24 24.

LA IMPORTANCIA DE LA CONEXION Y LA COHESION EN LA ELABORACION DE SISTEMAS DE INFORMACION.

UNAM – Dirección General de Bibliotecas Tesis Digitales Restricciones de uso

INDICE

C O N T E N I D O

3. CONEXION.

3.1. Factores que influyen en el grado de conexión.
  3.1.1. Tipo de enlace entre los módulos.
    - Enlace mínimo.
    - Enlace normal.
    - Enlace patológico.
  3.1.2. Complejidad de la línea de comunicación.
  3.1.3. Flujo de información.
    - Conexión de información o de entrada y salida.
    - Conexión de control.
    - Conexión híbrida.
  3.1.4. Momento en que los enlaces intermodulares son establecidos.
  3.1.5. Conexión de contenido.
  3.1.6. Conexión de ambiente común.

3.2. Recomendaciones para la disminución del grado de conexión de los sistemas.
  3.2.1. Tipo de enlace.
  3.2.2. Complejidad de la línea de comunicación.
  3.2.3. Flujo de información.
  3.2.4. Momento en que los enlaces intermodulares son establecidos.
  3.2.5. Conexión de contenido.
  3.2.6. Conexión de ambiente común.

4. COHESION.

4.1. Grados de cohesión.
  4.1.1. Cohesión coincidental.
  4.1.2. Cohesión lógica.
  4.1.3. Cohesión temporal.
  4.1.4. Cohesión procesal.
  4.1.5. Cohesión comunicacional.
  4.1.6. Cohesión secuencial.
  4.1.7. Cohesión funcional.

4.2. Medición del nivel de cohesión.

4.3. Recomendaciones para incrementar el grado de cohesión de los módulos.

CONCLUSIONES.

BIBLOGRAFIA.

INTRODUCCION

I N T R O D U C C I O N.

- Por qué este tema.

Durante los años en los cuales se cursan las carreras de matemáticas o actuaría -cuando los estudios se orientan hacia la computación- se conocen varias metodologías para el desarrollo de sistemas; cada una de ellas propone criterios y acciones distintas para la elaboración de sistemas, desde su concepción inicial hasta el comienzo de su operación.

Algunos ejemplos de estas metodologías son : "Diseño de arriba hacia abajo", "Programación estructurada", "Diseño estructurado", "Construcción lógica de programas", "Diseño compuesto" (Tabla 0-1). En todas ellas el propósito fundamental es proporcionar una serie de pasos o etapas, que seguidas cuidadosamente conducen a la obtención de sistemas "eficientes". Donde las comillas son una forma de hacer notar que lo que se entiende por eficiencia depende fundamentalmente de los parámetros que se utilicen para medirla. Para algunas personas la eficiencia se refleja en el tiempo ocupado durante el procesamiento, mientras que para otras un sistema eficiente es

	ATRIBUTO				
METODO	GRAFICAS ESPECIALIZADAS	PROCEDIMIENTOS DEFINIDOS	COMPATIBILIDAD CON OTROS ESQUEMAS Y TECNICAS	AREA DE APLICACION	CRITERIO DE EVALUACION
DISEÑO ESTRUCTURADO	UTILIZA DIAGRAMAS ESTRUCTURALES PARA LA ARQUITECTURA DEL SISTEMA	UN MARCO ITERATIVO QUE DIRIGE EL DESARROLLO DE LA SOLUCION	UTILIZABLE CON CUALQUIER ESTRATEGIA DE DISEÑO MODULAR	SISTEMAS CUYO FLUJO DE INFORMACION PUEDE INDICARSE DE MANERA GRAFICA	UN CONJUNTO BIEN DEFINIDO DE TECNICAS HEURISTICAS DE DISEÑO
METODOLOGIA DE JACKSON	UTILIZA DIAGRAMAS EN FORMA DE ARBOLES PARA LA ESTRUCTURA DE LA INFORMACION	LINEAMIENTOS VAGAMENTE DEFINIDOS PARA RESOLVER ALGUNOS PROBLEMAS	UTILIZABLE CON OTROS METODOS DE ESTRUCTURACION DE INFORMACION	SISTEMAS ADMVOS. Y OTROS CON ESTRUCTURAS DE INFORMACION BIEN DEFINIDAS	VERIFICACION DEL BUEN COMPORTAMIENTO SEGUN SUPOSICIONES BASICAS
CONSTRUCCION LOGICA DE PROGRAMAS	UTILIZA DIAGRAMAS WARNIER PARA LA ESTRUCTURA DE LA INFORMACION	CONJUNTO BIEN DEFINIDO DE PROCEDIMIENTOS EN TODOS LOS NIVELES DE DETALLE	LA NATURALEZA DE LOS PROCEDIMIENTOS LIMITA LA COMPATIBILIDAD	SISTEMAS ADMVOS. Y OTROS CON ESTRUCTURAS DE INFORMACION BIEN DEFINIDAS	VERIFICACION DEL BUEN COMPORTAMIENTO SEGUN SUPOSICIONES BASICAS

