, ni ajuste de contadores o interruptores, ni diseño físico alguno. El procedimiento así escrito puede ser ejecutado perfectamente bien por un empleado, aunque se puede elegir presentar las instrucciones de diferente manera.

2. El procedimiento ha sido escrito como una jerarquía de "bloques" de instrucciones, como se muestra en la Fig. 5.16. Si se desea obtener un resumen del procedimiento solo hay que leer el bloque superior. Si se desean los detalles de cualquier parte específica, se debe leer solamente esa parte.

3. El subrayado de ítem definidos en el diccionario de datos pone de manifiesto el uso de términos que aún necesitan definición. En la Fig. 5.15 la condición "el destino es local" sin subrayar indica que aún no se ha realizado una definición estricta de "local". Si se hubiera realizado, habríamos hallado dicha condición expresada como "*destino-tipo* es *local*".

4. Sujetas a la necesidad de más definiciones, las instrucciones están completas. Ello incluye la descripción de la tarea paso a paso, a diferencia del árbol de decisión y la tabla de decisión, los cuales solo muestran la lógica de las ramificaciones respectivas.

5. La precisión e integridad del lenguaje estructurado se obtienen a costa de una presentación verbosa y poco usual, que define cada pequeño detalle para la persona que no está familiarizada con el procedimiento, pero que pronto resulta irritante para aquel que solo desea que le expongan los hechos básicos. El "lenguaje comprimido" es una derivación del lenguaje estructurado que trata este problema; lo discutiremos en la Sec. 5.4.4.

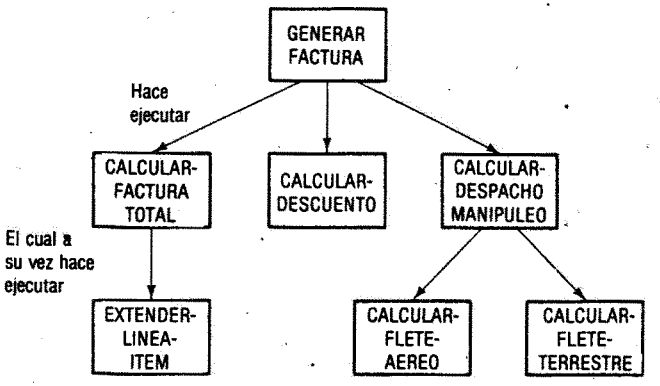


Figura. 5.16 Jerarquía de los bloques de funciones.

5.4.3 "Pseudocódigo"

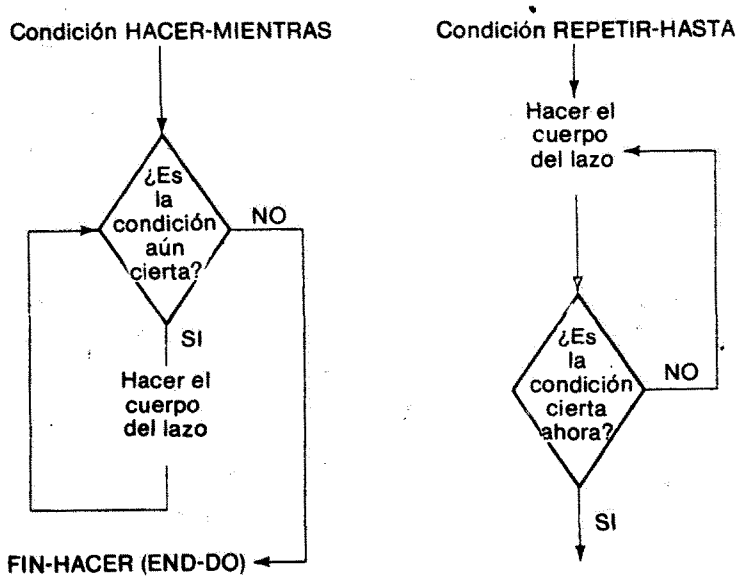
Hemos indicado que un problema definido en lenguaje estructurado no es un programa de computación. Sin embargo, una vez realizado el trabajo principal del diseño físico y definidos los archivos físicos, resultará muy conveniente poder especificar la lógica del programa físico utilizando las convenciones del lenguaje estructurado pero *sin llegar a la sintaxis en detalle de ningún lenguaje de programación en particular.* Esta notación "casi-código" se conoce como "pseudocódigo" [5.5] (en inglés, "pseudocode").

Para especificar un programa en "pseudocódigo" debemos ser capaces de manejar la inicialización y la terminación, leer y grabar los archivos, manejar fin de archivo y especificar contadores y señales. Así, si el diseñador físico ha decidido que GENERAR FACTURAS deberá ser un programa separado que lea un archivo llamado, digamos, EMBARQUES-DIARIOS, el "pseudocódigo" para el nivel superior de la jerarquía del programa será como el de la Fig. 5.17.

```
Inicializar el programa (abrir archivos, ajustar contadores)
Leer el primer registro-pedido
HACER-MIENTRAS haya más registros-pedido
        HACER-MIENTRAS haya más ítem en el pedido

                Calcular total-item
                Sumar total-item a total-factura

        FIN-HACER

        Calcular descuento
        Calcular cargo por despacho y manipuleo
        Calcular neto-factura, total-a-pagar
        Imprimir factura
        Grabar factura en archivo cuentas-a-cobrar
        Sumar detalle-factura a contadores resumen
        Leer próximo registro-pedido

FIN-HACER

Imprimir resumen de facturas del dia
Terminar programa
```

Figura 5.17 "Pseudocódigo" del nivel superior.

En "pseudocódigo" distinguimos la estructura del lazo "HACER-MIENTRAS" (DO-WHILE) respecto de la estructura del lazo "REPETIR-HASTA" (REPEAT-UNTIL), la cual tiene un significado especial. La estructura del lazo HACER-MIENTRAS implica que la prueba de terminación se hace antes de que se ejecute el cuerpo del lazo; la estructura REPETIR-HASTA implica que el cuerpo del lazo es ejecutado antes de hacer la prueba. En términos de diagrama de flujo estas estructuras son



La diferencia importante es que en una situación donde la condición es distinta de la esperada, la primera vez el lazo HACER-MIENTRAS no ejecuta el cuerpo de instrucciones, saltando directamente al FIN-HACER, mientras que el REPETIR-HASTA atravesará el cuerpo de instrucciones antes de efectuar la prueba y encontrar, por ejemplo, que no hay pedidos que procesar. En lenguaje estructurado, donde especificamos una lógica no física, no nos preocupa esta distinción. En el "pseudocódigo", donde estamos desarrollando la lógica del programa, debemos tener mucho cuidado.

El programa consiste de un lazo HACER-MIENTRAS dentro de otro lazo HACER-MIENTRAS, donde el lazo externo procesa cada registro de pedido. El diseñador ha especificado que después de la inicialización el programa deberá leer el primer registro del archivo de pedidos, de manera que en el caso que no existan pedidos en el archivo, la prueba de HACER-MIENTRAS fallará de entrada, el control saltará al FIN-HACER externo, no se imprimirá ningún resumen y el programa terminará normalmente.

Vemos, por lo tanto, que el "pseudocódigo" representa un diseño de programa muy detallado. Especialmente en un entorno donde todas las estructuras de datos y elementos de datos están definidos en un diccionario de datos legible por máquina, la tarea de traducir el "pseudocódigo" al COBOL o al BASIC-PLUS, digamos por ejemplo, es relativamente simple. A partir del lenguaje estructurado, en cambio, deberán tomarse varias decisiones importantes sobre el diseño físico antes de que pueda escribirse un programa, ya que el lenguaje estructurado representa solamente la lógica externa.

5.4.4 "Lenguaje comprimido" lógicamente

Comentábamos que las convenciones de lenguaje estructurado, al mismo tiempo que produce una exposición precisa y completa, abarca muchas palabras y una notación muy poco común, como para resultar ideal en una presentación a los usuarios. Esto nos lleva a preguntarnos si es posible obtener los beneficios del rigor del lenguaje estructurado, sin sus desventajas. Una vez que se comprenden las razones por las cuales el lenguaje estructurado elimina las ambigüedades y la falta de integridad, podemos desechar las partes molestas de la notación y escribir en un estilo equivalente de castellano, que sea "comprimido lógicamente", en el sentido de reflejar un análisis preciso y completo. La Fig. 5.8 muestra el proceso "generar facturas" escrito en "lenguaje comprimido".

Es admisible incluir árboles de decisión en una descripción en lenguaje comprimido, si se tiene la seguridad que será comprensible para cualquiera que utilice el documento. El lenguaje comprimido es familiar en su apariencia, y estructuralmente resulta equivalente al lenguaje estructurado. La ironía es que resulta difícil de escribir hasta tanto la lógica del proceso no haya sido resuelta en idioma estructurado y entendida en detalle en función de las cuatro estructuras (secuencia, decisión, caso y repetición).

Podemos resumir las convenciones del idioma comprimido como sigue:

1. Las operaciones secuenciales se presentan como instrucciones imperativas que se deben cumplir rutinariamente.

2. Las estructuras SI-LUEGO-SI NO-ENTONCES se representan con notación decimal y con sangría para mostrar cómo se anidan entre sí, por ejemplo

```
5.
    5.1
        5.1.1                    (
            5.1.1a
```

También pueden representarse como árboles de decisión.

3. Las condiciones SI NO se representan como "Para (explicación de la condición)".

4. Las estructuras de casos se representan como tablas.

Para generar una factura:

Paso 1: Calcular el total de la factura de la siguiente forma:

1.1 Tomar cada línea de la factura y multiplicar la cantidad del ítem por el precio unitario para obtener el total del ítem.

1.2 Sumar los totales de ítem para obtener el total de la factura.

Paso 2: Calcular el descuento (si hay alguno).

El valor del descuento está basado en el total de la factura de acuerdo con la siguiente tabla:

Factura total	% Descuento
Menor que $100	No
Mayor que $100 pero menor que $250	1%
Mayor que $250 pero menor que $1.000	2 1/2%
Mayor que $1.000	5%

Restar el descuento del total de la factura para obtener el importe neto de la factura.

Paso 3: Calcular el cargo por despacho y manipuleo (D&M).
(D&M) está basado en unidades, cada una cuesta actualmente 50 centavos.
Los pedidos especifican el flete aéreo o terrestre; cuando no se especifica flete, la omisión se considera flete terrestre.

3.1 Para los pedidos que especifican flete aéreo:

- Hasta 2 libras incluida: tarifa 6 unidades fijas.
- Más de 2 pero menor que 20: 3u/lb.
- 20 libras o más: 60 unidades + 2 unidades por cada libra que exceda las 20.

3.2 Para los pedidos que especifican flete terrestre (o que no especifican ningún método):

3.2.1 Para destinatarios locales:

a. Para servicio expreso: 2 u/lb
b. Para servicio normal: 1 u/lb

3.2.2 Para destinatarios externos al área local:

a. Para servicio expreso
   - peso mayor que 20 lb: 2 u/lb
   - peso menor o igual que 20 lb: 3 u/lb
b. Para servicio normal: 2 u/lb

Figura 5.18 Lenguaje comprimido.

5. Cuando se incluyen condiciones genuinas de excepción, la estructura (ver Sec. 5.1.1) "Acción-1 *salvo* condición *en cuyo caso* Acción-2" podrá utilizarse por razones de claridad.

Así cuando escribimos el resumen de la lógica para el proceso en el diccionario de datos (que repetimos más adelante) será suficiente emplear lenguaje comprimido con cláusulas "salvo" o "a menos que" (unless) para explicar el manejo de las excepciones. El lenguaje comprimido utilizado en "Verificar si crédito es OK" fue escrito como un resumen del lenguaje estructurado, el cual expresa todos los detalles de la lógica. La Fig. 5.19 muestra el lenguaje estructurado completo para "Verificar si crédito es OK."

VERIFICAR-CREDITO-OK | Proceso ref:

Descripción: Decidir adonde se embarcan los pedidos sin previo pago, o si debe requerirse al cliente pago previo.

Entradas	Resumen de lógica	Salidas
1-3 PEDIDOS D3-3 Historia de pago FECHA-APERTURA-CUENTA FACTURA * PAGO * BALANCE-EN-ORDEN	Recuperar historia de pago Si el cliente es nuevo, enviar pedido de pago previo. Si es cliente corriente (promedio de dos pedidos mensuales) OK el pedido, a menos que el balance esté vencido con más de 2 meses. Para clientes anteriores que no sean corrientes, OK los pedidos, a menos que tengan cualquier balance vencido.	3-C Pedido de pago previo [Recordatorio de balance] 3-D3 Nuevo balance en orden 3-6 Pedidos con crédito OK

Ref. física: Parte de la entrada del pedido en línea, OE 707

Detalles completos de eta lógica se pueden encontrar: Especificación funcional, Sección 7.2

Nombre del proceso: VERIFICAR SI CREDITO ES OK Ref:3

```
Por cada pedido:
    Recuperar          historia-de-pagos
    SI                 historia-de-pagos no se encuentra (nuevo cliente)
      LUEGO            generar requerimiento-de-prepago
    SI NO              (cliente existente)
      ENTONCES         calcular frecuencia-promedio-de-pedido
                           (cantidad promedio de pedidos por mes de los últimos
                           tres meses)
                       calcular saldo total vencido
                       SI      frecuencia-promedio-de-pedido es MQ 2.0
                               (cliente regular)
                         y-SI                antigüedad del saldo vencido más antiguo
                                             es mQ60 días
                           LUEGO             calificar pedido OK para crédito trasmitir
                                             flujo de datos 3-6
                                             (pedidos con crédito OK)
                         SI NO               (saldo vencido MI60 días)
                             ENTONCES        generar requerimiento-de-prepago más
                                             recordatorio-de-saldo-vencido
                       SI NO   (cliente no regular)
                         y-SI                saldo vencido es MQ cero
                           LUEGO             generar requerimiento-de-prepago
                         SI NO               (sin daldo deudor)
                             ENTONCES        calificar pedido OK para crédito trasmitir
                                             flujo de datos 3-6 (pedidos con crédito OK)
```

Figura 5.19 Lenguaje estructurado.

5.4.5 Pros y contras de las cuatro herramientas

La tabla 5.1 resume las fuerzas y debilidades relativas de las cuatro herramientas para lógica de procesos que hemos discutido en este capítulo. La tabla compara las cuatro herramientas lógicas en función de ocho consideraciones. Estas consideraciones pueden agruparse en tres áreas básicas:

1. ¿Cuán fáciles de usar son las herramientas para el analista? Se tabularon tres consideraciones: "verificación de la lógica", "representación (de la) estructura lógica" y "simplicidad". "Verificación de la lógica" describe la facilidad con la que cada herramienta asegura que se han cubierto todas las posibilidades lógicas. "Representación estructura lógica" describe hasta qué punto cada herramienta provee una representación gráfica de los cuatro tipos básicos de estructura (secuencia, decisión, lazo y caso). "Simplicidad" evalúa, para cada herramienta, la facilidad o dificultad para aprender cómo utilizarla.

2. ¿Cuán fáciles de usar son las herramientas para la comunidad de usuarios? Las evaluaciones de "verificación del usuario" se refieren a esto.

3. ¿Cuán fáciles de usar son las herramientas para el diseñador/programador? El encabezamiento "especificación del programa" describe la adaptabilidad de cada herramienta para este propósito. Las consideraciones de "legible por máquina" y "editable por máquina" describen la facilidad con que cada herramienta podrá ser procesada por un compaginador de texto y la facilidad con que podrá emplearse un paquete de "software" para verificar la consistencia y sintáxis de la lógica. "Cambiabilidad" describe la facilidad con la cual la herramienta permite cambiar la lógica debido, por ejemplo, a que se han modificado los requerimientos del usuario.

Tabla 5.1 Comparación de las herramientas para las estructuras de proceso.

Uso	Arboles de decisión	Tablas de decisión	Lenguaje estructurado	Lenguaje comprimido
Verificación de la lógica	Moderada	Muy buena	Buena	Moderada
Representación de la estructura lógica	Muy buena (pero solo decisión)	Moderada (solo decisión)	Buena (todo)	Moderada (todo, pero dependiendo del autor)
Simplicidad (fácilidad de uso)	Muy buena	Muy pobre a pobre	Moderada	Buena
Verificación del usuario	Buena	Pobre (a menos que el usuario esté capacitado)	Pobre a moderada (ligeramente jerga)	Buena
Especificación de programa	Moderada	Muy buena	Muy buena	Moderada
Legibilidad por máquina	Pobre	Muy buena	Muy buena	Muy buena
Validación por máquina	Pobre	Muy buena	Moderada (necesita sintáxis)	Pobre
Cambiabilidad	Moderada	Pobre (a menos que se trate de modificación de reglas sencillas)	Buena	Buena

Podemos resumir, a partir de la tabla, las situaciones más convenientes para utilizar cada herramienta:

Arboles de decisión se usan mejor para verificaciones de lógica o decisiones

moderadamente complejas de 10-15 acciones. Los árboles de decisión también son útiles para presentar a los usuarios la lógica de una tabla de decisión.

Tablas de decisión se usan mejor en problemas que involucran combinaciones complejas de hasta 5-6 condiciones. Las tablas de decisión pueden manejar cualquier cantidad de acciones; un gran número de combinaciones de condiciones puede hacer incómodo operar con tablas de decisión.

Lenguaje estructurado se usa mejor cuando el problema comprende la combinación de secuencias de acciones con decisiones o lazos.

Lenguaje comprimido se usa mejor en la presentación de una lógica moderadamente compleja, una vez que el analista está seguro de que no pueden aparecer ambigüedades.

5.4.6 ¿Quién hace qué?

A lo largo de este libro hemos distinguido los tres papeles de analista, diseñador y programador:

1. El analista establece las especificaciones funcionales lógicas.
2. El diseñador especifica cómo debe ensamblarse o armarse el sistema para satisfacer los requerimientos lógicos.
3. El programador implementa las especificaciones del diseñador.

Hemos dicho que estos papeles pueden ser desempeñados por una sola persona (el caso de un proyecto pequeño); por dos o tres personas, o por un equipo. El estilo de administración de proyectos de cada organización y la capacidad del personal disponible indicarán quién desempeña cada función en el proyecto. Uno de los puntos más discutidos en la división del trabajo es ¿Quién escribe la lógica en detalle? Tenemos claro que el papel del analista no incluye la escritura del "pseudocódigo"; esto es físico y deberá ser hecho por el diseñador o por el programador. ¿Pero corresponde al papel del analista escribir en lenguaje estructurado o deberá limitarse a los árboles de decisión y probablemente al lenguaje comprimido, dejando el análisis detallado de la lógica a los diseñadores/programadores? Por un lado, el analista es la persona que está en contacto más estrecho con las actividades de la organización y con los detalles de la lógica; por otra parte muchos programadores encuentran satisfacción en el trabajo de resolver el detalle de la lógica a partir de un claro pero conciso resumen lógico como el de la entrada del diccionario de datos; y en cambio cuando se les entregan definiciones de estructuras de datos y lenguaje estructurado, gritan "¡Me están convirtiendo en un mero codificador!"

Adoptamos la solución de compromiso, esto es, que el analista deberá estar entrenado y encontrarse cómodo con las tablas de decisión y el lenguaje estructurado, pero los deberá utilizar como herramientas de último recurso para poder llegar al fondo de un proceso complejo. El analista normalmente presentará la lógica del proceso al diseñador/programador en forma de árboles de decisión o lenguaje comprimido (las cuales, felizmente, son también las herramientas más útiles para comunicarse con los usuarios). *Si el diseñador o el programador los requiere,* el analista deberá estar preparado para proveer tablas de decisión o lenguaje estructurado. La prueba deberá ser siempre ¿es esta exposición lógica suficientemente clara como para poder escribir un programa a partir de ella?

BIBLIOGRAFIA

5.1 G. E. Whitehouse, *Systems Analysis and Design Using Network Techniques,* Prentice-Hall, Englewood Cliffs, N.J., 1973.

5.2 S. L. Pollack, H. T. Hicks, y W. J. Harrison, *Decision Tables: Theory and Practice,* New York, 1971.

5.3 K. R. London, *Decision Tables*, Auerbach, Philadelphia, Pa., 1972.
5.4 C. Böhm y G. Jacopini, "Flow Diagrams, Turing Machines, and Languages with Only Two Formulation Rules", *Communications of the ACM*, mayo de 1966.
5.5 P. Van Leer, "Top-down Development Using a Program Design Language", *IBM Systems Journal*, Vol. 15, Nº 2, 1976.

Ejercicios y puntos de discusión

1. Aplicar la prueba de indiferencia a la tabla de decisión extendida de la Fig. 5.13.
2. Convertir a lenguaje estructurado la tabla de decisión realizada en el ejercicio anterior.
3. La descripción siguiente corresponde al procedimiento seguido por el personal administrativo en el depósito de un distribuidor de repuestos eléctricos:

   Al recibirse un pedido del Departamento de Ventas verificar si cada ítem puede atenderse con el inventario corriente. Si se dispone de inventario suficiente, el empleado del depósito ajusta los registros de stock y trasfiere el ítem para su recolección y despacho. Cada vez que los registros de stock se modifican, el nuevo nivel de inventario se compara con el nivel de reposición que está asentado en la tarjeta de inventario del ítem. Si el nuevo nivel de inventario fuera menor que el nivel de reposición, el empleado del depósito llena un formulario de orden de compra, registra la cantidad pedida en la tarjeta de inventario del item y eleva el formulario de la orden de compra al Jefe de Compras para su aprobación y despacho al proveedor.

   Si hay existencia, pero insuficiente para atender el pedido (cumplimiento parcial), el empleado del depósito despacha los ítem disponibles, ajusta los registros de stock y confecciona un pedido pendiente de entrega por la cantidad requerida. El pedido pendiente de entrega se archiva por orden de número de parte, a la espera de la recepción de un embarque. Si el ítem ya fue solicitado, el empleado del depósito remite al Jefe de Compras un aviso de pronta entrega. Si el ítem no fue pedido, se prepara un formulario de orden de compra.

   Si no hay existencia del ítem, se confecciona un pedido pendiente de entrega, el cual se archiva de la forma ya indicada, y el empleado remite un aviso de pronta entrega al Jefe de Compras, esté o no pedido el ítem.

   Dibuje una tabla de decisión para indicar la lógica de este proceso. Cuando disponga de la tabla de decisión, presente la lógica en un árbol de decisión, y escríbala también en formato de lenguaje estructurado. Confeccione un conjunto de instrucciones, utilizando el lenguaje comprimido, para un empleado de depósito recientemente incorporado, y para verificar su claridad revíselas con un empleado administrativo.
4. Tome un programa que tenga una lógica medianamente compleja, y escriba la sección lógica en "pseudocódigo". ¿Esto le ayuda a comprender el programa?
5. Existe una cantidad de paquetes de "software" que convierten tablas de decisión directamente a codificación. A su juicio ¿por qué estos paquetes no tienen hoy un uso más amplio?
6. Si puede utilizar un software de procesamiento de tablas de decisión convierta la tabla de decisión de la tarifa del flete desarrollada en el Ejercicio 1 a codificación COBOL. Compare la salida con el lenguaje estructurado realizado en el Ejercicio 2.
7. La próxima vez que tenga que escribir un memorándum que contenga la palabra "si", redáctelo primero en lenguaje estructurado y luego conviértalo a lenguaje comprimido.
8. Si su empresa tuviera un plan de jubilaciones, analice la descripción de la política que especifica el monto de la jubilación, recurriendo a las herramientas lógicas descritas en este capítulo. ¿Cuál es la representación más fácil de entender?

6

DEFINIR EL CONTENIDO DE LOS ALMACENAMIENTOS DE DATOS

6.1 LO QUE SALE DEBE ENTRAR

Como se indicó en el Capítulo 2, una vez que hemos especificado las estructuras de los datos en los flujos de datos que salen de un almacenamiento de datos, sabemos cuál deberá ser el contenido mínimo de ese almacenamiento de datos. Para que un elemento de datos pueda ser extraído, en primer lugar deberá ser introducido en el almacenamiento. Podemos pues examinar los detalles de los flujos de datos que ingresan al almacenamiento de datos para controlar que se almacenen todos los elementos necesarios y para observar si algún dato a almacenar no va a ser usado nunca.

La Fig. 6.1 muestra parte de un flujo de datos de un sistema de personal.

Podemos ver, a partir de la naturaleza de los flujos de datos y los procesos relacionados que el almacenamiento de datos D5: EMPLEADOS-DETALLES debe contener la información de todos los empleados, sus nombres, direcciones y salarios. ¿Pero qué significa ésto en detalle? Podemos buscar cada uno de los cinco flujos de datos en el diccionario de datos para encontrar sus estructuras de datos componentes y buscar las estructuras de datos para hallar sus contenidos. Luego podemos comparar los contenidos de los flujos que entran en los almacenamientos de datos con los flujos que salen de los mismos, los que podrán ser como se muestra en la Fig. 6.2.

El próximo paso es hacer un borrador con el contenido de D5, basado en nuestro análisis de flujos de entrada y salida. Un primer intento podrá ser el que se muestra en la Fig. 6.3.

Comparando los flujos de salida con los de entrada, observamos los siguientes detalles:

1. EMPLEADO-DIRECCION claramente requiere detalles de la dirección actual, mientras que DIRECCION-CAMBIO implica que la dirección anterior también está almacenada. ¿Es necesario? ¿Alguna vez se desea saber la dirección anterior del empleado? No en los flujos de salida listados aquí, por lo menos.
2. EMPLEO-HISTORIA, por otro lado, requiere un listado de todos los distintos salarios percibidos por un empleado a través de su carrera, de manera que necesitaremos almacenar una cantidad variable de pares de salarios y fechas, y no solamente SALARIO-ANTERIOR y SALARIO-ACTUAL, como requiere la estructura de SALARIO-CAMBIOS.
3. EMPLEO-HISTORIA requiere un almacenamiento de datos similar a SALARIO-HISTORIA. ¡Pero no tenemos definido el flujo de datos para introducir cambios en TAREA-DENOMINACION! Según se ve en la Fig. 6.2, un nuevo empleado tiene una TAREA-DENOMINACION cuando ingresa y nunca la va a cambiar. Comparando los flujos de entrada y salida aparece una omisión realmente seria en el diagrama de flujo de

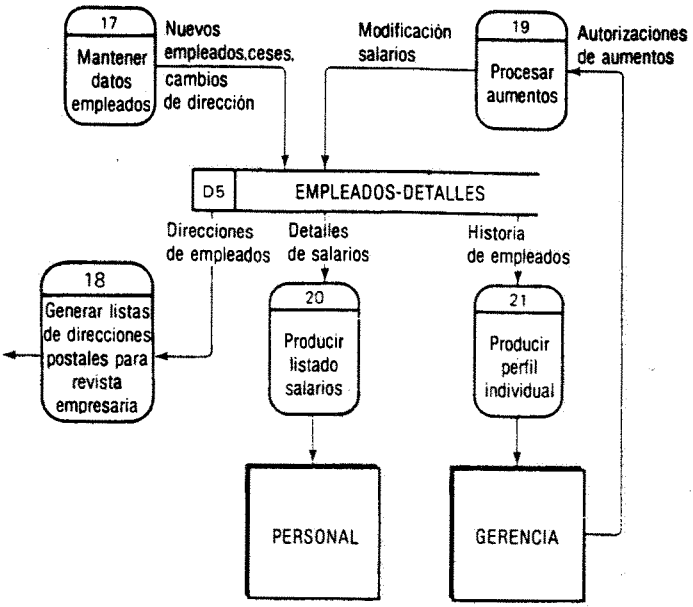


Figura 6.1 Diagrama de flujo parcial de un sistema de personal.

Estructura de datos contenida en:

Flujos entrantes a D5

NUEVO-EMPLEADO (17-D5)
- FECHA-INGRESO
- NOMBRE
- EMPLEADO-Nº
- DIRECCION
- TAREA-DENOMINACION
- SALARIO-INICIAL

CESANTIAS (17-D5)
- NOMBRE
- EMPLEADO-Nº

DIRECCION-CAMBIO (17-D5)
- NOMBRE
- EMPLEADO-Nº
- DIRECCIO*N-ANTERIOR
- DIRECCION-ACTUAL

SALARIO-MODIFICACION (19-D5)
- NOMBRE
- EMPLEADO-Nº
- SALARIO-ANTERIOR
- SALARIO-ACTUAL
- FECHA-EFECTIVA

Flujos salientes de D5

EMPLEADO-DIRECCION (D5-D18)
- NOMBRE
- DIRECCION

SALARIO-DETALLES (D5-D20)
- NOMBRE
- EMPLEADO-Nº
- SALARIO-ACTUAL

EMPLEADO-HISTORIA (D5-D21)
- NOMBRE
- EMPLEADO-Nº
- FECHA-INGRESO
- TAREA-HISTORIA*
  - TAREA-DENOMINACION
  - FECHA-EFECTIVA
- SALARIO-HISTORIA*
  - SALARIO
  - FECHA-EFECTIVA
  - [REVISION-RESUMEN]

} Para todos los empleados de la organización

Figura 6.2 Flujos entrantes y salientes de D5.

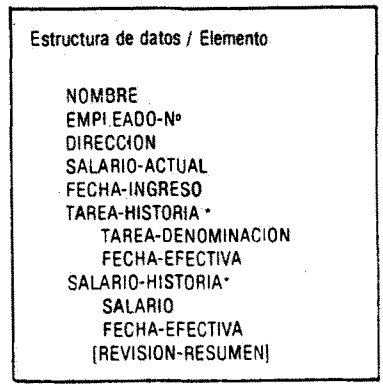


Figura 6.3 Estructura de D5 (primer borrador).

datos, que debemos corregir inmediatamente, junto con algunas redundancias menores en algunas estructuras de datos. Un comentario similar se aplica a REVISION-RESUMEN.

Para ver la realidad que reflejan las estructuras de datos, vamos a indicar algunos ejemplos del contenido, como se muestra en la Fig. 6.4. Examinando las historias de empleo de estas tres personas, vemos que un cambio de tarea normalmente (pero no siempre) implica un cambio de salario. Como ejemplo de una excepción a esta regla, Jones fue ascendido a supervisor el 08/01/76,sin aumento. Observamos también que cada empleado es objeto de una revisión en el aniversario de haber sido contratado y que el resultado de la misma se expresa como un único valor (una evaluación compuesta en una escala 0-5).

Contenidos de muestra de D5: EMPLEADO-DETALLES

NOMBRE	EMPLEADO NUMERO	DIRECCION	SALARIO ACTUAL	FECHA INGRESO	TAREA-HISTORIA TAREA	TAREA-HISTORIA DENOM.	SALARIO-HISTORIA SALAR.	SALARIO-HISTORIA FECHA-EFEC.	SALARIO-HISTORIA REV.
					FECHA	EFECTIVA			RES.
John P. Jones	26622	15 Corona Dr New York NY 10077	18200	12/01/74	Supervisor Prog. "sen". Programador	08/01/76 04/15/75 12/01/74	18200 15250 14500 12750	12/01/76 12/01/75 04/15/75 12/01/74	4.0 3.5
Mary A Worth	30604	2221 W 54 Paterson NJ 07070	21250	06/15/73	Gerente Jefe Turno Operador Mecanógrafo	11/01/75 12/15/74 07/01/74 06/15/73	21250 19000 16500 14000 9500 8500	06/15/76 11/01/75 06/15/75 12/15/74 06/15/74 06/15/73	4.5 4.8 4.5
Edward P Dullson	20927	491 East Highway New York NY 10066	8500	01/01/75	Empleado de correo	01/01/75	8500 8250	01/01/76 01/01/75	2.8

Figura 6.4 Ejemplo del contenido de D5: EMPLEADO-DETALLE.

6.2 SIMPLIFICACION DEL CONTENIDO DEL ALMACENAMIENTO DE DATOS MEDIANTE INSPECCION

¿Puede simplificarse este primer borrador de la estructura? Primero veamos qué podemos hacer aplicando solo el sentido común. Podemos observar de inmediato alguna duplicación de datos. SALARIO-ACTUAL será siempre el valor más reciente de SALARIO-HISTORIA. ¿Entonces, por qué debemos almacenarlo como elemento de datos separado? En forma similar, FECHA-INGRESO siempre será el dato más temprano, tanto en TAREA-HISTORIA como en SALARIO-HISTORIA, ¿entonces, por qué conservarlo en forma separada? Esta observación sugiere una simplificación muy importante: como varios datos en TAREA-HISTORIA y SALARIO-HISTORIA son los mismos, podemos hacer una estructura de datos compuesta en base a TAREA-HISTORIA y SALARIO-HISTORIA, llamada SALARIO-TAREA-HISTORIA y hacerle una entrada todas las veces que cambie la tarea o el salario. La estructura puede definirse como

SALARIO-TAREA-HISTORIA*
    FECHA-DE-CAMBIO
    TAREA-DENOMINACION
    [SALARIO-RESUMEN]

y para el señor Jones, su contenido será

Fecha-Del-Cambio	Tarea-Denominación	Salario	Revisión-Resumen
12/01/76	Supervisor	18200	4,0
08/01/76	Supervisor	15250	
12/01/75	Programador "Senior"	15250	3,5
04/05/75	Programador "Senior"	14500	
12/01/74	Programador	12750	

Es justo decir que esta simplificación de la estructura se hace al precio de una lógica ligeramente más compleja en algunos procesos subsiguientes. Por ejemplo, en el proceso 20 "Producir listado salarios" debemos hacer algo más que recuperar nombre, número de empleado y salario actual (como vimos en el primer borrador). Se deberá buscar la entrada más reciente del grupo repetitivo SALARIO-TAREA-HISTORIA y extraer de allí el salario actual. Sin embargo, lo que perdemos en movimientos, lo ahorraremos con creces en rodeos o desvíos. Volviendo a la pequeña lógica adicional del proceso hemos obtenido tres ventajas:

1. Tenemos un almacenamiento de datos más simple y pequeño.
2. Cada vez que cambie SALARIO-ACTUAL, solo se tiene que registrar en un solo lugar en lugar de dos, como se ve en el borrador original. Esto reduce la posibilidad de que las dos entradas se desfasen, ya sea porque alguien se olvidó de realizar ambos cambios (en un archivo manual) o por que un error del hardware haga que una se actualice y la otra no. Hacemos notar como un principio general que la *redundancia* (duplicación, triplicación, etc.) *aumenta* el *riesgo de error.*
3. Tenemos mayor seguridad contra el cambio. Consideremos el proceso 21 "Producir perfil individual". La estructura de los flujos de datos de la Fig. 6.2 implica que el usuario quiere por separado los informes de la historia de tareas y la historia de salarios; por eso se ve el primer borrador del almacenamiento de datos de esa manera. ¿Pero qué sucede si el usuario cambia de idea y quiere un listado combinado con los cambios de tarea y de salario ordenados por fecha? Con la estructura original, la lógica del proceso deberá hacer la combinación y clasificación; con la nueva estructura simplificada, el procesamiento será casi trivial.

Tanto para los sistemas manuales como para los computarizados comúnmente es *más fácil y más barato modificar la lógica de un proceso que modificar la estructura de un almacenamiento de datos.* En consecuencia, *cuanto más simple y más general sea la estructura* de un almacenamiento de datos, tanto *más fácil y más barato* resultará hacer cambios en los datos.

6.3 SIMPLIFICACION DEL CONTENIDO DEL ALMACENAMIENTO DE DATOS MEDIANTE LA NORMALIZACION

En la sección anterior hemos simplificado el contenido del almacenamiento de datos buscando y eliminando los elementos de datos redundantes. Se puede obtener un nivel adicional de simplificación reorganizando el contenido para eliminar los grupos repetitivos, proceso que se conoce con el nombre de *normalización.*

Después de eliminar la redundancia, el contenido de D5 tiene la estructura

NOMBRE
EMPLEADO N°
DIRECCION
SALARIO-TAREA-HISTORIA*
    FECHA-DEL-CAMBIO
TAREA-DENOMINACION
SALARIO
[REVISION-RESUMEN]

¿Cómo podemos librarnos del grupo repetitivo SALARIO-TAREA-HISTORIA? El único camino es dividir la estructura en dos, las cuales serán, ambas, más simples. Se llega así a una estructura que contiene nombre y dirección (lo que ocurre solo una vez por cada empleado y a otra que contiene cada cambio de tarea o salario (lo que puede ocurrir varias veces por empleado). Cada estructura deberá contener EMPLEADO-N°, el único elemento de datos que especifica unívocamente a cada empleado. Ver Fig. 6.5.

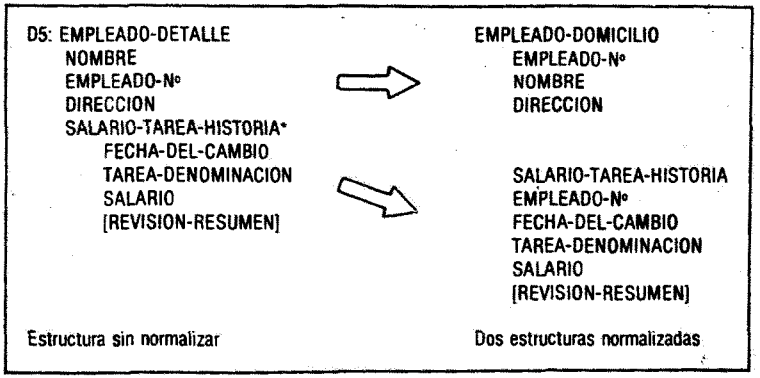


Figura 6.5 Normalización.

La Fig. 6.6 muestra cómo se presenta ahora el contenido del almacenamiento de datos de los tres empleados que se vieron en la Fig. 6.4. Se invita al lector a verificar que todos los flujos de datos listados en la Fig. 6.2 como salientes del almacenamiento de datos, pueden obtenerse en base a las estructuras de la Fig. 6.6, mediante una apropiada selección de los elementos de datos.

EMPLEADO-DOMICILIO

(EMPLEADO-Nº,	NOMBRE,	DIRECCION)
20927	Ed Dullson	492 E. Highway, New York, NY 10066
26622	John P. Jones	15 Corona Dr, New York, NY 10077
30604	Mary A. Worth	2221 W 54, Peterson, NJ 07070

SALARIO-TAREA-HISTORIA

(EMPLEADO-Nº, FECHA-DEL-CAMBIO, TAREA-DENOMINACION, SALARIO, REVISION-RESUMEN)

20927	01/01/76	Empleado de correos	8500	2.8
20927	01/01/75	Empleado de correos	8250	–
26622	12/01/76	Supervisor	18200	4.0
26622	08/01/76	Supervisor	15250	–
26622	12/01/75	Prog. sen.	15250	3.5
26622	04/15/75	Prog. sen.	14500	–
26622	12/01/74	Programador	12750	–
30604	06/15/76	Gerente	21250	4.5
30604	11/01/75	Gerente	19000	–
30604	06/15/75	Jefe Turno	16500	4.8
30605	12/15/74	Jefe Turno	14000	–
30604	07/01/74	Operador	9500	–
30604	06/15/74	Mecanógrafo	9500	4.5
30604	06/15/73	Mecanógrafo	8500	–

Figura 6.6 Contenido de las estructuras normalizadas de D5: EMPLEADO-DETALLES.

6.3.1 El vocabulario para la normalización

Los conceptos y técnicas de normalización han sido desarrollados por el Dr. E. F. Codd de IBM [6.1, 6.2] y otros, empleando términos de la matemática de los conjuntos. Por esta razón, el lector podrá encontrar algunos términos diferentes de aquéllos que hemos estado utilizando. Si bien este distinto vocabulario es, estrictamente hablando, innecesario, el planteo de la normalización se hará probablemente más y maś importante en los próximos años, a medida que las bases de datos se vuelvan más complejas y la simplicidad se haga más vital.

Vamos a explicar los términos que utilizaremos en el resto de este capítulo y en el siguiente, colocando el término equivalente más usual entre paréntesis, cuando sea de interés. Cada término está explicado en el glosario que se encuentra al final del libro. Una discusión más técnica se encuentra en Martin [6.3] y en Date [6.4].

En lugar de nuestro término *estructura de datos* se emplea la palabra *relación,* en el sentido que una estructura de datos expresa una relación entre elementos de datos. En lugar de nuestro término *elemento de datos* se utiliza el término *dominio,* significando el rango de valores que puede tomar un elemento de datos. Cada registro individual se denomina "tupla" ("tuple", en inglés) por analogía con "dupla". Una "tupla" (registro) con dos dominios (elementos de datos) se denomina 2-tupla; una "tupla" como "26622, 12/01/75, prog. sen., 15250, 3,5" con cinco dominios es una 5-tupla. Las relaciones (estructuras de datos) de los cuales forman parte estas tupias se denominan relaciones de grado 2 y de grado 5,

respectivamenente. La Fig. 6.7 ilustra estos términos. La relación es del grado 5, aun cuando en alguna "tupla" el valor del dominio REVISION-RESUMEN sea nulo.

Cada "tupla" en una relación (como cada registro en un archivo) debe tener una única clave mediante la cual puede identificarse. ¿Cuál es la clave en la relación SALARIO-TAREA-HISTORIA de la Fig. 6.7? EMPLEADO-Nº no es suficiente, ya que por la misma naturaleza de la forma de armar la relación habrá probablemente varias "tuplas" por empleado. Se puede formar una única clave *concatenando* (encadenando entre sí) EMPLEADO-Nº y FECHA-DEL-CAMBIO. Se podrá decir "Deme el registro del estado del empleado 30604, efectivo desde 11/01/75" y estar seguro de recuperar una sola "tupla". Es convencional mostrar la estructura y clave de las relaciones escribiendo

SALARIO-TAREA-HISTORIA (EMPLEADO-Nº, FECHA-DEL-CAMBIO
TAREA-DENOMINACION, SALARIO, REVISION-RESUMEN)

EMPLEADO-DOMICILIO (EMPLEADO-Nº, NOMBRE, DIRECCION)

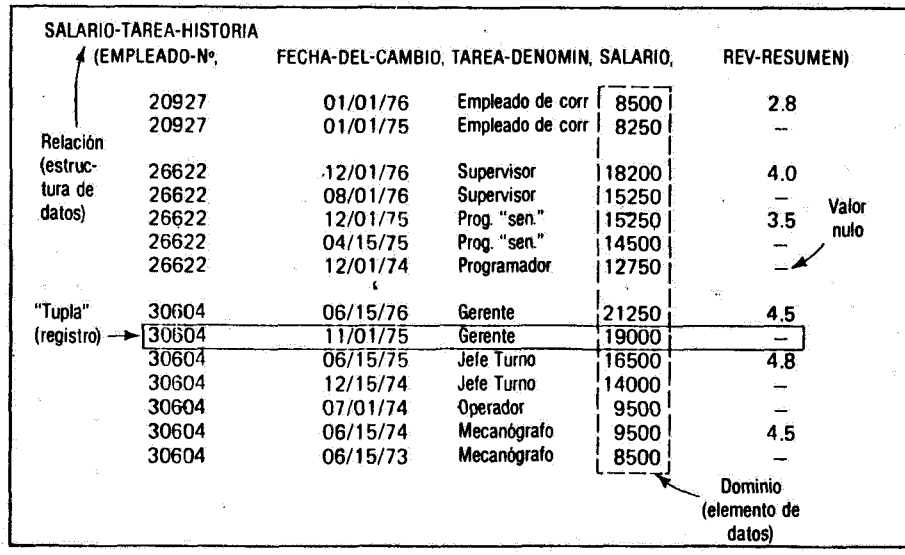


Figura 6.7 Términos de la normalización.

1. ¿Puede suprimirse cualquier elemento de datos, y a pesar de ello, la clave remanente seguir siendo unívoca para cada "tupla"?
2. ¿Existe alguna situación previsible en la cual esta clave candidata puede no ser unívoca?
3. ¿Alguna parte de la clave candidata tiene valores indefinidos?
4. De las claves candidatas remanentes, ¿cuál tiene menos dominios (elementos de datos)?

Figura 6.8 Pruebas para verificar si una clave candidata puede elegirse como clave principal.

Algunas veces no resulta obvio cuál puede ser la clave de la relación. Por ejemplo, a primera vista se puede pensar que otra *clave candidata* posible sería la concatenación de PERSONAL-NO, TAREA-DENOMINACION, Y SALARIO. En la Fig. 6.7 no tenemos duplicaciones de estas tres tomadas en conjunto. Pero EMPLEADO-Nº/TAREA-DENOMINACION/SALARIO sería una pobre elección como clave, debido a la posibilidad de que a un empleado pueda dársele una bonificación sin un aumento o una promoción, creando dos "tuplas" con el mismo valor de la clave. Elegimos como *clave principal* de la relación a la que tenga la menor cantidad de elementos de datos posible (obviamente el ideal es un elemento de datos) y con elementos de datos que no tengan valores indefinidos. (El criterio "no tengan valores indefinidos" excluirá a REVISION-RESUMEN de participar como clave candidata debido a que tiene el valor cero en varias "tuplas").

La Fig. 6.8 resume las preguntas que deben hacerse a una clave candidata en una relación.

6.4 ALGUNAS FORMAS NORMALIZADAS SON MAS SIMPLES QUE OTRAS

Codd estableció que existen tres tipos de relaciones normalizadas denominadas en orden creciente de simplicidad, primera forma normal, segunda forma normal y tercera forma normal. Definiremos cada una de ellas y discutiremos cómo reducir las relaciones a la forma más simple, la tercera normal.

6.4.1 Primera forma normal (1FN)

Cualquier relación normalizada (una estructura de datos sin grupos repetitivos) está automáticamente en 1FN, no importa cuan compleja sea su clave o qué interrelaciones deban existir entre los elementos de datos componentes. Las relaciones en la primera forma normal pueden tener dos tipos de complejidad:

1. Si la clave principal es concatenada, algunos de los dominios no-clave pueden depender de una sola parte de la clave, y no de la clave completa.
2. Algunos de los dominios no-clave pueden estar interrelacionados.

Esto es fácil de decir pero difícil de visualizar. Supongamos que estamos tratando de producir una versión normalizada de la estructura PEDIDOS, que describe la compra de libros a la Compañía CBM. Definiremos que un pedido consiste en el nombre del cliente, la fecha del pedido, el ISBN (código internacional) del libro pedido, el título, el autor, la cantidad de este título que ha sido pedida, y el importe total del pedido para un libro determinado. Podemos crear una relación normalizada:

LIBRO-PEDIDO (CLIENTE-NOMBRE, PEDIDO-FECHA, ISBN, TITULO, AUTOR
CANTIDAD, PRECIO, PEDIDO-TOTAL)

el subrayado indica que hemos elegido la clave concatenada CLIENTE-NOMBRE/ORDEN-FECHA/ISBN para identificar unívocamente cada pedido, haciendo una suposición razonable de que los clientes nunca piden el mismo libro dos veces el mismo día. Como hemos indicado, esta relación es de la *primera forma normal* en virtud de que no contiene grupos repetitivos.

La primera complejidad que quisiéramos suprimir es el hecho de que varios de los dominios no-clave (TITULO, AUTOR y PRECIO) pueden ser identificados con solo parte de la clave, el ISBN. El nombre del cliente y la fecha del pedido no tienen importancia a ese efecto. En otras palabras, si se da el ISBN, se sabe el TITULO, AUTOR y PRECIO sin

importar CLIENTE-NOMBRE y PEDIDO-FECHA. Con respecto a CANTIDAD, la situación es diferente: para conocer la cantidad de cualquier pedido en particular, se deberán conocer los tres dominios que están concatenados para formar la clave.

En el vocabulario de normalización CANTIDAD es *función completamente dependiente* de la clave concatenada íntegra. Por otra parte, TITULO no es función completamente dependiente, ya que solo se necesita saber parte de la clave (ISBN) para conocer el TITULO. Un dominio es función completamente dependiente si es dependiente de toda la clave. Es función dependiente, pero no completamente, si el valor del dominio puede ser determinado a partir de una parte de la clave. Este concepto nos permite definir la segunda forma normal.

6.4.2 Segunda forma normal (2FN)

Una relación normalizada está en la segunda forma normal si *todos* los dominios no-clave son funciones completamente dependientes de la clave principal. LIBRO-PEDIDO aunque es 1FN no es 2FN en la forma en que se encuentra. Para que LIBRO-PEDIDO sea 2FN debe librarse de la dependencia funcional parcial. Esto puede hacerse sacando los dominios que describen el libro y poniéndolos en una relación separada:

PEDIDO (CLIENTE-NOMBRE, ORDEN-FECHA, ISBN, CANTIDAD, ORDEN-TOTAL)
LIBRO ISBN, TITULO, AUTOR, PRECIO)

PEDIDO está ahora en *segunda forma normal;* cada uno de los dominios no-clave (CANTIDAD, PEDIDO-TOTAL) solo pueden especificarse si se conoce completamente la clave concatenada; es decir, todos los dominios no-clave son funciones completamente dependientes de la clave principal.

Podemos simplificar la situación más aún, porque PEDIDO todavía tiene una complejidad escondida dentro de ella; CANTIDAD y PEDIDO-TOTAL no son mutuamente independientes. Para cualquier PRECIO dado, la CANTIDAD determina el PEDIDO-TOTAL. Luego PEDIDO-TOTAL es función dependiente de CANTIDAD.

6.4.3 Tercera forma normal (3FN)

Una relación normalizada está en la tercera forma normal si

1. Todos los dominios no-clave son funciones *totalmente* dependientes de la clave principal y también
2. Ningún dominio no-clave es función dependiente de cualquier otro dominio no-clave.

Así, para transformar una relación 2FN a una relación 3FN se deben examinar cada uno de los dominios no-clave para ver si son independientes de cada uno de los otros dominios no-clave y suprimir cualquier dependencia mutua. En el caso de PEDIDO vemos que PEDIDO-TOTAL es de hecho un elemento de datos redundante, debido a que puede ser calculado. Así, podemos suprimirlo totalmente y expresar cada pedido por medio de dos relaciones, ambas en 3FN:

LIBRO-PEDIDO (CLIENTE-NOMBRE; PEDIDO-FECHA, ISBN, CANTIDAD)
LIBRO ISBN, TITULO, AUTOR, PRECIO)

Algunas veces solo podemos obtener la relación 3FN expresando la dependencia entre los dominios no-clave como una relación separada. Supongamos que tenemos una relación como la siguiente:

PROYECTO-ASIGNAR (EMPLEADO-Nº, TELEFONO, SUELDO-POR-HORA; PROYECTO-NO, FECHA-FINALIZACION)

La relación se utiliza para proveer la asignación de los empleados temporarios a los proyectos.

¿Está en 1FN? Si, es una relación normalizada.

¿Está en 2FN? Sí, ya que, puesto que la clave principal tiene un solo dominio, los dominios no-clave deberán ser funciones completamente dependientes de la misma.

¿Está en 3FN? No, existe una dependencia entre PROYECTO-NO y FECHA-FINALIZACION: Si se conoce PROYECTO-NO se conoce la FECHA-FINALIZACION del proyecto.

Podemos expresar PROYECTO-ASIGNAR en 3FN si la dividimos en dos relaciones:

TEMPORARIO (EMPLEADO-Nº, TELEFONO, SUELDO-POR HORA, PROYECTO-Nº)
PROYECTO (PROYECTO-Nº, FECHA-FINALIZACION)

La Fig. 6.9 resume las pruebas y pasos para trasformar relaciones no normalizadas en relaciones 3FN.

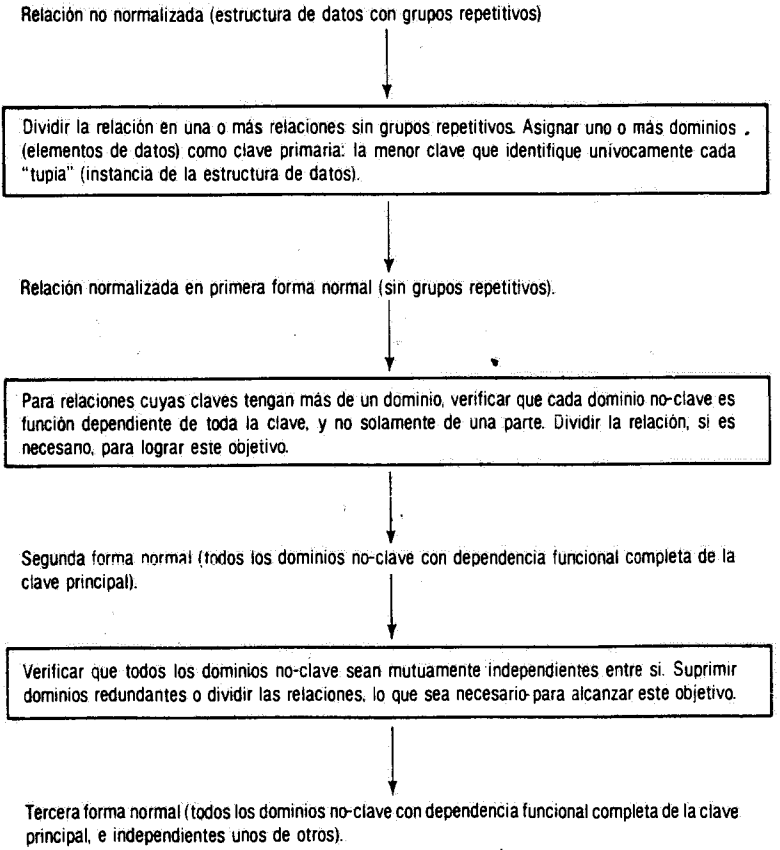


Figura 6.9 Transformación de relaciones sin normalizar en relaciones.