TABLA 0-1. COMPARACION DE ALGUNAS METODOLOGIAS DE DISEÑO (PETERS).

aquel en el que la cantidad de memoria utilizada es reducida. Durante los últimos años se ha comenzado a dar importancia a la sencillez con la que son creados los sistemas, lo cual ha producido una revolución en el área de diseño de sistemas; desde luego que en este caso es requisito indispensable dejar muy claro qué es lo que se entiende por sencillez.

En mi desempeño profesional dentro del área de sistemas de cómputo, me he convencido de que el último criterio mencionado es el de mayor importancia dentro del área administrativa, debido al hecho de que durante su vida útil, los sistemas son dinámicos. Es decir, constantemente están sufriendo modificaciones, las cuales surgen a partir de la necesidad de ajustar el funcionamiento de los sistemas a nuevos requerimientos por parte de sus usuarios, o también porque se precise adaptar los sistemas a cambios que se presentan en el medio en el cual se encuentran inmersos.

Cuando la estructura de un sistema es oscura o complicada, entonces las modificaciones -que son frecuentes- resultan en extremo difíciles y consumen muchísimo tiempo : se mencionan cifras que van del cincuenta al setenta por ciento del total del tiempo empleado en el área de sistemas [Pressmann,27], [Van Tassel,101]. Por lo tanto, todo esfuerzo que se haga en favor de la sencillez en la construcción de los sistemas, es

digno de consideración.

El presente trabajo está dirigido a resaltar dos aspectos que pocas veces se toman en cuenta durante el desarrollo de los sistemas. Estos dos aspectos son la conexión entre módulos y la cohesión modular. Cabe mencionar que la mayoría de las metodologías están apoyadas en estos dos conceptos, lo cual permite apreciar su importancia.

- A quién se dirige.

Cuando escribo el presente trabajo, recuerdo los semestres durante los cuales cursé las primeras materias de computación, y me doy cuenta que me faltó conocer muchos conceptos de gran importancia, entre ellos están los dos que dan nombre a esta tesis.

Es por eso que este escrito está dirigido a los alumnos que ya conocen algún lenguaje de alto nivel y comienzan a diseñar sistemas de ciertas dimensiones; alumnos que tal vez dentro de cuatro o cinco años se dediquen profesionalmente al diseño, mantenimiento u operación de sistemas de cómputo, y que deberán conocer ampliamente las bases teóricas sobre las que se sustentan dichas actividades.

- Qué pretende.

La principal intención del presente trabajo es dar a conocer a los alumnos interesados en el área de sistemas de cómputo, los conceptos de cohesión y conexión; además de ejemplificar de manera clara las ventajas y desventajas que la aplicación o falta de aplicación de dichos conceptos ocasiona.

Debe mencionarse que en esta tesis influye notablemente el conocimiento que tengo de sistemas de gran tamaño, que han estado operando durante muchos años en empresas importantes. Lo anterior se destaca porque sistemas de este tipo son los que quien ahora es estudiante encontrará durante su desempeño profesional, ya sea que le corresponda idearlos, crearlos o modificarlos.

- Forma en que se presenta el trabajo.

La exposición de este trabajo parte de las nociones generales y se dirige hacia temas particulares, hasta llegar a los dos conceptos que le dan nombre. Se ha procurado dar ejemplos en todos aquellos casos en que se considera necesario aclarar o situar la explicación, que en ocasiones resulta abstracta o

teórica debido a la naturaleza misma del tema.