Como ejemplo de algunas relaciones 3FN, consideramos EMPLEADO-DIRECCION y SALARIO-TAREA-HISTORIA, reproducidos en la Fig. 6.10. Se invita al lector a aplicar las pruebas y verificar que ambas relaciones están en tercera forma normal.

Como cuestión práctica, las relaciones 1FN y 2FN tienden a aparecer cuando tratamos de describir dos hechos reales subyacentes en la misma relación, como en el caso de PROYECTO-ASIGNAR y PEDIDO. Las relaciones más simples describen un aspecto del mundo real por vez.

SALARIO-TAREA-HISTORIA

(EMPLEADO Nº	FECHA-DEL-CAMBIO,	TAREA-DENOM,	SALARIO,	REV-RESUMEN)
20927	01/01/76	Emplead. correo	8500	2.8
20927	01/01/75	Emplead. correo	8250	–
26622	12/01/76	Supervisor	18200	4.0
26622	08/01/76	Supervisor	15250	–
26622	12/01/75	Progr. "sen".	15250	3.5
26622	04/15/75	Progr. "sen".	14500	–
26622	12/01/74	Programador	12750	–
30604	06/15/76	Gerente	21250	4.5
30604	11/01/75	Gerente	19000	–
30604	06/15/75	Jefe Turno	16500	4.8
30604	12/15/74	Jefe Turno	14000	–
30604	07/01/74	Operador	9500	–
30604	06/15/74	Mecanógrafo	9500	4.5
30604	06/15/73	Mecanógrafo	8500	–

Figura 6.10 Relaciones 3FN.

6.5 HACIENDO RELACIONES A PARTIR DE RELACIONES—PROYECCION Y UNION

Una vez que se han simplificado los contenidos de los almacenamientos de datos a unas pocas relaciones normalizadas, necesitamos ser capaces de utilizar estas relaciones para poder contestar consultas, así como para efectuar otras tareas de procesamiento. Esto significa que necesitamos un método para construir estructuras de relación más amplias y para la extracción de partes de nuestras relaciones almacenadas.

Han sido definidas dos operaciones que nos permiten describir lo que deseamos hacer. Se denominan *proyección* y *unión*.

6.5.1 Proyección

Un nombre más adecuado para esta operación sería el de *selección;* consiste en "proyectar" la relación completa sobre dominios seleccionados para finalizar con una relación solamente entre estos dominios. La Fig. 6.11 muestra la proyección de

SALARIO-TAREA-HISTORIA sobre EMPLEADO-Nº
SALARIO-TAREA-HISTORIA sobre EMPLEADO-Nº más TAREA-DENOMINACION

Como se puede ver, la proyección implica la extracción del dominio(s) elegido, seguido de la supresión de cualquier "tupla" duplicada. La proyección es la operación lógica que involucra el manejo de consultas tales como "Qué empleos ha tenido en la empresa el empleado 26622" o "Déme la historia del salario del empleado 30604."

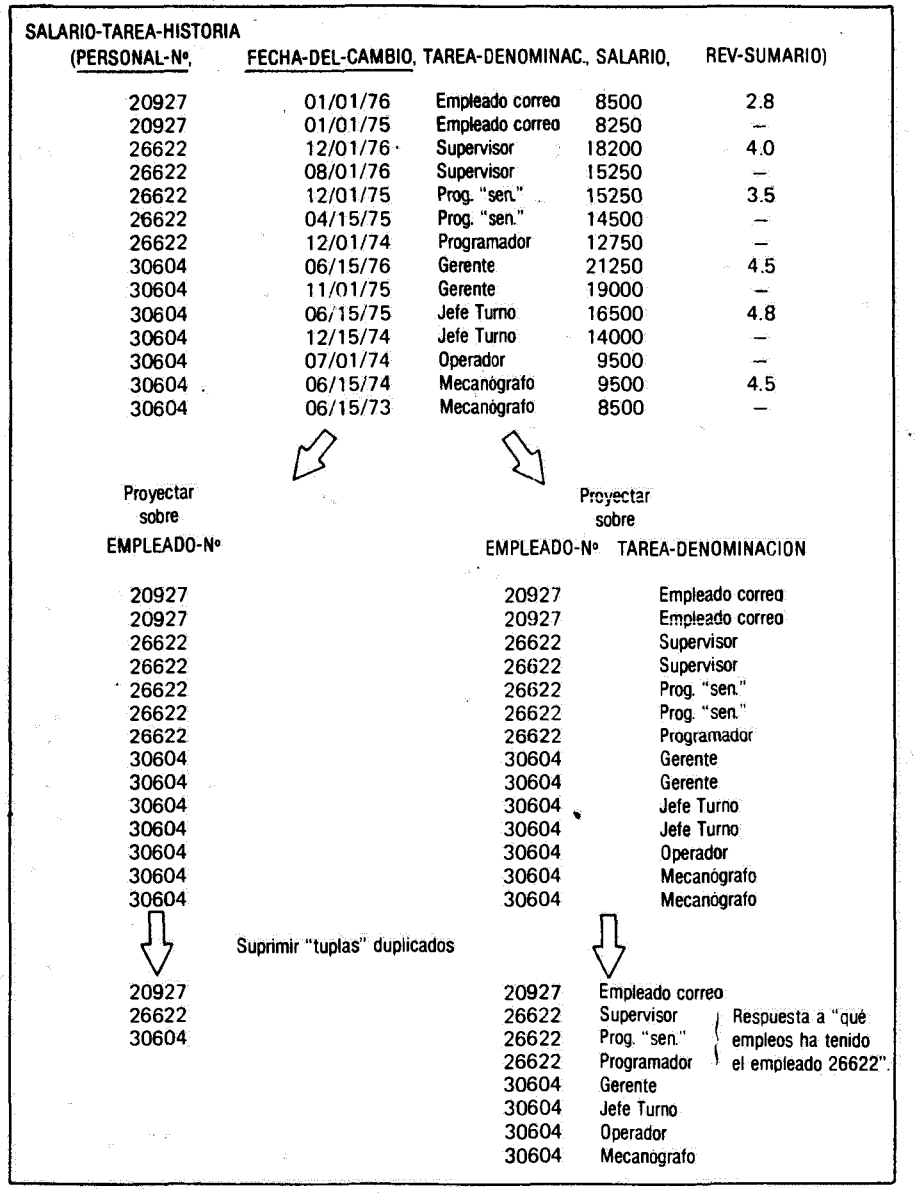


Figura 6.11 Proyecciones.

La figura ilustra también el hecho de que es perfectamente posible tener una relación con un solo dominio, tal como EMPLEADO-Nº, así como en la teoría se puede tener una estructura de datos con un solo elemento de datos.

6.5.2 Unión

Esta operación es la inversa de la proyección: colocar dos relaciones juntas para hacer una relación mayor. Las dos relaciones a ser unidas deben tener un dominio común. Supongamos que tenemos una relación que describe proveedores de productos, es decir,

PROVEEDOR (PROV-Nº, PROV-NOMBRE, CIUDAD)

con los siguientes valores

Proveedor:	Prov-No	Prov-Nombre	Ciudad
	013	Handy	New York
	061	Readymix	Pittsburgh
	062	Readymix	New York
	063	Readymix	Boston
	095	Friendly	Buffalo

y otra relación que describe el producto en sí mismo,

PRODUCTO (PRODUCTO-CODIGO, PROV-Nº, DESCRIPCION, CANTIDAD-DESPACHADA)

con los siguientes valores

Producto:	Producto-Código	Prov-No	Descripción	Cant-Despachada
	A123	013	Harina de maíz	100
	B671	013	Maíz	32
	A123	061	Harina de maíz	100
	C777	095	Alimento para ganado	50
	B671	062	Maíz	72
	A123	095	Harina de maíz	100

Podemos reunir estas dos relaciones sobre PROV-Nº para obtener la relación compuesta PRODUCTO-PROVEEDOR, como se ve en la Fig. 6.12. Las "tuplas" de PRODUCTO-PROVEEDOR han sido ordenadas por número del proveedor dentro del código del producto, aunque estrictamente hablando una relación no tiene por qué estar en secuencia. Nótese que el proveedor 063, Readymix de Boston, no aparece en la unión porque no nos ha despachado ningún producto. Nótese también que PRODUCTO no está en 3FN ¿Por qué no? ¿En qué forma está? ¿Cómo pueden convertirse todas las relaciones a 3FN?

El orden en el cual los dominios están listados no tiene importancia en la relación, aunque es conveniente que tengan la clave a la izquierda y los dominios no-clave agrupados de manera que tenga más sentido para la persona que lea la relación. Esto contrasta con nuestra notación de estructura de datos, en la cual tenemos que poner juntos a los miembros de un grupo repetitivo. Si tuviéramos que describir PRODUCTO-PROVEEDORES como una estructura de datos, se vería como sigue:

PRODUCTO-PROVEEDORES
PRODUCTO-CODIGO
DESCRIPCION
DESPACHOS* (1-)
PROV-Nº
PROV-NOMBRE
CIUDAD
CANTIDAD-DESPACHADA

6.6 LA IMPORTANCIA DE LA TERCERA FORMA NORMAL

3FN es la representación más simple posible que podemos efectuar de los datos. Representa en cierta manera el "sentido común inspirado" que a menudo, cuando reducimos los contenidos de los almacenamientos de datos a estructuras de datos en 3FN, y miramos el resultado, nos hace decir "¡Por supuesto! Estos son los elementos básicos de datos que describen a un proveedor (o una parte, o un empleado)." Los usuarios que no son técnicos encuentran las relaciones normalizadas, y 3FN en particular, de fácil comprensión; y después de todo, despojándonos del vocabulario especial, estamos representando todos los datos en forma de una tabla perfectamente común, ¿y qué puede resultar más familiar que esto?

Además de estos beneficios para el analista y el usuario, la tendencia en el diseño físico de las bases de datos, se dirige al uso de las relaciones normalizadas, a través de bases de datos "relacionales" más que con índices y jerarquías (como se describen en el próximo capítulo). La simplicidad básica de los datos en 3FN los hace mucho más flexibles y fáciles de cambiar comparados con otros métodos para organizar una base de datos física. Las bases de datos relacionales aún no se encuentran ampliamente difundidas debido a que el "hardware"

PRODUCTO:

PRODUCTO CODIGO	PROV-Nº	DESCRIPCION	CANTIDAD DESPACHADA
A123	013	Harina de maiz	100
B671	013	Maiz	32
A123	061	Harina de maiz	100
C777	095	Alimento p/ganado	50
B671	062	Maiz	72
A123	095	Harina de maiz	100

PROVEEDOR:

PROV-Nº	PROV-NOMBRE	CIUDAD
013	Handy	New York
061	Readymix	Pittsburgh
062	Readymix	New York
063	Readymix	Boston
095	Friendly	Buffalo

Juntar sobre PROV-Nº

PRODUCTO-PROVEEDORES:

PRODUCTO CODIGO	PROV-Nº	PROV-NOMBRE	CIUDAD	DESCRIPCION	CANTIDAD DESPACHADA
A123	013	Handy	New York	Harina de maiz	100
A123	061	Readymix	Pittsburgh	Harina de maiz	100
A123	095	Friendly	Buffalo	Harina de maiz	100
B671	013	Handy	New York	Maiz	32
B671	062	Readymix	New York	Maiz	72
C777	095	Friendly	Buffalo	Alimento para ganado	50

Figura 6.12 Ejemplo de una operación de unión.

requerido para procesar relaciones a velocidades aceptables, todavía es muy caro. Como el "hardware" se está abaratando rápidamente, y como la complejidad de los datos y la necesidad de cambio están creciendo, las bases de datos relacionales se están volviendo más y más importantes. Mientras tanto si podemos producir archivos y bases de datos utilizando estructuras 3FN, podrá resultar más fácil el cambio hacia las bases de datos relacionales.

Es así que como analistas podemos usar 3FN para matar tres pájaros de un tiro:

1. Podemos utilizar las relaciones 3FN como bloques de construcción básicos de los almacenamientos de datos que especifiquemos (realmente, el solo conocimiento de 3FN nos permite especificar estructuras más simples, con mayor facilidad).
2. Podemos utilizar 3FN como el medio standard para comunicar los contenidos de los almacenamientos de datos a los diseñadores físicos, ya sea que el eventual sistema esté orientado hacia una base de datos o a un archivo.
3. Podemos mostrar el contenido lógico de los almacenamientos de datos a los usuarios interesados en la forma de tablas familiares.

6.7 UN EJEMPLO PRACTICO DE 3FN

Para ver cómo trabajan las técnicas de normalización en la práctica, tomaremos una parte del sistema CBM definido en el Capítulo 3 y analizaremos los contenidos de los almacenamientos de datos de este subsistema. La Fig. 6.13 muestra la parte del sistema relacionada con la entrada del pedido, ingreso de pedidos para libros, control de dichos pedidos para ver cuáles pueden ser satisfechos y creación de una serie de "item despachables" por el depósito, junto con "item no despachables" que será necesario pedir. (En el Capítulo 9 supondremos que el analista y el diseñador han decidido hacerlo en un subsistema físicamente separado; y discutiremos el uso del análisis estructurado y de las herramientas de diseño estructurado para obtener un buen diseño físico. Utilizaremos las relaciones normalizadas de esta sección como una entrada para ese diseño.)

Como se puede ver en la Fig. 6.13 estamos interesados en cuatro almacenamientos lógicos de datos:

D1: CLIENTES
D2: LIBROS
D3: CUENTAS A COBRAR
D4: INVENTARIO

Asumimos, a los fines de este ejemplo, que los contenidos brutos de cada uno de estos almacenamientos de datos han sido definidos en el diccionario de datos, basándose en un examen de los flujos de datos hacia y desde cada almacenamiento de datos, como se discutió en la Sec. 6. Tomaremos por turno cada almacenamiento de datos y lo normalizaremos.

6.7.1 Normalización del almacenamiento de datos de CLIENTES

Supongamos que la entrada de CLIENTES del diccionario de datos especifica su estructura de la siguiente forma:

```
D1: CLIENTES
    ORGANIZACION-NOMBRE
        ORGANIZACION-DIRECCION* (1-)
            CALLE-CASILLA
            CIUDAD-MUNICIPIO
            ESTADO-CODIGO-POSTAL
```

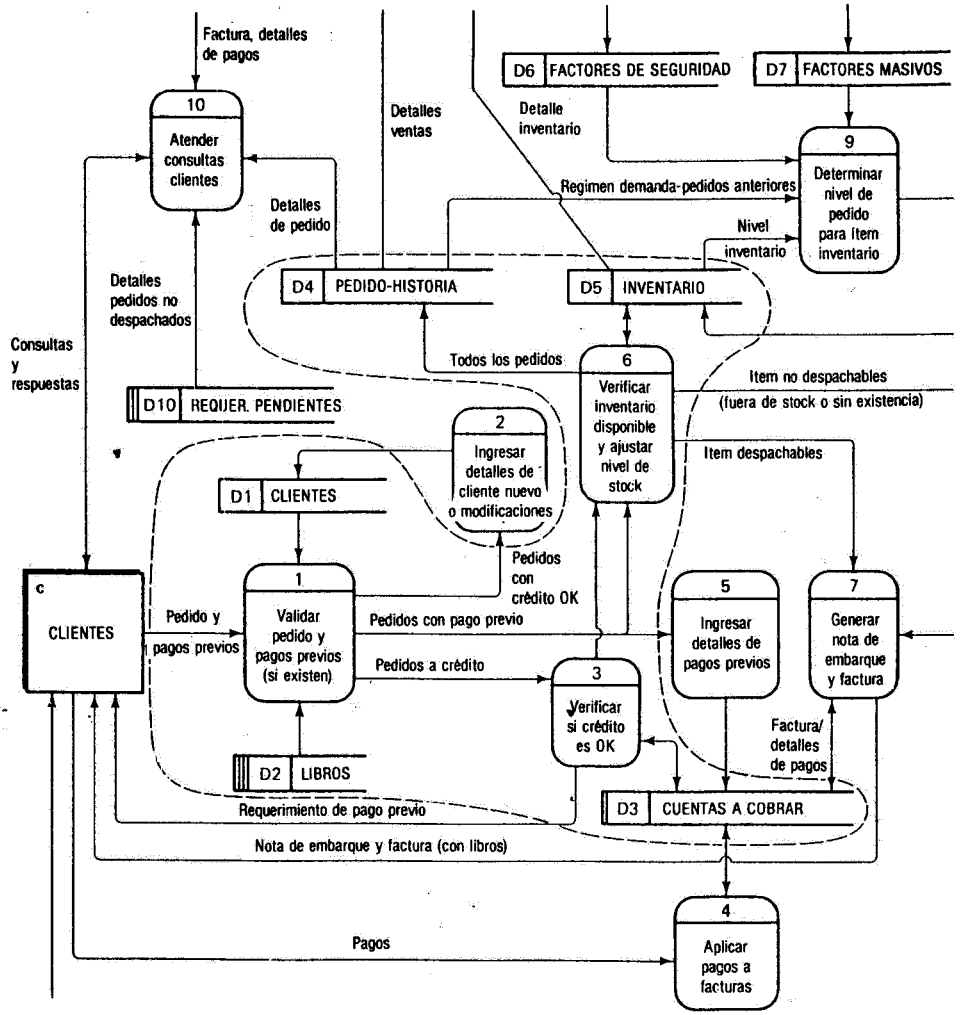


Figura 6.13 "Pedido-entrada", subsistema de la Compañía CBM.

TELEFONO
    AREA-CODIGO
    CENTRAL
    NUMERO
CONTACTO* (1-)
    CONTACTO-NOMBRE
    [EMPLEO-DENOMINACION]
    [TELEFONO-EXTENSION]

A partir de esta estructura de datos vemos que podemos esperar que cualquier organización, digamos IBM, tenga una o más ubicaciones (por ejemplo, White Plains, Palo Alto, Boca Ratón), cada una con su dirección. En cada una de las localidades podemos tener una cantidad de contactos, por ejemplo,

White Plains
  John Jones
  Mary Roe
Palo Alto
  Jim Smith
  Lucy Vélez
Boca Ratón
  Fred Smith

¿Es esta estructura de datos una relación en primera forma normal? Definitivamente, no; contiene un grupo repetitivo (CONTACTO) dentro de otro grupo repetitivo (ORGANIZACION-DIRECCION). De manera que nuestro primer paso será suprimir estos grupos repetitivos e identificar una única clave adecuada por cada "tupla" en la nueva relación. Un primer intento podría verse de la siguiente forma:

CLIENTES (ORGANIZACION-DIRECCION, CONTACTO-NOMBRE,
ORGANIZACION-NOMBRE, TELEFONO, EXTENSION, EMPLEO-DENOMINACION)

En esta relación, la clave es la concatenación de los atributos definidos por los dos grupos repetitivos originales. Como no tienen grupos repetitivos, está en 1FN. Nos resulta ligeramente insatisfactorio que para dos personas, ambas llamadas John Jones, que resulte que trabajan para organizaciones diferentes en un gran edificio, tengamos dos claves duplicadas. Para mayor seguridad, deberíamos incluir ORGANIZACION-NOMBRE en la clave. La relación sería entonces:

CLIENTES (ORGANIZACION-NOMBRE; ORGANIZACION-DIRECCION,
CONTACTO-NOMBRE, TELEFONO, EXTENSION, EMPLEO-DENOMINACION)

Ahora podemos identificar cada contacto unívocamente, excepto en el caso muy raro en que la misma *compañía* ¡tenga dos John Jones en la misma dirección! El precio que debemos pagar es una clave inmanejable; verdaderamente podría decirse que la relación es casi toda clave.

Dejando aparte esta consideración por el momento, ¿está la nueva relación en segunda forma normal, además de estarlo en 1FN? ¿Son todos los dominios no claves funciones completamente dependientes de la clave concatenada? A primera vista, podríamos decir que EXTENSION y EMPLEO-DENOMINACION están especificadas cuando conocemos CONTACTO-NOMBRE. En tal caso, esto implicaría que EXTENSION y EMPLEO-DENOMINACION no son funciones *completamente* dependientes de toda la clave. Sin embargo, hemos construido esta clave tan elaborada precisamente porque CONTACTO-NOMBRE puede estar duplicado. Consecuentemente, podemos decir que la clave completa es necesaria para especificar cada uno de los dominios no-clave y que la relación está entonces en 2FN.

¿Son todos los dominios no-clave mutuamente independientes? Sí, no hay forma segura de obtener TELEFONO, EXTENSION o EMPLEO-DENOMINACION, dado cualquier otro. ¡Luego la relación está no solamente en 2FN sino también en 3FN tal como está!

Aunque tenemos una relación 3FN, todavía nos resulta una clave inmanejable, que en raras circunstancias no es unívoca; como un refinamiento, podríamos crear un nuevo dominio para identificar unívocamente cada ubicación de cada organización, y llamarla tal vez,

ORG-ID. La misma podría corresponder a un número de cuenta donde los primeros cinco dígitos representan a la organización y los dos últimos representan la ubicación dentro de la organización. Por ejemplo, 10926 podría ser la ORG-ID básica de Distribuidora General Corp.; 1092601 podría resultar DGC de New York, 1092602 la sucursal de Chicago, etc. Dado este dominio podríamos dividir CLIENTES en dos relaciones 3FN:

ORGANIZACIONES (ORG-ID, ORGANIZACION-NOMBRE, ORGANIZACION-DIRECCION, TELEFONO-PRINCIPAL)

y

CONTACTOS (CONTACTO-NOMBRE, ORG-ID, CONTACTO-TELEFONO, EMPLEO-DENOMINACION)

Como toda clave simple o número de cuenta, ORG-ID deberá sufrir el hecho que, a menos que se hagan previsiones especiales, el usuario deberá conocer el código de ORG-ID para poder llegar a cualquiera de los registros que describen la organización o los contactos. (También deberemos suponer que si dos homónimos trabajan en la misma organización en la misma dirección, *ellos* ya habrán tenido bastantes problemas con la correspondencia confundida ¡para identificarse únicamente por sus nombres!

6.7.2 Normalización del almacenamiento de datos de LIBROS

En la entrada correspondiente a este almacenamiento de datos, en el diccionario de datos, encontramos la siguiente estructura:

LIBROS
- ISBN
- LIBRO-TITULO
- AUTOR
- ORGANIZACION-INSTITUCION
- EDITOR-NOMBRE
- PRECIO

En esta estructura no existen grupos repetitivos; está automáticamente en 1FN. Se podría pensar que AUTOR constituye un grupo repetitivo, pero una pequeña mirada nos mostrará que AUTOR es solamente un campo de longitud variable. Por ejemplo, si Smith y Jones son los autores de *Estructuras Estructuradas,* no es verdad que exista un libro de este título escrito por Smith y otro por Jones, como sería en el caso de un grupo repetitivo. Se podría argüir que podría haber una ORGANIZACION-INSTITUCION separada para cada AUTOR: Si Smith es de la Universidad de Podunk y Jones de la Distribuidora General Corp., deberíamos poder poner de manifiesto este hecho. En este caso, el analista ha observado que la mayoría de los autores que colaboran entre sí pertenecen a la misma institución y ha especificado que si existen diferentes instituciones solo se incluirá la institución del autor más caracterizado.

El ISBN define unívocamente el libro: luego, podemos escribir la relación 1FN como

LIBROS (ISBN, LIBRO-TITULO, AUTOR, ORGANIZACION-INSTITUCION, EDITOR, NOMBRE, PRECIO)

¿Son todos los dominios no-clave funciones completamente dependientes de la clave? Suponiendo que consideramos al comprobado ISBN como un único dominio, podría ser así. (En realidad, como se indicó en la Sec. 2.2, el ISBN tiene sub-códigos, uno de los cuales

especifica el EDITOR-NOMBRE, pero vamos a ignorarlo por el momento.) La relacioin entonces es 2FN.

¿Son todos los dominios no-clave mutuamente interdependientes? No, ORGANIZACION-INSTITUCION claramente depende de AUTOR, y por lo tanto debemos separarlos en una relación diferente. Ahora llegamos a dos relaciones 3FN.

LIBROS (ISBN, LIBRO-TITULO, AUTOR, EDITOR-NOMBRE, PRECIO)

y

AUTOR-INSTITUCION (AUTOR, ORGANIZACION-INSTITUCION)

6.7.3 Normalización del almacenamiento de datos de CUENTAS A COBRAR

La definición de la estructura de datos es:

ORG-ID
FACTURACION-DIRECCION
FECHA-APERTURA-CUENTA
FACTURA* (1-)
    FACTURA-NO
    FACTURA-FECHA
    FACTURA-IMPORTE
PAGO* (0-)
    CHEQUE-NO
    PAGO-FECHA
    PAGO-IMPORTE
SALDO-PENDIENTE
CANTIDAD-DE-PEDIDOS-A-LA-FECHA

En este almacenamiento de datos notamos que existen dos grupos repetitivos que representan las múltiples facturas que han sido remitidas y los diversos pagos recibidos. En la práctica, esta información solo se mantendrá en el almacenamiento de datos principal por un cierto período (comúnmente los 6 últimos meses o el año fiscal); después de este período los detalles de facturas y pagos se transferirán a un archivo donde se podrán obtener a requerimiento, de manera de no sobrecargar el almacenamiento activo de datos. Esta consideración práctica, sin embargo, no afecta a la *estructura* del almacenamiento de datos.

Como un primer intento para suprimir los grupos repetitivos, vamos a separar los elementos de datos relativamente estáticos de los más volátiles:

CUENTA-MAESTRO (ORG-ID, FACTURACION-DIRECCION, FECHA-APERTURA-CUENTA)
FACTURAS (FACTURA-Nº, ORG-ID, FACTURA-FECHA, FACTURA-IMPORTE)
PAGOS (CHEQUE-Nº, ORG-ID, PAGO-FECHA, PAGO-IMPORTE)

Observemos que FACTURA-NO es asignado secuencialmente por CBM a medida que se envían las facturas y en consecuencia define a cada factura en forma unívoca. El CHEQUE-NO es asignado por cada cliente y en consecuencia podrá no ser unívoco a menos que se combine con ORG-ID. ¿Pero qué pasa con SALDO-PENDIENTE y CANTIDAD-DE-PEDIDOS-A-LA-FECHA? Estos cambian todo el tiempo a medida que se reciben los pedidos y los pagos, y sin embargo, intuitivamente no podemos decir si pertenecen a FACTURAS o PAGOS. Si pudiéramos establecer que tienen completa dependencia funcional de la clave de una de las relaciones, tendríamos una buena razón para incluirlos en esta relación. En realidad, este es el caso; tanto SALDO-PENDIENTE como CANTIDAD-

DE-PEDIDOS-A-LA-FECHA tienen dependencia funcional de ORG-ID. En consecuencia podemos incorporar con seguridad ambos dominios en la relación

CUENTA-MAESTRO (ORG-ID, FACTURACION-DIRECCION, FECHA-APERTURA-CUENTA, SALDO-PENDIENTE, CANTIDAD-DE-PEDIDOS-A-LA-FECHA)

aunque sepamos que son volátiles.

6.7.4 Normalización del almacenamiento de datos de INVENTARIO

En el diccionario de datos encontramos esta especificación para el contenido de INVENTARIO:

ISBN
EDITOR-NOMBRE
CANTIDAD-PEDIDA
NIVEL-DE-PEDIDO

Como no existen grupos repetitivos podemos definir inmediatamente una relación 1FN, con ISBN como clave:

INVENTARIO (ISBN, EDITOR-NOMBRE, CANTIDAD-DISPONIBLE, CANTIDAD-PEDIDA; NIVEL-DE-PEDIDO)

Solo hay un dominio en la clave; luego la relación deberá ser 2FN; y como los dominios no-clave son independientes, también es 3FN.

6.7.5 Juntando las relaciones

La Fig. 6.14 muestra el análisis de los cuatro almacenamientos de datos dentro de sus relaciones componentes. Examinando las ocho relaciones en la Fig. 6.14 vemos que dos de ellas tienen a ORG-ID como claves (ORGANIZACIONES, CUENTA-MAESTRO) y dos tienen a ISBN como clave (LIBROS, INVENTARIO). ¿Podemos combinar estos dos pares y obtener todavía relaciones 3FN? Inspeccionemos los dominios de cada par para asegurarnos de que son mutuamente independientes. De hecho son independientes en ambos casos, de manera que podemos reducir de ocho relaciones a seis, como se muestra en la Fig. 6.15, empleando la operación UNION sobre la clave común.

También podríamos decidir disponer los archivos físicos en forma diferente; es de práctica común tener separada la información contable, por razones de seguridad y de control. Pero mediante la aplicación de la técnica de normalización, hemos completado los agrupamientos de los elementos de datos que describen las seis entidades básicas que están relacionadas con el subsistema de pedido-entrada (aparte de pedidos, por supuesto):

- Organizaciones
- Personas de las organizaciones
- Libros
- Personas que escriben libros
- Facturas
- Pagos

No obstante el caso de que los archivos reales ya estén diseñados (discutiremos los temas en

Almacenamiento de datos	Relación normalizada
D1: CLIENTES	
ORGANIZACION	(ORG-ID, ORGANIZACION-NOMBRE, ORGANIZACION-DIRECCION, TELEFONO-PRINCIPAL)
CONTACTOS	(CONTACTO-NOMBRE, ORG-ID, CONTACTO-TELEFONO, EMPLEO-DENOMINACION)
D2: LIBROS	
LIBROS	(ISBN, LIBRO-TITULO, AUTOR, EDITOR-NOMBRE, PRECIO)
AUTOR-INSTITUCION	(AUTOR, ORGANIZACION-INSTITUCION)
D3. CUENTAS A COBRAR	
CUENTA-MAESTRO	(ORG-ID, FACTURACION-DIRECCION, FECHA-APERTURA-CUENTA, SALDO-PENDIENTE, CANTIDAD-PEDIDOS-A-LA-FECHA)
FACTURAS	(FACTURA-NO, ORG-ID, FACTURA-FECHA, FACTURA-IMPORTE)
PAGOS	(CHEQUE-NO, ORG-ID, PAGO-FECHA, PAGO-IMPORTE)
D5: INVENTARIO	
INVENTARIO	(ISBN, EDITOR-NOMBRE, CANTIDAD-DISPONIBLE, CANTIDAD-PEDIDA, NIVEL-DE-PEDIDO)

Figura 6.14 Cuatro almacenamientos de datos.

ORGANIZACION-MAESTRO	(ORG-ID, ORGANIZACION-NOMBRE, ORGANIZACION-DIRECCION, FACTURACION-DIRECCION, TELEFONO-PRINCIPAL, FECHA-APERTURA-CUENTA, SALDO-PENDIENTE, CANTIDAD-PEDIDOS-A-LA-FECHA)
CONTACTOS	(CONTACTO-NOMBRE, ORG-ID, CONTACTO-TELEFONO, EMPLEO-DENOMINACION)
LIBRO-INVENTARIO	(ISBN, LIBRO-TITULO, AUTOR, EDITOR-NOMBRE, PRECIO, CANTIDAD-DISPONIBLE, CANTIDAD-PEDIDA, NIVEL-DE-PEDIDO)
AUTOR-INSTITUCION	(AUTOR, ORGANIZACION-INSTITUCION)
FACTURAS	(FACTURA-NO, ORG-ID, FACTURA-FECHA; FACTURA-IMPORTE)
PAGOS	(CHEQUE-NO, ORG-ID, PAGO-FECHA, PAGO-IMPORTE)

Figura 6.15 Seis relaciones de datos.

los Capítulos 7 y 9), empezaremos con una firma desde sus cimientos para comprender *qué* representan los archivos físicos a nivel lógico. Las relaciones 3FN son un modelo lógico de los almacenamientos de datos, tan libres de las consideraciones de implementación física como es posible. Mucho del esfuerzo y la dificultad del diseño de archivos físicos o bases de datos físicas, nace de la dificultad de ver qué elementos de datos en los archivos existentes verdaderamente *deben permanecer juntos,* qué elementos de datos describen *entidades diferentes,* qué elementos de datos se *superponen* o se *duplican,* y qué *omisiones* existen. La

construcción de un modelo relacional lógico nos permite seguir un largo camino para salvar estos problemas.

BIBLIOGRAFIA

6.1 F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM*, junio de 1970.

6.2 E.F. Codd, "Further Normalization of the Data Base Relational Model", en *Courant Computer Science Symposia*, Vol. 6 "Data Base Systems", Prentice-Hall, Englewood Cliffs, N.J., 1972.

6.3 J. Martin, *Computer Data-Base Organization*, Prentice-Hall, Englewood Cliffs, N.J., 1975.

6.4 C. J. Date, *An Introduction to Database System*, Addison/Wesley, Reading, Mass., 1975.

Ejercicios y puntos de discusión

1. Explicar con sus propias palabras los términos relación, dominio, "tupla", grado, clave candidata, y dependencia funcional.
2. Suponiendo que el almacenamiento de datos D5: EMPLEADOS-DETALLES de la Fig. 6.1 debiera ser fisicamente implementado como dos archivos, cada uno de ellos con una estructura 3FN como se ve en la Fig. 6.6, escribir el "pseudocódigo" para los procesos de la Fig. 6.1.
3. ¿En qué sentido es real que la normalización incrementa la redundancia?
4. ¿Qué operación proyección/unión se requerirá para contestar el pedido "Déme la historia de los salarios de Mary Worth"?, a partir de las relaciones de la Fig. 6.6.
5. La Compañía Alimentos Amistosos tiene su archivo de proveedores armado con esta estructura de datos:

   PROVEEDOR
   PROVEEDOR-NO
   PROVEEDOR-NOMBRE
   PROVEEDOR-DIRECCION
   PRODUCTO*
   PRODUCTO-CODIGO
   DESCRIPCION
   PRECIO

   No hay archivo separado de PRODUCTO.
   (a) Supongamos que Bighorn Inc., único proveedor del producto A679 cesa sus actividades y es eliminado del archivo. ¿Cómo podríamos averiguar la descripción de A679?
   (b) Supongamos que la gerencia de Friendly decide iniciar la comercialización de un nuevo alimento BETTACOW, al que se le da el código D302. ¿Cómo podemos conservar su descripción en el archivo mientras buscamos un proveedor?
   Convertir la estructura de datos anterior en relaciones 3FN. ¿Cómo se manejan las situaciones (a) y (b) con el modelo relacional?
6. A partir de las relaciones de las Figs. 6.14 y 6.15, producir una relación (o relaciones) 3FN para representar los pedidos de libros, recibidos de los clientes.

7

ANALISIS DE LOS REQUERIMIENTOS DE RESPUESTA

7.1 DESCRIPCION DE LAS FORMAS EN QUE SE UTILIZAN LOS DATOS

Las salidas provenientes de los almacenamientos de datos y definidas en los diagramas de flujo de datos son de la más variada naturaleza. Consideraremos los siguientes requerimientos de información:

1. "Deseo recibir cada mañana un listado con todos los pedidos recibidos el día anterior. El informe debe estar correcto hasta las 16.30 horas y a mi disposición a las 9,00 horas."
2. "Cada vez que el nivel de inventario de un ítem llegue a estar debajo del nivel de seguridad, déme el informe del estado de las órdenes de compra del ítem. Este informe debe producirse a la mañana siguiente."
3. "¿Cuál es el saldo deudor del cliente cuyo número de cuenta es 349, esta mañana? Lo tengo al teléfono."
4. "¿Cuánto negoció con nosotros el cliente 349 en los tres últimos meses hasta el viernes último? Tengo una reunión con él pasado mañana."
5. ¿Cuántas veces en los seis últimos meses el proveedor X dejó de cumplir las fechas prometidas de entrega? Lo tengo al teléfono."
6. "Necesito un informe que muestre qué porcentaje de nuestro presupuesto en los pasados cinco años financieros se realizó con firmas por menos de $ 1 millón anualmente. Tengo que hacer una presentación al directorio el mes próximo."
7. "Déme los nombres y teléfonos particulares de todos nuestros ingenieros electrónicos que hablen bien árabe, que hayan atendido el modelo 999 en los últimos seis meses y que no estén asignados a proyectos urgentes. Poner asterisco a los solteros, listarlos por orden de antigüedad y hacerlo en forma urgente. Deseo que vuele alguien esta tarde a Dhahran."

Este abanico de siete requerimientos abarca el espectro completo de aquellos que encontramos en la práctica, en un orden grosero de dificultad y costo de respuesta. A pesar de su aparente diversidad, tienen cuatro factores principales que los distinguen:

1. ¿Cuán predecible es la distribución en el *tiempo* del requerimiento? ¿Puede ser programado, como el requerimiento 1, o es activado por algún evento, como el requerimiento 2, o es a pedido como en los requerimientos 3 al 7? Si es activado o pedido ¿qué volumen de transacciones podemos predecir?
2. ¿Cuán predecible es la *naturaleza* del requerimiento? Mientras los requerimientos programables y activables son predecibles por definición, los requerimientos exigibles pueden variar ampliamente tanto en los elementos de datos que necesitan, como en el procesamiento necesario para estos elementos de datos. El requerimiento 3 sobre el saldo de

una cuenta, casi siempre podrá ser predecible por el usuario. Los requerimientos 4 al 7 son, en orden creciente, más difíciles de predecir, no solamente en los detalles sino también en su propia naturaleza. El requerimiento 6 sobre un análisis de negocios basado en la dimensión histórica del cliente jamás podrá haber sido prevista hasta que el gerente ejecutivo se volcó con entusiasmo a la promoción de negocios pequeños.

3. ¿Cuán *recientes* o *frescos* deberán ser los datos? ¿Será suficiente tener su valor correcto al finalizar el último mes, o será necesario que esté correcto hasta la última transacción realizada? En el requerimiento 3, sobre el saldo del cliente ¿deberá la información contener los cheques recibidos en el correo de la mañana?

4. ¿Cuán *rápidamente* deberá ser atendido el requerimiento?

Como se observó en el Capítulo 2, se llega al punto crítico cuando la respuesta es requerida en un lapso más breve que el que necesita el archivo físico o la base da datos para ser explorado de punta a punta o clasificado. Si el usuario de la información no puede esperar la respuesta mientras el archivo completo es explorado o clasificado, el diseñador físico tendrá que prever el acceso inmediato a los datos. El acceso inmediato y particularmente un requerimiento impredecible que requiere el acceso inmediato, plantea una complejidad y un costo de diferente magnitud que los predecibles y de acceso no inmediato. El requerimiento 3 es de acceso inmediato a un saldo del cliente, que hemos dicho que era razonablemente predecible. Muchos sistemas más bien modestos pueden proveer facilidades como ésta. La pregunta 7, que requiere tener acceso (probablemente) a varios archivos y que será muy difícil anticipar, será muy costosa en recursos de máquina si (como implica) debe ser contestada rápidamente.

Existe una cantidad de otros aspectos que el analista deberá investigar, como ya hemos visto, tales como seguridad (quién está autorizado a realizar requerimientos y/o a recibir la respuesta) y la importancia que para el negocio tiene este requerimiento (si es "bueno tenerlo" o si se "debe tener"). Los temas sobre predecibilidad, acceso inmediato y lo reciente del dato son las claves que más afectan al costo y a la utilidad del sistema.

Para ver por qué esto es así, revisaremos algunas de las técnicas relacionadas con la provisión de accesos inmediatos, antes de examinar las herramientas que el analista puede usar para resolver estos temas.

7.2 TECNICAS FISICAS PARA EL ACCESO INMEDIATO

7.2.1 Indices

Supongamos que nuestra empresa compra repuestos eléctricos a una gran cantidad de proveedores en todo el país. Tenemos un archivo de proveedores que contiene sus números de referencia, sus nombres, localidades y las iniciales del agente comprador responsable de negociar con ellos, como así también otra información tal como la dirección del pago, teléfono, etc., las que omitiremos por claridad. El archivo se verá de la siguiente forma:

Proveedor-Nº	Proveedor-Nombre	Ciudad	Comprador.
6293	Reliable	St. Louis	ALC
4421	Lightning	Chicago	TDM
6604	Quicktronic	Omaha	ALC
6498	Speedy	Chicago	PAS
2727	Eléctrico	St. Louis	TDM

y así sucesivamente.

Vamos a suponer también que tenemos el archivo organizado de manera que podamos tener acceso al azar basado en PROVEEDOR-NO. Esto es, dado un número, digamos 2727, podemos recuperar inmediatamente el registro de Eléctrico. Luego PROVEEDOR-NO es la clave principal. Podemos contestar inmediatamente preguntas tales como "¿Dónde está ubicado el proveedor 6604?" o "¿Quién es el agente comprador del proveedor 4421?", simplemente recuperando un registro determinado y presentándolo. Esto es cierto siempre que el archivo se mantenga manualmente, digamos en un fichero rotativo de tarjetas por número de orden del proveedor, o en un disco de acceso directo.

Pero si ahora deseamos tener respuesta a preguntas como "¿Quiénes son nuestros proveedores en Chicago?" o "Listar los proveedores de los que es responsable el agente comprador TDM", la vida se complica. Esencialmente tenemos dos alternativas, o bien clasificar el archivo en la secuencia del elemento de datos requerido, o bien buscarlo mediante la lectura de cada registro, decidiendo en cada caso si coincide con nuestro criterio de búsqueda o no. La Fig. 7.1 ilustra estas alternativas.

Ya sea que clasifiquemos o busquemos, se tendrá que leer cada registro por lo menos una vez, y eso lleva tiempo. Mientras que la recuperación de un registro dado un PROVEEDOR-NO puede llevar segundos; la clasificación y la búsqueda tomará algunos minutos, dependiendo de la capacidad del archivo. El tiempo regular para clasificar 10.000 registros de 100 caracteres cada uno en una IBM 370/158 puede ser de 2-3 minutos, y para recuperar uno de estos registros dada la clave, 2-3 segundos. Podemos estimar cuánto tiempo llevaría buscar manualmente a través de un archivo de tarjetas rotativo de 10.000 tarjetas; a 1 por segundo, sería unas 3 horas aproximadamente. Peor aún, si otra persona deseara tener acceso inmediato empleando la clave mientras se está desarrollando el proceso de clasificación, ello lo retardaría, si es que no lo impide del todo. Por esta razón, es muy tentador hacer todas las clasificaciones y las búsquedas por la noche, cuando nadie está trabajando con accesos en línea sobre los archivos. Esto crea la tendencia a dividir los accesos en "inmediato" o "para mañana temprano", si bien de hecho algunos requerimientos deberán contestarse en pocos minutos, si es estrictamente necesario.

Si se puede predecir el acceso inmediato sobre la base de elementos de datos distintos de la clave principal, y es bastante importante para los usuarios (esto lo tendrá que determinar el analista), se lo puede proveer mediante distintas formas. La más simple es construyendo un índice del archivo. Un índice por COMPRADOR podría ser:

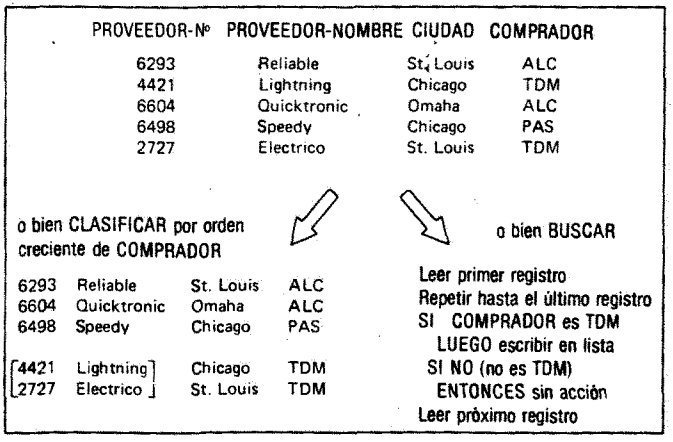


Figura 7.1 "Listado de proveedores de los cuales es responsable el agente TDM."

Comprador	Proveedor-Nº
ALC	6293, 6604
PAS	6498
TDM	2727, 4421

El índice deberá prepararse para el acceso al azar por las iniciales del agente comprador. Luego la respuesta al requerimiento "Listar aquellos proveedores de los cuales es responsable el comprador TDM", sería:

1. Buscar TDM en el índice (un acceso al azar) y recuperar los números de los proveedores.
2. Buscar los dos números de los proveedores (dos accesos al azar) en el archivo principal.

Luego, la existencia de este *índice secundario* nos permitirá remplazar una búsqueda de todo el archivo por una pequeña cantidad de rápidos accesos al azar.

La desventaja de tener un índice, es que ocupa mayor espacio en el archivo y cada vez que se cambia el agente comprador a un vendedor, el archivo deberá modificarse en forma correspondiente.

Podemos tener en el archivo un índice por cada uno de los elementos de datos no-clave. ¡Realmente, podríamos representar el archivo completo como un conjunto de índices! Ver Fig. 7.2. Esta estructura de archivo que se denomina *invertida,* de hecho contiene toda la información que había en el archivo original, si bien ahora no existirá un archivo "principal" como tal; el mismo ha desaparecido en los índices.

La estructura de archivo invertido es buena para el acceso inmediato al número del proveedor sobre la base de cualquier cosa que se pueda pensar. No es tan conveniente para contestar al requerimiento original "¿Dónde está localizado el proveedor 6604?" o "Déme todos los detalles del proveedor 2727."

INDICE-COMPRADOR		INDICE-CIUDAD		INDICE-PROVEEDOR-NOMBRE	
(Comprador. Proveedor-Nº)		(Ciudad: Proveedor-Nº)		Proveedor-Nombre:	Proveedor-Nº)
ALC	6293, 6604	Chicago	4421, 6498	Electrico	2727
PAS	6498	Omaha	6604	Lightning	4421
TDM	2727, 4421	St. Louis	2727, 6293	Quicktronic	6604
				Reliable	6293
				Speedy	6498

Figura 7.2 Archivo invertido.

En este último caso deberemos buscar en todos los índices para encontrar todas las referencias a 2727 y ¡luego armar el registro individual a partir de éstas! Realmente, no hay nada gratis.

La solución más utilizada en la práctica es la de mantener el archivo principal accesible por la clave principal, y proveer una pequeña cantidad de índices secundarios para manejar aquellos requerimientos que no puedan esperar la búsqueda de todo el archivo.

¿Pero cuáles son estos requerimientos? Los mismos deben ser predecibles por su naturaleza (de otra manera no sabremos que índices son necesarios), deben poder ser satisfechos por datos que estarán tan actualizados como el índice, y deben ser lo suficientemente importantes para la empresa como para que se justifique la molestia y el

costo de crear un índice especial. El analista de sistemas tiene la responsabilidad de tomar en cuenta estos factores de manera de poder elegir la mejor solución física.

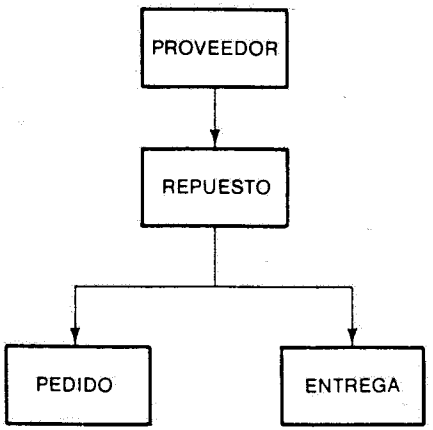
7.2.2 Registros jerárquicos

En el capítulo anterior se discutieron las formas para simplificar la estructura de los registros mediante la normalización, suprimiendo los grupos repetitivos. En el pasado, muchos diseñadores han tomado el camino opuesto; en lugar de simplificar los registros, ponían tanta información y estructuras como le eran posibles dentro de cada registro. En parte esto hacía factibles más accesos inmediatos; si teníamos no solamente el nombre del proveedor, cantidad, ciudad y agente comprador, sino también los repuestos que proveían con sus nombres, pedido y fechas de entrega, precios y peso, todos ellos en el mismo registro, se podía recuperar toda esta información con, en principio, un solo acceso inmediato.

Por ejemplo, el registro para Eléctrico podría estructurarse de manera que contenga mucha más información, como se muestra en la Fig. 7.3 (De hecho todos estos elementos de datos simplemente se disponen juntos en un registro largo; los hemos sangrado para mostrar su estructura.)

Eléctrico provee dos ítem, R23B y C117. Para cada repuesto, el número del repuesto, nombre, descripción y precio están dados en el segmento de repuesto, del registro. Cada repuesto tiene también una cantidad de segmentos de pedidos asociados con él y una cantidad de segmentos de entrega. La única forma por la que sabemos que la entrega NoE614 correspondió a la parte R23B es que su segmento sigue físicamente al segmento de repuesto, y viene antes que el próximo segmento de repuesto (para el repuesto C117).

Podemos hacer un esquema de la estructura de cada registro de la Fig. 7.3, de la siguiente forma



La estructura del diagrama implica que cada PROVEEDOR tiene uno o más REPUESTOS dependientes, cada REPUESTO tiene uno o más PEDIDOS dependientes, etc. Observamos que cualquier repuesto puede ser provisto por más de un proveedor; C117, el condensador, es provisto por ambos, Eléctrico y Lightining. El término para esta relación proveedor-repuesto es de *muchos-muchos* para distinguirla, digamos, de una relación compañía-empleado, que es de *uno-muchos*. Una única compañía puede tener muchos empleados, pero cada empleado tiene solo una compañía. Los registros jerárquicos pueden manejar ambas relaciones, uno-muchos y muchos-muchos.

Los registros jerárquicos pueden de este modo manejar una gran cantidad de complejidad y detalle y corrientemente se usan en un número de sistemas de administración de base de datos. El Information Management System (Sistema de Información Gerencial) (IMS) de IBM es tal vez el más conocido de estos sistemas. Con los sistemas jerárquicos, algunos tipos de acceso inmediato son fáciles. Por ejemplo, con los registros jerárquicos de la Fig. 7.3, podemos decir, "Déme el precio del repuesto C117 cargado por el proveedor 4421." La combinación del número del proveedor y del número de repuesto especifica *unívocamente* el segmento que contiene la información necesaria. Uno de los comandos provisto por IMS es "Get Unique", el cual dadas las claves necesarias recuperará el segmento específico. Podemos pensar en este tipo de comando como que "desciende a través del circuito jerárquico", iniciando con un proveedor específico (el nivel superior o segmento *raíz*) y moviéndose de este segmento a un segmento inferior, luego a otro inferior, etc. El requerimiento "¿Cuánto del repuesto C117 ha entregado el proveedor 2727 y cuándo? podrá ser contestado de esta manera.

Sin embargo, si deseásemos obtener otro dato distinto al que sigue la jerarquía, podríamos tener dificultades. Supongamos que quisiéramos contestar a la pregunta "¿Quién provee el repuesto C117?", tendríamos que buscar en la base de datos completa de todos los proveedores, controlando cada repuesto dentro de cada proveedor para ver si es el repuesto C117. IMS tiene el comando "Get next" para búsquedas de este tipo; la secuencia de comandos sería:

```
    Ir al comienzo de los datos
Lazo-Proveedor: Get next (obtener próximo) PROVEEDOR
    Get Next REPUESTO para el cual REPUESTO-Nº es C117
    IF (Si)      C117 es encontrado
      THEN (luego)     escribir detalles del proveedor en el informe
    ELSE (SI NO)       (este proveedor no provee repuesto C117)
      SO (entonces)    ir a lazo-proveedor
```

"Get next" nos permite desplazarnos a nuestra posición actual en la base de datos (cualquiera sea el último segmento recuperado) para recuperar el próximo segmento del tipo que se ha especificado. Por ejemplo, en la Fig. 7.3, si el último segmento recuperado fue PEDIDO-Nº 813, "Get Next REPUESTO" permitirá recuperar REPUESTO C117; "Get Next PROVEEDOR" permitirá recuperar PROVEEDOR 4421. Se desprende que podemos utilizar el comando "Get Next" para buscar en un archivo jerárquico o en una base de datos a fin de contestar cualquier información requerida (asumiendo que los elementos de datos necesarios están presentes, por supuesto). Sin embargo, volvemos al problema que encontramos en la sección anterior, donde, a menos que el requerimiento pudiera atenderse siguiendo el circuito jerárquico, habría que explorar efectivamente a través de toda la base de datos para asegurarse de haber encontrado todas las respuestas. Estas búsquedas pueden llevar un largo tiempo, especialmente con grandes bases de datos IMS. En la práctica, consultas como "¿Cuántos repuestos C117 han sido enviados desde el 1º de junio?" no pueden en realidad responderse inmediatamente, a menos de que se coloque algún tipo de índice.

En el IMS hay una cantidad de métodos para la creación de índices, o sus equivalentes, cada uno de los cuales involucra mayor o menor costo y complejidad. El diseño físico de una base de datos IMS es una tarea compleja, que comprende la ponderación de muchos factores de espacio requerido por los datos, frecuencia relativa de acceso, crecimiento de la base de datos, requerimientos de confiabilidad y seguridad, etc. El diseñador de la base de datos necesita que el analista de sistemas determine qué accesos inmediatos son necesarios, con qué frecuencia probable será necesario cada tipo de acceso, y la importancia relativa de cada acceso. Con esta información, será mucho más fácil para el diseñador de la base de datos seleccionar el mejor compromiso entre la capacidad de recuperación y el resto.