En la primera parte del trabajo se trata de establecer la terminología, y de situar el campo en el que se trabajará. Para ello, se dan ciertas definiciones básicas y se mencionan los principales tipos de estructuras de los sistemas, con el propósito de preparar el camino para el estudio de los sistemas modulares.

El siguiente capítulo se ocupa ya de lleno del tema de los sistemas modulares. Se inicia con un razonamiento que muestra que resulta más sencillo desarrollar un sistema a partir de su segmentación que si se considera de manera total, con esto se dá entrada al concepto de módulo, para llegar posteriormente a determinar las ventajas y desventajas que las modularizaciones presentan.

Una vez que se conocen las relaciones intermodulares e intramodulares, se presenta la definición formal de la conexión y posteriormente la de la cohesión, se dan varios ejemplos para cada uno de los distintos casos expuestos. Esta última parte es la más importante y constituye el núcleo de la tesis, y es el objetivo hacia el cual los capítulos anteriores nos han dirigido.

1. ESTRUCTURA DE LOS SISTEMAS COMPUTARIZADOS

1. Estructura de los sistemas computarizados.

Al estudiar los sistemas computarizados de información se observa que presentan diferentes aspectos, formas o estructuras. Desde luego que la estructura de los sistemas depende del punto desde el cual se les observe, siendo la estructura de control una de las más manejadas en el ámbito del diseño de sistemas.

En este capítulo se dará una definición de lo que aquí se entiende por estructura de un sistema computarizado de información. Para poder llegar a esa definición se requerirá el manejo de algunas otras, como las de sistema y programa. Una vez definido el término estructura, se establece una clasificación muy importante de los sistemas : monolíticos y modulares. Se describen y analizan los primeros, reservando los segundos para el siguiente capítulo, el cual se encuentra íntegramente dedicado a ellos, dada su gran importancia.

1.1. Definiciones.

- Sistema.

El primer término que se define para establecer el

terreno sobre el cual se elaborará todo el desarrollo de la tesis, es el de sistema. Se empezará por considerar diferentes definiciones, halladas en algunos diccionarios y libros, para posteriormente dar la definición que se empleará durante todo este trabajo.

Según el diccionario etimológico de Joan Corominas, sistema proviene de la voz griega systema, que significa 'conjunto', que a su vez se deriva de synistemi, 'yo reúno', voz en la cual aparece la forma histemi, 'yo coloco' (la cual está emparentada con la palabra latina stare, 'estar firme').

De acuerdo con el diccionario Larousse, sistema procede del griego, y significa "... Combinación de partes reunidas para obtener un resultado o formar un conjunto." Nótese que esta definición coincide con la anterior en dos ideas : la de reunir y la de conjunto.

Pressman [34] dice : "... un sistema se define como una colección de elementos relacionados de manera que permite la realización de un objetivo tangible." En la redacción de la última parte de la definición se observa que ésta comprende aspectos sobre la utilidad que los sistemas aportan.

Para el propósito de este trabajo, es suficiente con considerar a un sistema como un conjunto de elementos (ya sean éstos hombres, máquinas, procedimientos o inclusive otros

sistemas), reunidos o relacionados con el propósito de obtener un resultado determinado.

- Sistema computarizado de información.

Ahora, avanzando un poco más, y tomando como base la definición dada de sistema, se hablará de lo que deberá considerarse como sistema computarizado de información. Este será un sistema, cuyos elementos integrantes son "hombres, computadoras, programas y procedimientos", [Hartman,29]. Es decir, la definición completa seria : un sistema computarizado de información es un conjunto de hombres, computadoras, programas y procedimientos, reunidos o relacionados con el propósito de obtener un resultado determinado. Desde luego, el principal producto de estos sistemas es la información.

- Programa para computadora.

En la última definición se ha mencionado el término programa; pasemos a el análisis de su significado, para después situarlo en el contexto de la computación. En el diccionario etimológico de Corominas, encontramos que programa proviene del griego prógramma, el cual se deriva de prográphо, que significa 'yo anuncio por escrito'.

De el diccionario Larousse tomamos : "Programa. [...] Exposición que fija la linea de conducta a seguirse." Otra

definición que allí aparece es : " ... Conjunto de instrucciones preparadas de modo que un computador, [...] u otro aparato automático puedan efectuar una sucesión de operaciones determinadas."