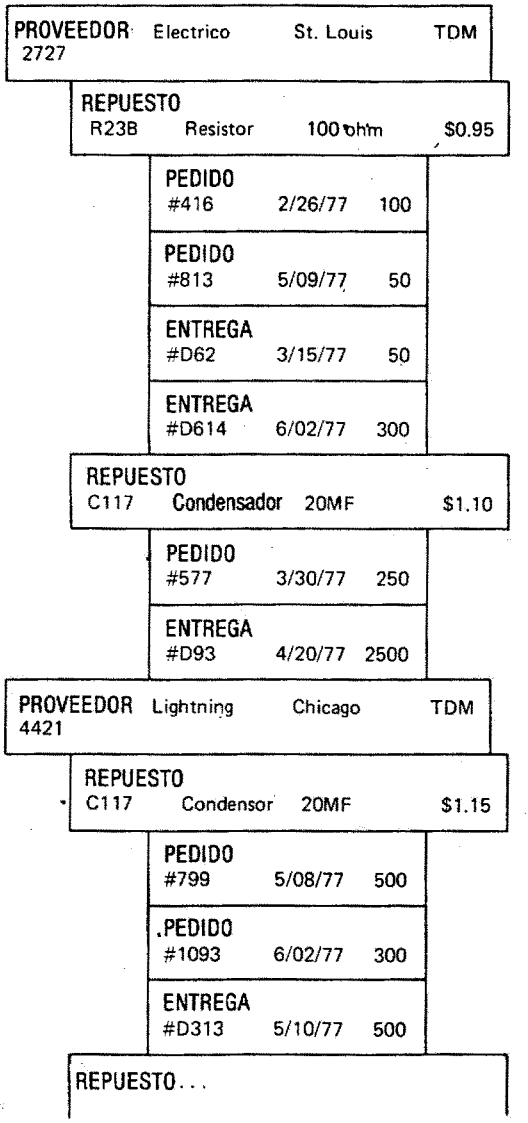


Figura 7.3 Estructura de un registro jerárquico.

7.3 CAPACIDAD DE LENGUAJE GENERAL DE CONSULTA

Aunque el analista haya especificado los índices apropiados para un archivo o base de datos, será muy difícil predecir en detalle la forma precisa en que el usuario deseará las respuestas a sus requerimientos de información.

Supongamos que tenemos un archivo de proveedores con un índice secundario para cada repuesto. El usuario podrá decir "Listar todos los proveedores que proveen repuesto C117" y podrá en principio obtener la respuesta inmediatamente. Pero supongamos que un día el requerimiento es "Déme todas las fuentes de provisión del repuesto C117 en California,

Arizona y Nuevo Méjico, en orden decreciente de la cantidad de repuestos que han entregado en los últimos tres meses". No hay nada que nos impida recuperar la información necesaria a partir del archivo de proveedores, pero vamos a necesitar un programa que compruebe el Estado donde está localizado cada proveedor, que acepte solo aquellos que están en los tres estados especificados, calcule las entregas hechas por los proveedores remanentes en los tres últimos meses y que clasifique los registros en orden decreciente por total entregado antes de listarlos. Este programa no es complejo, pero si debe ser escrito en un lenguaje normal de programación, digamos COBOL, su codificación y prueba podrán insumir un día o dos. Además, dado que no se previó un requerimiento de este tipo, no podemos tener la seguridad de que no volverá a repetirse. Así, estamos en posición de tener que escribir un programa "solo-por-una-vez" para contestar el requerimiento; el factor limitante de nuestra velocidad de respuesta al usuario no es la búsqueda del archivo sino ¡cuánto se tardará en escribir el programa!

Por esta razón, se ha desarrollado una cantidad de lenguajes de consulta simples, que permiten programar rápidamente una amplia variedad de requerimientos, en muchos de los casos por usuarios sin conocimientos de programación. Como un ejemplo consideremos el IQF (Interactive Query Facility) (Facilidad para Consultas Inter-activas), un paquete de "software" producido por IBM para ser utilizado con bases de datos tipo IMS [7.1, 7.2].

IQF permite al usuario sentarse frente a una terminal e ingresar requerimientos de información en inglés, supuesto que las palabras importantes de cada sentencia han sido definidas previamente al IQF. Por ejemplo, el usuario podría teclear:

> FROM THE PURCHASING DATA BASE, TOTAL THE DELIVERIES OF 20MF CONDENSERS IN THE LAST MONTH, AND LIST THE NAME AND CITY FOR SUPPLIERS OF 20MF CONDENSERS WITH DELIVERY LEAD TIMES LESS THAN ONE WEEK AND BULK DISCOUNTS OF GREATER THAN 10%.
>
> (A partir de la base de datos de adquisiciones, sumar las entregas de condensadores de 20MF en el último mes, y listar nombre y ciudad de los proveedores de condensadores de 20MF con tiempo de entrega menor que una semana y descuentos masivos mayores que 10%.)

Algunas de las palabras usadas en esta frase, tales como FROM, THE y OF (desde, el y de), habrán sido definidas previamente como palabras nulas, que el usuario puede colocar para hacer que el requerimiento le resulte legible y reconocible. Todos los otros términos, tales como PROVEEDORES, CONDENSADORES 20MF, etc., necesitarán ser definidos en el diccionario de datos privado del IQF. LIST y TOTAL (listar y sumar) son comandos IQF con obvio significado. El "software" IQF traduce los términos utilizados a sus significados predefinidos, realiza un método de búsqueda de la base de datos, arma los registros resultantes y lista los elementos de datos especificados en la terminal del usuario. Una vez que se ha hecho la definición de los términos (y cada usuario puede agregar sus propias definiciones), IQF permite básicamente a cada usuario escribir sus propios programas de consulta. Esta facilidad tan útil tiene la limitación que el usuario debe usar solamente los términos definidos; si tipea VENDEDORES en lugar de PROVEEDORES, IQF no sabrá lo que ello significa. El usuario también debe ser capaz de escribir sus requerimientos en una forma lógica, empleando los comandos IQF; algunos usuarios de lenguajes de consultas generales como IQF tienen un "asistente de información" especialmente entrenado para ayudarlo a traducir sus pensamientos a la forma requerida por el lenguaje.

IQF puede usarse en línea de manera que la respuesta al requerimiento pueda estar disponible en unos pocos segundos, siempre que se hayan previsto los necesarios circuitos de acceso dentro de la estructura de la base de datos, como se discutió en la sección anterior. El usuario no tiene forma de conocer, sin embargo, si la pregunta que introduce un lenguaje tipo corriente va a significar un acceso al azar o una búsqueda de toda la base de datos. Realmente, no tendría por qué saberlo; ¡los problemas de acceso deberán ser "transparentes" para él! De todos modos si 20 usuarios presentan simultáneamente 20 consultas distintas, cada una de las

cuales involucra búsquedas de punta a punta en una base de datos de tamaño razonable, se demorarían considerablemente entre sí e impedirían que alguien más pudiera utilizar la base de datos al mismo tiempo. Como consecuencia, IQF está equipado con una facilidad de advertencia; si el usuario plantea una consulta que IQF estima que insumirá una larga búsqueda, se lo notificará antes de comenzar dicha búsqueda, a fin de darle una oportunidad de trasladar la consulta a un horario nocturno, cuando la búsqueda pueda ser operada fuera de línea y entrañe una menor interferencia con los usuarios en línea. Alguna vez, por supuesto, el usuario dirá, "No me importa cuán larga sea la búsqueda o cuánto cueste, déme la respuesta"; es por eso que las facilidades de un lenguaje de consulta general pueden someter un centro de datos a cargas imprevisiblemente pesadas. Por otra parte, debe satisfacernos de que los usuarios puedan obtener valiosos resultados de la computadora (y esto sin gravar los recursos del elenco de programación). Por otra parte, se desea evitar una situación donde los usuarios estén consumiendo todo el tiempo del computador y degradando los tiempos de respuesta de otros sistemas, debido a que deben explorar la base de datos para hallar información, cuando se podría tener acceso inmediato a dicha información con solo proveerles un índice adecuado. Volvemos nuevamente a la moraleja de este capítulo, *el analista de sistemas deberá hacer todo lo que sea posible para prever qué tipos de accesos desearían tener los usuarios* y cuál es el valor de estos diversos tipos. Recién entonces se puede realizar un buen balance entre el costo de proveer índices secundarios y el costo de la exploración desde uno a otro extremo.

Esta moraleja resulta buena para todos los lenguajes de consulta que permitan la generación de informes en línea. Así como el IQF, IBM también comercializa el GIS (Generalized Information System) (Sistema de Información Generalizado), que es más extenso y con facilidades más poderosas.

GIS se asemeja más a un lenguaje de programación de alto nivel que el IQF, permitiendo a los usuarios construir archivos, modificar base de datos, hacer cálculos extensos, y asignar formato a los informes de diferentes maneras. IQF con su forma de tipo lenguaje corriente puede ser rápidamente aprendido por los empleados; GIS es más exigente y requiere mayor habilidad de programación, aunque puede ser aprendido por usuarios no programadores. Otros paquetes populares de consulta general comercializados por empresas independientes de "software" son CULPRIT (Cullinane Corp.), EASYTRIEVE (Pansophic), y MARK IV (Informatics). Las direcciones de estas firmas se encuentran al final del capítulo.

Es probable que veamos una expansión en el uso del "software" de consulta general en los próximos años, a medida que el "hardware" sea más barato, y que el "software" más avanzado y potente haga más fácil proveer acceso inmediato a cualquier elemento de datos del archivo, no solamente a aquéllos para los cuales se han establecido índices secundarios.

7.4 TIPOS DE CONSULTA

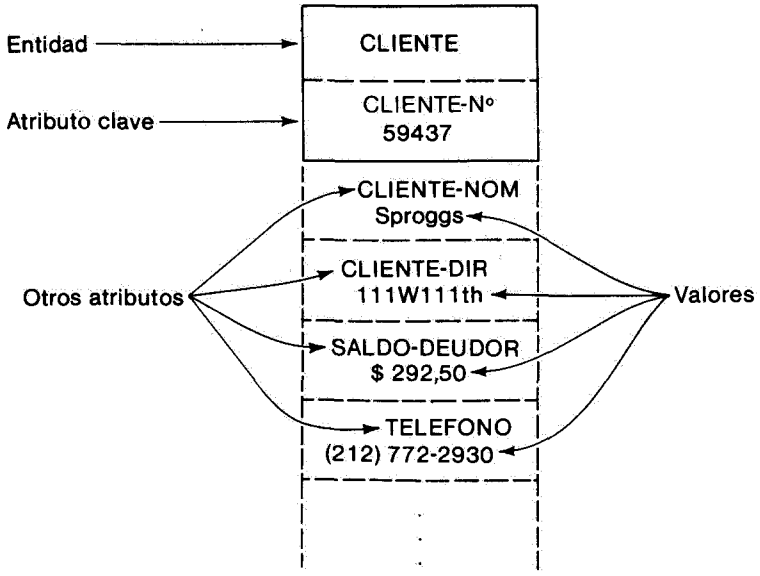
Antes de tratar de analizar las diferentes prioridades que los usuarios asignan a las consultas, será útil examinar los varios tipos de consulta distintos bajo el punto de vista lógico, que pueden presentarse. Podemos distinguir seis tipos básicamente diferentes de consultas [7.3] y observar algunas variaciones entre los mismos.

7.4.1 Entidades y atributos

En el mundo real las cosas tienen propiedades diversas; una máquina tiene tamaño, peso y número de identificación; una persona tiene nombre, dirección, número de empleado, fecha de ingreso, salario, etc. Cuando describimos datos, se define a una cosa o clase de cosas como una *entidad* y los elementos de datos que describen las propiedades de las cosas son *atributos* de la entidad. Podemos ver que los atributos de la entidad "cliente" son número de cliente, teléfono, dirección, etc. Un atributo se elige comúnmente como atributo clave, debido a que

identifica unívocamente a la entidad (si bien, como dijimos cuando consideramos la tercera forma normal, a veces será necesario combinar más de un atributo para formar una clave unívoca).

Podemos ilustrar este concepto como sigue:



Algunas veces es conveniente hacer la distinción entre *entidades básicas* (las cuales corresponden a cosas permanentes como ser clientes, empleados, proveedores y productos) y *entidades de transacción,* las cuales corresponden a cosas menos permanentes o eventos, que a menudo conectan dos entidades básicas. Así, una factura es una entidad que conecta al cliente con uno o más productos; una orden de compra es una entidad que conecta uno o más ítem a comprar con un proveedor.

7.4.2 Seis tipos básicos de consulta

Podemos describir los seis tipos básicos de consulta en función de entidades, sus atributos y los valores que pueden tomar los atributos.

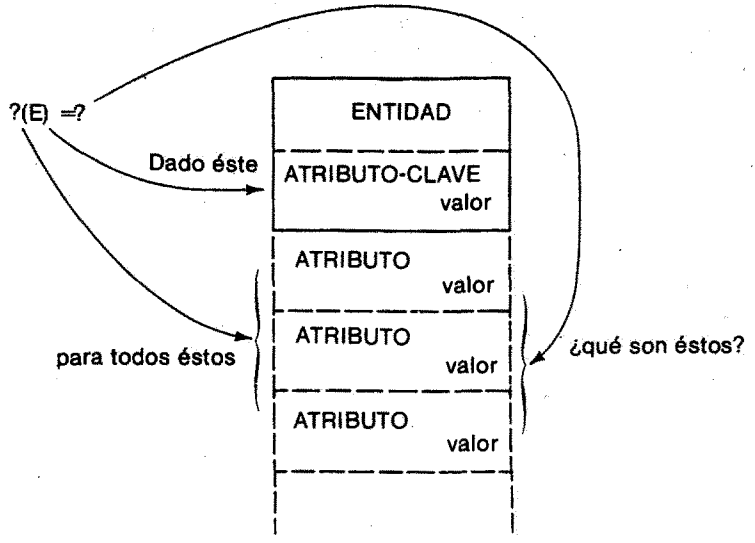
Consulta del tipo 1. ¿Dada una entidad particular (E), cuál es el valor de un atributo particular (A)? Por ejemplo,

"Dado el repuesto número B232 ¿cuál es el valor de su peso?

La notación convencional para este tipo de consulta es

$$A\,(E) = ?$$

donde A representa un determinado atributo de una entidad E (cuyo atributo clave es conocido) y el signo de interrogación significa el valor desconocido. La consulta puede mostrarse como sigue

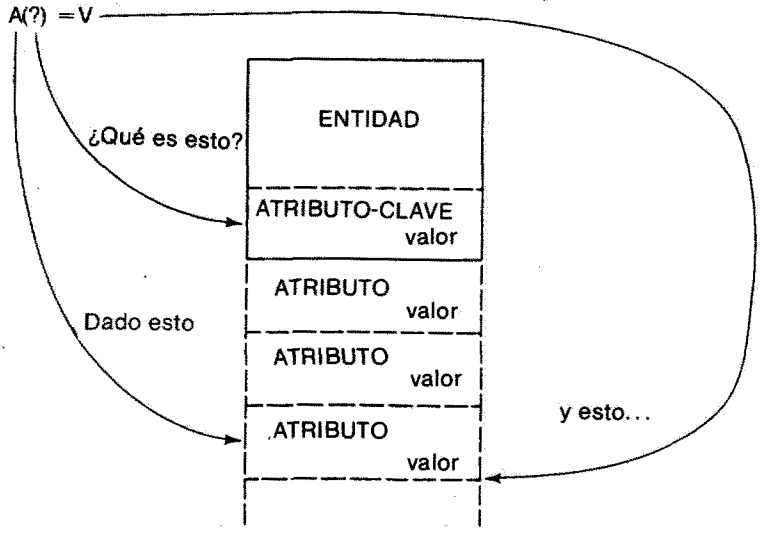


Consulta del tipo 2. ¿Dado un valor de un atributo particular, cuáles son las entidades que se especifican? Por ejemplo

"Dado un peso de dos libras (atributo) ¿qué repuestos (entidades) tienen ese valor?

"Dada la jerarquía de INGENIERO ¿qué empleados tienen esa jerarquía?"

Esta consulta es la inversa del tipo 1; en realidad la consulta del tipo 2 corresponde al caso del archivo invertido, que ya se discutió en la Sec. 7.2. El tipo 2 requiere un índice secundario si está relacionado con atributos no-clave. Podemos dibujar un diagrama como el siguiente:



A menudo se desea especificar más de un valor para la búsqueda; por ejemplo, se puede decir, "Dado un peso *menor que* 2 libras, ¿qué repuestos tienen este rango de valores? Así, la forma más general de la consulta tipo 2 es,

$$A(?) \begin{Bmatrix} = \\ \neq \\ > \\ < \end{Bmatrix} V$$

donde los símbolos tienen su significado usual (≠ Significa "desigual", > significa "mayor que", y < significa "menor que").

Consulta del tipo 3. Dada una entidad particular y un valor, ¿qué atributo(s) de la entidad coincide con este valor? Esta es una pregunta rara, empleada mayormente cuando varios atributos describen la misma propiedad en diferentes instantes de tiempo. Por ejemplo,

"Dado el empleado número 26222 y salarios de $15.000, ¿en qué años se excedió este valor?"

"Dado el producto número B232, ¿qué dimensión (si existe alguna) excede los 30 centímetros?"

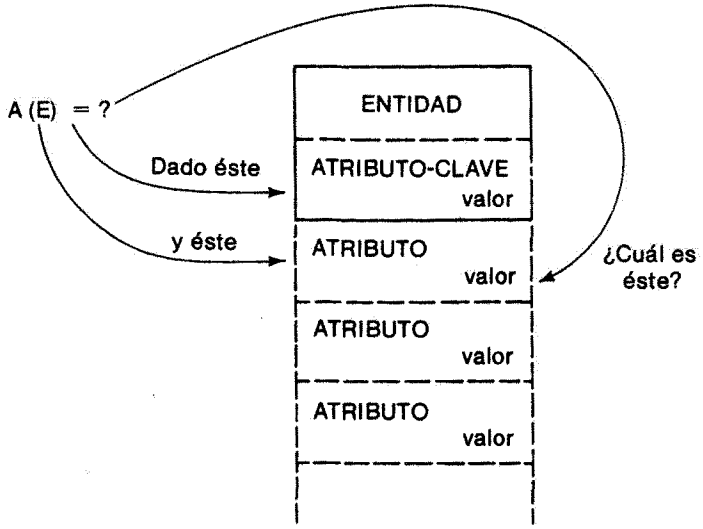
La notación es

$$?(E) \begin{Bmatrix} = \\ \neq \\ > \\ < \end{Bmatrix} V$$

Consulta del tipo 4. El tipo 4 es similar al tipo 1, excepto que se pregunta por todos los valores de todos los atributos, en lugar de un solo atributo.

"Listar los detalles del empleado número 26222."

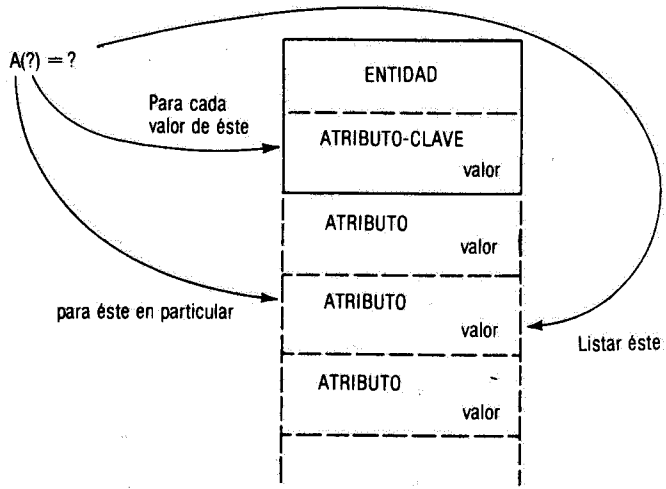
"Déme el perfil del repuesto número B232."



Consulta del tipo 5. Es similar al tipo 2, pero, como el tipo 4, es también global dado que pregunta por todos los valores de un atributo, para todas las entidades.

"Listar los pesos de todos los repuestos."

"Listar los salarios de todos los empleados."



Consulta del tipo 6. Es el equivalente global del tipo 3, preguntando por todos los atributos de todas las entidades que tienen un cierto valor.

"Listar todos los empleados que hayan ganado más de $22.500 en algún año."

"Listar todos los repuestos que tengan cualquier dimensión mayor que 15 centímetros."

La notación es del tipo

$$?(?)\left\{\begin{matrix}=\\ \neq\\ >\\ <\end{matrix}\right\}V$$

Como el tipo 3, no se presenta a menudo en la práctica.

7.4.3 Variaciones en los tipos básicos de preguntas

Valores múltiples. Dondequiera que se haga una pregunta por un valor en una consulta, siempre existe la posibilidad de poder combinar preguntas múltiples. Podemos decir, en una consulta del tipo 2, "¿Qué repuestos tienen un peso mayor que 2 libras *y* están pintados de colorado o azul *y* tienen una longitud mayor que 70 centímetros?" o "¿Qué empleados poseen el título de ingeniero electrónico y hablan fluidamente el idioma árabe?"

Ordenamiento y cuotas. Como un refinamiento posterior, la respuesta a la consulta podrá necesitar un ordenamiento, tanto ascendente como descendente sobre algún atributo. La consulta se podrá referir a una sola entidad o a un grupo específico de entidades que

satisfagan el criterio. "Déme todas las facturas sin pagar, ordenadas por importes decrecientes". "Déme cualquier proveedor de repuestos B232 ubicado en Chicago".

7.5 BUSQUEDA DE LAS NECESIDADES Y PREFERENCIAS DE LOS USUARIOS

Las necesidades de acceso inmediato a los datos de los usuarios surgen a lo largo de todo el período de estudio y de análisis del sistema, como parte del proceso iterativo de reunión de la información, de proposición de soluciones de prueba, de reunión de más información, etc. En algunos casos los gerentes usuarios no expresarán ninguna necesidad de acceso inmediato a los datos. Esto puede ocurrir por alguna de estas dos razones. Realmente habrá poco o ningún requerimiento empresario de acceso inmediato, como en el caso de una aplicación relacionada con un análisis histórico pre-determinado de eventos pasados. Más generalmente, la razón por la cual no se expresa se debe a que el usuario se ha acostumbrado a trabajar con los informes generados por un sistema en lote y no ha captado el hecho de que ahora la tecnología le permite tener una "ventana abierta en el computador". Después de todo, los últimos tiempos en que muchos gerentes tuvieron acceso inmediato fue allá por los días de la contabilidad de tarjetas de mayor, cuando podían recorrer un talonario de tarjetas, acceder por el nombre al registro de proveedores, e inspeccionar la historia de las transacciones. En estas circunstancias el analista tiene que hacer una tarea de actualización, explicando las posibilidades de la técnica actual y ayudando a los gerentes a pensar nuevamente cómo el acceso inmediato puede ser de valor para ellos.

Por otra parte, un número creciente de gente en la empresa y en el gobierno está muy bien enterada de la capacidad de recuperación de la información a través de la computadora y puede presentar al analista una elaborada "lista de deseos". El analista debe hacer frente al problema de entusiasmar al primer grupo de gerentes para que utilice más efectivamente el computador y de cuidarse de prometer al segundo grupo algo que no tenga una adecuada relación costo-efectividad.

7.5.1 El acceso operativo vs. el acceso informativo

Muchos de los accesos inmediatos a los almacenamientos de datos en un sistema tendrán relación con el procesamiento de transacciones de rutina, en lugar de responder a consultas de información. Algunos ejemplos de accesos *operativos* son:

1. Recuperar la dirección de un cliente para emplearla en la validación de un pedido.
2. Recuperar la historia de pago de un cliente para verificar el crédito.
3. Recuperar el nivel de inventario de un producto para procesar un pedido.

Típicamente estos accesos son necesarios para implementar algunas decisiones estratégicas de diseño; una vez que se ha decidido que el sistema tendrá entradas de datos en línea por CRT en el Departamento de Ventas, se desprende que se necesitará acceso inmediato a los archivos de los clientes por lo menos una vez por cada transacción. En forma similar, si uno de los objetivos del sistema es mantener actualizado el nivel de inventario durante el día, esto implica el acceso inmediato al archivo de inventario. El analista es responsable de definir la naturaleza de los accesos operativos, sus volúmenes probables (porque el diseñador necesita conocerlos), y cualquier consideración de seguridad involucrada. No está específicamente interesado en el valor para la empresa de cada acceso operativo; ya sea que se provea o no el acceso.

Por otra parte, muchos de los accesos que se han discutido en este capítulo son importantes pero no obligatorios. Su valor reside en la ayuda que prestan a los diversos niveles de administración para tomar decisiones más rápidas o para suministrar respuestas más rápidas a las consultas de los clientes. Si estos accesos *informativos* no son inmediatos, la empresa no

se parará; solamente no será conducida tan bien o tan armónicamente. Es con el acceso informativo, en oposición al operativo, que el analista puede hacer la gran diferencia costo-efectividad del sistema final; si los accesos informativos pueden ser identificados y los más importantes incorporados al diseño mediante índices secundarios o de otra manera, el conjunto de usuarios obtendrá el máximo beneficio del sistema. Si el analista no es capaz de predecir los accesos inmediatos que el usuario desea realizar, la información seguirá siendo necesaria, pero será provista no tan rápidamente y a un costo mayor.

7.5.2 Obtención de un listado de deseos compuesto

Como se mencionó, en los contactos iniciales con los diversos gerentes y supervisores que serán los usuarios del sistema, el analista encontrará una mayor o menor demanda de accesos inmediatos de datos. Supongamos, que en lugar de la Corporación CBM, el analista está involucrado en las primeras etapas de un sistema de información de comercialización, esta vez para la Compañía Sunray Engineering, que fabrica instalaciones de energía solar para la industria de la construcción. Los vendedores de Sunray entran en contacto con contratistas y arquitectos para discutir sus proyectos de construcción y como resultado preparan las propuestas de costos para el equipamiento solar basado en los componentes normales que fabrica la compañía. Algunas propuestas terminan en pedidos, que son enviados a los clientes para ser instalados en edificios. El sistema de información a ser desarrollado deberá conservar un registro de estas propuestas, los pedidos subsiguientes y las entregas, alimentando el sistema de cuentas a cobrar con la información necesaria para facturar a los clientes una vez que se enviaron los productos. Como resultado de las discusiones iniciales con los diferentes miembros de la gerencia, el analista tiene las siguientes anotaciones sobre los probables usos de los almacenamientos de datos:

Contralor. Necesita poder analizar el valor de los pedidos a ser enviados en cualquier período futuro específico, predecir el importe a facturar y como consecuencia el flujo de caja.

Gerente de ventas. Desea saber los pedidos y propuestas pendientes para cualquier cliente, o cualquier vendedor y las comisiones de cualquier vendedor.

Gerente de planeamiento de productos. (un reciente egresado de la Escuela de Administración de Empresas de Harvard). Desea acceso en línea a la historia de los pedidos, en base al tamaño del pedido, mes de los pedidos (disposición y envío), componentes, tipo del uso final y región geográfica.

Supervisor administrativo de ventas. Desea poder encontrar el número del cliente, dado el nombre, y contestar las consultas del cliente sobre el progreso de los pedidos en proceso de envío.

Estos requerimientos se refieren y son adicionales a los flujos normales de datos de ingreso de propuestas, propuestas de precios, ingreso de pedidos, impresión de notas de embarque, etc. Suponemos, basados en el análisis de flujos de datos y del contenido de los almacenamientos de datos del tipo descrito en los Capítulos 3 y 6, que los almacenamientos de datos que describen al CLIENTE, PEDIDO, PROPUESTA y VENDEDOR han sido identificados con los atributos indicados en la Fig. 7.4. Obsérvese que PRODUCTO y PEDIDO tienen claves compuestas (cada una de ellas con dos elementos de datos) y que contienen varios grupos repetitivos, por ejemplo, COMPONENTES.

Podemos indicar los accesos deseados por varios gerentes dibujando flechas de entidad a entidad. Por ejemplo, el gerente de ventas desea acceso a PEDIDOs dado un CLIENTE y a PEDIDOs dado un VENDEDOR. Podemos ver estos accesos en la Fig. 7.5, donde las flechas representan los accesos inmediatos. Estas flechas no representan ninguna clase de flujo o ninguna clase de control o de jerarquía, sino solamente un requerimiento de capacidad de acceso a una entidad, dada una instancia de otra entidad. Los diagramas como el de la Fig. 7.5 son diagramas de datos de acceso inmediato (DIAD, en inglés, Data Inmediate Access

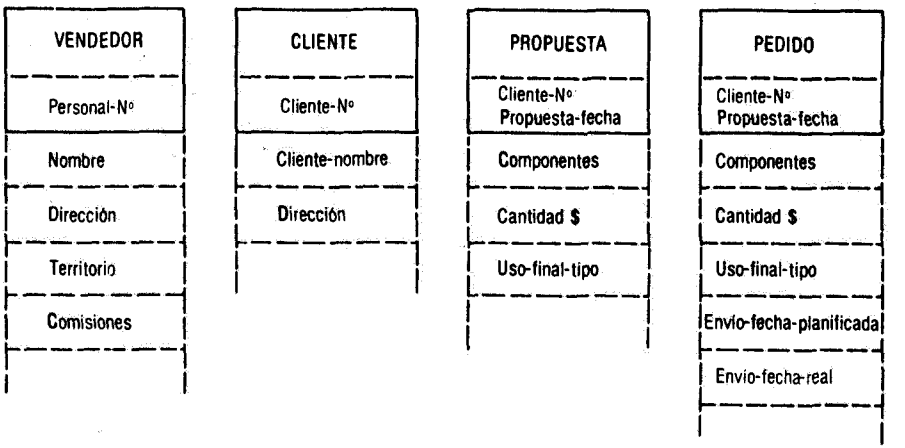


Figura 7.4 Entidades y atributos en Sunray Engineering.

Diagram), como los que habíamos descrito brevemente en el Capítulo 2. Son muy útiles para resumir todos los requerimientos de acceso inmediato de un grupo de gerentes antes de tomar una decisión acerca de la estructura del archivo físico o de la base de datos. Podemos observar que en la Fig. 7.5 el acceso inmediato desde CLIENTE a PEDIDOs parece lógicamente muy simple, ya que CLIENTE-Nº forma parte de la clave de PEDIDO. Esto puede no ser físicamente muy conveniente ya que algunos sistemas de "software" de manejo de archivos no pueden recuperar todos los registros que contienen un elemento de datos que es solo parte de la clave. Por otra parte en IMS la recuperación puede ser simple si los segmentos se disponen adecuadamente. El punto es que *no debemos preocuparnos* aún por la implementación física; debemos analizar todos los requerimientos para acceso inmediato en esta etapa y ocuparnos de la mejor forma de implementación cuando conozcamos valores y volúmenes.

Podemos notar, también, que no surge una forma obvia de acceso a PEDIDOs desde VENDEDOR. Se puede incluir la clave de cada PEDIDO como un atributo de vendedor, o

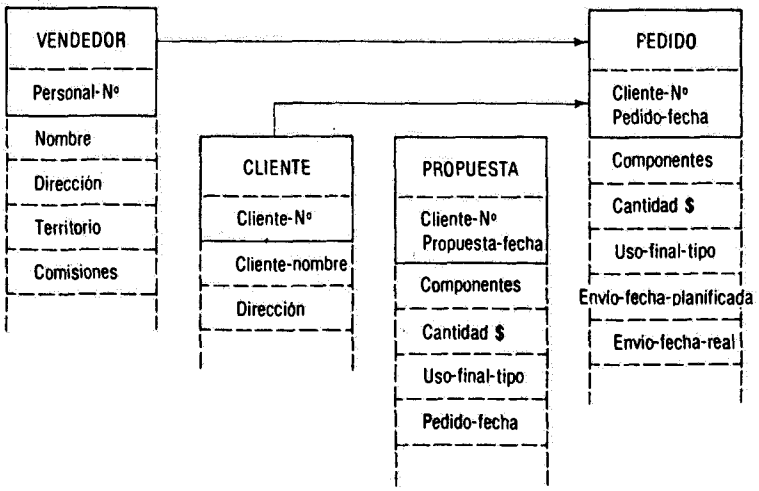


Figura 7.5 Las flechas indican los accesos inmediatos.

podemos incluir el número de VENDEDOR como un atributo de cada PEDIDO y crear un índice secundario. Por ahora no nos interesa cual.

El usuario pide algunos accesos que implican solamente la recuperación de una entidad basada en su atributo clave. El requerimiento del gerente de ventas para tener acceso a las comisiones de cada vendedor podría ser el siguiente: dado el número del vendedor, debe poder recuperarse el registro especificado, que incluye la comisión. No es necesario indicar especificamente un requerimiento como éste pues surge de la presencia del atributo.

Por otra parte, cuando una entidad deba ser recuperada, basándose en un atributo no-clave (la situación del índice secundario discutido en la Sec. 7.2), debemos indicar este hecho para la creación de una nueva entidad a partir del atributo. Por ejemplo, supongamos que el gerente de ventas desea acceso a cada vendedor por el nombre en lugar del número; se deberá indicar este hecho como se muestra en la Fig. 7.6.

El requerimiento del contralor también puede representarse de esta manera, ya que consiste en tener acceso a PEDIDOs sobre la base de un atributo no-clave de PEDIDOs. La Fig. 7.7 muestra el agregado de este camino de acceso inmediato.

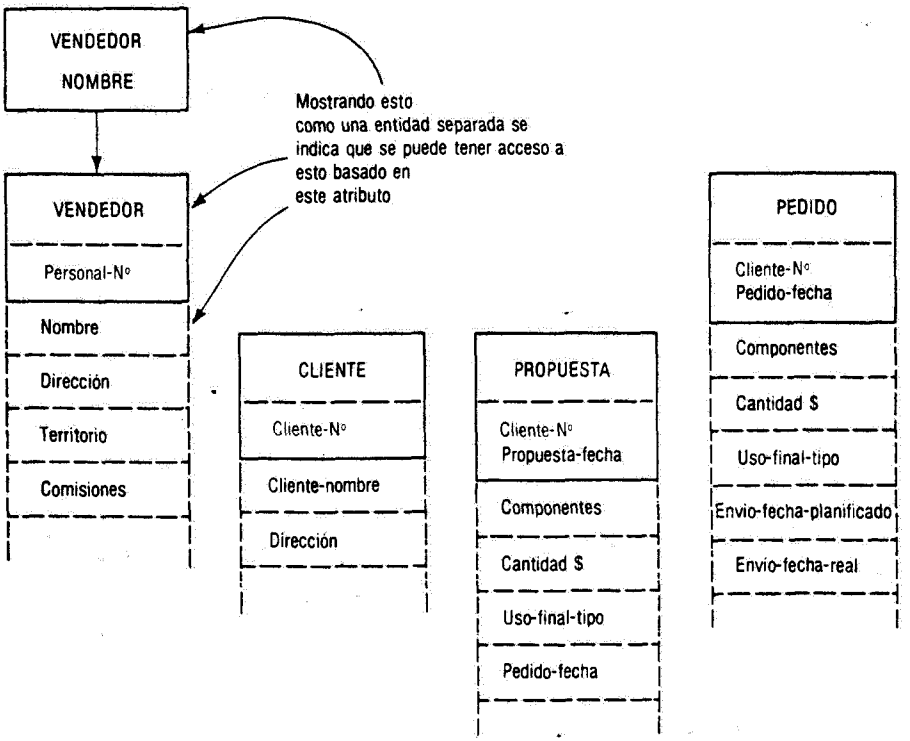


Figura 7.6 Diagrama de Datos de Acceso Inmediato, mostrando parte de los requerimientos del gerente de ventas.

La Fig. 7.8 muestra el conjunto completo de consultas que representan la suma total de las listas de deseos del contralor, gerente de ventas, gerente de planeamiento de productos y supervisor administrativo de ventas.

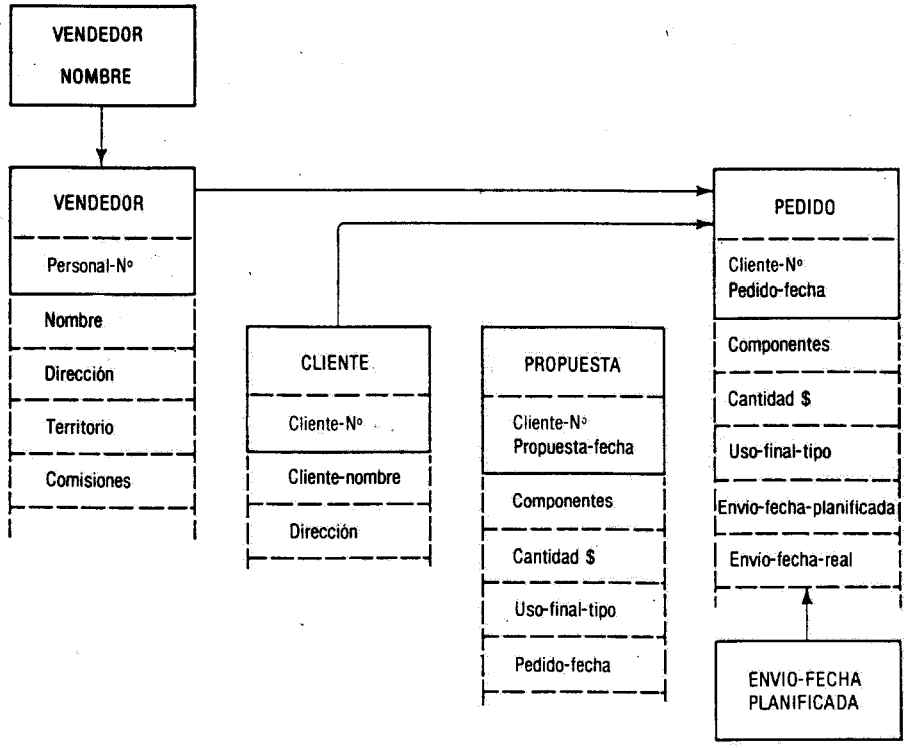


Figura 7.7 Diagrama de Datos de Acceso Inmediato con los requerimientos del contralor, agregados.

7.5.3 Refinación del listado de deseos

Si no se objeta el presupuesto del proyecto y si cada gerente usuario expresa una fuerte necesidad de acceso inmediato, la única información adicional que necesitará el analista es la frecuencia probable de cada acceso. A menudo, sin embargo, no todos los pedidos podrán ser satisfechos a bajo costo. Entonces el analista deberá enfrentar el problema de resolver cuáles accesos son menos valiosos y pueden ser degradados a una respuesta nocturna. Una forma de hacerlo es presentando el diagrama de acceso inmediato por turno a cada usuario, para discutir el significado de todos los diversos accesos. Aunque algunos de ellos no serán de interés para algún gerente, es importante mostrar la descripción completa a todos los responsables de tomar decisiones, explicando cómo cada acceso adicional aumenta el costo del sistema. Esta presentación, si se hace con mucho tacto, estimulará la imaginación de los gerentes para los posibles usos del sistema, a la vez que minimizará las posibilidades de fricción y de desilusión, en el caso que el sistema final no cumpla con toda la lista de pedidos del ejecutivo.

La presentación de un diagrama de datos de acceso inmediato puede ser seguido por un cuestionario que normalmente es llenado por el analista en conversación con cada gerente. La Fig. 7.9 muestra unas notas breves realizadas por un analista para explicar este cuestionario, y la Fig. 7.10 muestra el ejemplo de una página del cuestionario para Sunray Engineering.

Las preguntas están encuadradas para que cada gerente pueda poner lo más rápido y fácilmente posible un valor aproximado a cada respuesta y dar una idea de la frecuencia

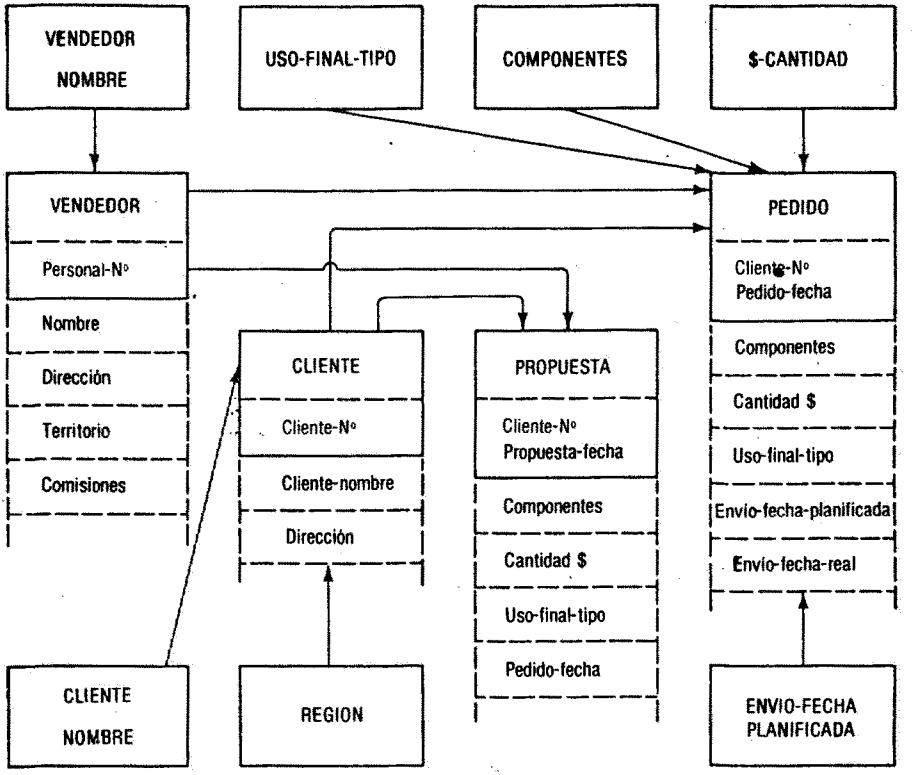


Figura 7.8 Diagrama de Datos de Acceso Inmediato compuesto, de Sunray Engineering.

Estamos en el proceso de decisión acerca del costo-efectividad de distintos dispositivos de un sistema integrado de comercialización. De la información que usted y otros gerentes entreguen, redactaremos una lista de todas las posibles consultas de información que se pueden formular al sistema. Sería de mi agrado revisarlas con usted, ver cuáles encuentra útiles y qué clase de uso podrá hacer de ellas cuando el sistema esté disponible.

Cada pedido de información ha sido especificado en función de cómo debe ser provista por el sistema (en un formulario de consulta o tecleada en una terminal), y cómo se la devolverá el sistema. Por ejemplo, podrá dar al sistema "Cliente número" y desear que le conteste "Detalles de la última factura". Por cada consulta, le solicito que me dé una idea de cuántas veces deseará hacerla y una indicación de qué valor tiene para usted obtener la respuesta al instante, como alternativa de tener el informe a la mañana siguiente, o como alternativa de disponer de un listado para consulta (digamos los nombres y números de clientes).

Estoy seguro de que usted sabe que cuesta mucho más una respuesta inmediata, en segundos, comparada con la preparación de la respuesta durante la noche para entregarle un informe al día siguiente, o con la provisión de un listado estático. Debemos estar seguros de que no estamos incorporando al sistema dispositivos "lindos-para-tener", de manera que para cada requerimiento yo le preguntaré cuánto cree que vale para usted tener la respuesta al instante, suponiendo que la podría tener mañana por la mañana al costo de $1 (no le estoy diciendo que éste será el costo, sino que busca ser una medida del valor para usted). Por supuesto, si tiene que tener la información al instante para su tarea, la pregunta carece de significado; por favor dígame si éste es su caso.

Figura 7.9 Breves notas para el cuestionario de acceso inmediato.

Nombre del Gerente _____ Posición _____ Fecha de la entrevista _____ Analista _____

Basado en este ítem que conoce...	Desea que el sistema le diga...	¿Cuál es la frecuencia media probable de este tipo de consulta en cuanto a usted concierne?*	Si pudiera tener esta información mañana por $1, ¿aproximadamente ¿cuánto valdría para usted tenerla inmediatamente?
Nombre del vendedor	Pedidos no despachados de este vendedor		
Nombre del vendedor	Propuestas pendientes de este vendedor		
Nombre del cliente	Pedidos no despachados para este cliente		
Nombre del cliente	Historia de pedidos despachados para este cliente		
Nombre del cliente	Propuestas pendientes para este cliente		
Dos fechas cualesquiera	Todos los pedidos con fecha de envío planificada en este intervalo		
Componente número	Pedidos que utilizan este componente, analizadas por mes		

* > 1/min = muy alta, > 10/hora = alta, > 10/día = mediana, > 10/semana = baja, ≤ 10/semana = muy baja

Figura 7.10 Ejemplo Sunray Engineering.

probable. No se está tratando de comparar la importancia de una respuesta con respecto a las otras, sino solo el valor relativo aproximado. A pesar de ello, los gerentes tienden a dar valores más altos a los accesos más importantes; podrán agregar comentarios tales como "No lo necesito muy a menudo, pero cuando lo necesito, lo necesito pronto y a cualquier precio" (por ejemplo, un análisis de la tarifa exitosa de la reciente propuesta, como guía para cotizar en forma competitiva).

Con accesos relacionados con datos que solo se modifican lentamente, tales como recuperar el número del cliente dado el nombre del cliente, la alternativa está en comparar la búsqueda de la respuesta en un listado, contra tener la respuesta presentada o impresa en una terminal. Con 50 clientes el acceso a la terminal parecerá un lujo; con 5000 clientes la búsqueda en el listado podrá insumir mucho tiempo. Le corresponde a cada gerente decidir sobre el valor relativo que tiene la facilidad para él y su gente.

Cuando se han comparado los resultados de los cuestionarios de las entrevistas con cada gerente, puede obtenerse por lo general una buena indicación del valor global de cada respuesta inmediata, junto a una indicación de la frecuencia total probable de utilización. Si el cuestionario no puede ser contestado con toda claridad debido a la gran variedad de usos que los gerentes pueden hacer de los datos, es muy posible que la solución esté dada por la disponibilidad de un lenguaje general de consulta.

7.6 CONSIDERACIONES SOBRE SEGURIDAD

Mientras que la cuestión de seguridad de datos contra robo, destrucción o cambios concierne al analista y al diseñador en cada etapa del diseño [7.4], la misma toma particular importancia cuando se analizan los requerimientos de acceso inmediato. Los documentos confidenciales generados por computadora pueden guardarse bajo llave y desmenuzarse

cuando no se los requiere más. Los datos confidenciales en un almacenamiento de datos al que se ha provisto de acceso inmediato vía terminal pueden —a menos que se tomen precauciones— ser leídos por personas no autorizadas, sin dejar rastros.

En consecuencia, el analista deberá establecer *quién* está autorizado para realizar cada acceso. Por ejemplo, el gerente de ventas deberá tener acceso a las comisiones ganadas por los vendedores. ¿Podrá el supervisor administrativo de ventas tener también este acceso? Si se dispusiera de un acceso a las propuestas pendientes y a sus importes en dólares, ¿quién detendría a alguna persona pagada por la competencia a que leyera esta información en un CRT?

El diagrama de acceso inmediato y el cuestionario, una vez confeccionados, son útiles como herramientas para considerar cada acceso por separado y decidir sobre los aspectos de seguridad de cada acceso. Esta es una cuestión que el analista deberá aclarar con la gerencia, explicando que es posible, dado el "software" adecuado, permitir a determinadas personas "ver" parte del almacenamiento de datos, pero no todo su contenido. También podría ocurrir que todas las terminales tengan acceso al nombre, dirección, teléfono y territorio del vendedor, pero que solamente algunos de los que utilizan la terminal en la oficina del gerente de ventas, provistos de la palabra clave —conocida solamente por el gerente de ventas, el presidente y el contralor— pudieran ver los importes de las comisiones individuales. La gerencia podría decidir que un amplio grupo de personas pueda tener acceso a las estadísticas de los importes en dólares correspondientes a las propuestas, pero que solo el vendedor de la cuenta tenga acceso al importe específico de una propuesta específica.

APENDICE

Paquetes generales de consultas

CULPRIT-Cullinane Corporation - 20 William Street - Wellesley, MA 02181
EASYTRIEVE - Pansophic - 709 Enterprice Drive - Oakbrook, IL 60521
MARK IV - Informatics - 65, Route 4 - Riveredge, N.J. 07661
Esta no es una lista exhaustiva: Consultar otros paquetes en las publicaciones especializadas.

BIBLIOGRAFIA

7.1 *IQF General Information Manual, GH20-1074,* IBM Corp., White Plains, N.Y.
7.2 J. Martin, *Principles of Data-Base Management,* Prentice-Hall, Englewood Cliffs, N.J., 1976, Cap. 17.
7.3 J. Martin, *Computer Data-Base Organization,* Prentice-Hall, Englewood Cliffs, N.J., 1975, Cap. 5.
7.4 J. Martin, *Security, Accuracy and Privacy in Computer Systems,* Prentice-Hall, Englewood Cliffs, N.J., 1973.

Ejercicios y puntos de discusión

1. En los términos de este capítulo, ¿qué relación existe entre las páginas amarillas y las páginas blancas de su guía telefónica? ¿Es posible realizar otras inversiones de los datos de la guía telefónica?
2. Revise los sistemas de su empresa que clasifican o exploran archivos para producir informes. ¿Existen sistemas en los que sería deseable tener un índice secundario y proveer acceso inmediato a los datos?
3. Si cualquiera de los sistemas de su empresa usa índices secundarios, averigüe cuál es la magnitud del índice comparado con el archivo principal. ¿Con qué frecuencia se reorganiza el índice? ¿Se emplea el índice con fines operativos o informativos? ¿Cuántos accesos se hacen en un día promedio?
4. Si tiene acceso a un archivo o base de datos jerárquicos, dibuje un diagrama de su estructura. ¿Qué previsiones hay que hacer para accesos que no descienden a través del camino jerárquico?
5. Familiarícese con una de las facilidades de consulta general mencionadas en este capítulo. En las

empresas que la han empleado, ¿en cuánto se ha aliviado la carga del departamento de programación? ¿Qué factores limitan su uso más amplio? ¿Cuántos meses-hombre gastó su empresa el año pasado en la confección de programas para informes "por una única vez"?

6. Tome un sistema del que sepa que tiene accesos múltiples a los datos almacenados y describa cada acceso en función de los tipos de la Sec. 7.4.
7. Dibuje un diagrama de datos de acceso inmediato del sistema (o de cualquier otro sistema que tenga accesos inmediatos múltiples). Reúna la información que pueda sobre el valor de los diversos accesos a la empresa.
8. Hable con uno de los expertos en diseño de base de datos de su empresa y pregúntele a el (o ella) qué información le agradaría, como ideal, que el analista de sistemas le proveyera para poder realizar el diseño óptimo en lo que hace a costo-efectividad.

8

EMPLEO DE LAS HERRAMIENTAS: UNA METODOLOGIA ESTRUCTURADA

En los Capítulos 3 al 7, examinamos en detalle las diversas herramientas y técnicas del análisis estructurado de sistemas. En este capítulo repasaremos el uso al cual se pueden aplicar en la primera parte del desarrollo del sistema, comenzando por el punto donde se plantea por primera vez la cuestión del esfuerzo de desarrollar un sistema, continuando con las fases de estudio y análisis, y concluyendo con el diseño físico. (Las técnicas de diseño estructurado, que utiliza el producto de nuestro análisis, se describen en el próximo capítulo.) Para ello bosquejamos una *metodología para el desarrollo de sistemas estructurados,* esto es, una aproximación general para la construcción de sistemas comerciales de procesamiento de datos, construidos alrededor del análisis y diseño estructurado.

8.1 EL ESTUDIO INICIAL

A medida que las computadoras se hacen más y más baratas, nuevas organizaciones se encuentran con que pueden obtener ventajas de la automatización, aun aquellas que tienen menos de 50 empleados. Si una organización no tiene todavía una computadora, las herramientas de análisis estructurado son útiles para analizar el sistema manual existente, para entenderlo con claridad previamente a la toma de decisión sobre si se debe instalar una computadora, y decidir qué partes del sistema se automatizarán.

La secuencia de actividades establecidas en este capítulo es de aplicación al desarrollo de cualquier sistema de información, esté automatizado o no. Debemos puntualizar, sin embargo, que la metodología ha sido desarrollada en primer término para situaciones donde ya existe un sistema computadorizado, vinculado con sistemas manuales, procedimientos administrativos y quizás con otros sistemas computadorizados, y se trata de efectuar algunas mejoras en este conjunto de sistemas.

Las preguntas que deberán contestarse en un estudio inicial son:

— ¿Qué tiene de malo la situación actual?
— ¿Qué mejoras son posibles?
— ¿Quién será afectado por el nuevo sistema?

En una organización de cualquier tamaño es usual tener una corriente continua de requerimientos por parte de los gerentes a fin de mejorar el servicio de procesamiento de datos. Mientras que algunos de estos requerimientos puedan lograrse mediante una mejor *operación*, como ser mejorar el tiempo de respuesta o utilizar una facilidad de consulta existente, y otros puedan lograrse *perfeccionando* los sistemas existentes, como ser proveer un nuevo informe a partir de datos ya existentes, muchos otros implican el *desarrollo de un nuevo sistema.* Nos interesan principalmente estos nuevos desarrollos.

Muchas organizaciones encuentran que la demanda de nuevos sistemas es varias veces superior a sus posibilidades de construirlo. Realmente, a medida que se implementen más y más sistemas, la tarea de mejorarlos adecuadamente ocupa más y más tiempo del personal disponible, en algunos casos del 50-60% del equipo humano de desarrollo. En consecuencia, los sistemas que se elijan para su construcción deberán ser aquellos que presenten el máximo beneficio para la mayoría de los gerentes de la organización, con la esperanza que se ajusten bien al plan global de desarrollo del procesamiento de datos en la organización. El estudio inicial (a veces denominado *evaluación de requerimientos*) es en parte un proceso de filtro para suprimir los requerimientos de desarrollo que no van a ser de suficiente utilidad y puedan hacerse en forma relativamente rápida y barata (ya que realizar un estudio involucra personal y tiempo, de los que se está escaso).

Este estudio inicial podrá insumir de 2 días a 4 semanas (excepcionalmente más). El analista deberá estudiar los requerimientos (si son escritos) y reunirse con los gerentes para obtener los antecedentes de la situación y comenzar a analizar el valor probable del nuevo sistema. Deberá hacer preguntas como "¿Puede indicar qué cantidad de ingresos estamos perdiendo a consecuencia de las deficiencias del sistema actual?" y "¿En qué costos estamos incurriendo, que podrían evitarse mediante un mejor sistema?" y "¿Es posible hacer una estimación en dinero sobre el servicio mejorado que podríamos dar con un mejor sistema?". Cuando el sistema se necesita para poder cumplir con un requerimiento estatutario, debe determinar la fecha en la cual se lo requiere, así como también las penalidades por demora. Los gerentes deberán ser invitados a definir en términos muy generales los beneficios intangibles que ven surgir del sistema mejorado e identificar otras áreas de la organización que se verán afectadas. Si es necesario, los gerentes de esas otras áreas deberán ser entrevistados para obtener una información similar.

Es útil tener presente las razones fundamentales que dan lugar al desarrollo del nuevo sistema. Las organizaciones pueden estar aprovechando una oportunidad (para aumentar ingresos, bajar costos, o mejorar servicios) o estar reaccionando a una presión, ya sea estatutaria o de la competencia. Por ejemplo, cuando un banco es el primero del área en instalar una facilidad que permita a los clientes pagar facturas mediante un teléfono con teclado, entrando el código del pagador seguido del importe de la factura, está aprovechando claramente una oportunidad de incrementar los ingresos a través de la mejora de un servicio. La competencia no lo está forzando a hacerlo, ni tampoco el gobierno. La justificación del sistema deberá basarse en estimaciones empresarias del valor del nuevo servicio para atraer depósitos. El segundo banco del área que instale este sistema lo hará en respuesta a la presión competitiva, tal vez como resultado de ver cómo los depositantes mueven sus cuentas hacia el primer banco. En las reparticiones públicas el aspecto de los ingresos es menos común, y la reducción de costos, sumado a la mejora del servicio al electorado, son las justificaciones más usuales. El acróstico IRACIS (Increase Revenue, Avoid Cost, Improve Service - Aumentar Ingresos, Bajar Costos, Mejorar Servicios) ha sido sugerido a modo de resumen de todos estos objetivos empresarios generales de los sistemas.

Dentro de estas categorías generales, hay varias *formas* en que el nuevo sistema puede contribuir:

1. Proveyendo la información existente, pero más rápidamente, por ejemplo, proveyendo un acceso inmediato a los datos que ahora aparecen en un informe semanal.
2. Proveyendo información más actualizada (fresca), por ejemplo, proveyendo datos sobre ventas de la noche anterior en lugar de las del último mes, o proveyendo balances de cuenta hasta el último minuto en lugar de balances de cierre del día de ayer.
3. Proveyendo información más exacta, ya sea en términos de exactitud aritmética, como en el caso de un informe que se presenta sujeto a ajuste cuando existen transacciones que no han podido ser procesadas; o en términos que le permitan representar en forma más aproximada el mundo real, como en el caso en que las cantidades de inventario están conciliadas con las cantidades actualmente disponibles y en trámite de pedido. Algunos

usuarios hablan de mayor exactitud en la información y quieren significar información más actualizada.

4. Proveyendo información basada en más elementos de datos, como en el caso de un sistema que nunca ha contado con la historia del entrenamiento de un empleado, y que es remplazado por otro que permite obtener detalles sobre los cursos realizados y la experiencia ganada. Esto implica la creación de un nuevo almacenamiento de datos o un agregado al contenido de uno existente.

5. Proveyendo información basada en nuevas funciones lógicas. Cuando se calcula la proyección de la tendencia de ventas a partir de los valores de ventas mensuales existente empleando el análisis de series de tiempo, los almacenamientos de datos permanecen iguales pero se ha introducido un nuevo proceso lógico.

Por supuesto, frecuentemente un requerimiento se expresa como una combinación de estos tipos: ¡El usuario desea capturar más información que en la actualidad, obtenerla más actualizada, analizarla de una manera más compleja, y tener la respuesta más rápidamente! Acá le resulta útil al analista estar atento a estas dimensiones implícitas, de manera de poder fraccionar un requerimiento complejo en sus componentes.

Así como entrevistar a los gerentes apropiados, el analista deberá obtener y revisar toda documentación sobre las limitaciones del sistema actual. Deberá revisar el plan de desarrollo de procesamiento de datos (¡si existe alguno!) para ver cómo esta área se ajusta a él, y deberá hallar toda la información que le sea posible acerca de lo que están haciendo otras organizaciones similares en esta área.

Al finalizar el estudio inicial, el analista deberá estar razonablemente seguro acerca de la magnitud de los beneficios que podrán resultar de un nuevo sistema. Deberá saber si un requerimiento tuvo origen en un rapto de inspiración de un gerente, pero solo puede ahorrar $10.000 al año o si, por el contrario, es algo que puede hacer crecer la facturación de $100 millones entre el 1% y el 3%. Probablemente estará en posición de dibujar un diagrama de flujo de datos general del sistema existente y sus interrelaciones; si desea conocer más acerca del sistema actual durante el estudio inicial, el diagrama logico de flujo de datos es una herramienta muy útil para reunir todos los fragmentos.

Deberá ser capaz de estimar el tiempo y el costo necesarios para realizar un estudio detallado (la próxima fase del desarrollo) podrá tener una idea muy aproximada de cuánto podrán costar algunos posibles nuevos sistemas. Deberá ser muy cauto cuando indique cualquier total estimado del proyecto; existe una gran presión de los usuarios y de la gerencia superior en establecer presupuestos del proyecto lo antes posible. Con todo, nuestra experiencia indica que el precio en firme no puede darse hasta que se conozca el número y la complejidad de los módulos del sistema, esto es, hasta después de haberse producido un diseño físico firme. El mejor curso es indicar un amplio y confortable margen de costos y tiempos para todo el proyecto y entregar solamente una cantidad en firme en la próxima fase. Luego, sintetizando, el resultado de un estudio inicial podría expresarse como sigue:

> Una investigación inicial de dos semanas del sistema de compras actual indica que, basado en estimaciones del gerente X y del gerente Y, estamos pagando entre el 1% y el 2,5% más por las materias primas que lo que pagaríamos si pudiéramos centralizar los requerimientos y obtener la máxima ventaja posible de los descuentos por cantidad y pronto pago ofrecidos por los vendedores. Nuestra proyección de gasto en materias primas durante este año es de $20 millones, indicando la posibilidad de una economía potencial anual entre $200.000 y $500.000. Además, un sistema de compras mejorado podrá tener una cantidad de otros beneficios, que no podemos cuantificar todavía, incluyendo una reducción de compras urgentes y una reducción en el inventario. Basados en la experiencia de otras compañías, estimamos que desarrollar un sistema de este tipo costará entre $300.000 y $750.000 y llevará entre 15 y 30 meses. No se pueden estimar costos operativos con la información disponible. Aunque estas cantidades son necesariamente muy tentativas, nuestra conclusión es que este proyecto tiene suficiente potencial como para garantizar un estudio detallado, el cual permitirá:
>
> 1. Ajustar nuestras estimaciones de beneficios potenciales.
> 2. Establecer objetivos para un nuevo sistema.

3. Definir las funciones de un nuevo sistema y cómo se integrarán.
4. Ajustar nuestras estimaciones del costo de desarrollo.

El estudio detallado requerirá dos analistas durante 8 semanas, con un costo total de $16.000.

8.2 EL ESTUDIO DETALLADO

El resultado del estudio inicial deberá ser revisado por un nivel de gerencia apropiado. Muchas organizaciones cuentan con un comité de alta dirección que imparte instrucciones para el desarrollo del procesamiento de datos. Dependiendo de una combinación de prioridades, políticas y de los hechos establecidos en el estudio inicial, se podrá autorizar un estudio detallado. Deberá quedar claro a todos los interesados que el estudio detallado no representa un compromiso para la implementación del proyecto. En administración se dice, en efecto, "luce promisorio; cuénteme más". El estudio detallado cuenta con los hechos producidos por el estudio inicial para documentar las limitaciones del sistema actual con mayor detalle y mayor confiabilidad, y para llevar la comprensión de las funciones del sistema actual al nivel necesario para poder especificar un remplazo. Ahora discutiremos las actividades de un estudio detallado.

8.2.1 Definir con mayor detalle quiénes serán los usuarios de un sistema nuevo

Se puede concebir que la comunidad de usuarios está constituida en tres niveles:

1. Los gerentes de alto nivel con responsabilidad en los beneficios, cuyas áreas serán las afectadas y que se ven a sí mismos pagando el sistema. Este grupo ha sido denominado los *comisionados* ya que ellos comisionan el desarrollo del sistema de la función procesamiento de datos, así como serán posibles usuarios de sus salidas de información.
2. La gerencia intermedia y supervisores cuyos departamentos se verán afectados.
3. Los empleados administrativos y demás personal que trabajará directamente con el sistema mediante el uso de terminales o completando formularios de entrada e interpretando salidas para realizar sus tareas.

Los comisionados de sistemas, tales como el Presidente, el Vicepresidente de Ventas, o el Vicepresidente de Producción están habilitados para considerar el mediano plazo, el cuadro de 3-5 años y realizar esencialmente una decisión de inversión, a favor o en contra de un proyecto significativo, sobre esta base. Pueden evaluar beneficios intangibles, aunque no puedan valorizarlos. Tienden a interesarse mucho en los aspectos de un sistema que pueden incrementar su control real o aparente sobre la empresa, ya que una de las cosas de mayor importancia para un ejecutivo antiguo consiste en tomar la responsabilidad por alguna actividad con cuyos detalles no podrá mantenerse en contacto.

Cuando se considera que los sistemas servirán a más de un área funcional importante, la alta gerencia podrá desear diferentes cosas del sistema e impulsar a éste en diferentes direcciones. Es tradicional que los gerentes de ventas deseen tiempos cortos de entrega porque ello les permite dar mejor servicio al cliente y maximizar ventas, que es para lo que se les paga. Es igualmente tradicional que los gerentes de producción deseen tiempos largos de entrega porque ello les permite mejor programación de la producción y menor costo de producción, que es para lo que a *ellos* se les paga. Si el analista tiene mala suerte, él y el sistema se verán arrastrados a una lucha de poderes que le demostrará que ningún gerente se encuentra satisfecho con el informe borrador de objetivos y funciones. Evidentemente, el analista no puede resolver una situación como ésta: si sospecha que está tratando de servir a dos patrones que no están de acuerdo, deberá buscar el apoyo en su propia gerencia, la que

deberá, si es necesario, plantear la situación al gerente general para obtener una resolución.

Las gerencias intermedias y los supervisores tienden a estar menos preocupados con el cuadro estratégico y más interesados en el desempeño en el corto plazo de su departamento, respecto de su presupuesto. El analista debe soportar presiones para explicar las consecuencias de un nuevo sistema sobre los costos y el personal, y debe estar preparado para lidiar con la resistencia de los gerentes intermedios que temen que el desarrollo pueda quitarles poder y autonomía. Los gerentes intermedios a menudo están asediados por los problemas de conseguir y mantener personal capaz, y si el analista puede mostrar cómo el nuevo sistema puede facilitar este problema, ganará aliados.

Respecto del personal administrativo cuyo trabajo se verá impactado el analista deberá hacer todo lo posible para que su trabajo sea más placentero e interesante bajo el nuevo sistema. Durante las etapas de análisis y diseño, la gente deberá ser tranquilizada acerca del impacto del sistema sobre su trabajo mediante charlas y demostraciones. Como parte del estudio detallado, el analista deberá familiarizarse con el trabajo de oficina involucrado en el estudio. Si el tiempo lo permite y es factible, el analista deberá ejecutar personalmente una de las tareas (por ejemplo, la de empleado que atiende consultas telefónicas) por un corto período. Ello le dará una riqueza de información de primera mano que le sería difícil, si no imposible, obtener vía entrevistas.

Cuando el analista no se ha encontrado con la alta gerencia durante el estudio inicial, deberá tratar de obtener sus opiniones sobre objetivos y preferencias durante el estudio detallado. Lo ideal es que cada uno de los gerentes intermedios afectados sea entrevistado, pero usualmente el tiempo es tirano especialmente si están geográficamente separados. Por último el analista deberá obtener o preparar un organigrama actualizado de los departamentos de interés, sus gerentes/supervisores y sus funciones. Deberá determinar la cantidad de personal administrativo y su régimen natural de rotación, así como entender mejor sus tareas.

8.2.2 Construcción de un modelo lógico del sistema actual

A partir de la información del estudio inicial y de la información adquirida al definir la comunidad de los usuarios, el analista puede ahora producir un borrador del diagrama lógico de flujo de datos del sistema actual. Pero, ¿qué es exactamente el *sistema actual*? La preocupación de los usuarios tiene relación a menudo con los resultados empresarios de un área específica, digamos, compras. A medida que el analista investiga puede encontrar que están involucrados varios sistemas interconectados, algunos manuales, otros automatizados. En estos casos, puede existir el problema de *definir el límite del problema,* de decidir qué funciones deben ser consideradas como parte del estudio del sistema y cuáles no. Acá el diagrama lógico de flujo de datos es de gran utilidad; permite al analista reducir cada uno de los sistemas interconectados a una terminología común y ver cómo se adaptan. Algunas organizaciones requieren que sus analistas dibujen los diagramas de flujo de datos de cada sistema que se relacione con el área bajo estudio, especialmente los sistemas administrativos. Esto requiere tiempo y esfuerzo adicionales, pero tiene el beneficio de mostrar las funciones duplicadas y redundantes, dando al analista mayor confianza de que está dibujando los límites del sistema en el lugar correcto y mayor comprensión de las tareas administrativas que deberá modificar.

Cuando debe desarrollarse un diagrama de flujo de datos de un sistema automatizado existente, normalmente el analista tiene poca dificultad en identificar la naturaleza de las funciones y las entradas y salidas. Puede existir problema en especificar la lógica detallada de un proceso si está implementado en un lenguaje mnemotécnico pobremente documentado como el Autocode 1401 (¡Sí señor, todavía hay programas 1401 que están corriendo!) Aquí el analista tendrá que elegir entre el "trabajo de hormiga" de extraer la lógica de un programa (lo que a menudo no es una forma económica de usar el tiempo) y el de redifinir la lógica de la función por comparación del flujo de datos de entrada con el flujo de datos de salida y consulta

con un usuario conocedor acerca de la naturaleza de la lógica externa (empresaria) que se utiliza para transformar estas entradas en salidas. El segundo método no es tan malo porque no nos interesa interiorizarnos de la lógica interna de un programa no documentado (la forma en que se ajustan y prueban interruptores, se acumulan contadores y se ejecutan lazos), sino solamente de las políticas empresarias externas, reglas y procedimientos que están incorporados en el programa.

Dado que puede no estar decidida aún la realización de un sistema, el analista puede decidir no investigar a fondo la lógica detallada de las funciones del sistema actual, sino identificar solamente las funciones del nivel de "Confeccionar facturas". Este temperamento puede tomarse también con otras funciones cualesquiera, que el analista sabe que no serán incluidas en cualquier nuevo sistema. Debe adoptarse una decisión similar de sentido común con respecto al diccionario de datos. Mientras se dibuja el diagrama de flujo de datos, el analista deberá identificar muchos elementos de datos y estructuras de datos. Si la estructura de los archivos del sistema actual está bien documentada, será una fuente adicional de definiciones de datos. Si estas definiciones son fáciles de establecer, el analista podrá usar cualquier manual o facilidades automatizadas que se encuentren disponibles para su captura. Si la definición de detalles requiere una cantidad desproporcionada de trabajo, el analista podrá identificar y poner nombre a los flujos de datos y procesos, con breves descripciones de cada uno, dejando los detalles para más adelante. Cuando el modelo lógico esté construido, el analista también deberá estar a la expectativa de cualquier entrada que parezca no ser utilizada, y de cualquier informe que no tenga valor en adelante o que pueda ser remplazado por informes de excepción o consultas. Este tipo de salidas redundantes y sus funciones asociadas podrán describirse brevemente.

8.2.3 Perfeccionar las estimaciones de IRACIS

El estudio inicial muestra el orden de magnitud del aumento de los ingresos/reducción de costos/mejora del servicio que podrá resultar de un sistema nuevo o mejorado. Cuando los beneficios potenciales de un sistema nuevo son obvios y excelentes o la penalidad por no tener un sistema nuevo es claramente inaceptable, no será necesario perder más tiempo en esta área. Vale la pena frecuentemente, sin embargo, hacer un segundo análisis a las limitaciones del sistema actual y a los beneficios potenciales, especialmente a la luz de las proyecciones del crecimiento en el volumen de las transacciones.

Por ejemplo, en el estudio inicial indicamos un ahorro potencial entre el 1 % y el 2,5 % en la compra de la materia prima, lo que significaba $20 millones en el año. ¿Cuál es la tendencia de crecimiento en la compra de la materia prima? ¿Existe algún modo de hacer un control cruzado del ahorro potencial? ¿Ahora que hemos tratado con la mayoría de los gerentes más importantes, podemos cuantificar algunos de los beneficios intangibles a los que nos hemos referido en el estudio inicial? ¿Si se han instalado sistemas similares en cualquier otra parte, podemos saber qué beneficios han dado? ¿Cómo se distribuyen los beneficios entre el efecto de centralizar las compras en forma masiva y el efecto de obtener descuentos por pronto pago? Será importante conocer los valores relativos de los beneficios que se podrán obtener de los diversos aspectos de un sistema nuevo, ya que los mismos podrán ser un indicador importante de aquellos aspectos que deberán ser implementados primero. Obviamente, si el 1,5 % del ahorro podrá hacerse en base al descuento por pronto pago, el cual parece ser un sistema relativamente simple de implementar, y otro 1 % podrá ser atribuido a la compra centralizada masiva, que además de ser un sistema más complejo requiere cambios grandes en la organización, deberemos tomar en cuenta estos hechos en el planeamiento posterior.

Al finalizar el estudio detallado se tendrá disponible la siguiente información:

— *Una definición de la comunidad de usuarios del nuevo sistema:*
Nombres y responsabilidades de los gerentes de máximo nivel

Funciones de los departamentos afectados
Relaciones entre los departamentos afectados
Descripciones de las tareas administrativas que serán afectadas
Cantidad de personal en cada tarea administrativa, régimen de ingresos y régimen natural de rotación.

— *Un modelo lógico del sistema actual:*
Diagrama de flujo de datos general (incluyendo los sistemas interrelacionados, si son de interés)
Diagramas de flujo de datos detallados por cada proceso importante
Especificación lógica de cada proceso básico a un apropiado nivel de detalle
Definiciones de datos a un apropiado nivel de detalle

— *Una presentación de aumento de ingresos/reducción de costos/mejora de servicios que podrán obtenerse mediante un sistema mejorado,* incluyendo:
Premisas
Volúmenes de transacciones actuales y proyectadas y cantidad de datos acumulados
Estimación en pesos de los beneficios cuando sea posible
Presión de la competencia/factores de índole estatutario (si existen).

— *Revisión del costo estimado del probable sistema de remplazo y un presupuesto en firme del costo/tiempo para la próxima fase (con la definición del "menú" de alternativas posibles).*

Esta información puede presentarse como un informe a la gerencia, en cuyo caso deberá prepararse un corto resumen y tenerlo disponible en forma separada de la documentación de los detalles que ahora se posee. Resulta mejor hacer una presentación personal al grupo gerencial de interés, organizando la presentación alrededor del diagrama de flujo de datos completo del sistema actual. Frecuentemente es conveniente tener el diagrama completo dibujado como una gran ayuda visual, porque permitirá a los gerentes usuarios tener una idea clara de la situación y cómo se integran sus partes.

Un comentario típico es "por primera vez entiendo cómo es el sistema en conjunto". El analista debe estar preparado para recibir mayores críticas cuando presenta el diagrama de flujos de datos que cuando exponía según los métodos tradicionales; siendo más fácil de entender, también es más fácil para los usuarios detectar errores en el diagrama de flujo de datos (lo cual es, en rigor, uno de los objetivos de la presentación).

Como resultado de esta presentación, se deberá tomar una decisión respecto de continuar con el proyecto en su próxima fase, o dejarlo de lado. Algunas veces el grupo gerencial podrá decir, "Construya el mejor sistema nuevo que pueda por $n." Se deberá, en lo posible, requerir de ellos que difieran la fijación del presupuesto hasta que se completen los resultados del análisis de la próxima fase, la definición de las alternativas. En parte por esta razón el estudio detallado y la definición de las alternativas es tratado como una sola fase, y no se toma como punto de control para los gerentes usuarios al final del estudio detallado.

8.3 DEFINIR UN "MENU" DE ALTERNATIVAS

En el pasado fue una práctica común para los analistas y diseñadores estudiar el sistema físico actual, comprender algunas de sus limitaciones y a continuación ir directamente a proyectar un nuevo sistema físico de acuerdo con algunas de estas limitaciones. En parte, se adoptaba este temperamento debido a la dificultad que existía para producir un modelo lógico; la única manera de dar forma a los pensamientos de uno sobre un sistema era dibujar cursogramas físicos y era tal el esfuerzo para producir *cualquier* nueva solución que casi no quedaba energía para la investigación de alternativas. En parte, esta producción de una sola solución se debió a la sensación de una relación "médico-paciente" entre el profesional de procesamiento de datos y el usuario. Después de estudiar los síntomas del paciente, el médico prescribe la única solución que podrá curar el problema. Nosotros vemos que un modelo mejor

Nombre del Gerente _____ Posición _____ Fecha de la entrevista _____ Analista _____

Basado en este ítem que conoce...	Desea que el sistema le diga...	¿Cuál es la frecuencia media probable de este tipo de consulta en cuanto a usted concierne?*	Si pudiera tener esta información mañana por $1, ¿aproximadamente ¿cuánto valdría para usted tenerla inmediatamente?
Nombre del vendedor	Pedidos no despachados de este vendedor		
Nombre del vendedor	Propuestas pendientes de este vendedor		
Nombre del cliente	Pedidos no despachados para este cliente		
Nombre del cliente	Historia de pedidos despachados para este cliente		
Nombre del cliente	Propuestas pendientes para este cliente		
Dos fechas cualesquiera	Todos los pedidos con fecha de envío planificada en este intervalo		
Componente número	Pedidos que utilizan este componente, analizadas por mes		

* > 1/min = muy alta, > 10/hora = alta, > 10/día = mediana, > 10/semana = baja, ≤ 10/semana = muy baja

Figura 7.10 Ejemplo Sunray Engineering.

probable. No se está tratando de comparar la importancia de una respuesta con respecto a las otras, sino solo el valor relativo aproximado. A pesar de ello, los gerentes tienden a dar valores más altos a los accesos más importantes; podrán agregar comentarios tales como "No lo necesito muy a menudo, pero cuando lo necesito, lo necesito pronto y a cualquier precio" (por ejemplo, un análisis de la tarifa exitosa de la reciente propuesta, como guía para cotizar en forma competitiva).

Con accesos relacionados con datos que solo se modifican lentamente, tales como recuperar el número del cliente dado el nombre del cliente, la alternativa está en comparar la búsqueda de la respuesta en un listado, contra tener la respuesta presentada o impresa en una terminal. Con 50 clientes el acceso a la terminal parecerá un lujo; con 5000 clientes la búsqueda en el listado podrá insumir mucho tiempo. Le corresponde a cada gerente decidir sobre el valor relativo que tiene la facilidad para él y su gente.

Cuando se han comparado los resultados de los cuestionarios de las entrevistas con cada gerente, puede obtenerse por lo general una buena indicación del valor global de cada respuesta inmediata, junto a una indicación de la frecuencia total probable de utilización. Si el cuestionario no puede ser contestado con toda claridad debido a la gran variedad de usos que los gerentes pueden hacer de los datos, es muy posible que la solución esté dada por la disponibilidad de un lenguaje general de consulta.

7.6 CONSIDERACIONES SOBRE SEGURIDAD

Mientras que la cuestión de seguridad de datos contra robo, destrucción o cambios concierne al analista y al diseñador en cada etapa del diseño [7.4], la misma toma particular importancia cuando se analizan los requerimientos de acceso inmediato. Los documentos confidenciales generados por computadora pueden guardarse bajo llave y desmenuzarse

cuando no se los requiere más. Los datos confidenciales en un almacenamiento de datos al que se ha provisto de acceso inmediato vía terminal pueden —a menos que se tomen precauciones— ser leídos por personas no autorizadas, sin dejar rastros.

En consecuencia, el analista deberá establecer *quién* está autorizado para realizar cada acceso. Por ejemplo, el gerente de ventas deberá tener acceso a las comisiones ganadas por los vendedores. ¿Podrá el supervisor administrativo de ventas tener también este acceso? Si se dispusiera de un acceso a las propuestas pendientes y a sus importes en dólares, ¿quién detendría a alguna persona pagada por la competencia a que leyera esta información en un CRT?

El diagrama de acceso inmediato y el cuestionario, una vez confeccionados, son útiles como herramientas para considerar cada acceso por separado y decidir sobre los aspectos de seguridad de cada acceso. Esta es una cuestión que el analista deberá aclarar con la gerencia, explicando que es posible, dado el "software" adecuado, permitir a determinadas personas "ver" parte del almacenamiento de datos, pero no todo su contenido. También podría ocurrir que todas las terminales tengan acceso al nombre, dirección, teléfono y territorio del vendedor, pero que solamente algunos de los que utilizan la terminal en la oficina del gerente de ventas, provistos de la palabra clave —conocida solamente por el gerente de ventas, el presidente y el contralor— pudieran ver los importes de las comisiones individuales. La gerencia podría decidir que un amplio grupo de personas pueda tener acceso a las estadísticas de los importes en dólares correspondientes a las propuestas, pero que solo el vendedor de la cuenta tenga acceso al importe específico de una propuesta específica.

APENDICE

Paquetes generales de consultas

CULPRIT-Cullinane Corporation - 20 William Street - Wellesley, MA 02181
EASYTRIEVE - Pansophic - 709 Enterprice Drive - Oakbrook, IL 60521
MARK IV - Informatics - 65, Route 4 - Riveredge, N.J. 07661
Esta no es una lista exhaustiva: Consultar otros paquetes en las publicaciones especializadas.

BIBLIOGRAFIA

7.1 *IQF General Information Manual, GH20-1074,* IBM Corp., White Plains, N.Y.
7.2 J. Martin, *Principles of Data-Base Management,* Prentice-Hall, Englewood Cliffs, N.J., 1976, Cap. 17.
7.3 J. Martin, *Computer Data-Base Organization,* Prentice-Hall, Englewood Cliffs, N.J., 1975, Cap. 5.
7.4 J. Martin, *Security, Accuracy and Privacy in Computer Systems,* Prentice-Hall, Englewood Cliffs, N.J., 1973.

Ejercicios y puntos de discusión

1. En los términos de este capítulo, ¿qué relación existe entre las páginas amarillas y las páginas blancas de su guía telefónica? ¿Es posible realizar otras inversiones de los datos de la guía telefónica?
2. Revise los sistemas de su empresa que clasifican o exploran archivos para producir informes. ¿Existen sistemas en los que sería deseable tener un índice secundario y proveer acceso inmediato a los datos?
3. Si cualquiera de los sistemas de su empresa usa índices secundarios, averigüe cuál es la magnitud del índice comparado con el archivo principal. ¿Con qué frecuencia se reorganiza el índice? ¿Se emplea el índice con fines operativos o informativos? ¿Cuántos accesos se hacen en un día promedio?
4. Si tiene acceso a un archivo o base de datos jerárquicos, dibuje un diagrama de su estructura. ¿Qué previsiones hay que hacer para accesos que no descienden a través del camino jerárquico?
5. Familiarícese con una de las facilidades de consulta general mencionadas en este capítulo. En las

Cuando se incluyen nuevos flujos de datos y estructuras de datos, los mismos deberán ser agregados al diccionario de datos.

Cuando se deban incluir nuevos elementos de datos en almacenamientos de datos existentes, los elementos deberán ser definidos y las definiciones del almacenamiento de datos, actualizadas.

Cuando se involucran nuevos procesos, su lógica deberá ser especificada al nivel de detalle necesario para estimar el esfuerzo probable requerido para su programación. Por ejemplo, si la nueva función es "Preparar informe de inversión de capital," con los flujos de datos entrantes y salientes especificados, será adecuado un resumen de la lógica. Si la nueva función es "Determinar el trayecto óptimo para los camiones de reparto", se necesitará una descripción más extensa y detallada.

Para desarrollar este modelo lógico, se deberán elegir, entre los objetivos fuertemente establecidos, aquellos más exigentes. En esta etapa, el analista está tratando de definir un sistema que satisfaga todos los pedidos de todos los usuarios.

La realidad aparece cuando el analista lleva a cabo un análisis de acceso inmediato para cada uno de los almacenamientos de datos para los cuales le fue requerido algún acceso inmediato. El analista confecciona una cuestionario de acceso inmediato a partir del borrador del diagrama de acceso inmediato, descripto en el Capítulo 7 y obtiene estimaciones de los miembros relevantes de la comunidad usuaria, acerca de los valores relativos y frecuencia probable de los diferentes accesos de la información. Consolida esta información para obtener un cuadro general con el valor y frecuencia agregados a cada acceso inmediato.

8.3.3 Producir diseños físicos tentativos alternativos

En este punto el analista deberá decidir si va a tomar o no la responsabilidad del diseño físico. La filosofía de la organización podrá ser tener a la misma persona haciendo el análisis y el diseño (y posiblemente codificando) o alternativamente hacer que el analista produzca la especificación funcional lógica en la forma que ya se describió, y luego la entregue a otra persona para el diseño físico, con las revisiones mutuas apropiadas. Desde nuestro punto de vista, el segundo temperamento se extenderá cada vez más, ahora que se dispone de técnicas para presentar con precisión "qué" se requiere que el sistema haga sin entrar en "cómo" lo hace.

Aun cuando la misma persona haga diseño y análisis, es una buena práctica "usar el sombrero del analista" hasta el punto que justamente hemos alcanzado en el desarrollo (la especificación de la función lógica) y luego "colocarse el sombrero del diseñador" y revisar las especificaciones funcionales pensando que ellas han sido producidas por cualquier otro para ver cómo se pueden implementar con la mejor relación costo/efectividad.

El analista y el diseñador, entonces —ya sea una cabeza o dos— trabajan juntos para idear varios sistemas alternativos que permitan lograr diversas versiones de los objetivos para distintas inversiones de costo y tiempo. Deberán dibujar límites tentativos alrededor de diferentes conjuntos de funciones del diagrama lógico de flujo de datos propuesto. Deberán considerar la implementación de todos los accesos inmediatos requeridos, o solamente los más importantes, o bien ninguno. Las consideraciones y técnicas de diseño se discutirán con más detalle en el capítulo siguiente.

Los siguientes son algunos tipos comunes de soluciones tentativas que podemos considerar:

1. Un sistema en lote, remplazando a un sistema manual o de cinta por otro en el cual los archivos, aunque sigan siendo posiblemente secuenciales, se puedan colocar en disco. Es relativamente rápido y barato, y los beneficios son normalmente la reducción del tiempo de retorno, el agregado de algunos elementos de datos extras a los nuevos archivos e informes, y la confección de programas más modificables (este tipo de solución se usa a menudo para remplazar a los primitivos sistemas de segunda generación).

2. Un sistema de entrada de datos fuente, con actualización durante la noche, en el cual las transacciones son ingresadas vía terminal en el área del usuario y validadas en línea, construyendo así un archivo de transacciones durante el día, que se utiliza para actualizar la base de datos principal durante una corrida nocturna. Es más costoso que el tipo 1, y los beneficios en general incluyen mayor seguridad (porque los errores pueden ser corregidos en el momento en que ingresan, por personal que sabe lo que está haciendo), mejora en el tiempo de proceso (porque las corridas de información no tienen que esperar que los errores atraviesen ciclos repetitivos de "validar, rechazar, corregir, validar"), y la facultad de consultar sobre el estado de los negocios de la empresa al cierre del día de ayer.

3. Un sistema de entrada de datos en línea, actualizado inmediatamente, con consulta en línea, en el cual cada transacción es validada y corregida en línea y, una vez aceptada, usada inmediatamente para actualizar la base de datos. Estos sistemas son relativamente costosos en función del hardware requerido, del desarrollo y mantenimiento del software, y de la seguridad y precauciones de recuperación que deben tomarse para prevenir la degradación de la base de datos. Los beneficios, por supuesto, pueden ser considerables; no solamente los errores pueden corregirse en el momento de la entrada, sino que los datos de los archivos son frescos al máximo posible. En un sistema de reservas de pasajes aéreos, la disponibilidad de ubicaciones para un vuelo registra hasta el último billete vendido, lo cual pudo tener lugar hace unos pocos segundos. En un sistema bancario en línea si un cliente tiene $100 en su cuenta y retira $80 a través de un cajero distribuidor de efectivo en línea, su saldo queda inmediatamente indicado como $20, y cualquier intento posterior para retirar más que este importe es rechazado. Este tipo de sistema, en general, permite al personal operativo responder a los cambios del cuadro empresario que tienen lugar minuto a minuto a través de todo el día. Esto faculta a la gerencia a consultar sobre la situación de hace pocos minutos, en lugar de la que existía al cierre de ayer.

4. Un sistema distribuido del tipo descripto en la Sec. 4.7, en el cual minicomputadoras locales o terminales inteligentes permiten la entrada de datos fuente y una limitada cantidad de validaciones contra los datos contenidos en los archivos del microcomputador o de la terminal. Los informes operacionales tales como facturas, podrían ser confeccionados por cada procesador local; de cuando en cuando (a menudo cada noche) los procesadores locales trasmiten información a un computador central. El computador central emplea la información (que puede consistir en los detalles completos de las transacciones o solo en un resumen de los datos) para actualizar su base de datos y a su turno puede retrasmitir información a los computadores locales para que actualicen sus propios archivos. Estos sistemas distribuidos ofrecen muchos de los beneficios del sistema centralizado del tipo 3. Cada gerente local tiene acceso a sus propios datos y puede actualizarlos inmediatamente si se justifica. La gerencia central tiene acceso a los datos correctos al cierre de la actividad la noche anterior (o más reciente si las computadoras cambian información más frecuentemente). Debido a que las minicomputadoras y terminales son tan baratas, un sistema distribuido tiende a ser menos costoso que un sistema centralizado de potencia similar. Cuando una empresa está dispersa geográficamente el sistema centralizado necesita líneas telefónicas más caras y vulnerables para permitir la entrada y el acceso de datos fuente. Asimismo, si un sistema centralizado se cae, todo el procesamiento se detiene. Si una microcomputadora local o la central de un sistema distribuido se caen, los efectos son menos drásticos.

5. Un sistema que utiliza una minicomputadora dedicada en lugar de compartir la UCP central existente. Este sistema no debe ser necesariamente distribuido, pero se puede aprovechar la ventaja del bajo costo de las minicomputadoras para implementar un sistema que no se caiga cuando se caiga el sistema central, y no deteriore su tiempo de respuesta o su rendimiento total cuando el sistema central esté excesivamente cargado. Este sistema a menudo es apropiado cuando un gerente tiene una aplicación que no necesita mucha comunicación con ninguna otra aplicación, pero requiere un alto nivel de servicio.

6. Por último, se deberá tener en mente que una alternativa puede ser diseñar un sistema manual mejorado con o sin un sistema automatizado. Frecuentemente, el diagrama de flujo de

datos de un sistema administrativo presenta redundancias y duplicaciones que han ido creciendo a través de los años y que pueden eliminarse ventajosamente. También podemos sacar en conclusión (preferiblemente antes de la acumulación excesiva de detalles) que los objetivos de los usuarios no requieren ningún cambio en el sistema de información, sino que pueden ser atendidos con cambios de organización o de personal. Tal vez, la razón que los deudores morosos sean tantos no sea porque la información necesaria para reducirlos no se encuentre disponible sino porque aun cuando el gerente de créditos rechaza un pedido, el gerente de ventas se apersona al presidente y consigue anular el rechazo. En esta situación podemos utilizar toda nuestra capacidad como consultores y sugerir que el gerente de créditos documente el costo de estas decisiones y silenciosamente haga llegar esta información al presidente.

Naturalmente, estas seis posibles alternativas son solo sugerencias que no deben seguirse ciegamente. Su consideración es muy útil, así como es útil hojear un catálogo de modelos de casas de vacaciones antes de comenzar a discutir los requerimientos de la casa que pensamos construir. Frecuentemente el analista y el diseñador seleccionarán una parte de una alternativa, y una parte de otra alternativa, y así siguiendo hasta llegar a un diseño físico tentativo.

Lo ideal es que los diseños físicos tentativos para un nuevo sistema se seleccionen de la siguiente forma:

1. Un sistema de bajo presupuesto, de implementación razonablemente rápida que incluya solo los objetivos de mayor presión para el usuario, pero que haga factible un posterior desarrollo hacia una solución más elaborada:

(la solución de la "hamburguesa")

2. Un sistema de mediano presupuesto, con una escala de tiempo mediana, que incluya la mayoría substancial de los objetivos, si bien no los más ambiciosos:

(la solución del "pollo frito")

3. Un proyecto de presupuesto muy alto, con un tiempo largo, que incluya todos los objetivos del usuario y tenga el máximo impacto sobre la gestión empresaria:

(la solución del "bife a la Chateaubriand")

Por cada posible alternativa tentativa, el analista y el diseñador deberán realizar estimaciones groseras del probables costos y listar los beneficios, cuantificando estimaciones todo lo que sea posible. Las estimaciones de costo y tiempo podrán basarse en costos y duración conocidos de proyectos similares o en la estimación de los costos de los componentes del "hardware" y del "software" que requiere cada sistema; preferiblemente ambos.

Luego, al final de este paso el analista está en posición de decir a la comunidad de usuarios algo como lo que sigue:

> Por aproximadamente $50.000-$70.000 y aproximadamente en 6-9 semanas de trabajo, podremos desarrollar el Sistema A. Este les dará los mismos informes que en la actualidad pero dentro de los 3 días después de finalizado el mes, en lugar de los 10 días como en el presente. Acelerará el ciclo de facturación, despachando las facturas con un tiempo medio de retraso de solo 2 días después del despacho y proveerá recordatorios automáticos, lo cual permitirá reducir el período medio de cobranza a 20 días en lugar de los 38 actuales. Por encima de ello mejorará nuestra posición de efectivo en $500.000, con una economía anual en intereses bancarios de

$35.000. Este nuevo sistema permitirá hacer el análisis de cliente por dimensión y por industria, que no se puede lograr en el presente. Será mucho más fácil de modificar y de extender que el sistema actual.

Por aproximadamente $150.000-$250.000 y aproximadamente en 12 a 18 meses de trabajo, y empleando una minicomputadora de $50.000-$70.000, podemos desarrollar el Sistema B. Este permitirá a nuestros empleados encargados de recibir los pedidos telefónicos controlar el crédito de los clientes y verificar el estado del inventario, actualizado hasta la noche anterior, antes de aceptar un pedido; y brindará a la gerencia la facultad de recuperar cualquier cuenta de cliente sobre una base inmediata. Además de los beneficios del Sistema A, los beneficios adicionales del Sistema B son la reducción de los morosos en un estimado 50% (equivalente a $20.000 el último año), el despacho de los pedidos en el mismo día en el caso de tener existencia, una intangible mejora del servicio al cliente, e intangibles mejoras en las ventas.

Por aproximadamente $600.000-$900.000 y aproximadamente en 2,5-4 años de trabajo podemos desarrollar el Sistema C, el cual integrará el procesamiento de pedidos y el control de inventarios, de forma tal que el vendedor, utilizando una terminal portátil pueda discar sobre nuestro computador desde la oficina del cliente, obtener una fecha de entrega y precio para los ítem solicitados por el cliente, ingresar de inmediato los detalles del pedido y obtener en el acto una confirmación para el cliente. El Vicepresidente de Ventas estima que esta facilidad posibilitará incrementar las ventas entre el 2% y el 5% y permitirá la reubicación de aproximadamente 30-35 empleados en la Gerencia de Ventas y Compras. Sujeto al control gerencial el Sistema C podrá ordenar automáticamente componentes y materias primas cuando el inventario disponible caiga por debajo de un nivel de seguridad especificado.

Obviamente, las características y beneficios de cada alternativa deberán registrarse con gran detalle, pero aunque imperfectas, las tres alternativas anteriores presentan el tipo de decisión de inversión que puede justificadamente pedírsele al empresario. Vemos que esta es una decisión de inversion (cuánto debemos invertir para obtener cuánto beneficio y en qué tiempo) y no una decisión técnica. No estamos pidiendo a los usuarios que decidan entre los méritos del último producto anunciado por IBM y la arquitectura de 17 bits mejorada de Digital Datawhack. El analista y el diseñador han realizado el trabajo de trasformar los elementos de cada diseño tentativo en *beneficios* para los usuarios.

8.4 UTILIZAR EL "MENU" PARA OBTENER EL APOYO DE LOS USUARIOS QUE TOMAN DECISIONES

Una vez que se formuló el "menú" debe ser presentado al gerente(s) que deberá(n) tomar la decisión de la inversión. Podrá ser un grupo formal existente, denominado tal vez, Comité de Orientación de Procesamiento de Datos, o Grupo de Política de Automatización; si no existiera tal grupo, el analista regresará a los usuarios que toman decisiones y que fueran identificados como partes de la comunidad usuaria en la Sec. 8.2.1. Vale la pena tomarse el trabajo de reunir simultáneamente, a todo el grupo en una sala, si es posible, y hacer una presentación formal de las alternativas empleando las herramientas del análisis estructurado de sistema, como ayudas visuales. Si los responsables de tomar decisiones están dispersos en un área geográfica extensa o tienen otras actividades programadas, el analista puede llegar a la conclusión de que vale la pena hacer una serie de presentaciones a cada uno de ellos. Si esto no se justifica económicamente, se deberá redactar un informe y hacerlo circular para sus comentarios, reconociendo que éste es el modo menos satisfactorio para llegar a un consenso sobre las alternativas.

Asumiendo que los responsables de tomar decisiones puedan reunirse en un grupo, el analista, acompañado por su gerente, deberá hacer una presentación ante el grupo cubriendo los siguientes puntos:

1. El sistema actual (si existe) recorriendo el diagrama de flujo de datos.
2. Las limitaciones del sistema o situación actual. Si el mismo grupo de gerentes recibió una presentación al final del estudio detallado, las limitaciones deberán resumirse. De lo

contrario, vale la pena perder 5-10 minutos en explicar los valores del IRACIS y las cosas que hay detrás, ya que esto es una entrada importante para los usuarios que toman decisiones.

3. El modelo lógico del nuevo sistema, recorriendo el diagrama lógico de flujo de datos que represente la alternativa más amplia del menú, y puntualizando las nuevas funciones que deberán ser incorporadas.

4. Cada uno de los sistemas alternativos que componen el "menú", explicando para cada uno

— Las partes del diagrama general de flujo de datos que deberán implementarse
— La forma en que el sistema aparecerá ante el usuario, en función de las terminales, informes y facilidades de consulta (el diagrama de acceso inmediato podrá utilizarse como ayuda visual). Deberá emplearse la menor cantidad de jerga posible.
— Los beneficios estimados de la alternativa.
— Las mejores estimaciones actuales del costo y del tiempo que insume la implementación de la alternativa.
— Una especificación del factor de riesgo respectivo.

5. Una consulta sobre cuál de las alternativas representan a los ojos de los usuarios la mejor solución de compromiso respecto del costo/beneficio.

El grupo gerencial podrá tener una cantidad de preguntas que formular, surgidas a raíz de esta presentación; podría serles posible alcanzar de inmediato una decisión en consenso sobre la alternativa a seleccionar, o podrán requerir al analista que evalúe algunas soluciones de compromiso alternativas. Podrán pedir un informe para reconsiderarlo y revisarlo por sí mismos; en cualquier situación, tiene lugar un diálogo que termina cuando se alcanza el consenso sobre la naturaleza del sistema que deberá ser construido y la asignación al analista y al diseñador de un presupuesto firme de costo y tiempo (dentro del rango de las estimaciones ya cotizados).

Como se indicó al comienzo de la Sec. 8.3, este proceso compromete muy directamente a los gerentes usuarios en la selección de los objetivos del proyecto y en la asignación de recursos. Existe una mayor probabilidad de que la comunidad de usuarios se convierta en la "dueña" del proyecto en lugar de verlo como "otra donación del personal de procesamiento de datos", algo que el personal de computación arma para atender a sus propios objetivos, que a su vez pueden servir o no a las necesidades de la empresa. Este compromiso hacia el proyecto y participación de los principales gerentes usuarios se ha identificado muchas veces como uno de los factores claves del éxito o fracaso de los proyectos de procesamiento de datos.

8.5 REFINACION DEL DISEÑO FISICO DEL NUEVO PROYECTO

Cuando los responsables de tomar decisiones han asumido un compromiso, el analista y el diseñador trabajan juntos para traducir el modelo lógico y el diseño físico tentativo en un diseño físico firme. Este proceso comprende cuatro actividades que se superponen.

8.5.1 Refinación del modelo lógico

Típicamente, el modelo lógico en esta etapa consiste en el diagrama general de flujo de datos, entradas del diccionario de datos a nivel lógico para cada flujo de datos principal, estructura de datos, almacenamiento de datos, y procesos, y un diagrama de acceso inmediato por cada almacenamiento de datos, cuando sea de interés. Comúnmente la lógica detallada de cada proceso no ha sido aún especificada, a menos que sea crítica para la estimación de costos. Deberán desarrollarse diagramas de flujo de datos detallados, entrando en el

tratamiento de errores y excepciones y en todo otro proceso aún no especificado, según lo descripto en el Capítulo 3. Los contenidos de las entradas del diccionario de datos deberán ser ahora revisados y completados si fuera necesario, como se describió en el Capítulo 4. Los informes y formatos de pantalla podrán simularse a partir de los elementos de datos definidos en el diccionario de datos. La lógica de procesos deberá definirse a un nivel lógico externo, especificando las reglas y procedimientos que deberán incorporarse en el sistema, como se describe en el Capítulo 5. Los contenidos de los almacenamientos de datos lógicos deberán analizarse y simplificarse, como se describe en el Capítulo 6.

Este modelo lógico deberá revisarse en detalle con los usuarios. A menudo se designa un enlace en representación de los usuarios, típicamente un gerente intermedio con conocimiento de la aplicación, y tiene la responsabilidad de aprobar las especificaciones de detalle. Se ha observado en diversas empresas que cuanto más importante es el representante del sector usuario y cuanto más tiempo le dedique al proyecto, mayor será la probabilidad del éxito consiguiente. Por ejemplo, si el asistente del gerente de ventas puede ser asignado con medio tiempo para colaborar con el analista en la definición de los requerimientos de detalle del nuevo sistema, su participación y autoridad hará probablemente que cada miembro de la comunidad usuaria se dedique a fondo a pensar en sus requerimientos y a proveer al analista una información pronta y completa. Algunas empresas han ido más lejos, designando a este gerente usuario principal como "gerente de proyecto" y asignándole el grupo necesario de procesamiento de datos como recurso para desarrollar el sistema. En toda circunstancia, se debe hacer todo lo posible para obtener la participación activa de personal usuario importante.

8.5.2 Diseñar la base de datos física

En base al contenido del almacenamiento de datos lógicos, a la información contenida en el diccionario de datos sobre los volúmenes de transacciones, y al análisis del acceso inmediato, el diseñador físico deberá adoptar un compromiso casi definitivo sobre los contenidos y organización de los archivos físicos y/o base de datos. Se habrá hecho un diseño tentativo de archivo/base de datos para el ejercicio del "menú", y éste ahora deberá refinarse y probarse contra el modelo lógico. Si el sistema va a utilizar un archivo o una base de datos existente, el diseñador deberá comprobar que los datos disponibles y la organización sean conocidos y se adapten a los flujos de datos planificados en el modelo lógico.

El tema del diseño físico del archivo/base de datos es extenso y complejo, y escapa al objeto de este libro su tratamiento en detalle; algunos balances de compromiso se discuten en el capítulo siguiente. El mejor libro reciente sobre el tema es el de James Martin [8.1]. Como analistas, más que como diseñadores físicos, nos concierne asegurarnos que hemos especificado toda la información necesaria para el proceso de diseño, de manera tal que el diseñador físico pueda realizar el mejor trabajo en el aspecto costo/efectividad.

8.5.3 Establecer la jerarquía de las funciones modulares que deberán programarse

Una vez que se han especificado los archivos físicos, los procesos y flujos de datos entran en el ámbito de los subsistemas físicos. Por lo tanto, una vez tomada la decisión de validar en línea las transacciones que ingresan, y construir un archivo con las transacciones aceptadas como entrada de un proceso de actualización nocturna de la base de datos principal, se sigue que todos los procesos y flujos de datos involucrados en validar transacciones y construir el archivo de transacciones pueden considerarse como el *subsistema de entrada de pedidos.* Este subsistema terminará siendo implementado como un programa en línea, o como varios programas. Antes de tomar la decisión acerca de qué "paquetes" físicos resultarán en

definitiva necesitamos estructurar cada subsistema como una jerarquía de módulos, en la cual cada módulo tenga una función *claramente* definida. Este concepto, y las técnicas para trasformar un diagrama de flujo de datos en una estructura modular se describen en el próximo capítulo.

8.5.4 Definir las nuevas tareas administrativas que se interconectarán con el nuevo sistema

Las tareas administrativas que requerirá el nuevo sistema están determinadas por:

1. Dónde se dibuja el límite del sistema automatizado en el diagrama de flujo de datos, y
2. La elección física que se haga de la entrada y la salida. Entonces, si la validación de las transacciones se hace en línea, deben considerarse las tareas administrativas del ingreso de la entrada en una terminal (probablemente una CRT), de la interpretación de las respuestas de la terminal y del tratamiento de dichas respuestas. Si la validación de las transacciones para verificar que sus datos están completos y la decisión acerca del crédito de los clientes deben hacerse manualmente, seguidas por la perfoverificación de los datos de las transacciones a partir de un formulario de entrada, esto involucra un conjunto de tareas administrativas completamente diferentes.

Deberá especificarse cada tarea administrativa así como los manuales de consulta y de entrenamiento necesarios para los empleados. Algunas empresas definen un *subsistema personal* consistente de todos los flujos de datos y procesos que serán implementados manualmente y asignan un gerente o analista de documentación/entrenamiento como responsable del mismo. Muchas empresas hacen al analista responsable del diseño e implementación de este subsistema personal. Especialmente donde los empleados administrativos no han trabajado antes con computadores, el subsistema personal es un componente crítico que puede amenazar el éxito de todo el sistema. Los procedimientos administrativos deberán diseñarse de manera que puedan llevarse a cabo por los empleados, lo cual implica que dichos procedimientos deberán ser probados completamente, lo mismo que el "software". A pesar de todo lo bueno que sea el análisis o el diseño del software, si un sistema requiere que los obreros de fábrica tipeen largos mensajes en las terminales, o exige a los gerentes que recuerden los códigos numéricos de cada Estado y ciudad del país, casi con seguridad está sentenciado al desuso y al fracaso. Para detalles de diseño de subsistemas personales, consultar Gilb y Wainberg [8.2] y Martin [8.3].

8.5.5 Nota sobre estimaciones

Cada una de las cuatro actividades anteriores está orientada a llegar al punto en que es posible dar una firme estimación de costos de desarrollo y operación de un nuevo sistema. Los componentes principales de estos costos son:

1. El tiempo profesional y el tiempo de prueba del computador requeridos para desarrollar los módulos que han sido definidos. A medida que el personal se hace más costoso y el "hardware" más barato, estos elementos del sistema de costos se convierten rápidamente en el factor principal, alcanzando el 70-80% del costo en algunos proyectos de minicomputadoras.
2. La velocidad de la UCP, tamaño de la memoria, y almacenamiento auxiliar (disco) necesario para operar con el volumen de transacciones y la base(s) de datos física(s) planificada(s) y satisfacer los requerimientos de producción y de tiempos de respuesta, tal como se especifican en los objetivos del sistema.
3. Las terminales, costos de las comunicaciones de datos (líneas arrendadas, cargo por discado) y otros equipamientos periféricos necesarios para implementar el sistema. Cuando

interesa el costo del "hardware" se lo puede expresar tanto como el precio de compra total o como el costo mensual equivalente, elegido a menudo como el 1/40 al 1/60 del precio de compra, ya que el equipamiento de computación a menudo se deprecia en 40, 50 o 60 meses.

4. El tiempo profesional requerido para desarrollar la documentación del usuario e impartirle el entrenamiento correspondiente. Este es normalmente un item menor, pero específico.

5. El tiempo del personal administrativo que interactúa con el sistema. Como muchos sistemas permiten liberar parte del personal existente, la reducción de la carga de trabajo administrativo algunas veces se expresa como un beneficio del sistema, en lugar de clasificar la carga del trabajo remanente como un costo operativo. Claramente, si el sistema actual emplea 30 personas y el nuevo sistema requiere 10 a un costo promedio de $1.000 por mes en ambos casos, el cambio puede ser expresado tanto como una economía de $30.000 por mes con un costo de $10.000 por mes o como una economía neta de $20.000.