Edward Yourdon [1979,30] dice que un programa es : "... una secuencia precisa y ordenada de instrucciones y conjuntos de instrucciones que en total, definen, describen, dirigen, o de otra forma, caracterizan la ejecución de alguna labor." Y allí mismo, "Un programa para computadora es entonces un programa que quizás por medio de transformaciones intermedias, puede dirigir a una computadora en la ejecución de la labor."

El mismo Yourdon [S.F.,1] cita a K. K. Kolence,

> .. La definición más frecuentemente utilizada - la de que un programa es una secuencia de instrucciones - obliga a uno a ignorar el papel de la información en el programa. Una definición mejor es la de que un programa es un conjunto de transformaciones y otras relaciones sobre conjuntos de datos y estructuras que los contienen ...

Procurando considerar el punto de vista anterior, se puede decir que un programa para computadora es un conjunto de transformaciones y otros tipos de relaciones sobre grupos de datos y sobre las estructuras que los contienen, definidas de manera que un computador las pueda efectuar.

Con lo dicho en los párrafos anteriores, se tienen ya definidos los siguientes términos :

- Sistema.
- Sistema computarizado de información.
- Programa para computadora.

Una vez establecidos estos tres términos, se está en posibilidad de mencionar y estudiar la forma o estructura de los sistemas o de los programas. Aquí se debe entender que se trata de sistemas computarizados de información y de programas para computadora.

- Estructura de un sistema.

La palabra estructura proviene del latín structura, y significa el arreglo o la disposición de las diversas partes de un todo. Para situar esta definición en el ámbito de los sistemas, se puede decir que la estructura de un sistema está conformada por los módulos -posteriormente se dará una definición formal de módulo, por el momento basta la noción intuitiva que se tenga- o segmentos que constituyen un programa y todas las líneas que sirven de comunicación entre ellos.

1.2. Diferentes tipos de estructuras.

Es claro que la estructura de un sistema no es única, depende del punto de vista que se tome para realizar su estudio. Entre los diversos tipos de estructuras que se distinguen en un sistema, existen tres que resaltan por su importancia (Fig. 1-1). A continuación se da una breve descripción de cada una de ellas.

- estructura referencial, es la que está basada en las referencias explícitas efectuadas en un lugar del sistema a cosas situadas en otro lugar. En ella están incluidas todas las menciones que se hagan de procesos, variables o estructuras, que en la construcción del sistema han sido ubicados en partes diferentes a aquella en la que se encuentra la mención.

- estructura comunicacional, es la que está constituida por el flujo de información entre diferentes partes del sistema. En este tipo de estructura están los parámetros, los indicadores, las variables de estado, etcétera. Es decir, en este tipo de estructura se considera toda transmisión de información dentro del sistema.

- estructura de control, es la que considera el flujo de control o activaciones sucesivas entre partes diferentes

PRINCIPALES ESTRUCTURAS DE LOS SISTEMAS

- REFERENCIAL
- COMUNICACIONAL
- DE CONTROL

FIGURA 1-1

SISTEMA MONOLITICO
(SIN DIVISIONES CLARAS)

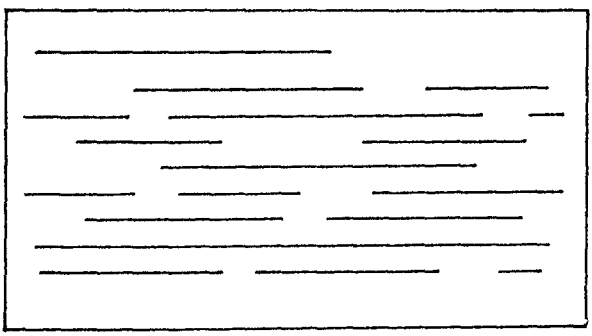


FIGURA 1-2

del sistema. Aquí tenemos todo aquello conocido como 'llamados a subrutinas', 'activaciones de procesos paralelos' o 'corrutinas', 'interrupciones', etcétera.