6. El tiempo profesional del personal requerido para mantenimiento o ampliación del sistema durante su vida útil.

Diferentes empresas han desarrollado esquemas distintos de costos para poner un valor monetario a estos diversos componentes. Ejemplos representativos son:

1K de memoria real por hora (incluyendo todos los costos de operaciones y mantenimiento)	50 centavos
1 X eje de disco 3330 en línea por hora (100 millones de bytes)	$6 ($1.000 por mes)
1 hora de tiempo profesional	$30

Tomando un ejemplo *muy* simple, basado en estas cifras, el sistema requiere 60 hombres/mes para desarrollarlo, empleando 4 horas/día de tiempo de prueba para los últimos seis meses del proyecto y correrá en una región de 150K, 8 horas/día, con cuatro ejes de disco 3330 permanentemente en linea, necesitando dos personas de tiempo completo para el mantenimiento de programas; los costos son los siguientes:

Desarrollo:	
Tiempo profesional: 60 hombres mes x 20 días/mes x 8 horas/día x $30	$290.000 aprox.
Tiempo de prueba en computador: 6 meses x 20 días x 4 horas/día x 150K x $0,50	$36.000
	$325.000 aprox.
Operación (por mes):	
UCP: 8 horas/día x 20 días x150K x0,50	$12.000
Disco: 4 ejes x $1.000	$ 4.000
	$16.000 por mes
Mantenimiento (por mes): 2 personas x 20 días x 8 horas x $30	$ 9.600

En este análisis se ignora el costo de los puntos 3, 4, y 5 anteriores, por supuesto, pero es

indicativo. Como puntualizamos en el capítulo próximo, cuando usted lea estas líneas, el costo del tiempo profesional habrá aumentado y el costo del hardware habrá disminuido substancialmente. Cada analista debe establecer valores realistas para su propia empresa y estimar la tendencia de los costos operativos durante la vida del proyecto.

Estas estimaciones, por supuesto, tienen como punto de partida las estimaciones de la mano de obra, capacidad de memoria y tiempo de corrida. ¿De dónde provienen estas estimaciones? Existe una cantidad de esquemas más o menos bien probados para estimar la mano de obra necesaria para un proyecto de programación (ver, por ejemplo, el trabajo de Aron [8.4]), pero todos ellos asumen que el *estimador conoce cuántos programas deben escribirse y cuán largos son.* IBM ha construido una base de datos de 60 proyectos, que abarcan tamaños desde 4000 líneas de código fuente hasta 467.000 líneas, y ha estudiado el efecto de muchos factores sobre el esfuerzo requerido y la velocidad de desarrollo del sistema [8.5]. Los proyectos medianos producen 20.000 líneas de código fuente en 11 meses, con un promedio de seis personas, para un trabajo total de 67 hombre-mes, con un costo de prueba de $36.000 aproximadamente. Partiendo de la base de datos es posible predecir con cierta certeza el número promedio de líneas de codificación a producir por hombre-mes en un proyecto, basado en factores tales como el uso de la programación estructurada, la experiencia profesional, la facilidad de comunicación con el cliente, y más de otras veinte variables. Así, una vez que el gerente de proyecto conoce la cantidad de líneas de codificación que se van a producir, puede predecir la productividad que se puede esperar del personal que tiene disponible y determinar la duración y costo del proyecto.

Por ello es que la construcción del modelo lógico y las técnicas de diseño jerárquico que se describen en el próximo capítulo son tan útiles. Las técnicas nos permiten identificar las funciones y módulos que se necesitarán programar; el diseñador puede emplear entonces su experiencia para estimar la cantidad de líneas fuente que se necesitarán para implementar cada módulo y así obtener una estimación de la tarea de programación total.

También es, por supuesto, muy útil tener un resumen de los costos, esfuerzo y duración requeridos por similares proyectos anteriores, de manera que puede hacerse una estimación por comparación y verificarla con la estimación realizada módulo por módulo. Realmente, sospechamos que la diferencia entre alguien que hace buenas estimaciones y otro que no es tan bueno reside principalmente en las bases de datos de costos y duraciones de anteriores proyectos que mantienen en su memoria. Una cantidad de empresas, como IBM, están pensando en formalizar sus experiencias para ponerlas a disposición de cualquiera que necesite trabajar en estimaciones en una forma fácilmente utilizable. Si se dispone de algo parecido a un "catálogo de Sears-Roebruck" de proyectos anteriores (con la descripción de cada proyecto, la naturaleza de los informes y los medios de consulta en línea, los costos de personal y el tiempo de máquina, la experiencia del personal en proyectos y técnicas similares, y otras variables de interés), puede ser extraordinariamente valioso para "acotar" el proyecto que se está tratando de estimar. Cada caso descrito en el "catálogo" puede compararse con el problema o sistema que se está considerando y decidir si es más simple, más complejo, o aproximadamente igual. Esto nos dará una buena indicación sobre el costo y duración relativa que el proyecto actual probablemente insumirá dentro de una variación del 20-30%. Realmente, durante el estudio inicial y detallado este tipo de comparaciones será el único método para estimar el alcance del proyecto, en vez de "introducir un dedo húmedo en el aire".

Los usuarios, en forma muy comprensible, presionan fuertemente a los analistas y gerentes de proyecto para que produzcan estimaciones tempranas de los costos totales de proyectos, y considerando, como se ha indicado, que no es realmente posible producir una estimación firmemente basada hasta haber realizado el diseño jerárquico, es notable que muchas empresas no hayan confeccionado aún tal "catálogo Sears-Roebruck". Si no lo tiene, le recomendamos éste como un proyecto de gran valor potencial.

8.6 ULTIMAS FASES DEL PROYECTO

Como este es fundamentalmente un libro de análisis y no de diseño o implementación, no llevaremos la metodología estructurada más allá de dicho punto. El diseño estructurado y las técnicas de implementación se discuten en el próximo capítulo para demostrar cómo se relacionan con el análisis estructurado. Para completar podemos distinguir las siguientes fases en el desarrollo subsiguiente del sistema:

- — Confección de un plan de implementación, incluyendo planes para la prueba y aceptación del sistema.
- — El desarrollo concurrente de los programas de aplicación, el subsistema personal, y la base de datos/funciones de comunicaciones de datos (cuando sean de interés).
- — La conversión y carga de la base(s) de datos.
- — La prueba y aceptación de cada parte del sistema.
- — Prueba del sistema bajo cargas reales para asegurar que satisface los criterios de operación previsto en los objetivos del sistema, en términos de tiempo de respuesta y de rendimiento.
- — Previsiones para la operación activa del sistema.
- — Medición del desempeño del sistema para identificar los cuellos de botella y los ajustes necesarios para su solución.
- — Comparación de las facilidades globales de sistema y de su desempeño con los objetivos originales, y acción para resolver cualquier diferencia, donde sea posible.
- — Análisis de los requerimientos de mejoras, asignación de prioridades a los mismos, y colocación del sistema en estado de "mantenimiento".

El analista frecuentemente actúa como agente de los usuarios en las últimas fases, así como un arquitecto verifica la construcción de un edificio para asegurarse de que se han seguido los planos y se han empleado los materiales de la calidad aprobada. El analista deberá participar en las especificaciones de las pruebas de aceptación y posiblemente en la generación de los datos de prueba.

Cuando deba utilizarse un diccionario de datos automatizado en el proyecto, puede ser muy conveniente la generación de datos de prueba, ya que los valores aceptables e inaceptables de cada elemento de datos de entrada han sido definidos en el diccionario de datos, y a partir de los mismos se pueden armar a su vez transacciones aceptables e inaceptables.

El modelo lógico debe mantenerse actualizado a través del diseño y la implementación, en especial los diagramas de flujo de datos. Servirán como una herramienta básica para planificar mejoras, especialmente aquellas que involucran nuevas funciones.

BIBLIOGRAFIA

8.1 J. Martin, *Computer Data-Base Organization*, Prentice-Hall, Englewood Cliffs, N.J., 1975.
8.2 T. Gilb y G. M. Weinberg, *Humanized Input: Techniques for Reliable Keyed Input*, Prentice-Hall, Englewood Cliffs, N.J., 1976.
8.3 J. Martin, *Design of Man-Computer Dialogues*, Prentice-Hall, Englewood Cliffs, N.J., 1973.
8.4 J. D. Aron, *The Program Development Process*, Addison-Wesley, Reading, Mass., 1974.
8.5 C. E. Walston y C. P. Felix. "A Method of Programming Meassurement and Estimation". *IBM Systems Journal*, Vol. 16, Nº 1, 1977.

Ejercicios y puntos de discusión

1. Revisar el enfoque normal de los proyectos de desarrollo que se emplea en su empresa (ya sea formal o informal). ¿En qué difiere de la metodología del Capítulo 8? ¿En qué se asemejan?

2. Producir un conjunto de objetivos fuertemente etablecidos para un sistema recientemente instalado con el que usted esté familiarizado.
3. Producir una definición de IRACIS real del sistema del Ejercicio 2, comparándolo con el sistema que remplaza.
4. Desarrollar un DFD lógico general para el sistema del Ejercicio 2 y confeccionar diseños tentativos para:
   (a) Un sistema más económico con menos benefricios que el sistema existente.
   (b) Un sistema más amplio con mayores beneficios que el sistema existente.
5. ¿En qué circunstancias sería inadecuado el enfoque "menú"?
6. Desarrollar el costo de desarrollo y el costo de operación corriente de los últimos sistemas implementados en su empresa, como se describe en la Sec. 8.5.5. Emplee hombre/hora y computadora/hora si no dispone de valores monetarios. Confeccione los diagramas de datos de acceso inmediato por cada uno de los sistemas que permiten algún acceso inmediato. ¿Existe alguna relación entre los costos del sistema y la cantidad de accesos inmediatos?
7. ¿Observa alguna conexión entre el valor de un sistema y el grado de participación del usuario en su desarrollo? ¿Cuál ha sido la experiencia de su empresa en obtener la participación del usuario? En su opinión ¿cuáles son las causas de que los usuarios no participen más plenamente en el desarrollo de los sistemas?

9

DERIVAR UN DISEÑO ESTRUCTURADO A PARTIR DE UN MODELO LOGICO

¿Qué queremos significar con la palabra diseño? ¿Que es diseño *estructurado*? En este capítulo vamos a tratar de contestar estas preguntas, demostrar cómo un diseño puede realizarse desde nuestro modelo lógico de un sistema y ver cómo los diseños mejorados permiten desarrollar más fácilmente los sistemas a través del denominado desarrollo *de arriba hacia abajo*.

Hay una gran confusión entre análisis y diseño, en parte porque hasta que fue posible producir el tipo de modelo lógico que hemos descrito era muy difícil separar el análisis ("qué" debe hacer el sistema) del diseño ("cómo" va a ser hecho). Definiremos al *diseño como el proceso (iterativo) de tomar un modelo lógico de un sistema junto con un conjunto de objetivos fuertemente establecidos para este sistema y producir las especificaciones de un sistema físico que pueda satisfacer estos objetivos*.

En los últimos años se ha logrado un gran progreso en las técnicas de diseño de sistemas. Desde una aproximación más o menos intuitiva al diseño —donde era casi un milagro ver *cualquier* forma de deducir las salidas requeridas a partir de las entradas dadas— han surgido algunos procedimientos y pautas bastante claros. Desde nuestro punto de vista, el conjunto de pautas de mayor utilidad lo originó Larry Constantine [9.1], como resultado de su trabajo en IBM, en Hughes Aircraft y en otros sitios a fines de la década de los 60, y fue perfeccionado por Myers [9.2] y Yourdon [9.3]. Este es un enfoque particular conocido como *diseño estructurado;* las herramientas lógicas que hemos discutido en este libro penetran profundamente en los conceptos de diseño estructurado. Para ver porqué el diseño estructurado (en oposición al diseño *de arriba hacia abajo* defendido por IBM o en oposición al diseño de abajo hacia arriba o diseño al azar) es la aproximación más útil, revisaremos los objetivos del diseño y relacionaremos el diseño estructurado con estos objetivos.

9.1 LOS OBJETIVOS DEL DISEÑO

El objetivo más importante del diseño, por supuesto, es entregar las funciones requeridas por el usuario. Si el modelo lógico exige la confección de cheques y el diseño no produce cheques, o no los produce correctamente, entonces el diseño es un fracaso. Pero, dado que son posibles muchos diseños correctos, hay tres objetivos principales que el diseñador tendrá que tener presente mientras desarrolla y evalúa un diseño:

- *Rendimiento,* cuán rápido permitirá el diseño realizar el trabajo del usuario dado un recurso particular de hardware.
- *Control,* la medida en que el diseño está protegido contra errores humanos, máquinas defectuosas, o daños intencionales.

— *Cambiabilidad*, la facilidad con la cual el diseño permite modificar el sistema, por ejemplo, satisfacer las necesidades del usuario de procesar diferentes tipos de transacciones.

Aunque ello no siempre es cierto, acontece que generalmente estos tres factores trabajan unos en contra de los otros; un sistema con controles de mucha seguridad tenderá a degradar su rendimiento, un sistema diseñado para muy alto rendimiento solo podrá ser cambiado con dificultad, etc. Vamos a rever algunos de los factores que están detrás de cada uno de estos objetivos del diseño, considerando con algún detalle la naturaleza de los sistemas modificables (Sec. 9.2). Luego en la Sec. 9.3, consideraremos las soluciones de compromiso entre los factores.

9.1.1 Consideraciones de rendimiento

El rendimiento normalmente se expresa en términos de

— *Volumen de procesamiento* (transacciones/cálculos por hora)
— *Tiempo de corrida* (para una tarea en lote, donde se procesa la misma cantidad de trabajo en cada corrida)
— *Tiempo de respuesta* (el tiempo entre que se pulsa la tecla de "entrada" en una terminal y el comienzo de la aparición de la respuesta del computador en dicha terminal).

Como la mayoría de los sistemas computadorizados están afectados básicamente a la manipulación o reducción de los datos (sistemas comerciales), está implícita en cada una de estas mediciones la cantidad real de memoria utilizada, ya que casi siempre es posible mejorar el volumen total de procesamiento o tiempo de respuesta, asignando una región mayor o más memoria real al programa(s).

Esto no es cierto en los sistemas orientados a algoritmos, tales como control de procesos en tiempo real, donde el rendimiento depende más de la velocidad del procesador central y de las instrucciones utilizadas en el proceso. Se invita a los lectores relacionados con dichos sistemas "desmenuzadores generadores de números", a pasar a la Sec. 9.1.3, ya que nuestra discusión de rendimiento y control es aplicable principalmente a los sistemas orientados a datos.

En los computadores modernos el diseñador, normalmente, está preocupado básicamente con el volumen o cantidad total de procesamiento y en segundo término con el tamaño de memoria. Se presenta una excepción con las pequeñas minicomputadoras y microcomputadoras, donde el objetivo primordial de rendimiento consiste en obtener la máxima cantidad de funciones en 4K bytes, o algo así.

Identificamos cinco factores que afectan el desempeño en volumen total de procesamiento de un sistema orientado a datos a nivel bruto, y podemos disponerlos aproximadamente en orden de importancia decreciente de la siguiente manera:

1. *La cantidad de archivos intermedios de un sistema:* Un archivo intermedio es aquel que se utiliza para "estacionar" datos entre programas, en lugar de un archivo que aloja un almacenamiento de datos de atributos sobre cierta entidad importante. Un archivo intermedio es grabado, y luego leído nuevamente por el próximo programa, después de lo cual es desechado sin afectar el procesamiento satisfactorio de los datos. Las instalaciones de tarjetas perforadas y computadoras de segunda generación con cintas magnéticas pero sin discos se vieron forzadas a correr sistemas diseñados con archivos intermedios en una secuencia programa-archivo-programa-archivo-programa. El diagrama de flujo físico de la Fig. 9.1 muestra un sistema de este tipo.

Por supuesto que con grandes computadores y archivos de discos ya no hace falta esta estructura de sistema; la creación y nueva lectura de todos los archivos innecesarios tiende a

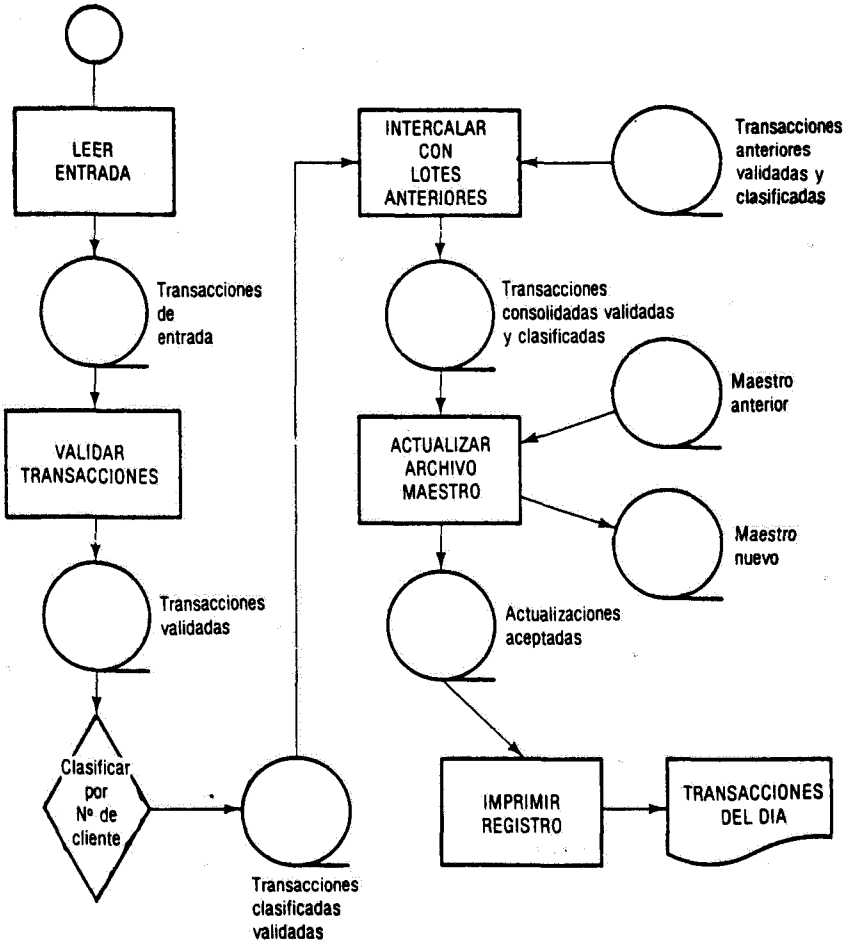


Figura 9.1 *Diagrama de flujo de un sistema físico.*

producir un sistema de corrida esencialmente lenta, aun cuando los archivos se saquen de cintas y se coloquen en discos. Sin embargo, el hábito de crear archivos intermedios es aún muy fuerte, y estos archivos a menudo se especifican innecesariamente por personas que obtuvieron su experiencia inicial de diseño de sistemas en los días de la segunda generación.

2. *La cantidad de veces que se pasa un archivo determinado:* Supongamos que un archivo contiene una mezcla de registros de tres tipos, para transacciones del tipo A, del tipo B y del tipo C. El diseñador puede elegir leer todo el archivo en procura del tipo A y procesar el tipo A, luego leerlo en procura del tipo B y luego del tipo C, o bien especificar un programa que lea el archivo una sola vez y atienda cada tipo de transacción al encontrarla. Ya sea que el archivo esté en cinta o en disco, la segunda estrategia nos dará el mejor rendimiento total de procesamiento, a igualdad de las demás condiciones. Clasificar un archivo o buscar de punta a punta para contestar una consulta (como se describió en la Sec. 7.2) requiere una pasada de archivo; ya hemos comentado que clasificar o buscar toma un tiempo considerable y puede degradar el rendimiento de otros sistemas que utilicen simultáneamente el mismo archivo.

3. *La cantidad de búsquedas contra un archivo de discos:* Dando por sentado que un

diseño tiene una cantidad mínima de archivos intermedios y los corre una cantidad mínima de veces, la próxima limitación del volumen total de procesamiento reside normalmente en la cantidad de movimientos de búsqueda realizado por el brazo de la cabeza móvil del disco. Siempre que el disco debe ser leído o grabado, el software administrador del disco deberá determinar dónde se encuentra alojado en el disco el registro apropiado y hacer que la cabeza lectora grabadora se mueva desde donde se encuentre hasta el lugar correcto para leer el registro. Este movimiento de búsqueda puede no tomar tiempo (si el brazo está justo en el lugar correcto) o insumir hasta 250 o 500 milisegundos (1/4 a 1/2 segundo) si el brazo tiene que moverse a todo lo ancho del disco en un modelo de unidad de disco lenta. Más aún, una lectura o grabación dada puede involucrar el examen de diferentes lugares del disco. Para tomar un caso simple, si los registros en el archivo se localizan a partir de un índice, la cabeza lectora grabadora deberá moverse primero hasta el índice y luego a la posición del registro deseado, que se encontró en el índice. Si la posición especificada no contiene el registro verdadero sino una dirección de desborde que indica donde encontrar el registro, será necesaria una tercera búsqueda. Con algunos métodos de acceso a discos y formatos de archivos, encontrar un registro puede involucrar de 15 a 20 búsquedas. Algunos de los factores que incrementan las búsquedas están fuera del control del diseñador debido a que forman parte del software de administración de disco que se utiliza. El diseñador puede asegurarse, por su conocimiento del software, de que el formato del archivo no da lugar a excesivas búsquedas; por ejemplo, el diseñador debe ser capaz de especificar reorganizaciones regulares de la disposición física para lograr desbordes mínimos en la búsqueda. Puede resultar posible especificar que el índice del archivo se encuentre en el medio del archivo, en lugar de estar en un extremo, lo cual acortará la distancia media de movimiento de la cabeza lectora grabadora. Ambas medidas son, sin embargo, esencialmente tácticas; lo más importante que debe hacer el diseñador es estructurar el sistema de manera que el archivo sea leído la menor cantidad de veces. Esto significa que, si el rendimiento es el objetivo principal, cuando el archivo maestro de clientes deba leerse para controlar la dirección, todos los datos de un registro deben permanecer en el sistema hasta que aparezca la necesidad de controlar el teléfono, o saldos pendientes, o la historia de un cliente o cualquier otro dato contenido en el registro. Estas consideraciones algunas veces entran en conflicto con otros objetivos del diseño, tales como simplicidad o cambiabilidad, pero el diseñador debe revisar el diseño con el objeto de minimizar lecturas, si puede hacerlo.

4. *El tiempo empleado en llamar programas y otros recursos del sistema:* Un sistema está compuesto por una serie de programas, cada uno de los cuales puede estar formado por sub-programas. En la mayoría de las computadoras cada programa principal es invocado por el lenguaje de control de trabajo, que en sí mismo forma un *mini-programa* invocado por el operador de la consola. Podemos proporcionar la *estructura de invocación* en la mayoría de los sistemas como una jerarquía, tal como se ve en la Fig. 9.2. La flecha de invocación a diferencia de la flecha de un gráfico de flujo o de un diagrama de flujo de datos, implica que el control se trasfiere de programa a programa, pero también que dicho control es devuelto cuando el programa invocado o llamado ha terminado su ejecución. Este tipo de jerarquía de control, en el cual una cantidad de pequeños módulos manejables (programas o secciones de programas) se combinan para formar un sistema, tiene muchas ventajas, como veremos al discutir la cambiabilidad de los diseños. Desde el punto de vista del rendimiento cada invocación toma un cierto tiempo, ya sea porque el computador (su sistema operativo) tiene que determinar el lugar de la memoria donde se encuentra el programa invocado, y trasferirle el control, o porque —lo que es más serio— si el programa no está en la memoria, el sistema operativo debe ir a proveerlo desde alguna otra biblioteca, ya sea vía un "CALL" (llamado) dinámico, o una extracción por superposición (overlay fetch), o en una operación de "paginado desde" en un sistema de memoria virtual. Genéricamente hablando, las invocaciones dentro de un programa (por ejemplo, un PERFORM en COBOL) involucran menos proceso auxiliar que las invocaciones de un programa por otro (por ejemplo, un CALL en COBOL). Por esta razón el diseñador puede decidir "empaquetar" dos módulos juntos en el mismo programa si se llaman entre sí frecuentemente.

5. *El tiempo insumido para ejecutar el programa real:* Después de haberse leído los datos necesarios y después que el programa apropiado ha sido invocado y ha recibido el control, se ejecutan las instrucciones reales de máquina generadas por el compilador a partir del programa fuente. Esta ejecución de la codificación generada, a menudo representa la menor proporción de todo el tiempo insumido por el sistema, y debe ser el último lugar en el que un diseñador o un programador consciente gaste tiempo en lograr mejoras. Muchos técnicos no aprecian la relativa poca importancia de una eficiente codificación de máquina, en estos días de grandes máquinas y sistemas operativos poderosos. Es difícil recordar que si la búsqueda promedio toma 30 milisegundos, una instrucción larga de máquina puede tomar 30 *microsegundos,* 1000 veces menos. Prestar atención a la eficiencia en la generación de codificación *solía* ser importante cuando las máquinas eran pequeñas y lentas; ocasionalmente puede ser aún importante en mini o microcomputadoras y en UCP asignadas a sistemas en tiempo real, donde son importantes las fracciones de segundo en los tiempos de respuesta. En general, sin embargo, el diseñador y el programador logran mayores mejoras de rendimiento dedicando su tiempo a los primeros cuatro factores, en lugar de preocuparse acerca de la cantidad de instrucciones en lenguaje compaginador (assembler) generadas por alguna sentencia. Citando a Ed Yourdon,

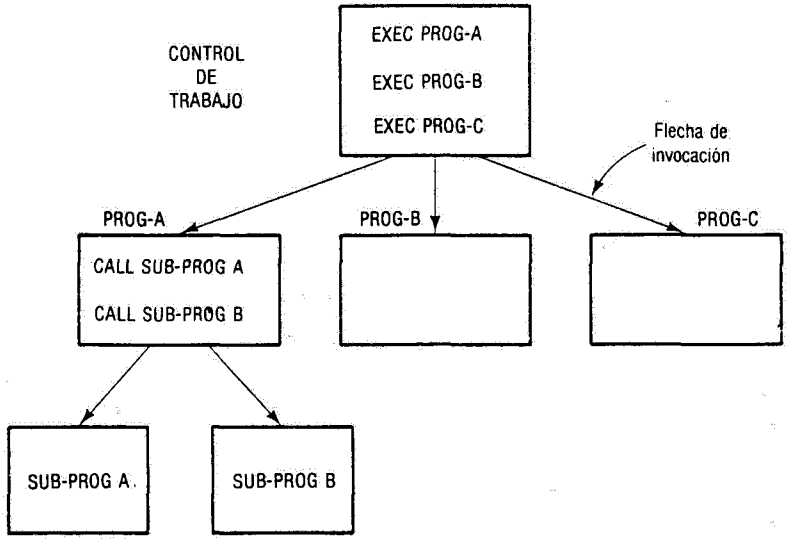


Figura 9.2 Estructura de invocación.

> ...uno debe siempre sonreír un poco ante las quejas de ineficiencia de los programadores COBOL trabajando bajo un sistema operativo tal como el OS/360. Si realmente estuviéramos interesados en la eficiencia, deberíamos estar codificando en octal en máquinas sin un sistema operativo... [9.4].

En realidad, a menos que podamos simular el sistema operativo en nuestras mentes, es insustancial "pre-optimizar" la codificación hasta que no veamos dónde se pierde la mayor parte del tiempo.

9.1.2 Consideraciones sobre control

Dependiendo de la naturaleza del sistema y la cantidad de dinero en juego, el diseñador necesitará introducir diversos tipos de controles. Obviamente un sistema donde se manejan grandes sumas de dinero o donde se almacena importante información secreta, necesita controles más fuertes contra errores o daños, que el sistema de distribución de libros.

Algunos aspectos comunes de control son:

1. *El uso de dígitos de verificación sobre números pre-determinados:* Hemos visto que ISBN tiene un dígito de verificación, que es el último de sus diez dígitos. Muchos bancos asignan el último dígito del número de la cuenta personal como dígito verificador. En el momento en que se asigna el número del cliente, el dígito verificador se calcula en base a los otros dígitos del número y a una fórmula adecuada. Luego, cada vez que se procese en el futuro el número de cuenta —en particular durante la entrada del dato— los dígitos del número, tal como ingresan, se utilizan para recalcular el dígito verificador, y la respuesta se compara con el dígito verificador ingresado. Si los dos no concuerdan uno o más dígitos han sido ingresados incorrectamente, y la transacción será rechazada. El uso de un dígito verificador requiere un pequeño procesamiento extra, pero éste se justifica ampliamente en razón de los errores que previene.

2. *El uso de los totales de control de lote:* Cuando las transacciones se ingresan al sistema en lotes, el total (tal vez la cantidad total de todos los pedidos del lote) es calculado manualmente antes de la entrada. Como parte de la validación de este lote, el diseñador deberá especificar que el computador calcule el total del lote y lo compare con el total calculado manualmente. Si no concuerdan puede ocurrir que una transacción se haya perdido o que haya entrado incorrectamente un importe.

3. *La creación de libros diarios y líneas de auditoría:* Es deseable que la gerencia o los auditores internos estén siempre en posición de decir qué transacciones ha procesado el sistema, y qué ha hecho con ellas. El diseñador puede especificar que cada transacción que entre al sistema se grabe en un registro o archivo diario, que podrá ser leído por los auditores a su comodidad. Algunas veces se llevará también un diario de *archivo de acciones* el que registrará cada cambio de cualquier archivo. La creación de estos libros diarios retardará algo al sistema y requerirá más recursos de hardware; el diseñador tendrá que encontrar una solución de compromiso con las necesidades de control y seguridad. Realmente, puede no tener mucho para elegir en materia de controles, ya que ellos pueden ser impuestos por los auditores. En los sistemas en línea, el archivo diario se crea por otra razón. Si el sistema llegara a fallar, algunas de las transacciones que habrían debido entrar desde terminales remotas se perderían; puede ser necesario hacer previamente una copia del archivo maestro y correr las transacciones del día contra él desde el archivo diario. Tomar copias de respaldo de los archivos o de puntos de control también representa un uso, por el diseñador de recursos de hardware para seguridad y control.

4. *Las limitaciones de los accesos a los archivos:* Muchos de los aspectos de diseño de seguridad y control están vinculados con las respuestas a las preguntas

"¿Quién puede acceder a estos datos?"
"¿Quién está autorizado a modificar estos datos?"

Los accesos pueden limitarse mediante el empleo de contraseñas, o palabras clave, las que deberán ingresarse antes de que una persona pueda utilizar una terminal, o mediante el software que prevenga contra usuarios no autorizados a usar ciertos programas, o que prevenga para que ciertos programas no puedan leer elementos de datos específicos de un archivo. También puede controlarse mediante la posesión física del propio archivo, como en el caso que la cinta maestra de la nómina se encuentre encerrada en la caja fuerte. Para una descripción detallada de estos temas y de toda la cuestión de seguridad y precisión en el diseño, ver Martin [9.5]. En general, podemos decir que el control y la seguridad cuestan dinero ya sea por el hardware y/o software adicional, y también que puede afectar el rendimiento. El

diseñador, guiado por el analista, tiene que encontrar la mejor solución de compromiso entre estos factores.

9.1.3 Consideraciones sobre la cambiabilidad

Hablamos acerca "del sistema" pensando en él como si fuera una cosa física estática, pero por supuesto nada podría estar más lejos de la verdad. Puesto que el sistema procesa datos sobre el mundo real, siempre que el mundo real "externo" cambie, el sistema puede necesitar cambiar. Nuevas líneas del negocio se introducen o se quitan, los precios y los esquemas salariales cambian, los gerentes usuarios cambian y los nuevos usuarios tienen ideas diferentes sobre sus requerimientos de información, las políticas cambian y nuevas leyes se sancionan.

Así como estos cambios son inducidos externamente, también la tecnología del procesamiento de datos está en agitación. Mayor potencia, hardware más barato, nuevos sistemas operativos y lenguajes, nuevas técnicas de comunicación de datos y de base de datos, todo ello podrá significar que el sistema debe ser cambiado de alguna manera. Por otro lado, no importa cuán cuidadosa sea la prueba que reciba el sistema, siempre habrá errores que solo aparecen una vez que el sistema está en estado de producción; estos errores deben encontrarse y eliminarse. Encontrar y eliminar un error involucra muchas veces las mismas actividades que hacer un cambio para beneficiar al usuario; el programador deberá localizar la parte del sistema que necesita ser reparada o cambiada, hacer el arreglo y asegurarse que trabaja bien. (Realmente, los tres tipos de cambios se trabajan juntos y con frecuencia se llama a esto *mantenimiento*, aunque hacer cambios que provean al usuario de funciones adicionales o mejores podría llamarse más apropiadamente, *perfeccionamiento*).

¿Existe alguna diferencia entre la remoción de errores después de poner el sistema en producción o removerlos *antes* de ponerlo en funcionamiento? Aparte del impacto de los errores en el entorno de la producción, las actividades involucradas son idénticas. Por lo tanto sacamos en conclusión que la *cambiabilidad* de un sistema es algo muy importante; si se pudiera encontrar una manera de diseñar el sistema más modificable, sería fácil de corregir, fácil de adaptar a hardware y software cambiantes, y fácil para incorporarle los cambios y mejoras solicitadas por los usuarios.

Por cambiabilidad queremos significar una medida del tiempo que lleva hacer cualquier cambio en el sistema, ya sea suprimir un error o instalar una mejora. Supongamos un sistema dado que ha sido diseñado de dos maneras diferentes y que se ha hecho el mismo conjunto de cambios en cada sistema. Si el diseño A requiere 20 hombres-horas para el cambio y el diseño B requiere 60 hombres-horas para introducir los mismos cambios, podemos decir razonablemente que el diseño A es tres veces más modificable que el diseño B.

Valores recientes muestran justamente cuán importante es el factor de cambiabilidad en el costo total del sistema. La Fig. 9.3 resume valores representativos que indican que para sistemas desarrollados en la actualidad, la mayor parte del costo a lo largo de su vida útil se invierte en modificar dichos sistemas, y que en el presupuesto de la mano de obra "visible" para su desarrollo, se gasta aproximadamente la mitad en los cambios correspondientes a pruebas y correcciones. No es de extrañar que en muchas instalaciones el esfuerzo que implica mantener el sistema existente "en el aire" interfiera con el desarrollo de nuevos sistemas aun cuando la empresa necesite de ellos con urgencia.

Corregir y cambiar el software son tareas con ardua exigencia de mano de obra; no es de gran ayuda lo que se podrá obtener del uso de la capacidad del computador, aparte del uso de las técnicas de depuración interactivas. Mientras, como se ve en la Fig. 9.4, el costo del hardware declina a un ritmo notable, el costo de la mano de obra profesional crece continuamente. Hay por lo tanto una gran *presión que irá creciendo mes a mes, para que los sistemas a diseñar sean lo más modificables que sea posible.* Después de todo, si podemos producir sistemas que sean dos veces más modificables que los actuales, ¡podríamos reducir

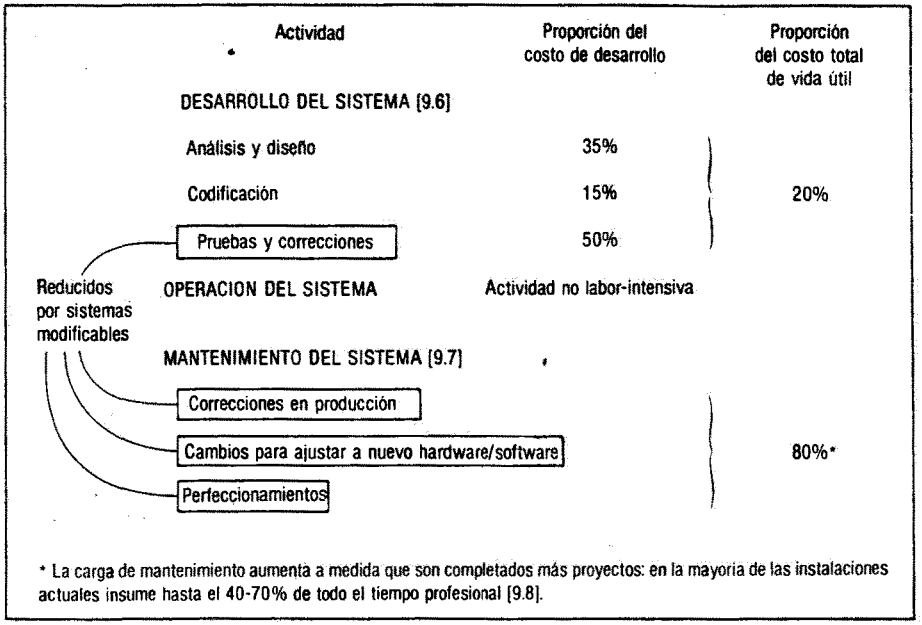


Figura 9.3 Resumen de la distribución del costo de la mano de obra a través del ciclo de vida de un proyecto.

	Precio de compra equivalente por 1K Memoria	Costo promedio por hora por Programador/Analista
1964	$2.000 (360/30)	
1969	$1.000 (S/3)	$20
Sep 1975	$ 260 (370/158)	
May 1976	$ 170 (370/158)	
Abr 1977	$ 110 (370/158)	$30
TENDENCIA:	DISMINUCION 20% por año	INCREMENTO 7% por año

Figura 9.4 Comparación del costo del hardware con el costo del personal (unidades monetarias en dólares estadounidenses).

potencialmente su costo total en su tiempo de vida en un 40%! y, lo que es más importante para muchos gerentes, podríamos reducir su costo de desarrollo en un 25 % (la mitad del costo de pruebas y correcciones). Estos son valores importantes.

9.2 DISEÑO ESTRUCTURADO PARA CAMBIABILIDAD

9.2.1 ¿Qué contribuye a que un sistema sea modificable?

Existe un acuerdo general en el sentido de que los sistemas más fáciles de cambiar son aquellos que están constituidos por módulos manejablemente pequeños, cada uno de los

cuales es independiente, hasta donde es posible, de cualquier otro, de manera que pueden sacarse del sistema, cambiarse, y reponerse sin afectar al resto del sistema.

¿Qué queremos significar con "módulo"?

¿Qué es "manejablemente pequeño"?

¿Qué es "independencia"?

Un módulo lógico es una función definida con un nombre que expresa el propósito de la función; los procesos que hemos definido en el diagrama de flujo de datos, como "Verificar cantidad despachada", "Calcular total de los pedidos", "Generar pagos", pueden considerarse módulos lógicos. Un módulo físico es una implementación particular de un módulo lógico, normalmente en términos de cierto grupo de sentencias en un lenguaje de programación, al cual se puede hacer referencia por un nombre. Por lo tanto un módulo físico puede ser un programa, un sub-programa, una sección COBOL, o aun un parágrafo COBOL. Cuando el lenguaje de control de trabajo permite dar un nombre a los conjuntos de sentencias y almacenarlos como un procedimiento catalogado, este procedimiento es un módulo físico. Estrictamente hablando, un conjunto de instrucciones a un operador de consola o a un empleado puede considerarse como un módulo físico. Obviamente, los módulos a menudo invocan a otros módulos, los cuales invocan a otros módulos, etc., como se muestra en el croquis de la Fig. 9.2. Dentro de un sistema, o dentro de un programa, los módulos caen naturalmente dentro de una jerarquía de jefe, sub-jefe, sub-sub-jefe, etc., hasta los módulos *de trabajo* que desempeñan tareas particulares.

Cuando decimos que queremos un módulo manejablemente pequeño significamos que una persona competente será capaz de tomar el listado de un módulo, leerlo y hacerse mentalmente un cuadro de su función interna mientras decide cómo cambiarlo. Algunos módulos pueden ser tan pequeños como de 10 líneas de codificación y otros (donde la función es simple en principio pero de larga ejecución, como el caso de un problema de programación lineal) pueden implicar algunos centenares de líneas; 50-100 líneas es un tamaño común para módulos de trabajo bien formados.

Los módulos que componen un sistema no pueden ser completamente independientes unos de otros o no existiría un sistema sino solamente un montón de módulos aislados. La tarea del diseñador es formar los módulos y diseñar sus interconexiones para minimizar la posibilidad del temido *efecto de onda*(*ripple*, en inglés) [9.9]. El efecto de onda se presenta cuando el diseñador (o implementador) ha permitido que las diversas partes del sistema se conecten de manera confusa; tal vez dos módulos comparten el mismo almacenamiento temporario, o una parte del programa lee un interruptor que se acciona en una parte completamente diferente del programa. Si existen muchos de estos sutiles acoplamientos intermodulares, el programador de mantenimiento tendrá una vida muy dura. Supongamos que un usuario necesite hacer un cambio que afecte al módulo A. El programador resuelve los detalles del cambio, modifica el módulo A y pone el sistema nuevamente en producción. ¡Ay! el cambio del módulo A de alguna manera causa un error en el módulo B, y este error no es percibido hasta mediados de la noche siguiente. El programador rastrea el error y modifica el módulo B para anularlo. Pero este cambio afecta el módulo C, y reparar el módulo C deteriora al módulo D... El efecto del cambio original provoca "ondas" a lo largo del sistema, como las ondas causadas al tirar una piedra en un estanque. Algunos sistemas están tan interconectados que se vuelven imposibles de mantener; el pobre programador persigue los errores por todo el sistema sin poder eliminarlos jamás.

Las normas del diseño estructurado son de gran ayuda para producir sistemas en los cuales el efecto onda se mantiene en un mínimo.

Otro factor que hace a la cambiabilidad es el grado con el cual el diseñador logra *aislar la función* en la menor cantidad de módulos posible. Esto resulta obvio, una vez expuesto; si el usuario quiere cambiar la política del cálculo de descuento y el descuento está calculado en un módulo (el cual es modificable sin originar el efecto onda), el cambio será relativamente fácil, rápido y barato de hacer. Si las diferentes partes del cálculo de descuento están distribuidas a través de varios módulos o si se calculan descuentos independientemente dentro de varios

módulos en el sistema, los cambios serán correlativamente más difíciles de introducir.

En último término, queremos que nuestras funciones estén contenidas dentro de *cajas negras*, donde la función producirá siempre resultados predecibles a partir de un conjunto de datos que pasen a través de la misma. Por caja negra, específicamente queremos decir:

1. Que produce resultados que pueden predecirse, vista desde los módulos que la invocan.
2. Que no necesitamos conocer la codificación *interna* específica del módulo para conocer su función.

La Fig. 9.5 resume los factores que hemos discutido para construir diseños modificables.

Sistema compuesto de una jerarquía de módulos caja negra.
Cada módulo manuablemente pequeño.
Cada módulo modificable sin crear un efecto onda.
Las funciones del usuario aisladas al menor número posible de módulos.

Figura 9.5 Factores que contribuyen a un diseño cambiable.

9.2.2 Derivar un sistema modificable desde un diagrama de flujo de datos

La experiencia del diseño estructurado y el examen de los sistemas que se conocen son cambiables, sugieren que los módulos en un sistema cambiable a menudo se parezcan a unidades de una organización militar. Cada módulo tiene su propia tarea, que desempeña cuando recibe órdenes superiores; se comunica solo con su jefe superior y con sus subordinados, a los cuales a su vez imparte órdenes. La Fig. 9.6 ilustra esta analogía.

En esta jerarquía modular el módulo "jefe de sistema" invoca al "jefe de entrada" con la orden implícita "Deme una transacción correcta". El jefe de entrada a su turno invoca al módulo "lector" con la orden "Consígame una transacción". El lector desempeña esta sola función en su vida y luego devuelve el control a su jefe superior, diciendo "Acá hay una transacción" o "Acá no hay más transacciones". Ignorando la posibilidad de fin de archivo por el momento, el jefe de entrada "llama" a un módulo "editor" le transfiere la transacción en bruto y le pregunta "¿Es ésta una transacción correcta?" El editor ejecuta su función y retorna con la respuesta. Si la transacción no es buena, el jefe de entrada decidirá si debe ser arreglada de alguna manera o si debe ser rechazada, y el lector recibirá la orden de obtener otra. La suma total de este proceso es que el jefe de entrada devuelve el control al jefe de sistema y le trasfiere una transacción correcta. El jefe de sistema pasa entonces esta transacción correcta al "jefe de trasformación", el cual, utilizando a cada uno de sus subordinados para que ejecute su función especial, calculará el resultado que deberá tener la salida correspondiente a la transacción correcta original. La trasformación deberá calcular el pago neto y las deducciones de un registro de pago, o determinar el plan de vuelo óptimo para un avión con los parámetros de carga y estado del tiempo, o determinar la cantidad de mercadería a pedir con la información de consumo y condiciones de descuento. Cualesquiera que sean los detalles, el módulo jefe de trasformación hará llegar a su jefe el resultado, el que será entregado al "jefe de salida" para su disposición. Obsérvese que los módulos de trabajo no se comunican directamente entre ellos, sino con sus jefes. Esta es una forma de simplificar el acoplamiento inter-modular y hacer fácil de comprender el comportamiento del sistema.

Los sistemas con este tipo de estructura, en los cuales un tramo de entrada maneja todas las funciones de entrada, un tramo de trasformación toma una entrada correcta y produce un resultado y un tramo de salida maneja toda la salida correspondiente a ese resultado, se denominan sistemas *centrados en trasformación*, por razones obvias. Su estructura

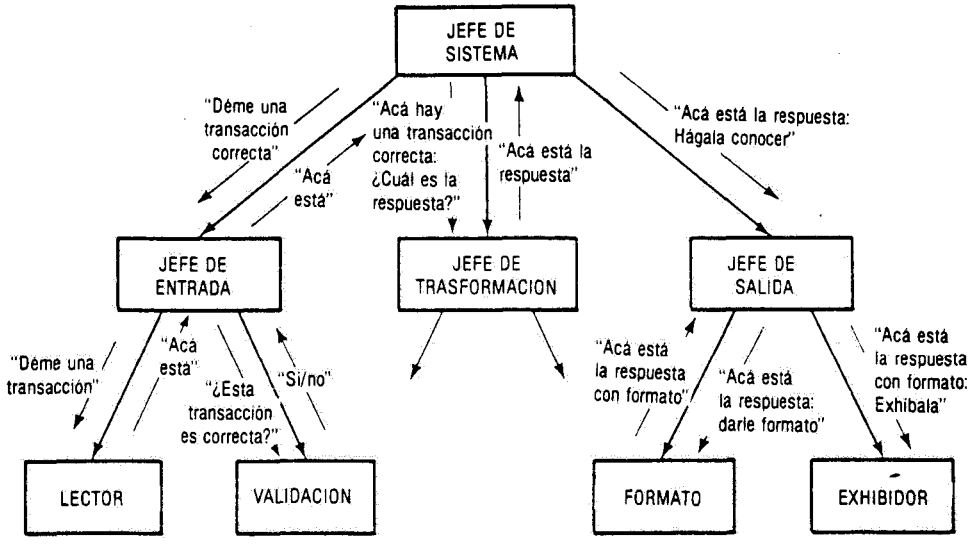


Figura 9.6 Analogía con la organización militar.

jerárquica puede establecerse casi directamente desde el flujo de datos a través del sistema como puede verse en la Fig. 9.7.

Para pasar del flujo de datos a una estructura jerárquica, empezamos por la forma más bruta de la entrada y la rastreamos a través del flujo de datos hasta alcanzar un punto donde no se puede decir que es una entrada. Igualmente se toma la salida final y se rastrea hacia atrás dentro del sistema hasta donde no se pueda concebir pensar más como una salida. La Fig. 9.8 muestra el flujo de datos con estos puntos marcados.

Una vez que estos *puntos de mayor abstracción* han sido identificados, siempre podremos crear un diseño centrado en trasformación mediante la especificación de un sistema que requiere la forma de entrada más abstracta, la transforma en la forma de salida más abstracta y la coloca en la salida. La Fig. 9.9 muestra el *nivel superior* de este tipo de jerarquía.

Nótese que en este caso hemos denominado a los módulos de acuerdo con sus funciones o las "órdenes que ellos obedecen" en lugar de llamarlos "jefe de entrada", etc. Esto es precisamente similar a la forma en la cual hemos descripto las funciones lógicas en nuestro diagrama de flujo de datos, utilizando un verbo activo y un objeto único.

Podemos completar los módulos del nivel inferior y marcar en este así llamado *diagrama de estructura* los datos que se han trasferido entre módulos. Es convencional mostrar cada estructura de datos o elemento de datos con una flecha pequeña con un pequeño círculo en blanco en el extremo: ○⟶ . Donde un módulo trasfiere una señal o interruptor de control a otro módulo, indicando al módulo receptor qué fue lo que sucedió o qué debe hacer, el elemento de control se indica con un círculo lleno: ●⟶ .

La Fig. 9.10 muestra la estructura hipotética de nuestro sistema utilizando estas convenciones. "Señal de validación" es un elemento de datos de control, operado por el módulo VALIDAR ENTRADA, tal vez a "sí", si la entrada pasa la validación o a "no" en caso contrario.

El sistema centrado en trasformación, comúnmente se deriva de un flujo de datos donde todas las transacciones siguen caminos iguales o muy similares. A menudo, especialmente en los sistemas comerciales, éste no es el caso, el flujo de datos muestra que el sistema maneja

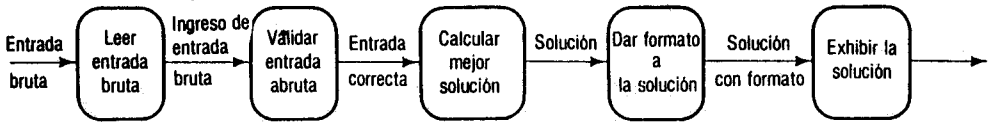


Figura 9.7 Flujo de datos hipotético.

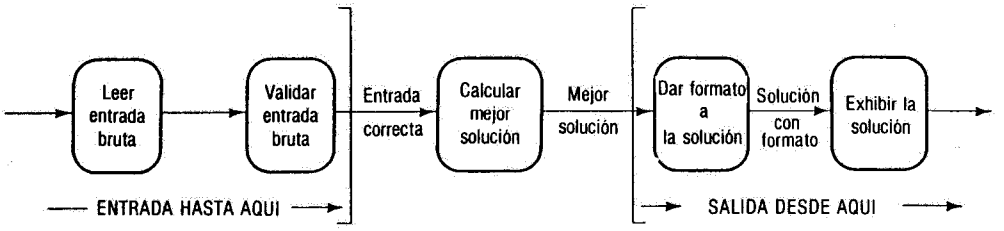


Figura 9.8 Puntos de mayor abstracción de entrada y salida.

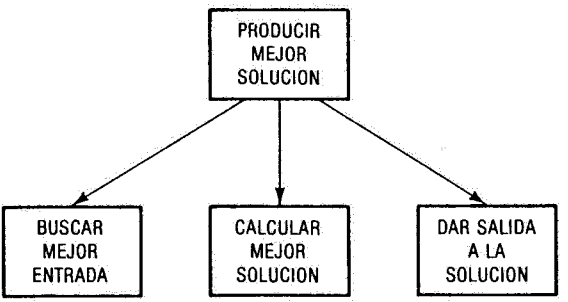


Figura 9.9 Jerarquía básica derivada de un flujo de datos.

varios tipos diferentes de transacciones, como se indica en la Fig. 9.11. Aquí se muestran cuatro tipos diferentes de transacciones (podrían ser más o menos), cada una de ellas tendrá que validarse en forma separada, y luego ser empleada para actualizar diferentes archivos maestros, según una lógica diferente, después de la cual se imprime un registro o diario para mostrar los detalles de cada transacción, junto con los contenidos de cada registro maestro antes y después de la actualización.

Una jerarquía modificable correspondiente a este módulo de flujo de datos se muestra en la Fig. 9.12. En este tipo de sistema el módulo ejecutivo principal invoca primero un módulo *analizador* que retorna con una transacción y su tipo; el ejecutivo principal entonces invoca a un *despachador* cuya tarea es dirigir la transacción a un subsistema apropiado establecido para cada tipo de transacción.

Este modelo de jerarquía se conoce *como centrado en transacción.* Como vimos cuando consideramos nuestro ejemplo de diseño, muchos sistemas de flujos de datos implican una combinación de ambos tipos de jerarquías, trasformaciones y transacciones, de manera que la clave reside en que en el punto de trasformación haya un camino de datos o varios. Al estudiar el diagrama de flujo de datos detallado, el diseñador puede determinar la estructura del nivel superior (centrada en trasformación o en transacción) y derivar una jerarquía de "primer corte"; luego el problema está en asegurar que los módulos de la jerarquía y la comunicación

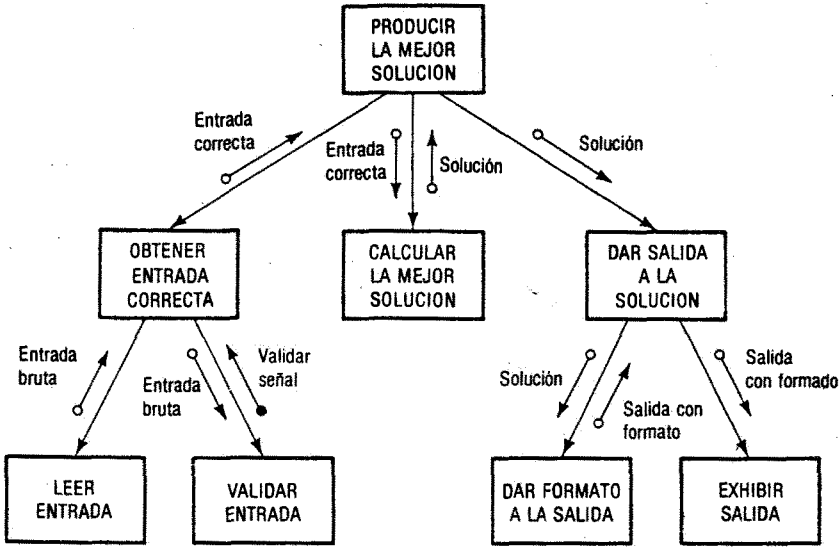


Figura 9.10 Diagrama de estructura para un sistema simple centrado en trasformación.

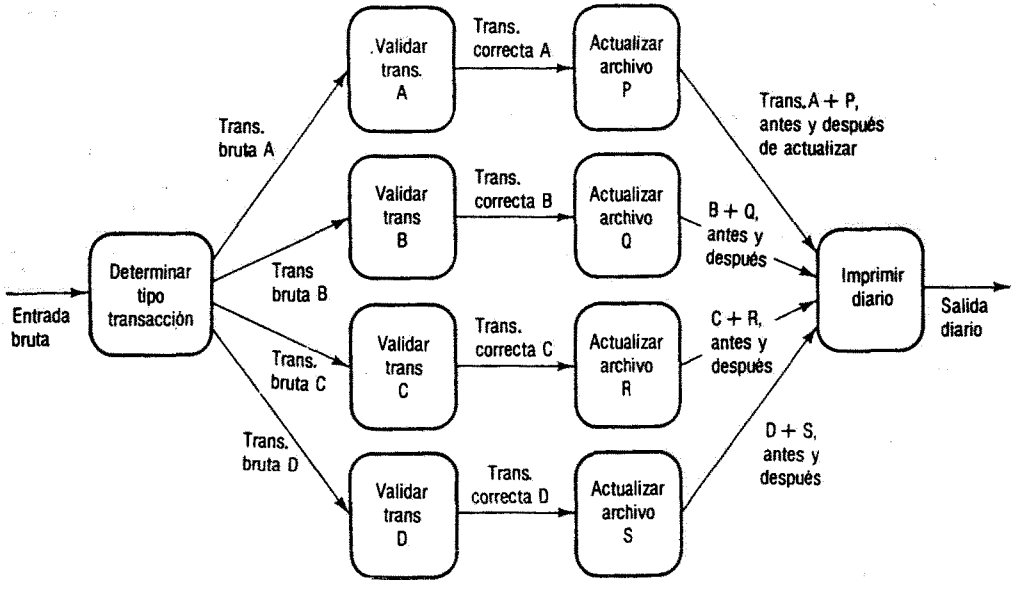


Figura 9.11 Flujo de datos con varios tipos de transacciones.

entre ellos es todo lo modificable que sea posible. Para hacer esto, necesitamos algunas guías de acción, como ser la forma de llegar al acoplamiento más modificable entre módulos, y qué constituye un módulo bien formado. Analizaremos por turno cada uno de estos factores.

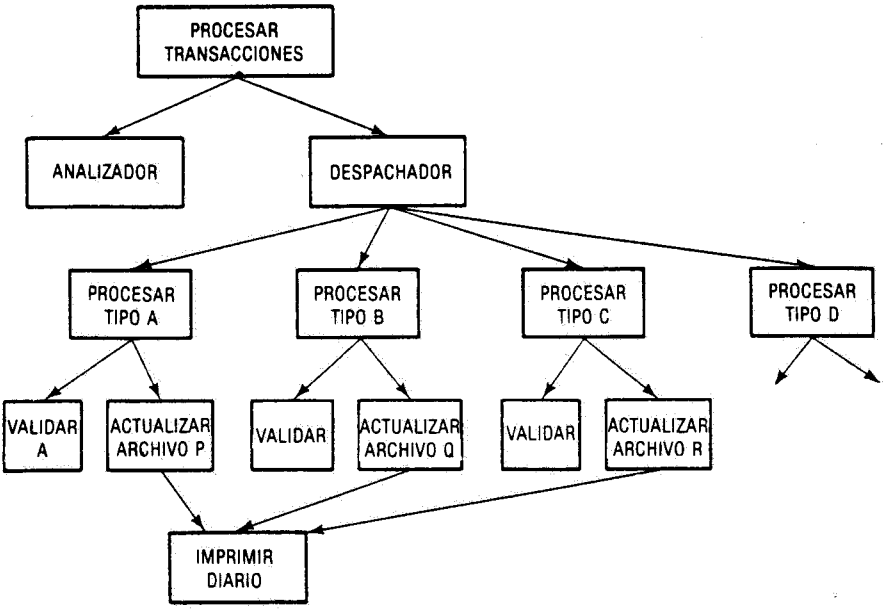
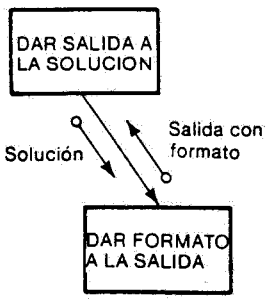


Figura 9.12 Jerarquía centrada en transacción.

9.2.3 Acoplamiento de módulos

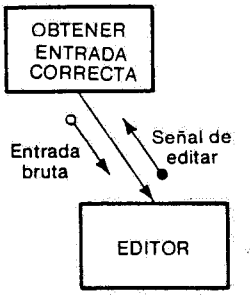
Como dijimos, uno de los objetivos del diseño en cuanto hace a la cambiabilidad es tener el menor acoplamiento posible entre los distintos módulos, sin afectar el funcionamiento al sistema. ¿Qué tipos de acoplamientos existen? ¿Son algunos peores que otros? Surgieron diversos estudios de acoplamientos con tipos ligeramente diferentes [9.2, 9.3, 9.10]; resumiremos los puntos sobre los cuales hay un acuerdo general.

1. *Acoplamiento de datos.* Está claro que la forma más deseada de acoplar es donde un módulo le trasfiere datos a otro como parte de la invocación o del retorno del control. Este es el mejor acoplamiento debido a que es acoplamiento mínimo. Los dos módulos que se muestran a continuación están acoplados por datos. En general un diseño en el cual se trasfieren entre módulos pocas porciones de datos es más modificable que otro en el que se transfieren muchos datos:



El acoplamiento es más claro y fácil de seguir cuando los elementos de datos se trasfieren como parámetros en una interfaz entre módulos, en vez de que el dato sea parte de un conjunto global o común, al que pueden tener acceso todos los módulos. Uno de los problemas de cambiabilidad que aparece en el COBOL proviene del hecho que todos los módulos (secciones o parágrafos) dentro de un programa COBOL dado, tienen acceso a la totalidad de la división de datos, especialmente el almacenamiento de trabajo. Por lo tanto cada módulo dentro del programa tiene, en teoría, acoplamiento-por-datos a cualquier otro dato. A menudo es valioso establecer en forma separada un área de almacenamiento de trabajo que pertenezca a cada módulo y no pueda ser utilizada por ningún otro módulo excepto para la trasferencia explícita de datos como se ha ilustrado más arriba. Esta situación también es válida para las sentencias de FORTRAN COMMON, pero es posible una implementación alternativa de FORTRAN mediante la trasferencia de parámetros explícitos).

2. *Acoplamiento de control.* Idealmente un sistema podrá diseñarse solo con acoplamientos de datos explícitos de módulo a módulo. Sin embargo cada vez que un módulo de trabajo lee, graba, o entra de alguna otra manera en contacto con el mundo exterior, tiene que informar a su jefe lo ocurrido; puede alcanzar fin-de-archivo, o detectar que una transacción es inválida, o que un número de cuenta no está en el archivo. Esta *información de retorno* involucra trasferir una variable de control, como vimos en la Fig. 9.10.



El acoplamiento de control tiene un efecto más serio sobre la cambiabilidad que el acoplamiento de datos y debe mantenerse al mínimo absoluto necesario para que funcione el sistema. A medida que aumenta la cantidad de interruptores y señales, se hace más compleja la tarea del programador de mantenimiento. Las señales de control trasferidas *hacia abajo* de la jerarquía, es decir, en el momento de invocación, son indicaciones de que el módulo invocado no es caja negra; será ejecutado en formas diferentes dependiendo de las señales de control. Esta situación no es deseable debido a que implica que el módulo invocado contiene una mezcla de funciones.

3. *Acoplamiento patológico/externo/interno.* Estos términos se refieren al acoplamiento rígido entre módulos en el cual un módulo apunta al interior de otro, ya sea para extraer algunos datos definidos dentro del segundo módulo, o para trasferir el control al interior del segundo módulo, o para modificar la forma de ejecutar dicho segundo módulo. Ver Fig. 9.13. Si, por ejemplo, un módulo de salida lee un contador de transacciones que pertenece a un módulo de entrada, el acoplamiento entre los módulos puede no ser visible, ya que el contador de transacciones no es trasferido hacia arriba ni hacia abajo en la jerarquía. El programador de mantenimiento puede modificar alegremente el módulo LEER sin darse cuenta del daño efectuado al módulo de salida. Cuando una situación se presenta tan raramente que no vale la pena trasferir datos o control a lo largo de toda la jerarquía, *puede* justificarse un acoplamiento severo de esta naturaleza. En general, sin embargo, cualquier forma de acoplamiento que no sean datos o control, deberá evitarse a toda costa; aquí es donde comienza la onda sin fin. Esto implica que cuando estamos diseñando los módulos debemos estar seguros de especificar explícitamente todos los datos o información de control necesarios para que el módulo funcione.

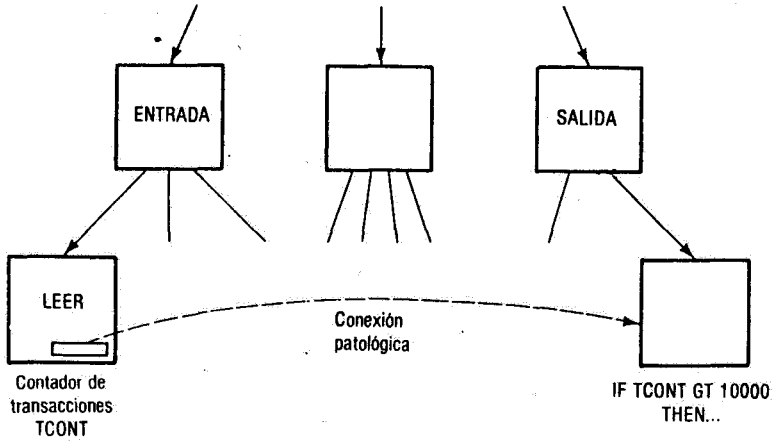


Figura 9.13 Conexión patológica.

9.2.4 Módulos bien formados: coherencia, cohesión, ligazón

Al mismo tiempo que se minimiza el acoplamiento debemos especificar módulos elegidos y formados lo mejor posible. Los términos empleados para describir la cualidad del módulo —coherencia, cohesión y ligazón— indican la medida en la cual todas las partes de un módulo se corresponden entre sí. Un módulo altamente cohesivo, cuyas partes contribuyan todas a una sola función, probablemente no necesite mucho acoplamiento con otros módulos; un módulo pobremente cohesivo, a menudo una combinación de partes no relacionadas, probablemente necesite un alto acoplamiento con otros. Luego el bajo acoplamiento y la alta cohesión van unidas de la mano y viceversa.

Diversos estudios identifican seis o siete tipos de cohesión entre módulos. Describiremos los tipos más generalmente aceptados, desde el peor al mejor.

1. *Cohesión coincidente* (la peor): No puede apreciarse que los elementos del módulo lleven a cabo ninguna función definible. Están allí por accidente. Cuando una organización impone como norma que los programas deberán ser modulares y que ningún módulo debe contener más de 50 sentencias, un programador tomará un listado de un programa de 2.000 sentencias y un par de tijeras y cortará el listado cada 50 líneas para hacer módulos de las partes. Estos módulos fueron cohesivos por coincidencia.

2. *Cohesion lógica* (sigue a la peor): En este tipo de módulos varias funciones semejantes, pero ligeramente diferentes, están combinadas juntas, configurando un módulo más compacto que si cada función fuera programada separadamente. Un ejemplo típico es un módulo que valida todos los tipos de transacciones empleando secciones comunes de codificación cuando corresponde y salteando otras partes de la codificación cuando no son requeridas por un tipo particular de transacción. Un módulo lógicamente cohesivo a menudo requiere un interruptor de control al ser trasferido desde el módulo que lo invoca, para indicarle cómo debe ejecutarse en esa ocasión específica. Las señales de control que se van trasfiriendo hacia abajo de la jerarquía son entonces a menudo indicios de la presencia de módulos lógicamente cohesivos. Los módulos de este tipo son a menudo difíciles de modificar, debido a que los circuitos lógicos que pasan a través de ellos son complejos. Deben ser remplazados por módulos de propósitos especiales a razón de uno por función.

3. *Cohesión temporal* (de moderada a pobre): Los módulos tales como "inicialización", "preparación previa" y "reiniciación automática" contienen una variedad de funciones cuyo único elemento común es el de ser ejecutados al mismo *tiempo.* La cambiabilidad se mejora aislando cada función en su propio módulo.

4. *Cohesión de procedimiento* (moderada): Este tipo de cohesión se encuentra donde los módulos han sido derivados de un diagrama de flujo y cada "pedazo" o procedimiento del diagrama ha sido armado en un módulo. Dentro de cada módulo se ejecutan varias funciones, pero por lo menos las funciones están relacionadas a través del flujo de control entre ellas.

5. *Cohesión de comunicación* (moderada a buena). Las funciones componentes de un módulo con cohesión de comunicación operan todas sobre la misma corriente de datos además de ser cohesivas de procedimiento. Algunas descripciones de módulos de comunicación podrían ser "Representar el resultado y grabar registro del diario", "Leer sentencia fuente y eliminar blancos" y "Calcular solución e imprimir resultado".

6. *Cohesión funcional* (la mejor): Un módulo funcionalmente cohesivo, que es nuestro ideal, realiza una y solo una función identificable. Una prueba para un módulo funcional es ver si puede ser acusado como de comunicación, de procedimiento, temporal o lógicamente cohesivo. Si es inocente de estos cargos, probablemente sea funcional. Los módulos funcionalmente cohesivos pueden comúnmente describirse por medio de una sola frase, con un verbo activo y un objeto único del tipo que hemos encontrado antes, tales como "Calcular la mejor solución", "Validar consulta" y "Representar respuesta". Una guía para determinar el nivel de cohesión de un módulo es escribir una sola oración comenzando con "El propósito total de este módulo es ..." y luego examinar esta oración. Si la misma no puede ser completada, el módulo es posiblemente solo cohesivo coincidentemente. Si la sentencia tiene un objetivo plural e incluye la palabra *todas*, por ejemplo, "Validar todas las transacciones" el módulo probablemente sea lógicamente cohesivo. Si la oración emplea palabras que guardan relación con tiempo o secuencia —primero, próximo, luego, después, de otra manera, comienzo— podrá ser cohesivo temporal, de procedimiento o de comunicación. Si la oración consiste en un solo verbo activo con un objeto no plural, el módulo probablemente sea funcionalmente cohesivo. Una de las razones para tomarse el trabajo en describir los procesos lógicos que identificamos durante el análisis en términos funcionales, es que haciéndolo así se facilita al diseñador la identificación de módulos funcionales a partir del diagrama de flujo de datos. Así como conociendo la tercera forma normal es más probable que el analista pueda identificar almacenamientos de datos lógicos simples, conociendo la cohesión funcional es más probable que el analista pueda identificar procesos simples.

9.2.5 Problemas de alcance de efecto/alcance de control

Suponiendo que hemos diseñado una jerarquía utilizando estas guías para minimizar el acoplamiento y maximizar la cohesión, hay otro refinamiento que considerar.

En algunos casos un módulo puede contener lógica para tomar una decisión sobre algún acontecimiento; basado en esta decisión el módulo invocará otros módulos. La Fig. 9.14 muestra un caso simple.

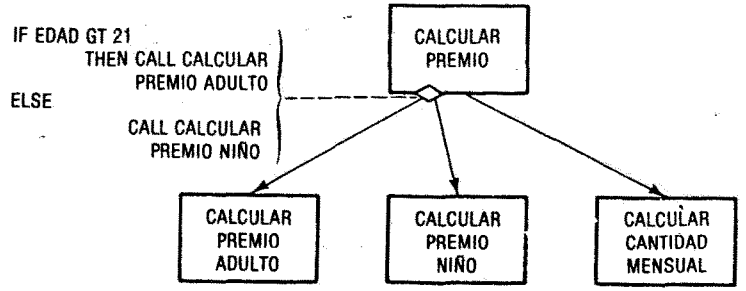


Figura 9.14 Diagrama de estructura indicando el alcance de efecto dentro del alcance de control.

El denominado *alcance del efecto* de la decisión son los dos módulos que están por debajo de CALCULAR PREMIO; la decisión tiene un efecto en base al cual será invocado uno de ellos. El llamado *alcance del control* de un módulo son todos los módulos que el llama, y todos los módulos llamados por ellos, etc.

En la Fig. 9.14 el alcance del control de CALCULAR PREMIO cubre a los tres módulos llamados por él. El alcance del efecto de la decisión sobre la edad está entonces dentro del alcance del control del módulo que toma la decisión; así es como debe suceder. Así como un oficial no impartirá órdenes a la tropa que no se encuentra bajo su mando, ningún módulo tomará una decisión que afecte a módulos fuera de su alcance de control.

Pero supongamos que el diseño fuera como el de la Fig. 9.15. Los pequeños rombos con flechas de invocación salientes indican que se toma una decisión en alguna parte del módulo, y cuáles serán los módulos subordinados que se invoquen dependerá de dicha decisión. Hemos indicado la lógica de la decisión en cada caso.

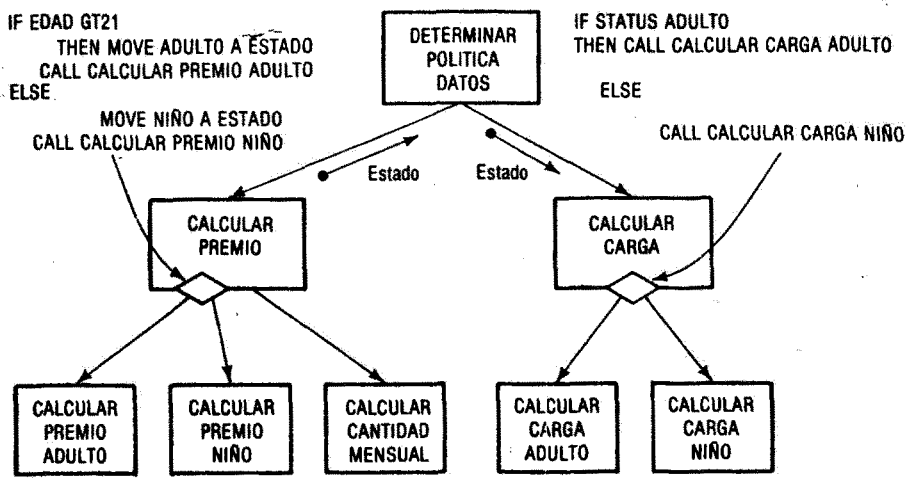


Figura 9.15 Diagrama de estructura indicando el alcance de efecto fuera del alcance de control.

Ahora el alcance de efecto de la decisión original es más amplio; cubre no solo los módulos llamados por CALCULAR PREMIO sino también a los dos llamados por CALCULAR CARGA, ya que CALCULAR PREMIO pasa un elemento de control, ESTADO, a CALCULAR CARGA, el cual determina qué módulo inferior se va a llamar. El alcance del efecto de la decisión no permanece más dentro del alcance del control del módulo que contiene la decisión. Esto no es deseable por varias razones: crea acoplamiento de control entre módulos (vía la señal de control ESTADO, en este caso), crea decisiones duplicadas las cuales pueden crear problemas de mantenimiento, y confunde el efecto de la decisión.

La estructura deberá diseñarse nuevamente para llevar el alcance del efecto de la decisión dentro del alcance del control del módulo que contiene la decisión, como se muestra en la Fig. 9.16. En esta estructura modular no es necesario ningún acoplamiento de control y la decisión sobre la edad se toma una sola vez.

El diseñador deberá estar vigilando la oposición del alcance de control/alcance de efecto a medida que el diseño avanza.

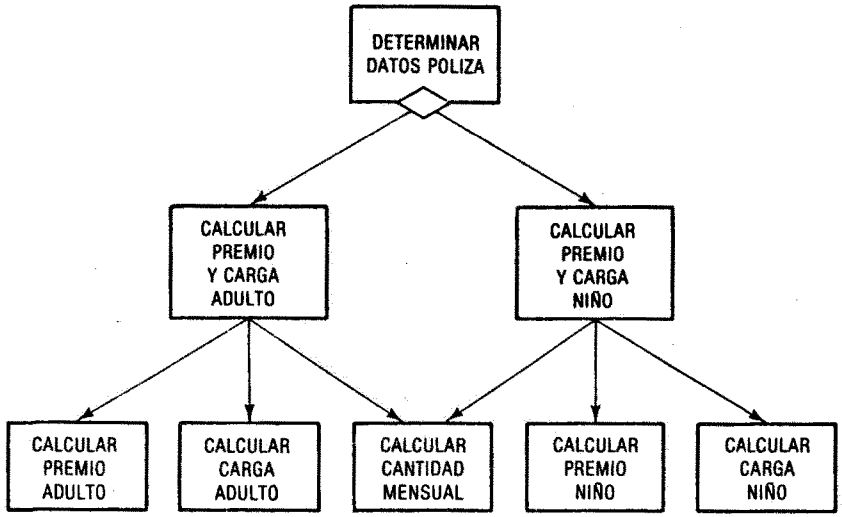


Figura 9.16 Diagrama de estructura indicando el alcance de control dentro del alcance de efecto.

Su presencia a menudo es señalada por variables de control innecesarias, como vimos en el ejemplo.

Derivar la estructura a partir del flujo de datos

- Centrada en trasformación donde entrada/salida es homogénea.
- Centrada en transacción donde diferentes tipos de estructuras de datos son procesados en forma diferente.

Minimizar el acoplamiento inter-modular

- Acoplamiento de datos en cuanto sea posible.
- Acoplamiento de control tan pequeño como sea necesario.
- Acoplamiento más riguroso solo bajo protesta.

Maximizar la cohesión del módulo

- Módulos funcionales en cuanto sea posible.
- Evitar módulos cohesivos lógicamente o por coincidencia a toda costa.

Verificar posibles oposiciones de alcance de control/alcance de efecto.

Figura 9.17 Guías para producir diseños modulares modificables.

Podemos resumir las guías principales del diseño estructurado tal como se indica en la Fig. 9.17.

9.3 LA SOLUCION DE COMPROMISO ENTRE CAMBIABILIDAD Y RENDIMIENTO

Ahora que hemos discutido los factores que contribuyen a un diseño modificable, podemos considerar el impacto de la cambiabilidad sobre el rendimiento y las soluciones de compromiso que el diseñador tiene que tomar.

Muchos diseñadores al encontrarse con técnicas de diseño estructurado por primera vez temieron que las mismas tuvieran un serio efecto sobre el rendimiento de cualquier sistema diseñado para ser modificable. Esto proviene principalmente de dos factores:

1. Diseñar un sistema como un conjunto de pequeños módulos modificables, en lugar de algunos programas grandes, podría involucrar una mayor estructura superior, causada por módulos que llaman a otros módulos.
2. Evitar módulos pobremente cohesivos podría significar dividir un módulo de propósito general lógicamente cohesivo, digamos, en varios módulos funcionales, dando como resultado programas que requerirían más memoria.

Además, aunque no lo discutimos específicamente, si se utiliza programación estructurada para la codificación de los módulos, se puede producir una cantidad relativamente mayor de instrucciones de máquina que si se utilizaran métodos más tradicionales con "ardides" de codificación para una mayor eficiencia de máquina. (Se acepta generalmente que la programación estructurada produce codificaciones que son más fáciles de leer y depurar; los estudios indican que la codificación estructurada puede ser a veces ligeramente más larga y más lenta que la codificación tradicional [9.11] pero a veces más reducida y veloz [9.12].

Estas preocupaciones acerca del rendimiento no pueden ocultarse bajo la alfombra, si bien en muchos aspectos no se justifica la preocupación sobre ello. Realmente, el hecho singular es que los diseños modificables pueden producir actualmente mejor rendimiento que los diseños tradicionales. Consideremos los siguientes hechos:

1. Hemos indicado que algunos de los factores más importantes del rendimiento de los sistemas orientados a datos eran los archivos innecesarios, pasadas de archivos y lecturas. El diseño estructurado no empeora estos efectos; por el contrario, la producción de la jerarquía de módulos lógicos alienta al diseñador a evitar archivos intermedios y a buscar cómo optimizar el acceso de archivos.
2. Hemos indicado que la cantidad de codificación generada y el tiempo perdido en la trasferencia del control entre módulos en la memoria principal era de mucho menor importancia en los sistemas orientados a datos que las trasferencias y accesos de archivos, excepto en el caso donde el llamado de un módulo origine la lectura de un archivo de biblioteca o una operación de paginado en un sistema de memoria virtual. Salvo estas excepciones, la producción de un diseño modificable afecta al factor de rendimiento de importancia mínima.
3. Es difícil conocer por anticipado dónde se producirán los cuellos de botella en el rendimiento de un sistema, en especial en un entorno operativo complejo. Lo que necesitamos ser capaces de hacer es implementar un sistema donde los problemas más importantes de rendimiento hayan sido resueltos, ensayar el sistema con datos reales y medir dónde se insume el tiempo, ya sea en la entrada/salida, en los cálculos o en la estructura superior del sistema. Una vez que conocemos *por medición* dónde están los cuellos de botella debemos ser capaces de modificar el sistema para eliminarlos. Típicamente, el tipo de cambios necesarios para mejorar el rendimiento será volver a escribir la tabla de búsqueda que consume tiempo en lenguaje "assembler", cambiar la estructura de superposición para asegurar que la sección frecuentemente llamada esté siempre en la memoria principal, o modificar los detalles de acceso a los archivos. En el pasado era difícil y costoso hacer esta *optimización de post-implementación.* ¿Por qué? Debido a que los sistemas habían sido diseñados de tal forma que eran difíciles de modificar ¡sin introducir el efecto onda! El hecho cierto de que ahora estemos diseñando nuestros sistemas para poder ser modificados significa que podemos optimizar sus rendimientos rápida y fácilmente una vez que sabemos qué partes necesitan mejorarse.

La solución de compromiso entre cambiabilidad y rendimiento es en consecuencia paradójica. Si el diseñador se preocupa mucho por los detalles finos del rendimiento del sistema durante el diseño, existe la posibilidad de introducir cohesión lógica y severo acoplamiento a través de la estructura modular, al intentar economizar memoria y tiempo de ejecución. Todos estos hechos harán dificultoso y costoso mejorar el rendimiento del sistema una vez que haya sido implementado.

Así, ¿cuál es la moraleja? El diseñador necesita adoptar una aproximación de seis pasos:

Paso 1. Obtener el diseño físico tentativo global del sistema y los archivos principales a partir del modelo lógico, con la menor cantidad de archivos intermedios, las menores pasadas posibles de archivos, y la menor cantidad posible de búsquedas en estos archivos suplementarios.

Paso 2. Incluir los controles necesarios, tratando de minimizar cualquier efecto degradante del rendimiento. (El diagrama de flujo de datos es una herramienta útil para presentar a los auditores para la discusión de los controles de auditoría).

Paso 3. Diseñar la estructura de módulos lógicos para máxima cambiabilidad, sin prestar atención en este paso al rendimiento del sistema final (salvo los aspectos ya tratados en los pasos 1 y 2).

Paso 4. Estudiar los módulos lógicos en la estructura jerárquica para estimar el tamaño de los módulos físicos que será necesario implementar. Observar aquellos módulos que se invocarán frecuentemente (en cada transacción o en cada elemento de datos), y observar aquellos que se invocarán menos frecuentemente durante la ejecución (por ejemplo, rutinas de tratamiento de errores, inicialización o terminación). Elegir un embalaje físico de los módulos lógicos en programas, subprogramas o secciones, de manera que, hasta donde sea posible, los módulos llamados frecuentemente estén en la memoria al tiempo de ser llamados; es decir, se minimizará la cantidad de búsquedas de las bibliotecas de programas.

Paso 5. Implementar el sistema como se especificó en el paso 4 sin comprometer adicionalmente la cambiabilidad del diseño, excepto en los casos donde el rendimiento es tan vital que sobrepasa la importancia de la cambiabilidad. En estos casos existe el compromiso en combinar módulos y compartir recursos, pero esto debe hacerse en la menor medida posible.

Paso 6. Ejecutar el sistema con datos representativos y utilizar un monitor de tiempo de ejecución para medir qué partes del sistema toman la mayor proporción del tiempo de ejecución o identificar qué módulos están causando problemas de memoria. Cambiar estas partes para hacerlas más rápidas o pequeñas, según sea el caso. En un entorno de memoria virtual, redistribuir el orden físico de los módulos en la biblioteca apropiada para lograr el *conjunto de trabajo* (aquellos módulos que se utilizan frecuentemente), todos dentro del menor número de páginas. Considerar la posibilidad de dejar fijo el conjunto de trabajo en memoria real, si el sistema operativo lo permite.

Repetir el paso 6 hasta que el sistema satisfaga los objetivos de rendimiento. El diseño de sistemas orientados a algoritmos, puede en principio seguir *incluso* los pasos delineados anteriormente. Particularmente en sistemas muy complejos, las pruebas y la depuración necesarias se verán favorecidas si se mantienen las funciones lo más aisladas posibles hasta que pueda demostrarse que *todas* las funciones del sistema trabajan correctamente, pudiendo comenzarse entonces con la optimización.

9.4 UN EJEMPLO DE DISEÑO ESTRUCTURADO

En esta sección aplicaremos las guías y técnicas discutidas anteriormente en el capítulo para el nuevo sistema de la Corporación CBM. Vamos a asumir que se ha seguido la metodología discutida en el Capítulo 8 y que los usuarios han optado por la alternativa del presupuesto medio, en el cual los pedidos de libros ingresan al sistema vía terminales CRT y se validan en línea; los archivos de ítem despachables y no despachables se generan durante el

día. Las notas de despacho se imprimen desde estos archivos durante la noche junto con las facturas. Los niveles de inventario se ajustan a medida que van entrando los pedidos, pero el control del inventario y de los puntos de pedido son atendidos por un subsistema separado que verifica el stock disponible al finalizar cada día. Compras, cuentas a cobrar y cuentas a pagar son subsistemas separados, en lote. Algunos informes son creados en modo lote; hay facilidades de consulta disponibles en algunos de los archivos.

9.4.1 Los límites del diseño

A los fines de este ejercicio diseñaremos el subsistema de entrada de pedidos cuyos límites se definen en la Fig. 9.18 (que muestra un extracto de todo el diagrama de flujo de datos). Para simplificar, diseñaremos una versión inicial que atenderá aquellos pedidos que no han sido pagados previamente, es decir, aquellos que solicitan crédito. Como muestra el diagrama de flujo de datos, la entrada de los detalles de nuevos clientes se encuentra fuera del límite de nuestro diseño. De hecho, cuando se observa que un pedido no proviene de un cliente existente, o se nota un cambio en la dirección, es encaminado hacia el supervisor de entrada de pedidos, el cual ingresa los detalles en el archivo de CLIENTES regresando después el pedido al sistema para su procesamiento normal. Tenemos entonces que diseñar un sistema que maneje los tres procesos:

1. Validar pedidos (omitiendo los pagos previos, por ahora)
2. Verificar que el crédito sea correcto
3. Verificar el inventario disponible y ajustar el nivel de stock.

La Fig. 9.19 muestra las explosiones de cada uno de estos procesos, junto con algún detalle del flujo de datos en el proceso manual de tomar el pedido escrito o telefónico y entrarlo en el CRT. Estos procesos están numerados M1, M2, etc. ya que son parte de un proceso que no aparece como tal en el diagrama global de flujo de datos.

9.4.2 Consideraciones del diseño del archivo físico

Antes de considerar la construcción de la jerarquía modular, consideraremos algunos de los factores de la especificación del archivo físico, ya que ellos clarificarán el DFD y el diseño modular.

1. *DM/1: AUTOR/TITULO-INDICE, en el proceso M-Entrada Manual.* Cuando analizamos el almacenamiento de datos que describía los libros en el Capítulo 6, indicamos que el único identificador de un libro era el ISBN. Al mismo tiempo, los clientes piden un libro especificando su título y autor y al editor cuando se lo conoce. Cuando vamos a ingresar los pedidos de libros tenemos, entonces, la necesidad de un acceso inmediato a LIBRO-TITULO o AUTOR o a ambos para encontrar el ISBN. Esto podrá hacerse proveyendo índices secundarios al archivo de LIBROS y permitiendo el acceso vía los CRT al Departamento Entrada de Pedidos. Las características del archivo deberán ser:

Cantidad de registros: alrededor de 1.500 libros sobre computación son los conocidos por CBM.

Cantidad de accesos: uno por cada ítem pedido.

Volatilidad (régimen de cambios del archivo): alrededor de 20 nuevos libros por mes.

Cuando vemos un archivo pequeño con baja volatilidad como éste, que requiere accesos múltiples, nos debemos preguntar si no *es* factible proveer a los usuarios una salida impresa del archivo pre-clasificado de diferentes formas en lugar de crear y mantener índices. Durante el diseño tentativo para la alternativa del presupuesto medio, el diseñador decidió ahorrar dinero de esta manera.

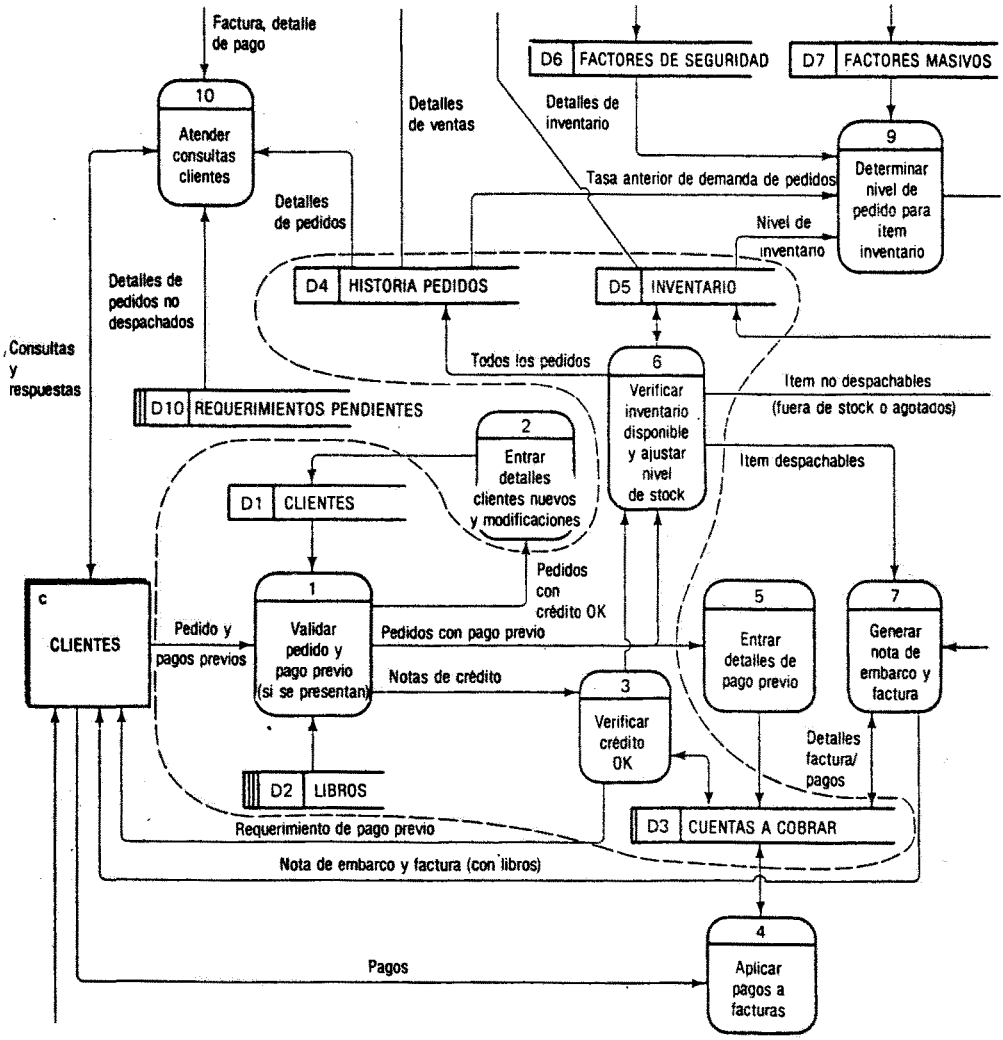


Figura 9.18 Subsistema de entrada de pedidos.

A cada empleado de entrada de pedidos se le provee un listado impreso de computadora con todos los títulos conocidos, con su autor, editor y el ISBN. Debido a las formas variadas en que la gente recuerda los títulos de los libros (este libro podría citarse como *Análisis estructurado, Herramientas de análisis de sistemas, Técnicas de sistemas estructurados,* etc.), el listado se provee como un índice de palabra clave del contexto (KWIC: KeyWord-In-Context) en el cual el título aparece una vez por cada palabra importante en él y la lista de títulos es clasificada por cada palabra. Un resumen de la sección S del índice KWIC es el siguiente:

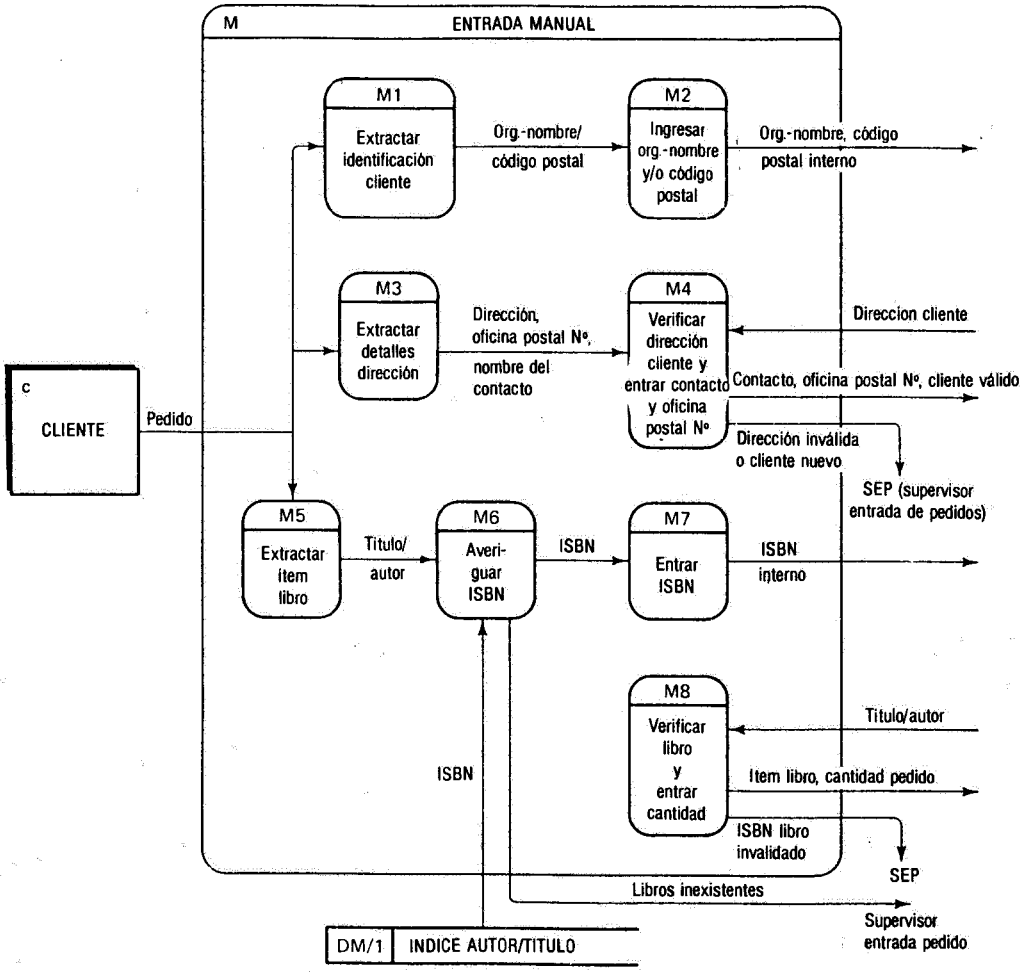


Figura 9.19 (parte 1) Explosiones.

•
•
•

SYSTEMS ANALYSIS AND DESIGN USING NETWORK TECHNIQUES
(Análisis y Diseño de Sistemas utilizando técnicas de red)

Whitehouse ISBN .....

STRUCTURED SYSTEMS ANALYSIS: tools and techniques
(Análisis Estructurado de Sistemas; herramientas y técnicas)

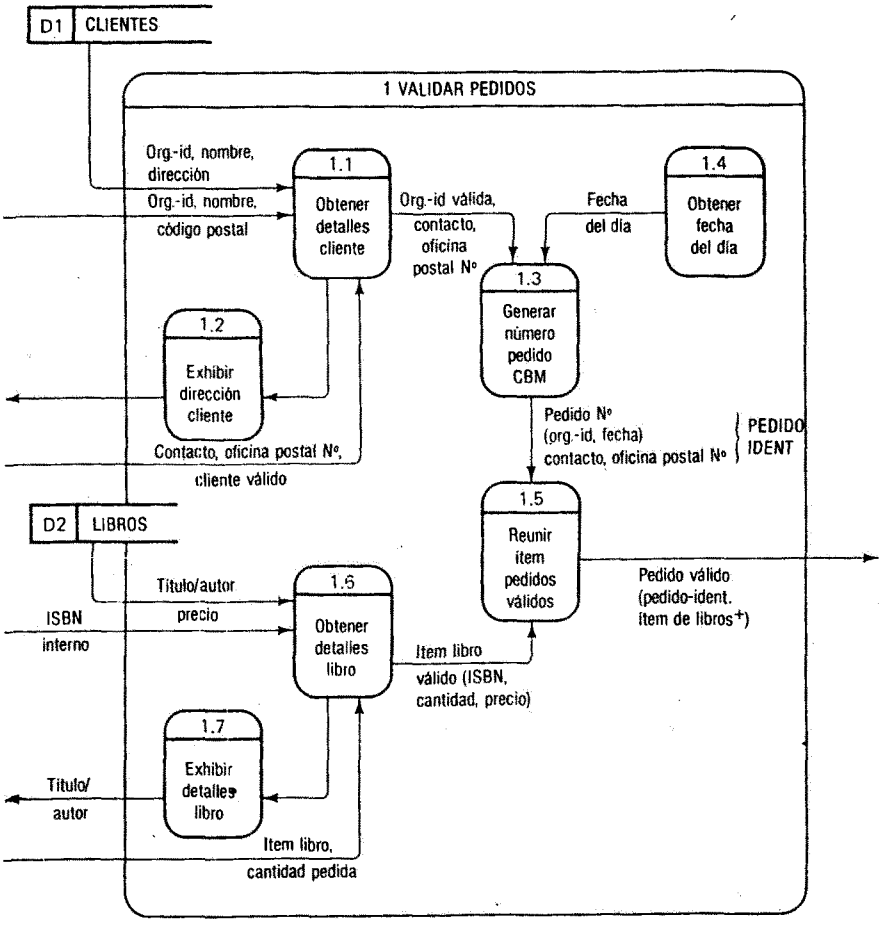


Figura 9.19 (parte 2)

Gane, Sarson ISBN .....

INTRODUCTION TO SYSTEMS SAFETY ENGINEERING
(Introducción a la Ingeniería de Seguridad de Sistemas)

Rodgers ISBN .....

INTRODUCTION TO GENERAL SYSTEMS THINKING
(Introducción al pensamiento general de sistemas)

Weinberg ISBN .....

•
•
•

El libro de Whitehouse también aparecerá bajo ANALISIS, DISEÑO, RED y TECNICAS; el libro de Weinberg también aparecerá bajo GENERAL y PENSAMIENTO, etc. Luego para localizar cualquier libro el empleado de entrada de pedidos podrá tomar la

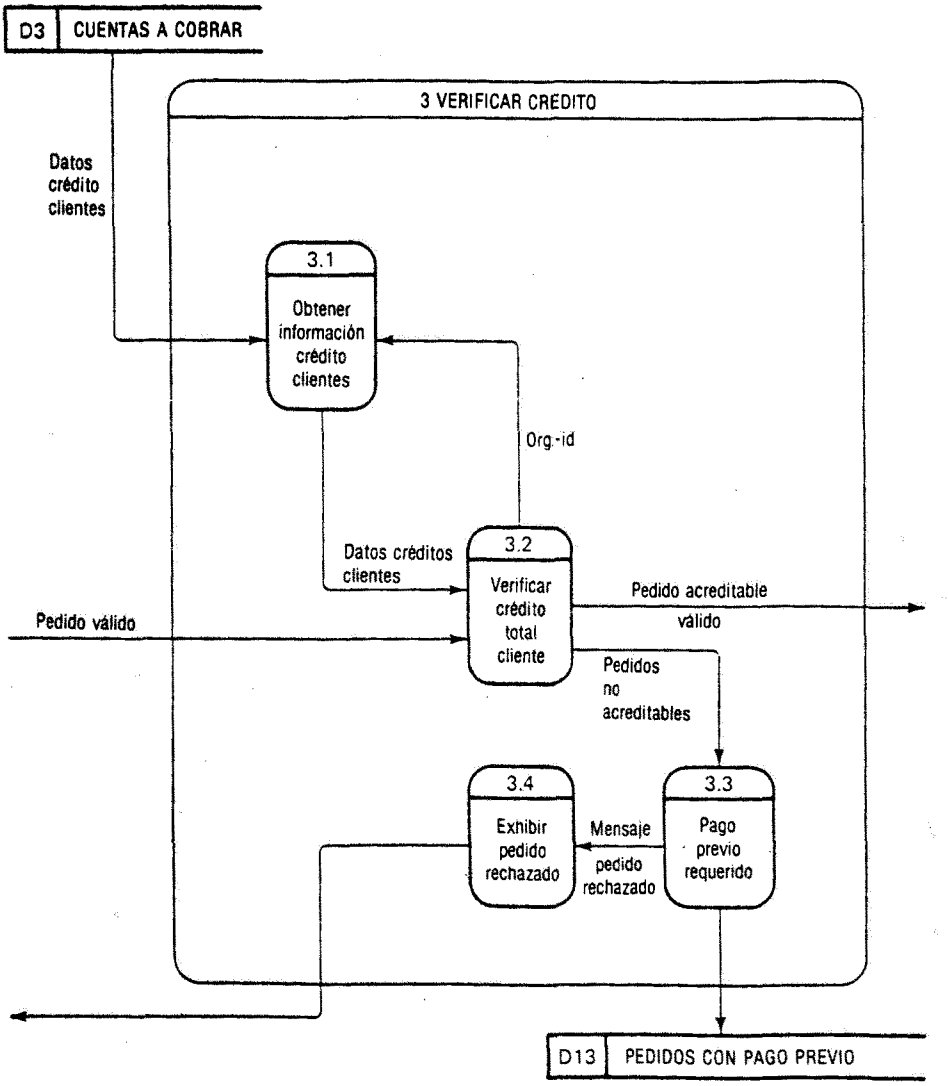


Figura 9.19 (parte 3).

palabra más significativa del título, buscar en el índice KWIC, y explorar a través de todos los libros que contengan esa palabra en cualquier lugar del título hasta localizarlo. El índice KWIC provee alguna ayuda para aconsejar a los clientes sobre libros referidos a un tópico dado.

Si el cliente especifica al autor pero es vago sobre el título o el título no se puede encontrar en el índice KWIC, el empleado de entrada de pedidos puede utilizar un segundo impreso con los libros listados en secuencia de AUTOR.

Las salidas impresas se producen una vez por mes (del archivo LIBROS), con la fecha actualizada impresa con claridad sobre las mismas; cada semana, cualquier agregado o

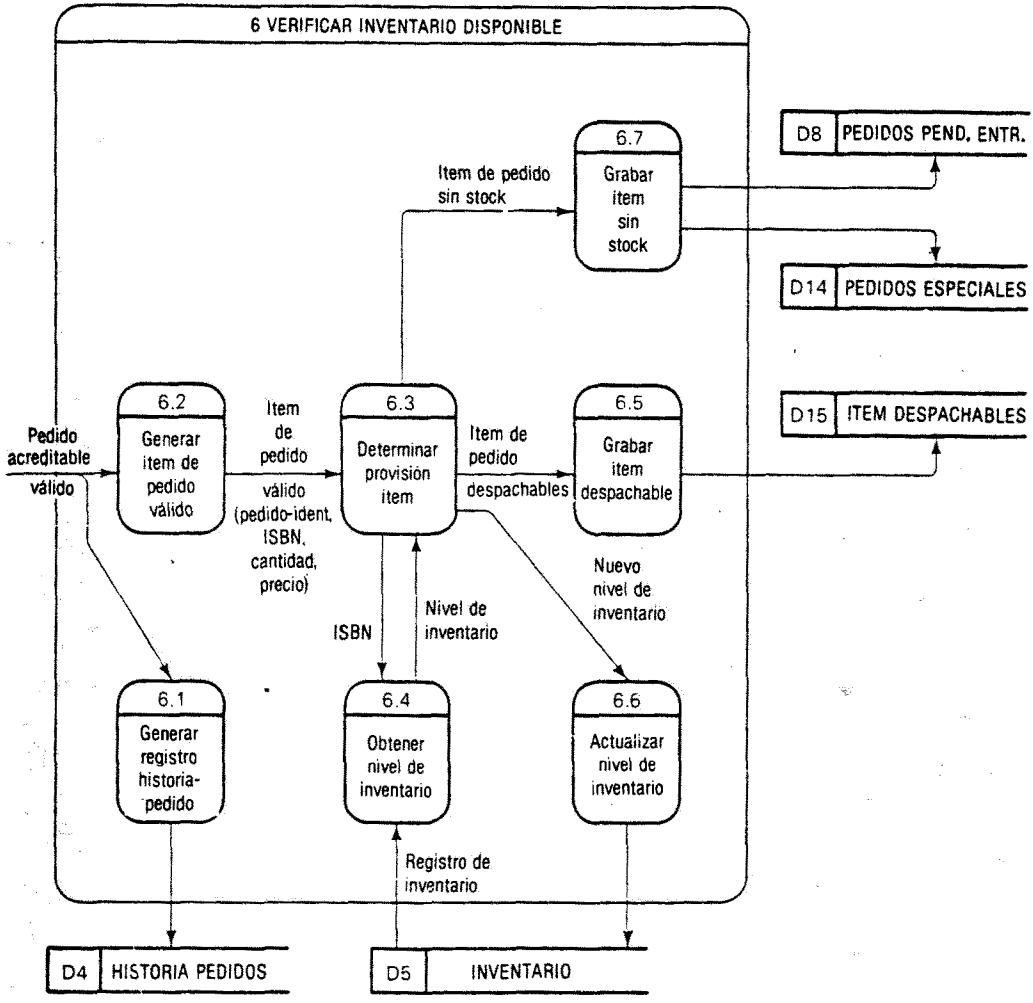


Figura 9.19 (parte 4).

cambio que haya tenido lugar circula como hojas actualizadas que deben ser agregadas al frente de cada impreso.

2. *D2: LIBROS, con acceso desde el proceso 1-Validar pedidos.* Una vez que el ISBN de cada libro ha sido determinado y entrado al sistema, los detalles del libro deben ser representados ante el empleado de entrada de pedidos como una doble verificación de que se utiliza el ISBN correcto. El archivo LIBROS contiene la misma información que el índice TITULO/AUTOR, con el agregado del precio actual de cada libro. Es un archivo pequeño

—1.500 registros con un promedio de 100 caracteres, o sea, 150.000 caracteres. Toda vez que, como mostró en el Capítulo 6, tanto LIBROS como INVENTARIO están organizados por ISBN, el diseñador consideró que ambos podían estar combinados y contenidos en un solo archivo físico, pero decidió lo contrario, ya que del total de esos 1.500 títulos, solo 100-200 se conservarán en inventario. También el archivo LIBROS, que será mantenido cada noche, será leído, pero no actualizado, de día. En consecuencia, no será necesario tener copias de respaldo con fines de seguridad. Los datos de INVENTARIO serán actualizados constantemente, por lo que será necesario copiarlo por seguridad a intervalos frecuentes, ya que resultaría muy inconveniente perder el registro de lo que hay en stock. Como objetivo menor en orden al rendimiento, podría ser ventajoso poner LIBROS e INVENTARIO en unidades de discos separadas y ocupando el resto de cada unidad con archivos de baja actividad tales como PEDIDOS-PENDIENTES-DE-ENTREGA, PEDIDOS-ESPECIALES o PEDIDOS-QUE-REQUIEREN-PAGO. En cada caso el archivo de alta actividad es pequeño y solo ocupará unas pocas pistas. La cabeza lectora grabadora deberá entonces perder la mayor parte del tiempo en ubicarse sobre el archivo activo, reduciendo el tiempo de búsqueda a un mínimo.