De todo lo anterior se puede notar que un mismo sistema posee varios tipos de estructuras. Para los propósitos que se persiguen en este trabajo se tratarán dos grupos de sistemas que se distinguen entre si por los diversos tipos de estructuras que cada uno posee, tanto referenciales como comunicacionales o de control. Estos dos grupos son los sistemas monolíticos y los sistemas modulares.

1.3. Sistemas monolíticos.

El término monolítico se aplica a cualquier sistema o segmento de él que está formado por piezas tan altamente interrelacionadas que se comportan como si se tratara de una sola pieza. Es decir, no existen divisiones claras -aunque sólo sean implícitas- dentro de todo el bloque de instrucciones, que sitúen o señalen los diferentes procesos que en él se llevan a cabo (Fig. 1-2).

En esta sección se mencionarán las ventajas y desventajas que este tipo de construcciones presenta. Se señalará la principal causa de las dificultades que se encuentran con dichas construcciones, para finalmente proponer una solución, la

cual nos acercará más al tema central de este escrito.

1.3.1. Ventajas y desventajas.

Como todas las cosas, los sistemas monolíticos tienen cualidades o ventajas, y también defectos o desventajas. Qué tan buena sea la creación o utilización de este tipo de sistemas, lo dirá el resultado del balance que se realice entre sus cualidades y sus defectos (Fig. 1-3). Para ello, se comenzará por mencionar sus ventajas.

Indiscutiblemente, la mayor ventaja que presentan los sistemas monolíticos es que su desarrollo resulta muy sencillo en términos del diseño. Esto debido a que no se requiere invertir tiempo en buscar una buena división o fragmentación del sistema. Inclusive el diseño mismo no necesita de mucha planeación, pues siempre hay forma de efectuar modificaciones sin estropear la estructura, pues ésta es elemental.

A lo anterior se agrega el hecho de que basta un conocimiento básico de programación para lograr diseños monolíticos, no es necesario haber realizado estudios ni siquiera medianamente profundos de diseño para encontrarse en posibilidades de idear un sistema de esta naturaleza.

Las dos situaciones mencionadas hacen que este tipo de sistemas se encuentre muy extendido entre los programadores con

PRINCIPALES VENTAJAS Y DESVENTAJAS DE LOS SISTEMAS MONOLITICOS

S I S T E M A S M O N O L I T I C O S

V E N T A J A S	D E S V E N T A J A S
- EL DISEÑO ES MUY SENCILLO	- DIFICULTAD PARA IDENTIFICAR LAS DIFERENTES FUNCIONES.
- SE REQUIEREN POCOS CONOCIMIENTOS DE PROGRAMACION.	- APARICION FRECUENTE DE EFECTOS LATERALES.

F I G U R A 1-3

poca experiencia o preparación en lo que a diseño de sistemas se refiere.

Pudiera parecer que las ventajas que ofrecen los sistemas monolíticos son suficientemente atractivas como para no verse en la necesidad de buscar otro tipo de estructuras. Sin embargo, las complicaciones que presentan son dignas de consideración, y a continuación se señalan algunas de las más representativas.

Cuando mejor se aprecian las desventajas de este tipo de sistemas es en el momento de intentar comprender su funcionamiento, y particularmente al efectuarles modificaciones que no sean de carácter trivial. El primer problema con que se encuentra quien tiene que modificar un sistema monolítico, radica en la gran dificultad que se presenta al querer identificar o aislar mentalmente las diferentes funciones que se efectúan dentro de ese único bloque, de quizás miles de instrucciones, que conforma el sistema.

Una vez que se ha logrado reconocer aquellas funciones que de alguna manera se encuentran diferenciadas del conjunto informe de instrucciones (pueden existir funciones cuyos componentes estén tan dispersos dentro del bloque, que resulte imposible identificarlas), aparece el segundo problema : efectuar correctamente la modificación. Esta tarea puede resultar muy penosa debido a los efectos laterales que durante su realización

pueden aparecer. Para apreciar mejor esta situación es conveniente detenerse un momento a considerar qué son los efectos laterales.

Se le llama efecto lateral a toda aquella consecuencia no prevista que se origina al efectuar una modificación a un sistema. Se puede notar que es muy probable que al efectuar una modificación dentro de un sistema al cual no se ha podido comprender por completo, aparezca algún efecto lateral. En el caso de los sistemas monolíticos, que son difíciles de comprender, la aparición de efectos laterales resulta una situación común.