Si la gerencia de CBM tiene éxito en atraer 1.000 pedidos por día, y cada pedido, digamos, por tres libros, y si muchos de los pedidos son telefónicos, habrá una actividad durante el período pico de la mañana (10, 30 a 11,30 horas) de 700-800 ítem que requerirán un acceso a LIBROS y un acceso a INVENTARIO. Esto significa un acceso cada 4 segundos en la hora pico, lo cual, sin llegar a ser abrumador, no es despreciable.

3. *D1: CLIENTES; con acceso desde proceso 1-Validar pedidos.* Dijimos en el Capítulo 6 que el almacenamiento de datos lógico CLIENTES consistía en registros que describen tanto a las empresas como a las personas dentro de las organizaciones. La única definición de una organización es la clave ORG-ID, consistente en cinco dígitos que especifican la organización, más dos dígitos que especifican una ubicación particular, más un dígito verificador. Habrá de 10.000-20.000 clientes de CBM en el nuevo sistema, contando cada ubicación como un cliente separado, con múltiples contactos potenciales en cada una. El diseñador ha decidido en este sistema de presupuesto medio, no almacenar el contacto pero sí ingresar el nombre de la persona que coloca el pedido (y el número del pedido del cliente, cuando exista) con los detalles del pedido. La estructura de datos del archivo CLIENTES será

ORG-ID
  ORG-CODIGO
  UBICACION-CODIGO
  DIGITO-VERIFICADOR
ORGANIZACION-NOMBRE
ORGANIZACION-DIRECCION
  CALLE
  CIUDAD
  ESTADO
  CODIGO-POSTAL
TELEFONO-PRINCIPAL

Observamos que esta decisión significa que no será posible recuperar los teléfonos de los CONTACTOS individuales; si necesitáramos hablarles tendríamos que hacerlo a través de su conmutador. Los registros de ORGANIZACION requieren un promedio de 150 caracteres, cada registro de CONTACTO requiere un promedio de 60 caracteres y hay un promedio de 1,8 contactos por ubicación. El diseñador ha reducido pues los requisitos del disco en línea de aproximadamente 5 millones a 3 millones de caracteres (20.000 organizaciones $\times 150 = 3$ millones de caracteres; $20.000 \times 1{,}8 \times 60$ contactos $= 2{,}16$ millones de caracteres). Un pequeño ahorro por una pequeña penalidad; éste es el contenido del diseño físico.

Muchos clientes desconocen su ORG-ID y por eso muchos pedidos no lo traerán. (Con cada despacho, el analista prevé enviar formularios de pedido pre-impresos con el ORG-ID del cliente, pero ello representará la minoría de los pedidos recibidos). ¿Cómo hace el empleado de entrada de pedidos para encontrar el ORG-ID de cada cliente de manera tal que el pedido pueda ser procesado? Podemos adoptar la modalidad que se describió recién para encontrar el ISBN y dar a cada empleado un listado de CLIENTES actualizado regularmente. Si bien ello es factible para 1.500 libros, se vuelve engorroso para 20.000 clientes. En consecuencia, el diseñador ha decidido proveer una capacidad de índice secundario dentro del archivo de CLIENTES por ORGANIZACION-NOMBRE y/o CODIGO-POSTAL, como se muestra en el DIAD de la Fig. 9.20. Esto significa que el acceso inmediato delineado en la Fig. 9.21 es factible para el empleado de entrada de pedidos.

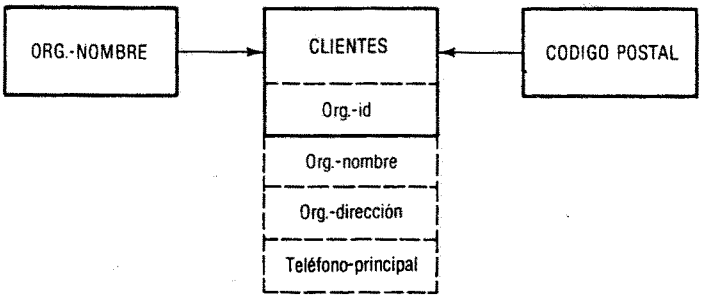


Figura 9.20 DIAD.

Entrar	El sistema representa
CODIGO-POSTAL	Todas las organizaciones cuyas direcciones tienen ese código postal
ORGANIZACION-NOMBRE	Todas las organizaciones con este nombre
ORGANIZACION-NOMBRE y CODIGO-POSTAL	Todas las organizaciones con este nombre en este código postal

Figura 9.21 Accesos inmediatos.

Típicamente, el empleado tecleará el nombre de la organización como aparece en el pedido, usando ciertas abreviaturas normales, más el código postal de la dirección del pedido y recuperará el registro de toda la organización, que le mostrará el ORG-ID. Supuesto que la dirección que se exhibe en la pantalla coincide con la dirección del pedido, se pueden ingresar los detalles del pedido. En el diagrama de flujo de datos de la Fig. 9.19, los procesos correspondientes son M2, 1.1. 1.2. y M.4.

4. *D3: CUENTAS A COBRAR, con acceso desde el proceso 3-Verificar crédito.* Aunque la parte estática de CUENTAS A COBRAR podría combinarse con el registro de CLIENTES sobre una base lógica, los auditores de CBM requieren que los registros contables se mantengan separados de los otros archivos y solo puedan tener acceso desde terminales instaladas en el Departamento de Contabilidad. La gerencia requiere que la historia de los últimos 6 meses de facturas y pagos se mantenga en línea; cada seis meses los registros correspondientes al período de antigüedad de 7 meses deben ser grabados en cinta y almacenados en un depósito. En consecuencia CUENTAS A COBRAR deberá armarse

como un archivo físico separado de CLIENTES. (Hacemos notar que un sistema administrador de base de datos con dispositivos apropiados de seguridad nos permitirá satisfacer a los auditores y combinar aun así los datos de CLIENTES con CUENTAS A COBRAR).

CUENTAS A COBRAR será un archivo algo mayor. Deberá contener 20.000 cuentas con la información maestra por cada ORG-ID. Debemos admitir 1.000 órdenes por día, las cuales crean 1,2 facturas en promedio (algunos pedidos se llenarán en dos o tres partes y cada parte deberá facturarse separadamente), de lo cual resultará un máximo de 1.000 pedidos ×120 días (6 meses) ×1,2 facturas, o 144.000 facturas en el archivo en todo momento. Algunos pagos cubrirán más de una factura; es razonable suponer unos 100.000 pagos. Considerando 100 caracteres por cada encabezamiento de registro y 30 caracteres por cada registro de pago, tendremos una capacidad de archivo de

20.000 encabezamientos	×100	=2 millones
144.000 facturas	×30	=4,3 millones
100.000 pagos	×30	=3 millones
Total de caracteres:		9,3 millones

El archivo tendrá acceso desde el subsistema de entrada de pedidos una vez por cada pedido válido, y por supuesto, desde otros subsistemas vinculados con facturas y pagos.

5. *D15: ITEM DESPACHABLES, creado por el proceso 6-Verificar inventario disponible.* Puede considerarse que el objetivo principal de todo el subsistema de entrada de pedidos es el de producir este archivo, que tiene un registro por cada ítem que sea válido, parte de un pedido de un cliente con crédito y en stock de acuerdo con los registros de computadora. Este archivo se utilizará como la entrada principal del subsistema de despacho, el cual creará notas de despacho basadas en las cantidades reales embarcadas del stock y alimentará al subsistema de facturación con información del cumplimiento de cada pedido.

La estructura de datos del archivo es:

PEDIDO-IDENTIFICACION
 PEDIDO-NUMERO
  ORG-ID
  PEDIDO-FECHA (tal como la recibió el sistema)
 CONTACTO-NOMBRE
 [CLIENTE-COMPRA-NUMERO] (ingresar solo si está disponible)
ISBN
CANTIDAD-PEDIDA
CANTIDAD-DESPACHABLE

Debemos observar que en realidad es un archivo intermedio, creado porque hemos descompuesto el sistema total en subsistemas.

6. *Otros archivos creados por el subsistema. D13:* PEDIDOS QUE REQUIEREN PREVIO PAGO contiene aquellos pedidos de clientes que no tienen crédito válido de acuerdo con la lógica de la política del proceso 3.2. Contiene detalles de los pedidos que han sido colocados y es la entrada a un subsistema que representa cada pedido, con la historia del crédito de los clientes para que el Departamento de Contabilidad determine finalmente si el crédito va a ser acordado o no.

D8: PEDIDOS PENDIENTES DE ENTREGA Y D14: PEDIDOS ESPECIALES tienen la misma estructura que ITEM DESPACHABLES pero reflejan los casos donde el ítem no tiene existencia o es uno de los 1.300 títulos que no se encuentran en el inventario.

D8: HISTORIA PEDIDO contiene un registro completo por cada pedido válido, sea despachable o no. Se emplea como base del análisis de demanda de libros y en el informe a la gerencia. Se utilizará para consultas de clientes, cuando este subsistema sea implementado.

9.4.3 Ubicación de la trasformación central en el diagrama de flujo de datos

Si revisamos el diagrama de flujo de datos de la Fig. 9.19 a la luz de los archivos físicos, resulta difícil a simple vista saber cómo corresponde el mismo al modelo simple de entrada-trasformación-salida descrito en la Sec. 9.2.

El subsistema manual descompone cada pedido en los ítem de libros componentes para ingresarlos y validarlos contra el archivo de LIBROS. El proceso 1.5 combina nuevamente los ítem en un pedido válido (con los precios agregados a cada ítem) de manera que la cantidad total del pedido se pueda calcular y utilizar para decidir si puede acordarse al cliente este crédito adicional. (Cuando el diseño se extienda a la atención de los pagos previos, necesitaremos el importe total de cada pedido para verificar que se ha enviado el importe correcto con el pedido). Una vez que se ha establecido el crédito, el pedido debe descomponerse nuevamente en ítem por el proceso 6.2 para verificarlo contra el inventario.

¿Dónde se hace visible por primera vez como salida la corriente de salida principal de ITEM DESPACHABLES?: en el flujo de datos de salida "Item despachables" del proceso 6.3. A partir de este razonamiento podemos ver que el proceso 6.3 representa la función central del sistema y los puntos de mayor abstracción de entrada y salida están a ambos lados del mismo, como se muestra en la Fig. 9.22. Esto puede parecer algo raro ya que implica que la mayoría del sistema está en el tramo de entrada y muy poco en el tramo de salida. ¿Con todo, no es lo que uno esperaba de un sistema de entrada de pedido? Ahora sabemos que podemos derivar una estructura de alto nivel de la jerarquía modular estableciendo un módulo que llama por un ítem de pedido válido, lo compara con el inventario y graba registros apropiadamente. Además, vemos que tenemos en el tope de la jerarquía un centro de transacción, antes que un centro de trasformación. El flujo de datos se divide en tres trayectorias diferentes que dependen de la situación del inventario. La Fig. 9.23 muestra dicha estructura de alto nivel.

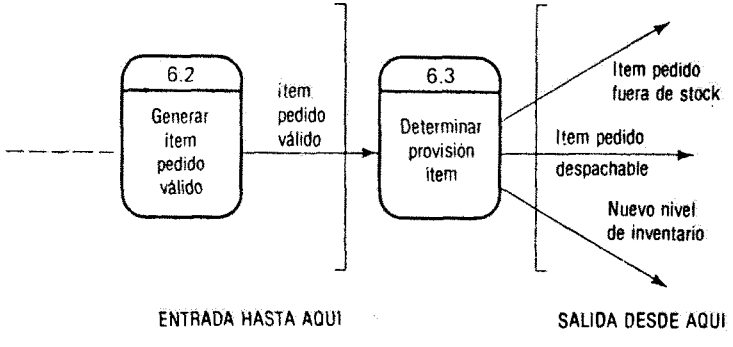


Figura 9.22 Proceso central del subsistema de entrada del pedido.

9.4.4 Perfeccionamiento del diseño desde arriba hacia abajo

El módulo DETERMINAR PROVISION ITEM juega el papel de *analizador*, determinando qué tipo de transacción está siendo atendida. La función de despachar está contenida en el módulo principal PROCESAR PEDIDOS, como se ve en el símbolo de decisión que llama a módulos que atienden el caso donde el ítem es, o bien despachable, o bien sin existencia, o ambos (en caso en que no haya suficiente cantidad de ítem para completar el pedido). La flecha curva que atraviesa a todas las flechas de invocación, indica que existe un lazo dentro de PROCESAR PEDIDOS; el cual invoca a todos los módulos inferiores cada

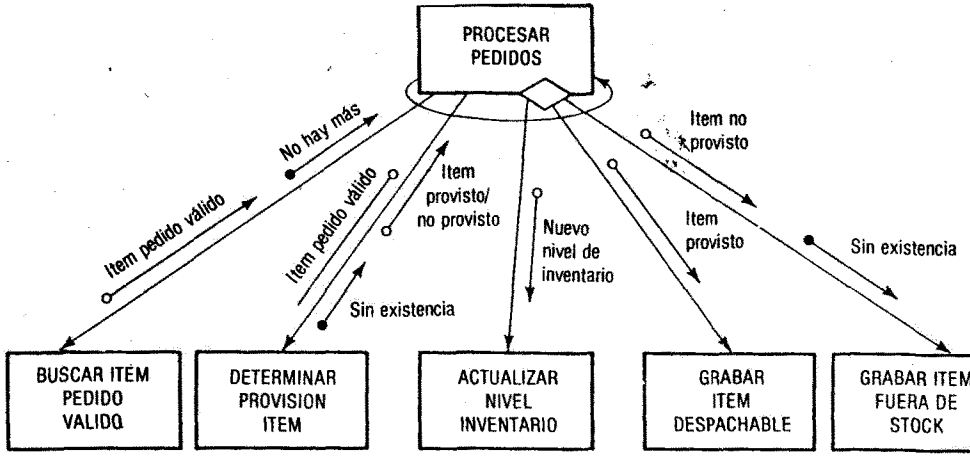


Figura 9.23 Estructura del nivel superior.

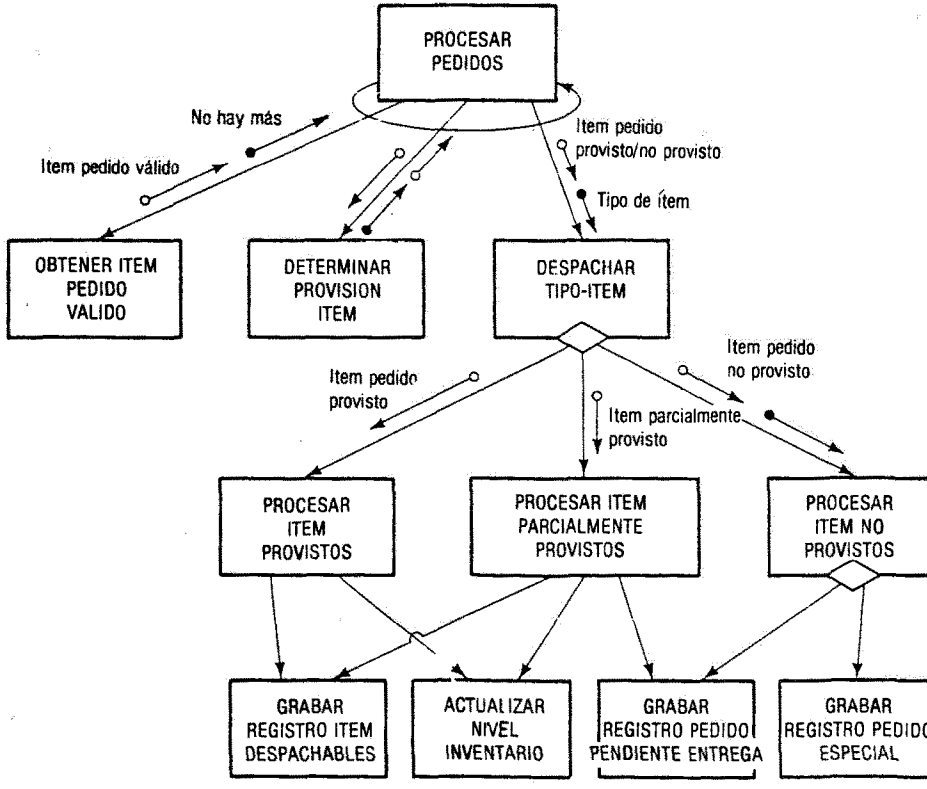


Figura 9.24 Perfeccionamiento o refinación del diseño.

vez que un ítem es procesado. ACTUALIZAR INVENTARIO no será llamado si el ítem está fuera de stock o si el inventario ya está en cero (sin existencia). El próximo perfeccionamiento del borrador de este diseño, como se ve en la Fig. 9.24, especifica las funciones del tramo de salida en mayor detalle. Creamos un módulo de despacho que invoca un módulo separado para tratar cada una de las tres circunstancias, ya sea que un ítem de pedido pueda cumplirse completamente (la situación más frecuente), o que pueda completarse parcialmente, o que no pueda cumplirse de ninguna manera. El despachador sigue siendo muy simple —realmente trivial— ya que solo tiene que llamar al módulo apropiado; la lógica para determinar las distintas acciones está contenida en esos módulos que son llamados. Realmente el despachador es tan simple que aun mostrándolo como un módulo separado en el diagrama de estructura podemos pensar que será implementado como parte de la codificación de PROCESAR PEDIDOS. El término que se emplea para esto es *inclusión lexicográfica* de un módulo en otro; existen dos módulos lógicos dentro de un módulo físico. Esto se muestra en el diagrama de estructura mediante un triángulo achatado en la parte superior del módulo, como se ve en la Fig. 9.25.

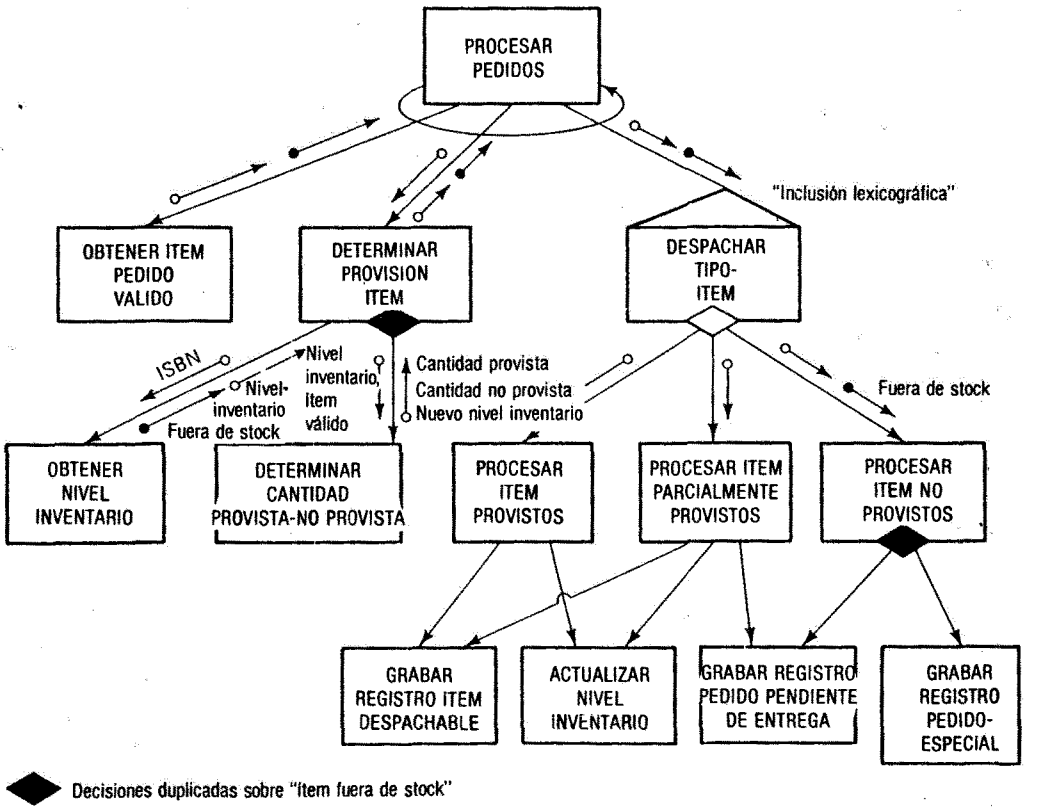


Figura 9.25 Aparición del problema de alcance de control/alcance de efecto.

En la Fig. 9.25 el analizador ha sido fracturado adicionalmente (o *factoreado*) en un módulo que obtiene el nivel de inventario para el ítem que está siendo procesado, y un segundo módulo DETERMINAR CANTIDAD PROVISTA/NO PROVISTA que es llamado si el ítem está ordinariamente mantenido en stock.

Ahora percibimos que tenemos en nuestras manos un conflicto de alcance de control/alcance de efecto. Una decisión es tomada en DETERMINAR PROVISION ITEM para saber si el ítem está o no incluido en el inventario (basado en BUSCAR NIVEL INVENTARIO y considerando que no hay registro de inventario). Pero esta misma decisión se hace nuevamente en PROCESAR ITEM NO PROVISTOS para decidir si llama GRABAR REGISTRO PEDIDO PENDIENTE DE ENTREGA o GRABAR REGISTRO PEDIDO ESPECIAL.

¿Cómo podemos resolver este conflicto y suprimir la toma de decisión duplicada? Podemos procesar los pedidos especiales como parte de DETERMINAR PROVISION ITEM, pero ello confundiría la función del módulo creando un módulo cohesivo de procedimiento. DETERMINAR PROVISION ITEM Y CUANDO EL ITEM ES UN PEDIDO ESPECIAL, CREAR EL PEDIDO ESPECIAL. Una mejor aproximación es hacer que el módulo central principal maneje la decisión solo una vez, llamando un módulo que atienda solo pedidos especiales. La solución se muestra en la Fig. 9.26 donde se ha incorporado la función analizadora y la función DETERMINAR CANTIDAD PROVISTA se ha desplazado a PROCESAR PEDIDOS, ya que podemos ver que ambas son triviales.

¿Hemos hecho el módulo de control principal muy complicado? Invocamos otros seis módulos —llamados *su espectro de control*— de manera que debemos ser cuidadosos en no introducir demasiada complejidad. Vamos a expresar la lógica que necesitamos en PROCESAR PEDIDOS mediante el empleo de pseudocódigo. La Fig. 9.27 muestra este resultado.

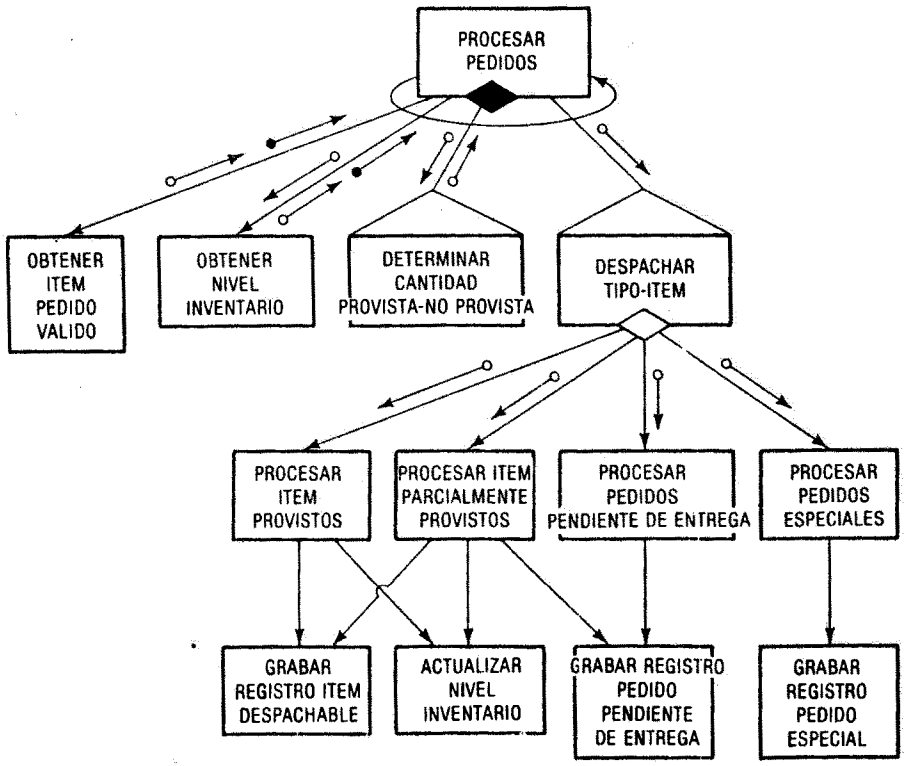


Figura 9.26 Supresión de toma de decisiones duplicadas.

```
HACER MIENTRAS MAS ENTRADA

   OBTENER ITEM PEDIDO VALIDO
   OBTENER NIVEL INVENTARIO
   SI          NO REGISTRO INVENTARIO
     LUEGO     HACER RUTINA-PEDIDO-ESPECIAL
   SI NO       (INCLUIDO EN INVENTARIO)
     ENTONCES SI          NIVEL-INVENTARIO ES CERO
                LUEGO     HACER RUTINA-PEDIDO-PENDIENTE-ENTREGA
              SI NO       (EXISTE STOCK)
                ENTONCES RESTAR PEDIDO-CANTIDAD DE NIVEL
                           INVENTARIO DANDO NUEVO-NIVEL-
                           INVENTARIO
                         SI          NUEVO-NIVEL-INVENTARIO
                                       MIE CERO
                           LUEGO     HACER RUTINA-PEDIDO-REPOSICION
                         SI NO       (PEDIDO PARCIALMENTE REPUESTO)
                           ENTONCES RESTAR NIVEL-INVENTARIO DE PEDIDO-
                                      CANTIDAD DANDO CANT-Nº-REPUESTA
                                    HACER RUTINA-PEDIDO-PARCIALMENTE-
                                      REPUESTO
FIN HACER
```

Figura 9.27 Pseudo-codificación para PROCESAR PEDIDOS.

Hasta aquí nos hemos dedicado en especial al tramo de salida del sistema. Es tiempo de volver al más extenso tramo de entrada —todo aquello que está subordinado a BUSCAR ITEM PEDIDO VALIDO— y diseñarlo. La Fig. 9.28 muestra el primer paso; BUSCAR ITEM PEDIDO VALIDO ha sido factoreado en BUSCAR PEDIDO VALIDO y AISLAR PROXIMO ITEM. Cuando el último ítem de cada pedido ha sido procesado, AISLAR PROXIMO ITEM pasará la señal "No hay más ítem" a BUSCAR PEDIDO VALIDO, después de lo cual éste llamará al próximo pedido para luego llamar nuevamente a AISLAR PROXIMO ITEM.

¿Qué está pasando en BUSCAR PEDIDO VALIDO? Podemos ver en el diagrama de flujo de datos que están involucradas varias funciones; ellas estarán bajo el control de BUSCAR PEDIDO VALIDO, que es un subconjunto de la estructura, como se observa en la Fig. 9.29. Vemos que CREAR HISTORIA-PEDIDO es una de esas funciones, una salida subsidiaria que justamente surge de la corriente de datos de entrada en este punto. Estrictamente hablando, CREAR HISTORIA-PEDIDO no tiene nada que hacer con BUSCAR PEDIDO VALIDO, o con BUSCAR ITEM PEDIDO VALIDO en ese aspecto. Es un ejemplo de *efecto lateral*, algo que se requiere al módulo, "aparte" de su propia función. Un programador de mantenimiento observando la codificación de BUSCAR ITEM PEDIDO VALIDO no podrá saber por el nombre del módulo que la historia del pedido fue creada como parte de una de sus sub-funciones. ¿Podemos diseñar de nuevo la estructura de manera que CREAR HISTORIA-PEDIDO se ubique en un lugar más natural y comprensible? Podríamos trasladar la lógica de BUSCAR ITEM PEDIDO VALIDO hacia el interior de PROCESAR PEDIDOS e invocar CREAR HISTORIA-PEDIDO desde allí, pero ello involucrará dos lazos anidados, como se ve en la Fig. 9.30; un lazo para atender pedidos y otro lazo para atender ítem. Este es un diseño indebidamente complejo; aun así, su simplificación mediante la confección de un módulo que atienda el procesamiento de ítem, como se muestra en la Fig. 9.31, implica cierta delegación del trabajo real de manejar al sistema, a fin de operar algo que, después de todo, es meramente una función de registración.

No, debemos reconocer mejor que CREAR HISTORIA-PEDIDO es una salida lateral subordinada a la función principal del sistema y aceptar que sea invocada por BUSCAR

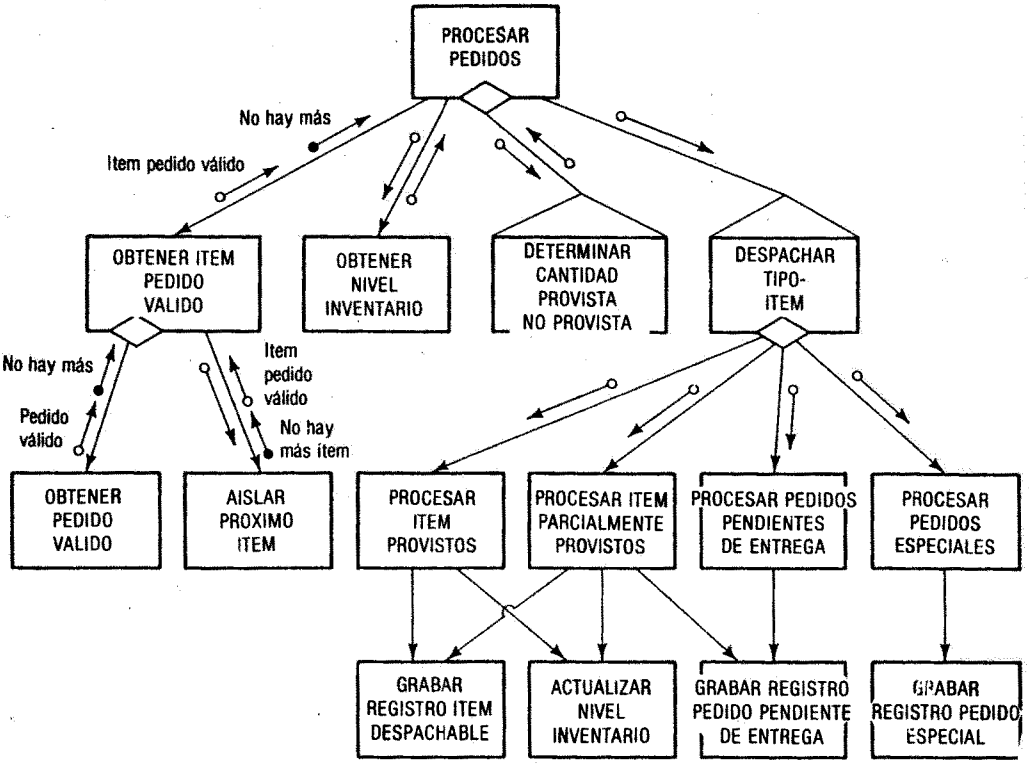


Figura 9.28 Iniciación del diseño del tramo de entrada.

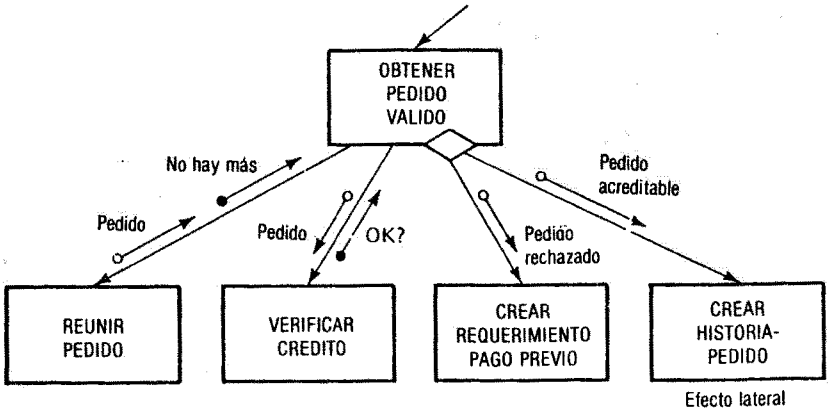


Figura 9.29 Funciones de OBTENER PEDIDO VALIDO.

PEDIDO VALIDO, haciendo así a BUSCAR PEDIDO VALIDO un módulo cohesivo de comunicación. Lo que tenemos que hacer para evitarle sorpresas indeseables al programador de mantenimiento es documentar la existencia y posición estructural de CREAR HISTO-

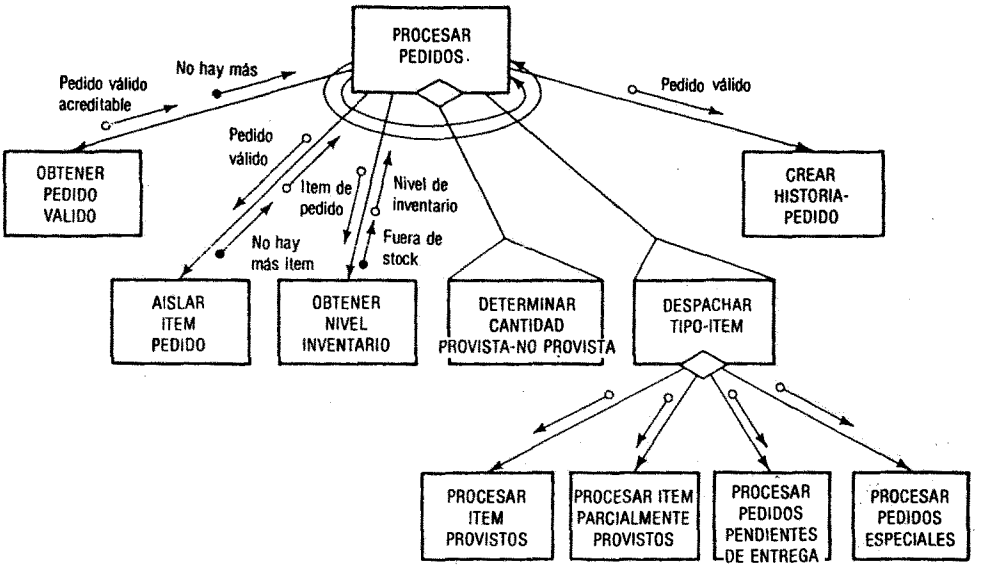


Figura 9.30 Un intento para resolver el efecto lateral causado por CREAR HISTORIA-PEDIDO.

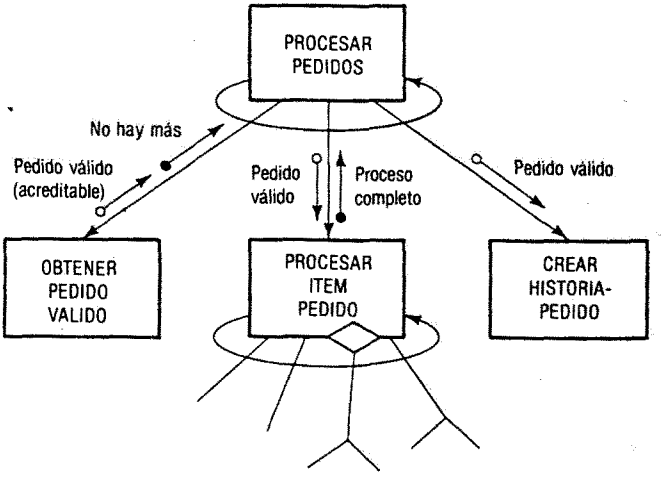


Figura 9.31 Diseño del lazo simplificado.

RIA-PEDIDO en el punto de BUSCAR ITEM PEDIDO VALIDO donde es invocado BUSCAR PEDIDO VALIDO y también en PROCESAR PEDIDOS. Además, tendríamos que cambiarle el nombre al módulo BUSCAR PEDIDO VALIDO para que represente verdaderamente su función.

Ahora estamos en condición de colocar los módulos del nivel inferior del tramo de entrada y llegar al diseño completo del subsistema tal como se muestra en la Fig. 9.32.

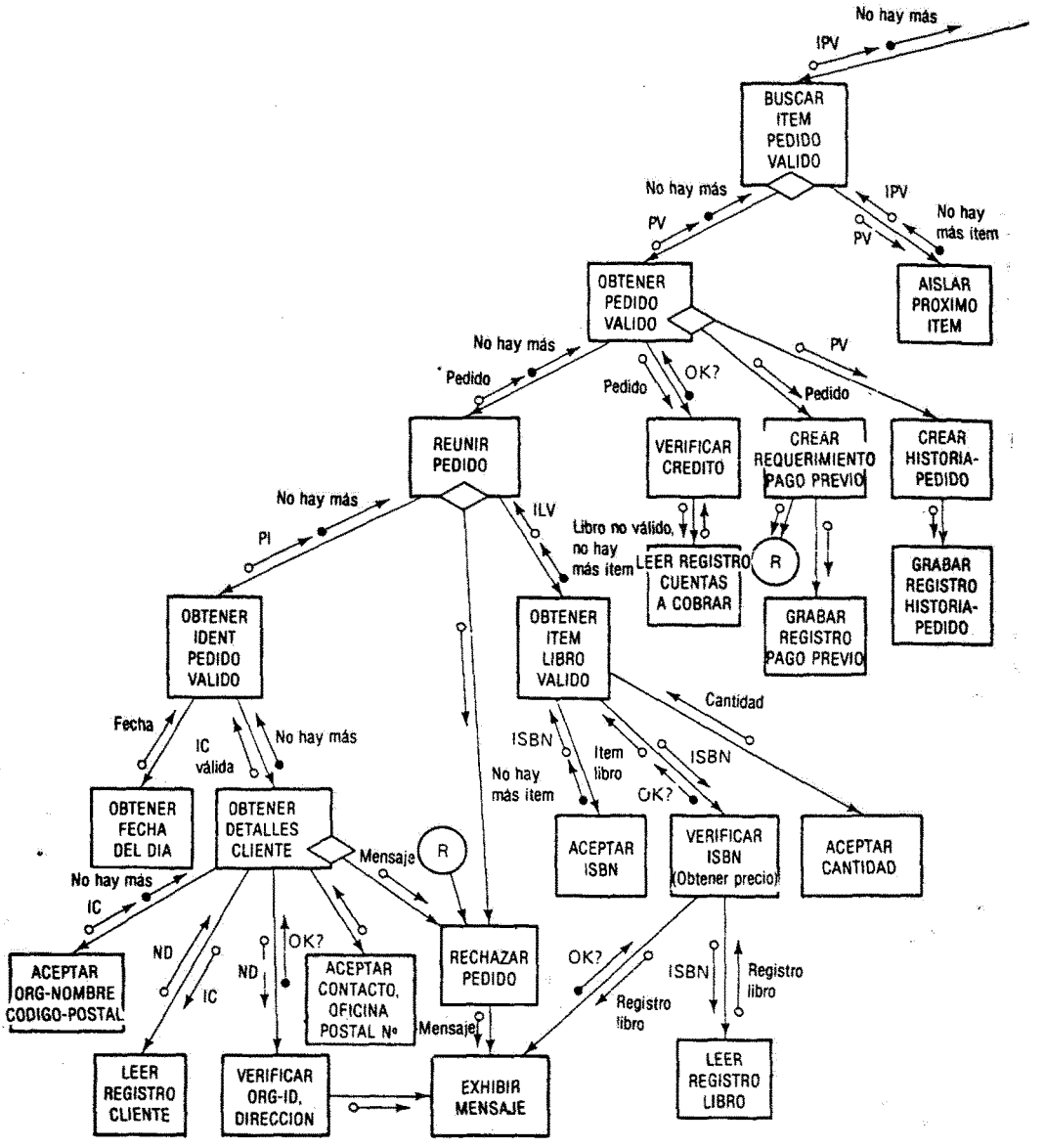
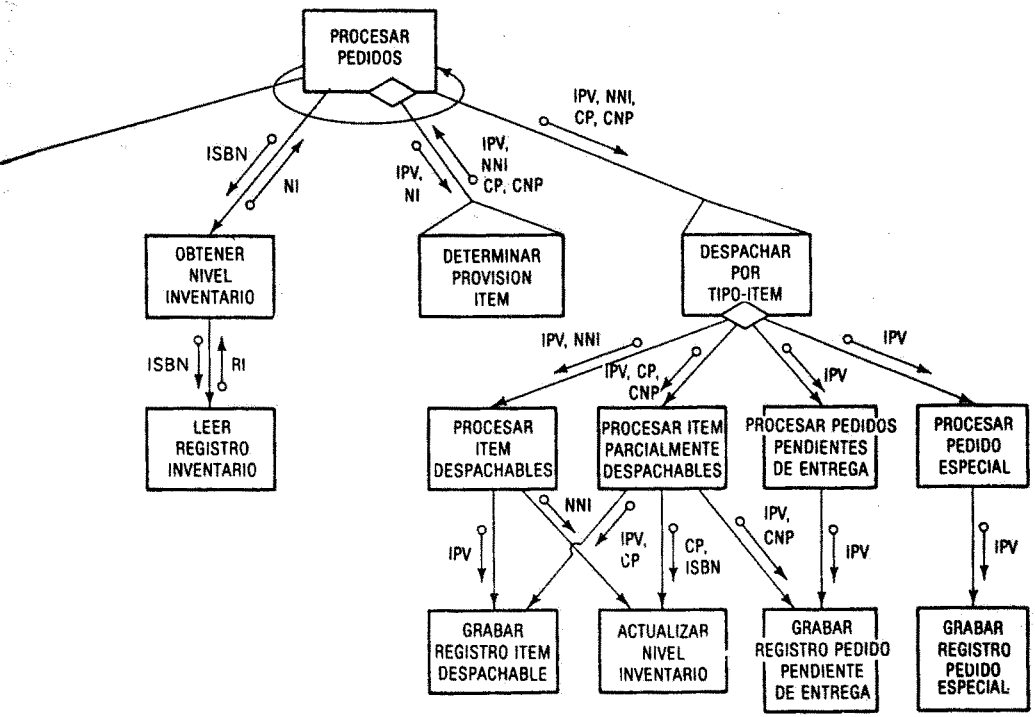


Figura 9.32 Diseño completo del subsistema.



Abrev.	Nombre	Descripción
CNR	Cantidad no provista	Cantidad de ítem de libro de pedidos pendientes de entrega
CR	Cantidad provista	Cantidad de ítem de libro a despachar
IC	Identificación cliente	Nombre organización y/o código postal
ILV	Ítem libro válido	ISBN, precio, cantidad
IPV	Ítem pedido válido	Pedido-identidad, ítem libro válido
ISBN	International Standard Book Number	Identificador de 10 dígitos del libro
ND	Nombre y dirección	Nombre organización, dirección organización, org-id
NI	Nivel inventario	Nivel actual del inventario
NNI	Nuevo nivel inventario	Nivel inventario menos cantidad del pedido
PI	Pedido-identidad	Org-id, fecha [contacto, Oficina Postal Nº]
PV	Pedido válido	Pedido-identidad, ítem libro válido +
RI	Registro inventario	ISBN, nivel inventario, cantidad del pedido, nivel de pedido

Obsérvese la utilización de la tabla resumen del diccionario de datos para reunir las abreviaturas de las diversas estructuras de datos y elementos de datos que se trasfieren en el sistema.

Este diseño, por supuesto, puede continuarse perfeccionando; por ejemplo, mediante la expansión posterior de los módulos de nivel inferior. Sin embargo, aparte de estas pocas instancias que hemos observado, está compuesto de módulos funcionalmente cohesivos los

cuales están acoplados, principalmente por el pasaje de datos, con unas pocas variables de control que "informan hacia atrás" qué ha sucedido. Para comprobar el grado de cambiabilidad del diseño, resulta interesante considerar los efectos de diversos cambios que puede solicitar el usuario. ¿Cuántos módulos necesitarán modificarse si el usuario quiere alterar las reglas de concesión de crédito? ¿Cuántos módulos tendrán que modificarse si todos los pedidos, válidos y no válidos, debieran ser grabados en el archivo HISTORIA-PEDIDO? ¿Qué pasa si el usuario decide eliminar los envíos parciales y despachar todo el pedido o nada, con excepción de una confirmación? También podemos ver que el agregado de la lógica que trata el pago previo es más claro que cuando empezamos.

Suponiendo que implementamos el sistema así definido, ¿qué podríamos hacer si el tiempo de respuesta fuese demasiado grande como para ser aceptado? Primero, podemos empaquetar los módulos lógicos del tramo de entrada en menos módulos físicos, incluyendo subordinados lexicográficos. Si esta estructura es aún muy larga para procesar, podemos considerarla descompuesta en dos subsistemas, en el punto en que se produce un pedido válido. El empleado de entrada de pedidos podrá invocar un programa que trate con los pedidos válidos. Por separado, podemos invocar la estructura de nivel superior que pueda trasferir los pedidos válidos desde el archivo de HISTORIA-PEDIDOS y procesarlos contra el inventario. Como los empleados de entrada de pedidos no tienen nada que ver con el hecho de que un pedido sea actualmente despachable o no, ésto reduciría la cantidad de accesos de archivo y la cantidad de módulos invocados. La cuestión es que el diseño puede optimizar su rendimiento de muchas maneras.

9.5 DESARROLLO DE ARRIBA HACIA ABAJO

Una vez que el diagrama de flujo de datos se ha terminado y se han dibujado los diagramas de estructura de todos los subsistemas, se los puede utilizar para planificar la implementación *de arriba hacia abajo* del sistema. Esto es, en vez de definir programas y construir subsistemas como unidades completas y comprobar que las unidades trabajan juntas después de estar listas (la aproximación de desarrollo *de abajo hacia arriba*) *podemos iniciar el desarrollo produciendo una versión "esqueleto" en bruto del sistema, que acepte algunos datos simples de entrada, los procese de alguna manera muy simplificada a través de tantos subsistemas como sea posible, y genere alguna salida muy sencilla.* Después que esta versión esqueleto esté trabajando en nuestra computadora real, podemos agregar más complejidad a cada subsistema y hacer que *éste* funcione. Veamos cómo se puede aplicar este concepto a CBM y después discutir porqué tiene ventajas sobre el procedimiento tradicional.

9.5.1 Posibles versiones de arriba hacia abajo del sistema CBM

Versión 1. Nuestra pretensión con la versión 1 será tener algo trabajando lo más rápido posible y que involucre la mayor cantidad posible de subsistemas y sus correspondientes interfaces. Una versión 1 realista debe tener las siguientes funciones:

— Aceptar pedidos de 1 a 10 clientes desde una sola CRT.
— Aceptar pedidos por solo 10 títulos, un ítem por pedido.
— Aceptar siempre que el crédito del cliente es bueno.
— Decir siempre que hay 100 copias en stock de cada título.
— Escribir una nota de despacho y factura en papel simple de computadora, sin actualizar CUENTAS A COBRAR.

¡Realizar estas funciones no requerirá mucho esfuerzo y presentará una tarea bastante simple para el empleado de entrada de pedidos! Se demostrará que podemos leer y grabar en

la pantalla CRT, grabar ítem despachables en el archivo intermedio y volver a leer el archivo como entrada del subsistema de despacho y facturación. El módulo VERIFICAR CREDITO de la Fig. 9.32 podrá ser el denominado módulo simulado, que se compone de dos líneas:

```
MOVE 'OK' TO CREDIT-STATUS  (Mover "OK" a ESTADO-CREDITO)
EXIT (SALIDA)
```

En forma similar, para todos los otros módulos bobos o simulados que no sean requeridos por la versión 1, se requerirá solo lo suficiente de sus codificaciones como para permitir compilar y ejecutar los distintos programas. Mientras que la lógica en el módulo principal PROCESAR PEDIDOS deberá ser completa, el módulo PROCESAR PROVISIONES PARCIALES podrá contener simplemente las sentencias

```
DISPLAY 'INVOCADO MODULO PROVISION'
EXIT.
```

y si este mensaje aparece mientras probamos la versión con pedidos de menos de 100 copias, sabremos que algo ha andado mal.

Versión 2. Una vez que la versión 1 ha sido desarrollada y demostró que funciona, el grupo de proyecto implementará la versión siguiente, que implica mayor cantidad de subsistemas y emplea más interfaces. La versión 2 podrá tener las siguientes funciones:

- Aceptar pedidos de los mismos 10 clientes (como la versión 1, probando la interfaz del archivo CLIENTES).
- Aceptar pedidos por cualquier cantidad de libros de un determinado editor, suponiendo que hay normalmente existencia en stock.
- Aceptar siempre que el crédito del cliente es bueno.
- Escribir en formato correcto la nota de despacho y la factura.
- Introducir y cambiar niveles de pedido del archivo INVENTARIO.
- Crear una orden de compra por 100 copias del editor de cualquier libro por debajo del nivel de pedido.

Esto involucrará la implementación de partes de los subsistemas de control de inventario y de compras, así como algo más de los subsistemas de entrada de pedidos y de despacho; y empleará mayor cantidad de archivos e interfaces de programas. Proveerá a los usuarios de ejemplos realistas de tres de los documentos importantes que salen de la empresa y lo hará en una temprana etapa del desarrollo.

Versiones subsiguientes. Una vez que la versión 2 funciona de acuerdo a lo especificado, se puede desarrollar la versión 3, seguida de la versión 4, etc., cada una de ellas con más funciones del sistema definitivo. Es conveniente planificar la implementación del sistema completo en una serie de tales versiones, cada una de las cuales puede llevar de 1-3 meses, en función de la escala del proyecto. Contar con este tipo de referencias cada 30-60 días, y poder mostrar evidencias tangibles de progreso a intervalos regulares, motiva al grupo de proyecto e infunde confianza a los usuarios.

9.5.2 ¿Por qué desarrollar de arriba hacia abajo?

¿Por qué realizamos la implementación de esta manera? La razón principal es que el desarrollo de arriba hacia abajo evita uno de los problemas más dificultosos, de mayor pérdida de tiempo y más costosos en el que la implementación "normal" (de abajo hacia

arriba) tiende a caer, esto es, el *problema de las interfaces* o "hacer que las partes se integren". El enfoque "normal" de la implementación ha sido dividir el sistema en programas, especificar los programas, escribir y probar cada programa por separado, ensamblar los programas en subsistemas y luego al final del proyecto, ensamblar los subsistemas en un sistema operante. El sistema está formado por los módulos de detalle de nivel inferior, con los niveles más altos de control, tales como el JCL (Job Control Language, lenguaje de control de trabajo), agregándose en último término; de ahí la denominación *desarrollo de abajo hacia arriba*. Especialmente, si los diferentes subsistemas son desarrollados por diferentes equipos o personas, en muchos proyectos, al integrarse los subsistemas, aparece un error en alguna parte de la interfaz entre dos subsistemas. Debido a la falta de comunicación de un cambio en las especificaciones, o debido a especificaciones vagas, la interfaz del subsistema A producido por el equipo A no se adapta a la interfaz requerida por el subsistema B producido por el equipo B. Todos los errores son desagradables, pero los errores de interfaz son los peores, por tres razones:

— Tienden a involucrar mayores cambios en las codificaciones existentes que, digamos, los errores lógicos. Si se hace mal un cálculo de descuento tendremos que corregir un programa; si el subsistema control de inventario se ha codificado utilizando el formato de registro erróneo para un pedido y en consecuencia no puede aceptar los datos del subsistema de entrada de pedidos, tendremos que codificar y compilar nuevamente y volver a probar todos los programas que utilizan este registro en uno u otro de los subsistemas.
— Tienden a ser detectados uno después del otro; hasta que el error de la primera interfaz no se haya resuelto, no se puede proceder a realizar pruebas de integración para ver si hay otros errores. Esto extiende el período de prueba en forma impredecible.
— En el desarrollo de abajo hacia arriba los errores se encuentran en su mayoría después de que se ha gastado la mayor parte del presupuesto del proyecto y de su tiempo, y cuando la fecha de entrega del sistema está próxima o ya se ha cumplido. Esto significa que siempre existe el riesgo de que justo cuando los usuarios esperan la entrega del sistema, éste se ve suspendido por un período impredecible de tiempo para arreglar una cantidad de errores de interfaz.

El gerente de proyecto que tenga mala suerte con los errores de interfaz durante el desarrollo de abajo hacia arriba, cayendo en una serie de ellos, se encuentra en la situación de tener que concurrir a ver a los usuarios y decirles "Sé que hace cuatro semanas les prometí que el sistema estaría listo en dos semanas, y también sé que hace dos semanas les prometí que el sistema estaría listo para hoy, pero nos hemos encontrado con *otro* problema...". Los usuarios no lo comprenderán, se irritarán y rápidamente perderán confianza en el gerente de proyecto que ha desperdiciado una cantidad sustancial de su dinero y parece no tener el control de la operación.

El desarrollo de arriba hacia abajo —que es el opuesto al de abajo hacia arriba— crea primero la lógica de alto nivel y todas las interfaces importantes del sistema, *antes* de que se haya escrito mucho de la codificación detallada. Las versiones 1 y 2 han sido proyectadas para ejercitar las interfaces, de manera que si aparece algún problema, será detectado y corregido relativamente pronto en el proyecto antes de que requiera mucho trabajo adicional. Esto tiene el mayor efecto en la uniformidad de la implementación posterior; los horrores de la integración son en gran medida desterrados. Hace posible mostrar a los usuarios evidencias tangibles de progreso a intervalos regulares. El desarrollo de arriba hacia abajo también es una gran ayuda para que el analista pueda *presentar modelos* del sistema a los usuarios, ya que es posible mostrar una versión temprana, como ser la entrada de pedidos vía una CRT o consultas al archivo HISTORIA-PEDIDO, y preguntar a los usuarios si eso es lo que tienen en mente. Aunque esta demostración normalmente no puede tener lugar hasta alcanzar aproximadamente entre el 65-70% del total del proyecto —cuando ya es muy tarde para hacer grandes cambios— es aún muy valiosa para probar el diálogo hombre-máquina, dando

tiempo a los usuarios para que asimilen la naturaleza del nuevo sistema y descubran errores de concepto que no se han evidenciado en el modelo lógico. Especialmente, cuando los usuarios no tienen experiencia con sistemas en línea, mostrarles una versión temprana hace a las discusiones previas de requerimientos (y al diagrama de acceso inmediato) mucho más reales. A menudo, esto estimulará a los usuarios a preguntar por mejoras del sistema en las que de otra manera no hubieran pensado. Si estas mejoras pueden ser incorporadas dentro del presupuesto del proyecto, todos ganan.

9.5.3 El papel del analista

Incluimos esta discusión de desarrollo de arriba hacia abajo en un libro de análisis, en parte, debido al valor e interés de la técnica, pero principalmente al impacto que el procedimiento tiene sobre los usuarios y a la contribución que el analista necesita para lograr un proyecto de arriba hacia abajo exitoso.

Mientras que el diseño estructurado y la codificación estructurada se ocultan a la comunidad usuaria (excepto en cuanto a que ellos producen sistemas más rápidos y mejores), el desarrollo de arriba hacia abajo hace que el proyecto completo luzca diferente. En un proyecto normal, los usuarios están incluidos en el estudio y en las especificaciones funcionales, después de lo cual desciende un silencio de muerte mientras el equipo de proyecto regresa a su cueva para diseñar, codificar y probar el sistema. Puede que los usuarios por muchos meses no escuchen nada, salvo informes de estado, sobre sus sistemas hasta el inicio de las pruebas de aceptación y del entrenamiento del usuario. En un desarrollo de arriba hacia abajo trascurre un período mucho más corto después de la especificación funcional para que los usuarios sean invitados a "venir y ver la versión 1". De ahí en más tienen lugar a intervalos (regulares) demostraciones y aceptación de versiones a lo largo de la última tercera parte de la duración del proyecto.

En esta situación de mucha mayor visibilidad, el analista tiene una cantidad de papeles que jugar:

1. *El analista deberá ayudar a proyectar las versiones de arriba hacia abajo para el plan de implementación.* Como vimos antes en esta sección, el diagrama de flujo de datos y los diagramas de estructura de los subsistemas son las herramientas principales para planificar las versiones tempranas. Una guía es imaginar la lógica más simple de todas las transacciones y resolver la cantidad de implementación mínima necesaria para procesar esta transacción con la mayor cantidad de interfaces posibles. El analista está en una buena posición para saber cuál es la transacción lógica más simple, el diseñador conocerá la cantidad de implementación necesaria para procesarla. Una vez que se han establecido las versiones tempranas, el analista podrá ver si es posible crear versiones posteriores, aunque parciales, que provean algún servicio útil a los usuarios. Por ejemplo, sería posible empezar proveyendo capacidad de consulta a algunos archivos *antes* de que se complete la versión total del sistema. Sería posible comenzar produciendo listas de correo desde el archivo de CLIENTES *antes* de que esté instalado el sistema completo de entrada de pedidos. El analista conoce qué partes son valiosas para el usuario y puede aportar ese conocimiento al plan de implementación.

2. *El analista deberá explicar el concepto de desarrollo de arriba hacia abajo a la comunidad usuaria para asegurar la colaboración armónica.* Como se puntualizó al comienzo de esta sección, el desarrollo de arriba hacia abajo es mucho más visible a los usuarios. El analista deberá esmerarse cuando explique el plan a todos los usuarios, que serán impactados por cada versión, detallando qué hará cada versión y también qué *no hará.* Si esta explicación y orientación no están bien realizadas, surgen dos peligros. El primero es que los usuarios podrán desilucionarse por la versión temprana que ven, debido a su función limitada, y perderán confianza en el sistema. El segundo es el peligro opuesto, que

queden tan impactados con la version 1 que quieran comenzar a usarla en su empresa el lunes siguiente. El analista puede explicar el procedimiento de versiones por comparación con la construcción de una casa, diciendo, "La versión 1 es comparable con la visión de la estructura de la casa solamente, sin paredes, ni techo, ni interiores. Nos permite asegurarnos que estamos construyendo la casa en el lugar correcto y que tiene el tamaño adecuado, pero ustedes no se quejarán porque penetra la lluvia ni esperarían mudarse el lunes próximo". La misma comparación es útil para explicar el impacto de los cambios que los usuarios querrán hacer. Cambiar un formato de informe es como querer las paredes de la sala pintadas de otro color; es agradable saberlo de antemano, y de cualquier manera es una de las últimas cosas que se suelen hacer en la construcción (y en el desarrollo de arriba hacia abajo), pero si el cambio se desea después de realizado el trabajo, no es conveniente. Algunos cambios son comparables a trasformar un dormitorio en otro baño —un importante trabajo estructural que implica tiempo y gastos—. Otros cambios son comparables a que el usuario diga, "Quiero la casa como está, pero ¿podría desplazarla tres metros más lejos del camino?"

3. *El analista deberá asegurarse el apoyo de la gerencia para el desarrollo de arriba hacia abajo de manera que el hardware esté disponible cuando se lo necesite.* Típicamente la versión 1 deberá estar implementada al completarse alrededor del 60-70% del proyecto, como lo mencionamos anteriormente. Luego, en un proyecto de 12 meses de duración que se inicia en enero, el modelo lógico puede aparecer en marzo/abril y los diagramas de estructura en mayo/junio, y la versión 1 podría entregarse a fines de julio y tal vez con cinco versiones más que seguirán a intervalos de cuatro semanas. Si el sistema tiene que tener funciones en línea, esto significa que por lo menos una terminal —con facilidades de comunicaciones y acceso al hardware requerido— necesita estar disponible a más tardar en junio. ¡Los gerentes que firman los cheques para el hardware, esperan, sin embargo, que no lo van a necesitar hasta noviembre! El analista debe "vender" los beneficios de esta estrategia de desarrollo a la alta gerencia y asegurarse de que no existen obstáculos para obtener el hardware y el software necesarios. Cuando el hardware se está desarrollando al mismo tiempo que el sistema (digamos una terminal de propósitos especiales) es importante poder obtener un prototipo en la etapa más temprana posible y probar las interfaces con él. A este requerimiento de tener algún hardware más temprano puede contraponerse el hecho de que la carga de prueba en la parte final del proyecto tiende a ser menor con el desarrollo de arriba hacia abajo que con el desarrollo de abajo hacia arriba, debido a que, con el desarrollo de arriba hacia abajo, evitamos la carga de rehacer tareas y pruebas, que es típica de la fase de integración de un proyecto de abajo hacia arriba y que a menudo conduce a emplear más y más tiempo de prueba para entregar el sistema.

4. *El analista deberá ejercer presión para la frecuente y completa integración de los subsistemas.* Una vez que se ha comenzado a codificar y probar de arriba hacia abajo, existe frecuentemente la tentación de "diluir" el concepto. Por ejemplo, el programador que trabaja en el subsistema entrada de pedidos puede preferir desarrollar y probar su subsistema solo, después de demostrar que la versión 1 trabaja. Puede desarrollar el subsistema de arriba hacia abajo, pero si no se reúne regularmente con el resto del personal que está desarrollando otros subsistemas y comprueba que todas las interfaces siguen funcionando, se perderán en gran medida las ventajas del desarrollo de arriba hacia abajo. En forma similar, si algunas piezas del hardware no están disponibles y la interfaz del hardware es simulada, o no es probada junto con las otras, aumenta el riesgo de problemas de integración. El analista, como representante de los usuarios, puede ejercer presión para inducir a la prueba completa de arriba hacia abajo, asegurándose de que el equipo de proyecto se tome tiempo para probar cada interfaz en el sistema final, tan temprana y frecuentemente como sea posible. El personal que es más renuente a integrar sus subsistemas con el resto del sistema es probablemente el que más lo necesita.

5. *El analista deberá ver que el subsistema personal sea desarrollado de arriba hacia abajo y operado juntamente con las versiones del sistema de aplicación.* Aunque hayamos prestado bastante atención a las interfaces software/software y software/hardware, la interfaz

hombre/sistema es igualmente crítica y tiene la misma probabilidad de presentar problemas de integración. Uno de los beneficios del desarrollo de arriba hacia abajo es que el manual del usuario y la capacitación del usuario pueden desarrollarse a lo largo de las distintas versiones del sistema. Tomando el caso de la versión hipotética del sistema CBM que hemos discutido, la documentación y la capacitación necesarias para entrar por cada cliente un libro entre diez, no son, evidentemente, extensas, y sin embargo la versión 1 provee una oportunidad de entrenar a alguien para interactuar con una terminal CRT aun cuando no haya visto una anteriormente, dentro de una situación muy simple, libre de riesgo. Las versiones posteriores permiten desarrollar gradualmente en complejidad la documentación y la capacitación, y verificarlas con las pruebas del subsistema. Un corolario de este concepto es que cada versión debe ser ejercitada con un empleado usuario en la terminal, utilizando la documentación producida por la gente responsable del subsistema personal. ¡No es bueno tener la versión ejecutada por un programador senior utilizando codificaciones y procedimientos que solo el/ella conoce!

6. *El analista deberá actuar como representante de los usuarios en la aceptación de cada versión del sistema.* El plan de implementación deberá especificar la entrega de cada versión en función de objetivos fuertemente establecidos. El día fijado para la entrega de cada versión, el analista y representantes apropiados de los usuarios deberán supervisar y ejercitar el sistema para ver si, en efecto, se han alcanzado los objetivos de la versión. Naturalmente esta ejercitación se hace más elaborada a medida que las versiones posteriores suministran más funciones, hasta que la versión final recibe la aceptación completa de la prueba y entra en producción. El analista actúa como asesor técnico del usuario en este proceso y deberá asegurarse de que todos los miembros de la comunidad usuaria en condiciones de proveer información útil sobre cada versión, tengan la oportunidad de hacerlo. Deberá coordinar la realimentación de los usuarios y presentarla al equipo de proyecto. Para hacer esto deberá reforzar los comentarios realizados en los puntos 4 y 5, asegurándose de que todas las interfaces hayan sido ejercitadas adecuadamente, incluso las del personal.

9.5.4 Resumen

El desarrollo de arriba hacia abajo es una aproximación excitante. Planificada y realizada adecuadamente puede evitar muchos de los problemas que han significado una plaga por años en el desarrollo de sistemas. En el estudio realizado por Walston y Felix sobre productividad dentro de IBM [9.13], los proyectos que no utilizaron el desarrollo de arriba hacia abajo obtuvieron una productividad promedio de 196 líneas de codificación por hombre-mes; los proyectos que lo utilizaron promediaron 321 líneas por hombre-mes, es decir, una mejora de aproximadamente el 60%. Tal vez el aspecto más importante, al menos para nosotros como analistas, es la creciente participación de los usuarios a medida que el sistema se construye, tanto en función de los comentarios valiosos y de la ayuda que pueden suministrar a medida que evoluciona el sistema, como en función de la confianza que adquieren sabiendo que se está construyendo para ellos el sistema correcto y que el mismo hace progresos tangibles.

BIBLIOGRAFIA

9.1 W. P. Stevens, G. J. Myers, y L. L. Constantine, "Structured Design", *IBM Systems Journal*, Vol. 13, Nº 2, 1974.

9.2 G.J. Myers, *Reliable Software Through Composite Design*, Petrocelli/Charter, New York, 1975.

9.3 E. Yourdon y L. L. Constantine, *Structured Design*, Prentice-Hall, Englewood Cliffs, N. J., 1979.

9.4 E. Yourdon, *Techniques of Program Structure and Design*, Prentice-Hall, Englewood Cliffs, N.J., 1975.

9.5 J. Martin, *Security, Accuracy, and Privacy in Computer Systems,* Prentice-Hall, Englewood Cliffs, N. J., 1973.
9.6 F. P. Brooks, *The Mythical Man-Month,* Addison-Wesley, Reading, Mass., 1975.
9.7 B. Boehm, "Software Engineering", *IEEE Transactions on Computers,* Vol. C-25. Dec. 1976.
9.8 H. Mills, "Software Development", *IEEE Transactions onf Software Engineering,* Vol. SE-2, Dec. 1976.
9.9 F. M. Haney, "Module Connection Analysis: A Tool for Scheduling Software Debugging Activities", *Proceedings FJCC,* Vol. 41, 1972.
9.10 R. C. Tausworthe, *Standardized Development of Computer Software,* Prentice-Hall, Englewood Cliffs, N.J., 1977.
9.11 R. J. Weiland, "Experiments in Structured COBOL", in *Structured Programming in COBOL-Future and Present,* ed. H. Stevenson. ACM Publications, New York, 1975.
9.12 P. Kraft and G. M. Weinberg, "The Professionalization of Programming", *Datamation,* Vol. 21, Oct. 1975.
9.13 C. E. Walston and C. P. Felix, "A Method of Programming Measurement and Estimation", *IBM Systems Journal,* Vol. 16, Nº 1, 1977.

Ejercicios y puntos de discusión

1. Identificar un sistema que emplee una gran cantidad de archivos intermedios. ¿Aproximadamente qué proporción del tiempo promedio de operación de ese sistema se consume en la lectura y en la grabación de dichos archivos o esperando que el operador monte los volúmenes?
2. ¿Cuál es el tiempo de búsqueda más largo y el promedio de las unidades de disco de su instalación? ¿Cuál es el tiempo de ejecución más largo y el promedio de una instrucción de máquina en su computador?
3. Si tiene acceso a un monitor de ejecución de programas, del tipo PPE o SUPERMON utilícelo en un programa con el que esté familiarizado para observar dónde el programa utiliza más tiempo. ¿Le sorprendieron los resultados?
4. ¿Qué proporción del tiempo profesional total de su instalación utilizan las actividades de mantenimiento?
5. Tome un sistema que se encuentre en producción desde hace varios años y evalúe los hombre-horas empleados en su mantenimiento. ¿Cuánto tiempo aprecia que el sistema continuará en uso antes de ser remplazado definitivamente? ¿Cuál será el trabajo total de mantenimiento durante la vida del sistema en relación con el trabajo de desarrollo original?
6. Si tiene acceso a un sistema escrito en forma de módulos, grafique la estructura del sistema y evalúe los tipos de acoplamiento y de cohesión involucrados. ¿Cómo correlaciona sus conclusiones con la posibilidad de cambio del sistema?
7. Algunos módulos de propósitos generales son lógicamente cohesivos, como ser las funciones generales de validación, los operadores generales de entrada/salida, etc. ¿Puede pensar en algunos ejemplos de módulos funcionalmente cohesivos que aún puedan considerarse de propósitos generales, en el sentido que pueden utilizarse en una cantidad de lugares de un sistema y de distintos sistemas? ¿Cuáles serían los beneficios de tener una biblioteca con módulos funcionalmente cohesivos, tipo caja negra, de propósitos generales?
8. Desde su punto de vista ¿podría reducirse la posibilidad de cambio en beneficio del rendimiento? En este caso ¿podría definir en qué circunstancias?
9. A partir del diagrama de flujo de datos de la Fig. 9.19 y del gráfico de estructura de la Fig. 9.32 amplíe el diseño para atender pagos previos, verificando que el pago previo corresponda a la cantidad correcta, y grabando un registro en el archivo CUENTAS A COBRAR que conecte el importe con la identidad del pedido.
10. A partir de suposiciones razonables, diseñe un subsistema de compras para CBM.
11. En el gráfico de estructura de la Fig. 9.32 marque los módulos que deberían haberse implementado para entregar la hipotética versión 1 que se especifica en la Sección 9.5.1. Clasifique cada módulo como F (requiere implementación total), P (requiere implementación parcial) o S (para ser implementado solamente como una simulación).
12. Realizar un plan de implementación de arriba hacia abajo para un sistema entregado recientemente, con el que se encuentre familiarizado. ¿Cuáles serían los beneficios probables que se habrían obtenido de haberse seguido este plan?

10

LA IMPLEMENTACION DEL ANALISIS ESTRUCTURADO DE SISTEMAS EN SU EMPRESA

En primer lugar vamos a considerar en este capítulo los pasos que deben tomarse en una organización de desarrollo de sistemas para implementar las herramientas y técnicas discutidas en este libro, y luego veremos los beneficios que razonablemente se pueden esperar juntamente con los problemas que experimenta el personal.

10.1 LOS PASOS EN LA IMPLEMENTACIÓN DEL ANALISIS ESTRUCTURADO DE SISTEMAS

10.1.1 Revisión de las reglas fundamentales para la administración de proyectos

La cantidad de trabajo abarcada en esta área será muy variable, ya que depende del uso o no de una metodología formal de desarrollo de sistemas y en el caso afirmativo de lo que prescriba esta metodología para la fase de análisis.

Como respuesta a los problemas que presenta el desarrollo de sistemas, algunas empresas han adoptado conjuntos formales de procedimientos para el desarrollo de sistemas, ya sea, elaborándolos por sí mismas o adoptando un "paquete" metodológico desarrollado por firmas consultoras. Cada metodología especifica —como lo indicamos en el Capítulo 8— la secuencia de actividades que deben seguirse al desarrollar el sistema, los productos a ser desarrollados en cada etapa y los controles gerenciales a aplicar. Típicamente, deberán especificar la conducción de un estudio de factibilidad, el contenido del informe del estudio de factibilidad y el equipo gerencial que deberá revisar el estudio de factibilidad y autorizar el trabajo siguiente. A continuación del estudio de factibilidad deberá especificarse una fase de diseño general y una fase de diseño detallado, continuando con la codificación, las pruebas unitarias, las pruebas de los subsistemas y la prueba del sistema.

Si ya se tiene este tipo de metodología, será necesario revisarla cuidadosamente para detectar si sus prescripciones o prohibiciones entran en conflicto con el uso de las técnicas estructuradas de sistemas. Las siguientes preguntas se encuentran entre aquellas que necesitamos considerar:

1. ¿La presente metodología prescribe o estimula el diseño físico prematuro? Si es así, ¿cómo puede modificarse la metodología para permitir la construcción del modelo lógico antes que el diseño físico?
2. ¿La metodología estimula la *sobre-documentación*, por ejemplo, escribir exhaustivamente sobre los detalles narrativos? ¿Cómo puede modificarse la metodología para

permitir que las herramientas gráficas del análisis estructurado de sistemas tomen el lugar —donde sea posible— de las narraciones? Aunque el uso del diagrama de flujo de datos, el diccionario de datos y las demás herramientas no eliminan la necesidad de *todas* las narraciones en la especificación, puede reducir considerablemente el volumen de las descripciones narrativas necesarias, supuesto que su uso sea aceptable en el contexto de la metodología.

3. ¿La presente metodología permite el desarrollo de arriba hacia abajo? Aunque específicamente no está relacionado con la fase de análisis, este hecho tiene su impacto sobre el analista —como se indicó en el Sec. 9.5—. Algunas metodologías requieren que se complete todo el diseño detallado antes de que pueda iniciarse cualquier codificación y solo supervisan la entrega de una versión del sistema (la final) empleando el completamiento de las pruebas unitarias y las pruebas de los subsistemas como puntos de control gerencial. Por supuesto, esto hace algo dificultoso el empleo del desarrollo de arriba hacia abajo, ya que el procedimiento de arriba hacia abajo permite codificar los módulos de alto nivel en un sistema a encarar, antes de que se complete el diseño detallado de los módulos de bajo nivel, y se puedan entregar series de versiones de trabajo más bien que fases de pruebas completas.

El último punto da lugar a un aspecto más fundamental y sutil, el del enfoque en *línea recta* como opuesto al enfoque en *espiral.* En el pasado hemos tendido a suponer que el desarrollo de un proyecto bien administrado va en "la línea recta" desde el estudio de factibilidad a través del análisis, del diseño, la prueba, aceptación y la operación. La Fig. 10.1 muestra este esquema

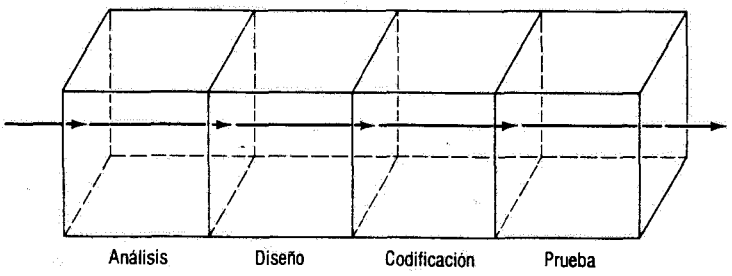


Figura 10.1 El "ideal" del progreso del proyecto.

A pesar de lo cristalino y manejable que puede ser el enfoque en línea recta, no parece corresponder a la realidad del desarrollo de sistemas. Aun un proyecto bien administrado, conducido por personal competente, necesita manejarse iterativamente, comenzando por algún análisis, luego un pequeño diseño, luego retroceder para un análisis más detallado para seguir con algo más de diseño, luego —tal vez— la codificación de la versión 1, luego seguir con más diseño, etc. La trayectoria de este tipo de proyecto puede representarse con una espiral, como se muestra en la Fig. 10.2.

El concepto de espiral se introdujo en nuestra discusión de una metodología estructurada en el Capítulo 8. Allí describimos la construcción de un diagrama de flujo de datos global antes de la construcción de flujos de datos detallados y la construcción de un diseño tentativo seguido de perfeccionamientos o refinaciones sucesivos. Este enfoque —desde nuestro punto de vista— refleja la realidad de los dificultosos problemas que afrontamos en el desarrollo de sistemas; así como desarrollo de arriba hacia abajo, hacemos diseño de arriba hacia abajo y análisis de arriba hacia abajo. En cada caso y en cada nivel construimos un esqueleto, primero lógico y después físico, vemos cómo funciona dicho esqueleto, y luego retrocedemos para armarlo poniendo la carne sobre los huesos.

¿Cómo podemos ejercer el control gerencial sobre una actividad en espiral? Las metodologías convencionales identifican típicamente los hitos de los proyectos como

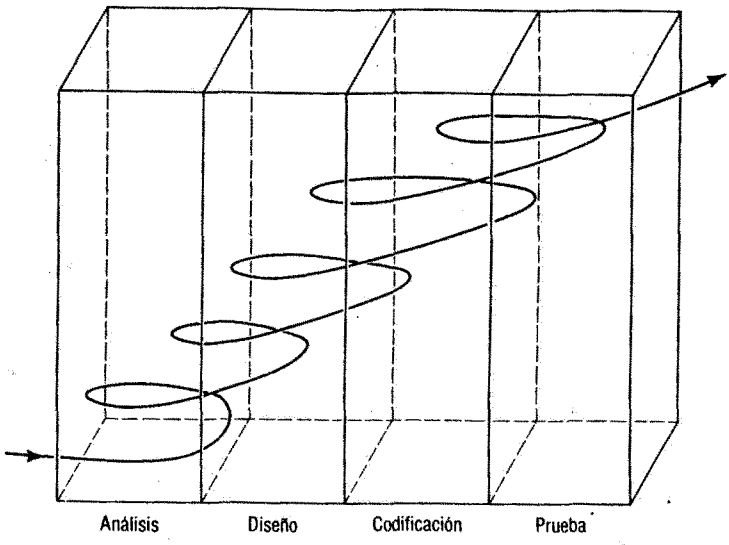


Figura 10.2 La realidad de los proyectos en espiral.

"análisis terminado" o "diseño completo". Los informes de estado intermedios requieren que el analista o el gerente de proyecto estimen el grado en que se ha completado la actividad, tal como "completado el 40% del análisis" o "terminado el 90% de la codificación". Este criterio es poco significativo, especialmente cuando el proyecto está marchando en espiral. El control gerencial no puede seguir basado en *actividades* y deberá basarse en *entregas;* en lugar de decir "¿Hasta dónde llegó usted en el análisis?", los gerentes dicen, "Muéstreme la última versión del diagrama de flujo de datos" o "Muéstreme el diagrama de estructura". Aunque discutimos la metodología estructurada en el Capítulo 8 en función de una serie de fases, solamente lo hicimos por conveniencia; lo que se está construyendo realmente en un proyecto estructurado es un conjunto de entregas de refinación creciente, que llegan a la entrega de cada una de las versiones de arriba hacia abajo del sistema. Aquí, por supuesto, las entregas son mucho más definidas que en el proyecto tradicional y el control gerencial es correlativamente más estrecho; si el plan de implementación especifica que la versión 3 será entregada el 31 de diciembre, y consiste en el procesamiento de dos transacciones que producen 7 informes, y si a las 5 de la tarde de ese día el jefe del proyecto no puede presentar esa facilidad a su gerente, el proyecto estará atrasado. No es cuestión de que aparezca "90% completo"; la versión se entrega o no se entrega. El lapso que lleve entregar la versión —después de la fecha especificada— es una medida exacta del atraso del proyecto y del tiempo que deberá desplazarse la fecha de entrega final.

Por lo tanto, las reglas fundamentales del control gerencial de un proyecto necesitan modificarse en el entorno estructurado para poder poner énfasis en la entrega de productos —diagramas de flujo de datos, diagramas de estructura, versiones de las funciones— más bien que en el grado en que completan las actividades.

10.1.2 Establecer normas y procedimientos para el uso del diccionario de datos y de otro software

Deberá decidirse entre desarrollar o adquirir el recurso de un diccionario de datos automatizado, si es que ya no existe uno. Si las técnicas de diccionario de datos son manuales

deberán confeccionarse los formularios y procedimientos de control. Si se dispone de capacidad de edición de textos, deberá adaptársela para proveer una cierta medida de la capacidad de diccionario de datos, como se describió en el Capítulo 4. Los empleados que utilicen el editor de textos necesitarán estar capacitados en las convenciones de los diccionarios de datos.

Es probable que se pueda disponer de un software más extenso para el soporte del análisis; será muy conveniente tener un trazador de diagrama de flujo de datos para el mantenimiento y fácil actualización de los diagramas, y software, para la lectura y verificación de la lógica de procesos en formato de lenguaje estructurado. Un esfuerzo de pioneros es, en este sentido, el proyecto ISDOS de la Universidad de Michigan [10.1], que ha desarrollado un software que permite a los analistas expresar flujos de datos, estructuras de datos y lógica de procesos en un lenguaje casi corriente, denominado Problem Statement Language (Lenguaje de Declaraciones de Problemas) (PSL). Las sentencias PSL son procesadas por el Problem Statement Analyser (Analizador) (PSA), el cual verifica la corrección y consistencia de las sentencias PSL y construye una base de datos de información sobre el proyecto, a partir del cual se pueden producir informes y diagramas.

10.1.3 Capacitación de los analistas en el uso de herramientas y técnicas

Hemos tratado de exponer en este libro las herramientas y técnicas del análisis estructurado de sistemas, del modo más simple, si bien realista, posible. El empleo fluido de las herramientas lleva estudio y práctica. Mientras que las reglas y convenciones pueden ser aprendidas con bastante facilidad, digamos, con una semana de práctica, el esfuerzo mayor significa comenzar a pensar en un nivel lógico, más bien que en términos de implementación física. Hemos observado un fenómeno paralelo en nuestros seminarios de diseño estructurado; el aspecto más difícil es ver al sistema como una jerarquía en lugar de verlo como un diagrama secuencial que muestre la secuencia de los sucesos. Tanto en el análisis como en el diseño estamos pidiendo a las personas que piensen en los problemas con un mayor nivel de abstracción que antes, y esto lleva tiempo y perseverancia.

Tanto como fluidez en el uso de las herramientas lógicas, los analistas necesitan familiarización con el nuevo soporte del software o con los procedimientos manuales del diccionario de datos.

Si se deben modificar las reglas fundamentales de los proyectos, los analistas deberán explicarlo a la comunidad usuaria, para lo cual aquellos deben ser informados sobre la nueva metodología y pensar en las implicaciones que ésta traerá a los usuarios.

Si se va a utilizar el desarrollo de arriba hacia abajo, los analistas deberán estar informados a fondo sobre el concepto y el plan de implementación de cada proyecto con el que estén relacionados, ya que les corresponde una participación significativa en la mayor intervención del usuario,que implica el desarrollo de arriba hacia abajo. Así como nosotros hemos llegado en este libro a cierto detalle del diseño estucturado, cada analista deberá comprender los principios del mismo y ser capaz de criticar un diseño a la luz de estos principios, aun cuando no sea él quien hizo el diseño. Esto puede lograrse estudiando este libro y las referencias sobre diseño estructurado o con una práctica formal.

10.1.4 Orientar a los usuarios en los nuevos procedimientos

Dado que las nuevas técnicas y enfoques mejoran la comunicación con los usuarios y los compromete más en la administración del proyecto, las mismas son, en general, bienvenidas por la comunidad usuaria. Al mismo tiempo, las nuevas ideas representan un cambio en las reglas de "cómo puede ser desarrollado algo alrededor de esto", y como tal generan desconfianza, a menos que se expliquen claramente sus implicaciones y beneficios.

El informe escrito a los usuarios, comúnmente hecho al comenzar cada proyecto *estructurado,* deberá cubrir los siguientes puntos:

— La notación de los diagramas de flujo de datos (y posiblemente el árbol de decisión).
— El concepto de presentar un "menú" de sistemas alternativos para la consideración de los usuarios a cargo de la decisión (si esto es de aplicación al proyecto en cuestión).
— El concepto de desarrollo de arriba hacia abajo.
— Una explicación de la participación necesaria por parte de los usuarios.
— Una reafirmación de que las nuevas técnicas *remplazan* al enfoque existente, en lugar de imponer aún un mayor esfuerzo, y de que el proyecto generará menos, y no más,papel escrito para la revisión de los usuarios.

La mayoría de las herramientas involucradas —el diccionario de datos, el diagrama de datos de acceso inmediato, el lenguaje estructurado— deberán dejarse de lado hasta que cada usuario tenga que revisar un producto que las utilice. El analista deberá estar enterado de quién ha recibido información sobre cada herramienta, y estará preparado para explicar la que va a utilizar antes de mostrársela al usuario por primera vez.

De igual manera, el concepto de desarrollo de arriba hacia abajo deberá ser explicado nuevamente cuando el analista presente el plan de implementación a los usuarios.

¿Deberá capacitarse a los usuarios para que confeccionen los diagramas de flujo de datos y escriban luego sus políticas en lenguaje estructurado? Nos inclinamos a favor de ello, supuesto que cada individuo usuario desee hacerlo. En otras palabras, no podemos requerir a los usuarios que expresen sus necesidades en un modelo lógico, pero podemos *inducirlos* a hacerlo cuando están dispuestos a dedicar el tiempo y el esfuerzo necesarios. Hemos tenido algunos usuarios no técnicos asistiendo a nuestros seminarios de análisis estructurado. Aunque ellos han encontrado que los detalles técnicos de los sistemas físicos estaban más allá de sus conocimientos han encontrado, sin excepción, en el modelo lógico un concepto fácil de entender y una manera valiosa para poder pensar sobre los sistemas que necesitan. El diagrama de flujo de datos y el árbol de decisión aparecen como las técnicas más fáciles de adquirir por parte del personal no técnico. ¡Paradójicamente, podría ser por su total ignorancia de los detalles técnicos que se les hace más fácil comenzar pensando a nivel lógico!

Cuando se asignan específicamente ejecutivos como enlaces en representación de los usuarios, es deseable que éstos sean capacitados en el empleo de todas las herramientas del análisis estructurado de sistemas. Ello les dará habilidad para pensar con más precisión sobre sus negocios y sus requerimientos, para comunicar sus ideas al analista y a los diseñadores en forma normalizada y para ser un crítico bien informado de los modelos lógicos producidos por el personal de procesamiento de datos.

10.2 BENEFICIOS Y PROBLEMAS

Con la codificación estructurada o el desarrollo de arriba hacia abajo, es posible cuantificar algunos de los beneficios que resultarán: mejora de la productividad de líneas de codificación por día, uso más controlable de tiempo de prueba, etc.

Con el diseño estructurado, los beneficios son exactamente tan reales, pero más difíciles de cuantificar. Podemos pedirle a un grupo de programadores de mantenimiento que compare la cambiabilidad de un sistema que utiliza diseño estructurado con la de otro que no lo haga; teóricamente podemos medir el costo de mantenimiento de un grupo de tales sistemas y compararlo con el de un grupo de sistemas no estructurados. Un estudio inédito sugiere que un sistema que utilice diseño estructurado es unas *siete* veces más fácil y barato de modificar que los diseños tradicionales [10.2]. Otros estudios tienden a confirmar este dramático resultado pero son anecdóticos; Bill Inmon [10.3] comenta que en un sistema de diseño estructurado, "Los mayores cambios requerían cuatro días y los cambios que le siguen en magnitud, menos de un día".

Con el análisis estructurado de sistemas los beneficios son aún más difíciles de medir. ¡Realmente, en algunos casos, si el trabajo de análisis está perfectamente hecho, el único resultado será la ausencia de problemas! Tenemos una cantidad de comentarios subjetivos y anecdóticos que parecen ser fielmente representativos de la experiencia del personal con las herramientas de análisis estructurado de sistemas. Ver por ejemplo [10.4]. En la siguiente sección se resumen estos beneficios.

10.2.1 Beneficios del empleo del análisis estructurado de sistemas

1. Los usuarios obtienen una idea más vívida del sistema propuesto por intermedio de los diagramas lógicos de flujo de datos que de las descripciones y cursogramas de los sistemas físicos. Como lo entienden, tienen una actitud más positiva hacia el proyecto. De esta forma se reduce notablemente la probabilidad de construir un sistema, que siendo excelente, no satisfaga las necesidades de los usuarios.

2. La presentación del sistema en términos de flujos de datos lógicos destaca los aspectos mal interpretados y conflictivos mucho antes de lo normal. El comentario que se hace es "con especificaciones narrativas, cualquiera traza mentalmente su propio diagrama de flujo de datos". Una vez que este flujo de datos mental ha sido volcado al papel y hecho público, se hacen obvias muchas diferencias entre las ideas que las personas mantenían privadamente sobre el sistema. Por ejemplo, una narración escrita puede especificar que "Deberá crearse un archivo HISTORIA-PEDIDO CONTENIENDO DETALLES DE TODOS LOS PEDIDOS PROCESADOS". De la manera como está redactada esta frase podría ser perfectamente aceptable por los usuarios. Sin embargo, cuando comiencen a revisar el diagrama de flujo de datos, verán más claro partiendo del lugar del diagrama donde se origina el flujo de datos de la historia del pedido, exactamente qué se incluirá en la HISTORIA-PEDIDO. ¿Sólo los pedidos despachados? ¿O todos los pedidos, sean o no despachables? ¿O incluye todos los pedidos, sean despachables o no, incluso aquellos rechazados por falta de crédito? Esta despiadada exposición de vaguedad a través del diagrama de flujo de datos significa que tiene lugar mucha mayor discusión sobre el flujo de datos que sobre la especificación de una típica narración. Es, sin embargo, una discusión muy productiva. Hacer cambios en el papel es muy barato comparado con hacerlos en la codificación.

3. Las interfaces entre el nuevo sistema y los sistemas administrativos y/o los sistemas automatizados, existentes, se ven claramente mediante el diagrama de flujo de datos; y la necesidad de documentar los detalles de los flujos de datos en el diccionario de datos, en una etapa temprana, fuerza a una clara definición de estas interfaces. Algunas empresas especifican que debe dibujarse un diagrama de flujo de datos no solo para el sistema bajo estudio, sino también para cada uno de los otros sistemas —administrativos o automatizados— con los cuales interactúan. Este ejercicio, aunque involucra un considerable trabajo, muestra las duplicaciones de funciones y señala las oportunidades de incluir tareas administrativas dentro del nuevo sistema.

4. El uso del modelo lógico elimina cierta cantidad de duplicación de esfuerzo que tiene lugar en los proyectos tradicionales. Típicamente, el representante de los usuarios y el analista debían, en el pasado, trabajar juntos para producir una especificación narrativa del sistema. Una vez redactada la especificación narrativa, el grupo de diseño y programación debía volver sobre ella, repitiendo una buena parte del trabajo de definición de la lógica y de los datos. Es digno de atención que las herramientas de análisis estructurado de sistemas sean igualmente valiosas para usuarios y técnicos. Una vez que los usuarios estén de acuerdo con el flujo de datos, el análisis de acceso inmediato y la lógica de la política, estos documentos podrán ser utilizados directamente como entrada del diseño físico. Esta ventaja es particularmente notable para el diseñador de la base de datos, quien anteriormente tenía que seguir la especificación narrativa para extraer los requerimientos de ítem de datos y de accesos. Ahora

se les presenta un diccionario de datos, relaciones en 3FN y un análisis de acceso inmediato.

5. El uso del diccionario de datos para contener el glosario de ítem del proyecto ahorra tiempo debido a la resolución rápida de los casos donde el personal llama a las mismas cosas con nombres diferentes o cuando un término significa distintas cosas dependiendo del contexto. Estos usos de las palabras se consideran naturales por el personal de la comunidad usuaria, ya que ellas forman parte de su vida diaria, pero son muy desconcertantes para los analistas.

En pocas palabras, los beneficios se reducen a dos:

— Mostrar claramente qué es lo que usted va a hacer, de manera que todos puedan estar seguros de que va a construir el sistema correcto.
— Resolver las alternativas y detalles con la menor pérdida de tiempo posible.

Expresado así, suena algo trivial, pero sin embargo, ¿cuánto tiempo y dinero se han perdido en los últimos 20 años porque no fuimos capaces de hacer estas dos cosas?

10.2.2 Problemas potenciales

Los beneficios de las nuevas herramientas analíticas no son gratis, por supuesto; existen ciertos costos y problemas potenciales asociados con su introducción. En parte son los problemas asociados con cualquier cambio; en parte son el resultado de la mayor formalidad y disciplina de las herramientas lógicas.

1. Se requiere orientar a los usuarios y capacitar a los analistas, tal como se discutió en la sección previa. Dado que la introducción del análisis estructurado de sistemas es percibida como "cambiar de reglas", debe explicarse con claridad a cada uno cuáles son las nuevas reglas y cómo mejoran el juego.

2. El esfuerzo, formalidad y grado de detalle requerido, especialmente en la construcción del diccionario de datos, son resistidos a menudo. Es cuestión de hacer una inversión de esfuerzo durante el análisis, en aras de una mayor armonía en el proyecto posterior, de hacer correctamente las cosas desde un comienzo de manera de no tener que rehacerlas. En parte, la resistencia proviene de los usuarios, debido a que los proyectos anteriores no han requerido definiciones tan claras de términos y significados; el equipo de proyecto había sufrido así que los usuarios continuaran con su vieja terminología chapucera. La resistencia podrá surgir a raíz del intento de una definición con excesivo detalle muy temprano; como se comentó en el Capítulo 8, se deberá tomar una decisión finamente balanceada acerca del nivel de detalle de la documentación especialmente en las funciones del sistema actual que no van a ser incluidas en el nuevo sistema. Pero debe afrontarse un hecho: hacer un buen análisis significa esfuerzo tanto para el usuario como para el analista. La compensación es que resulta un esfuerzo más productivo.

3. Podrá existir cierta inquietud por parte de los programadores, de que al recibir especificaciones detalladas de lógica en lenguaje estructurado "perderán toda la diversión de programar; convirtiéndose en meros codificadores". Estos miedos se pierden cuando los diseñadores y programadores ven que el análisis estructurado de sistemas les permite realizar una tarea mayor al recibir un modelo lógico para trabajar. Nuestra discusión sobre las soluciones de compromiso de diseño del Capítulo 9 nos aclaró cuánto trabajo resta por hacer después de concluir el modelo lógico. Creemos que los problemas surgen debido a que hasta que los programadores no tienen cierta experiencia de trabajo con modelos lógicos, no pueden apreciar la diferencia entre la lógica externa de validación del crédito —digamos— expresada en lenguaje estructurado, y la lógica interna del módulo físico. La lógica externa está dada por el analista, la lógica interna se debe diseñar completamente.

4. Por último, aparece un problema en ciertas empresas después de su primera experiencia positiva con el análisis estructurado de sistemas. "Qué vergüenza no haber tenido

estas herramientas en el proyecto XYZ; hemos trabajado en él por seis meses y no finalizamos su análisis. ¿Podemos usar las herramientas de análisis estructurado de sistemas en XYZ ahora que hemos hecho parte del camino?" La lección de esta experiencia parece ser que resulta provechoso utilizar las técnicas estructuradas para análisis, diseño y desarrollo a partir de cualquier punto de un proyecto. Aunque parezca a los usuarios del proyecto XYZ que vamos a tirar por la borda el trabajo de los últimos seis meses y volver a empezar (lo cual no es cierto), quedarán rápidamente convencidos cuando vean las mejoras en marcha. Citemos a un analista de una compañía de seguros, "Una vez que haya visto lo que estos nuevos métodos pueden hacer, no querrá continuar empleando los viejos métodos".

BIBLIOGRAFIA

10.1 D. Teichroew y E. Hershey, "PSL/PSA; A Computer-Aided Technique for Structured Documentation and Analysis", *IEEE Transactions on Software Engineering,* Vol. SE-3, enero de 1977.

10.2 Larry Constantine, articulos no publicados.

10.3 Bill Inmon, "An example of Structured Design", *Datamation,* Vol. 22, marzo de 1976.

10.4 W. James Kain, "The Practice of Structured Analysis", articulo presentado al Life Office Management Association Systems Forum, Atlanta, marzo de 1977.

GLOSARIO

Acceso inmediato. Recuperación de una parte de los datos de un almacenamiento de datos más velozmente que por medio de la lectura de todo el almacenamiento buscando esa parte del dato o clasificando el almacenamiento de datos.

Acoplamiento de contenido. Una forma rigurosa de acoplamiento, donde un módulo hace una referencia directa al contenido de otro módulo.

Acoplamiento de control. Una forma de acoplamiento mediante la cual un módulo trasfiere a otro una o más señales o interruptores, como parte de la invocación o retorno del control.

Acoplamiento externo. Una forma rigurosa de acoplamiento inter-modular, en la cual un módulo se refiere a elementos internos de otro módulo y dichos elementos han sido declarados accesibles por otros módulos.

Administrador de datos (administrador de base de datos). Una persona (o grupo) responsable del control y la integridad de un conjunto de archivos (bases de datos).

Agregado de datos. Una determinada colección de ítem de datos (elementos de datos) dentro de un registro. *Ver también* grupo.

Alcance del control (de un módulo). Todos aquellos módulos que son invocados por un módulo, y todos aquellos invocados por sus niveles inferiores, etc. El "departamento" del cual el módulo es el "jefe".

Alcance del efecto (de una decisión). Todos aquellos módulos cuya ejecución o invocación depende del resultado de la decisión.

Alias. Un nombre o símbolo que se da a una cosa y que no es su nombre propiamente dicho.

Almacenamiento de datos. Cualquier lugar de un sistema donde se almacenan los datos entre transacciones o entre ejecuciones del sistema (incluye archivos manuales y legibles por máquina, bases de datos y tablas).

Arbol de decisión. Un gráfico de ramificaciones que muestra las acciones que resultan de las diversas combinaciones de condiciones.

Archivo invertido. Aquel que provee múltiples índices de los datos; los propios datos pueden estar contenidos dentro de los índices.

Argumento. Un valor que se emplea como entrada a algún proceso, a menudo trasferido a través de una interfaz módulo/módulo.

Argumento de búsqueda. El valor(s) del atributo empleado(s) para recuperar algún dato de un almacenamiento de datos, ya sea a través de un índice o mediante una búsqueda. *Ver también* argumento.

Atributo. Un elemento de datos que contiene información sobre una entidad.

Base de datos. Una colección de datos interrelacionados almacenados juntos con redundancia controlada para servir a una o más aplicaciones; los datos están almacenados de manera que sean independientes de los programas que los usan; se emplea un procedimiento común y controlado para agregar nuevos datos, y para modificar y recuperar los existentes dentro de una base de datos. [James Martin, Ref. 7.2].

Base de datos relacional. Una base de datos construida únicamente con relaciones normalizadas.

Clave. Un elemento de datos (o grupo de elementos de datos) empleado para encontrar o identificar un registro ("tupla").

Clave candidata. Un atributo o grupo de atributos cuyos valores identifican unívocamente cada "tupla" de una relación y para la cual no puede quitarse ninguno de los atributos sin destruir la identificación unívoca.

Clave primaria. Clave que identifica unívocamente un registro ("tupla").

Cohesión de coincidencia. Utilizada para describir un módulo que no tiene una relación significativa entre sus componentes (aparte de aparecer en un mismo módulo). La cohesión más débil de un módulo.

Cohesión de comunicación. Utilizada para describir un módulo donde todos los componentes operan en la misma estructura de datos. Una cohesión buena, pero no ideal en un módulo.

Cohesión de procedimiento. Empleada para describir un módulo cuyos componentes forman dos o más bloques en un diagrama de flujo. No es tan buena como la cohesión de comunicación o la funcional.

Desarrollo de arriba hacia abajo. Una estrategia de desarrollo mediante la cual los módulos ejecutivos de control de un sistema se codifican y prueban primero para integrar una versión "esqueleto" del sistema, y después que se ha probado el funcionamiento de las interfaces del sistema, se codifican y prueban los módulos de los niveles inferiores.

DFD. *Ver* diagrama de flujo de datos.

DIAD. *Ver* diagrama de acceso inmediato de datos.

Diagrama de acceso inmediato de datos (DIAD). Una representación de los caminos de acceso inmediato a un almacenamiento de datos que muestra lo que los usuarios quieren recuperar del almacenamiento de datos sin llevar a cabo la búsqueda o la clasificación.

Diagrama de flujo de datos (DFD). Una representación de los flujos de datos a través de un sistema de cualquier tipo, que muestra las entidades externas que son fuente o destino de los datos, los procesos que trasforman los datos, y los lugares donde los datos son almacenados.

Diccionario de datos. Un almacenamiento de datos que describe la naturaleza de cada parte de los datos empleados en el sistema, que a menudo incluye las descripciones de procesos, entradas del glosario, y otros ítem.

Directorio de datos. Un almacenamiento de datos, usualmente legible por máquina, que indica *dónde* se almacena un dato en un sistema.

Diseño. El proceso (iterativo) de tomar un modelo lógico de un sistema con un conjunto de objetivos fuertemente establecidos para este sistema, y producir la especificación de un sistema físico que satisfaga dichos objetivos.

Diseño estructurado. Un conjunto de guías para producir una jerarquía de módulos lógicos que representan un sistema altamente modificable. *Ver también* diseño.

Dominio. El conjunto de todos los valores de un elemento de datos que forma parte de una relación. Es un equivalente efectivo de un campo o de un elemento de datos.

EE. *Ver* entidad externa.

Efecto lateral. La disminución de la cohesión de un módulo debido a que realiza algunas funciones menores "laterales" y que no forman parte de la función principal del módulo.

Elemento de datos (ítem de datos, campo). La unidad de datos más pequeña que tiene significado para el propósito que se trata.

Elemento de datos continuo. Aquel que puede tomar tantos valores dentro de su rango que no resulta práctico enumerarlos, por ejemplo, una suma de dinero.

Elemento de datos discreto. Aquel que toma solo un número limitado de valores, cada uno de los cuales tiene generalmente un significado. *Ver también* elemento de datos continuo.

En línea. Conectado directamente a la computadora de manera que la entrada, la salida, los accesos de datos y los cálculos pueden tener lugar sin ninguna intervención humana.

Entidad. 1. Entidad externa: fuente o destino de datos en un diagrama de flujo de datos. 2. Algo sobre lo cual se almacena información en un almacenamiento de datos, por ejemplo, clientes, empleados.

Entidad externa (EE). *Ver* entidad.

Espectro de control. El número de módulos directamente invocados por otros módulos. No debería ser ni muy alto (excepto en el caso de un módulo despachador) ni muy bajo.

Estructura de datos. Uno o más elementos de datos con una forma de relación particular, que generalmente se utilizan para describir alguna entidad.

Factorear. Una función o módulo lógico se factorea cuando se lo descompone en funciones o módulos menores componentes.

Físico. Relacionado con la forma particular en que los datos o la lógica es representada o implementada en un momento particular. A una declaración física no se le puede asignar más de una implementación del mundo real. *Ver también* lógico.

Funcional. 1. Cohesión funcional: empleada para describir un módulo donde todos sus componentes en

conjunto contribuyen a la ejecución de una sola función. 2. Dependencia funcional: un elemento de datos A es funcionalmente dependiente de otro elemento de datos B si dado el valor de B se determina el correspondiente valor de A.

Grado (de relación normalizada). El número de dominios que integran la relación (Si existieran siete dominios, la relación es de grado 7).

Gráfico de estructura. Un modelo lógico de una jerarquía modular, que muestra invocación, comunicación inter-modular (datos y control), y la ubicación de lazos y decisiones importantes. Ver Fig. 9.32.

Grupo (ítem). Una estructura de datos compuesta de un pequeño número de elementos de datos, con un nombre, referida como un conjunto. *Ver también* agregado de datos.

HIPO (proceso jerarquico de entrada salida). Una técnica gráfica similar al diagrama de estructura que muestra un modelo lógico de una jerarquía modular. Un diagrama HIPO generalizado indica la jerarquía de los módulos: los detalles de entrada, procesamiento y salida de cada módulo se indican en un diagrama de detalle separado, a razón de uno por módulo.

Indice. Un almacenamiento de datos que como parte del proceso de recuperación toma información sobre los valores de algún atributo(s) y retorna con información que permite que el/los registro(s) con esos atributos sea(n) recuperado(s) rápidamente.

Indice secundario. Un índice de un almacenamiento de datos basado en algún atributo distinto de la clave primaria.

IRACIS: Acróstico en idioma inglés que significa mayores ingresos, gastos evitables, servicio mejorado.

Lenguaje comprimido. Una herramienta para representar políticas y procedimientos con la menor ambigüedad posible., *Ver también* lenguaje estructurado.

Lenguaje estructurado. Una herramienta para representar políticas y procedimientos en lenguaje corriente en forma precisa, empleando las estructuras lógicas de la codificación estructurada. *Ver también* pseudocódigo.

Item de datos. *Ver* elemento de datos.

Léxicográfico. Hace al orden en el que se escriben las sentencias de un programa. El módulo A está *léxicográficamente incluido* en el módulo B si las sentencias de A se encuentran incluidas en las sentencias de B en el listado fuente.

Lógico. 1. No físico (de una entidad, sentencia, o gráfico): factible de ser implementado en más de una forma, y que expresa la naturaleza subyacente del sistema referido. 2. Cohesión lógica: empleado para describir un módulo que realiza un número de funciones similares aunque ligeramente diferentes - una cohesión pobre de módulos.

Módulo. 1. Módulo lógico: una función o un conjunto de funciones referidas por un nombre. 2. Módulo físico: una secuencia de sentencias contiguas de programa, limitada por un elemento vecino, y referida por un nombre.

Normalizada (relación). Una relación (archivo), sin grupos repetitivos, de manera que los valores de los elementos de datos (dominios) puedan ser representados como una tabla bidimensional.

Patológica (conexión). Una forma rigurosa de acoplamiento intermodular donde un módulo se refiere a algo que se encuentra dentro de otro módulo. *Ver también* acoplamiento de contenido.

Primera forma normal (1FN). Una relación sin grupos repetitivos (una relación normalizada) pero que no satisface las pruebas más exigentes de la segunda y tercera forma normal.

Proceso (transformación). Un conjunto de operaciones de transformación de datos, lógica o físicamente, de acuerdo con alguna lógica de proceso.

Programación estructurada (codificación). La construcción de programas empleando un pequeño número de construcciones lógicas, cada una con una entrada, una salida, en una jerarquía enlazada.

Pseudocódigo. Una herramienta para la especificación de un programa lógico en un formato legible, en forma similar al lenguaje corriente, pero sin observar las reglas sintácticas de ningún lenguaje de programación en particular.

Relación. Un archivo representado en formato normalizado como una tabla bidimensional de elementos de datos.

Segmento. Un grupo de (uno o más) elementos de datos; la unidad de datos accedida por el software IMS. *Comparar* grupo, agregado de datos.

Segunda forma normal (2FN). Una relación normalizada en la cual todos los dominios no-clave son completamente dependientes de la clave primaria.

Subsistema personal. Los flujos de datos y procesos, dentro de un sistema integral de información, que son manejados por personas: la documentación y capacitación necesarios para hacer funcionar este tipo de subsistema.

Tabla de decisión. Un gráfico tabular que presenta la lógica que relaciona diversas combinaciones de condiciones con un conjunto de acciones. Generalmente se tratan en la tabla todas las combinaciones posibles de condiciones.

Tercera forma normal (3FN). Una relación normalizada en la cual todos los dominios no-clave son completamente dependientes de la clave primaria y todos los dominios no-clave son mutuamente independientes.

TSO. Opción de tiempo compartido: un dispositivo del software de IBM que permite el ingreso y la edición de programas y de texto a traves de terminales en linea.

"Tupla". Un conjunto especifico de valores para los dominios que permiten realizar una relación. El término "relacional" para un registro. *Ver también* segmento.

Volatilidad. Una medida del régimen o tasa de cambio del contenido de un archivo, en especial en términos de agregados de nuevos registros y de eliminación de los antiguos.

Este libro se terminó de imprimir el 20 de abril de 1987
en Gráfica Yanina, República Argentina 2686, V. Alsina, Bs. As.