1.3.2. Causa de las dificultades.

Ahora es necesario concentrarse en las dificultades de los sistemas monolíticos que han sido mencionadas, y buscar qué es lo que las genera o produce, para que posteriormente se esté en condiciones de encontrar algún tipo de remedio.

Entre las causas de la dificultad para manejar los sistemas monolíticos, destaca la limitación humana para trabajar con muchas situaciones simultáneamente [Miller,81-97], situándose el límite promedio en siete situaciones o aspectos distintos que pueden ser considerados de manera simultánea, sin que ello represente algún problema.

SITUACION COMUN EN LOS SISTEMAS MONOLITICOS

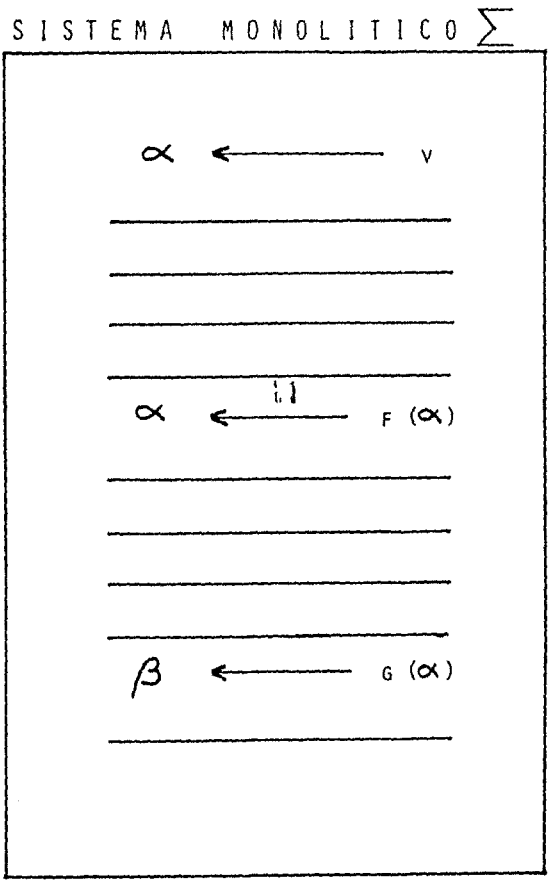


FIGURA 1-4

Debido a que dentro de un sistema monolítico son muchos los aspectos que deben considerarse a lo largo de toda la ejecución, no es de extrañar que estos sistemas sean tan difíciles de comprender, y por tanto de manipular.

Como ejemplo que ilustre este tipo de situaciones, puede mencionarse el siguiente hecho, que frecuentemente aparece al analizar sistemas monolíticos (Fig. 1-4) : supóngase que al iniciar la ejecución del sistema, llámese éste Σ, la variable α toma el valor v; después, en la mitad de la ejecución, ésa misma α interviene en algún cálculo y es modificado su valor; finalmente, poco antes de que termine la ejecución, el valor de α sirve de base para obtener o calcular el de alguna otra variable β. En un caso como el anterior, la persona que intenta comprender el funcionamiento del sistema Σ no debe olvidarse en ningún momento de la variable α, ya que prácticamente permanece activa durante toda la ejecución. Ahora, si se multiplica esta situación por el número de casos de este tipo que existan, se percibirá la magnitud real del problema.

1.3.3. Soluciones.

Sabiendo que la causa principal de la complejidad de un sistema monolítico está representada por el número de situaciones distintas que deben considerarse de manera simultánea, parece simple poner el remedio : bastaría con disminuir tanto como sea

posible el número de dichas situaciones. Es decir, dividir o segmentar el sistema de tal manera que cada una de las partes pueda ser analizada sin preocuparse (o cuando menos, sin preocuparse mucho) por las otras, y que además el sistema continúe realizando su función de manera satisfactoria. (Fig. 1-5)

Sin embargo, para llevar a la práctica este remedio, que conceptualmente resulta muy sencillo, es necesario adentrarse en algunos aspectos teóricos que a primera vista no resultan evidentes, y que forman el cuerpo principal de el presente trabajo.

SEGMENTACION DE UN

SISTEMA MONOLITICO

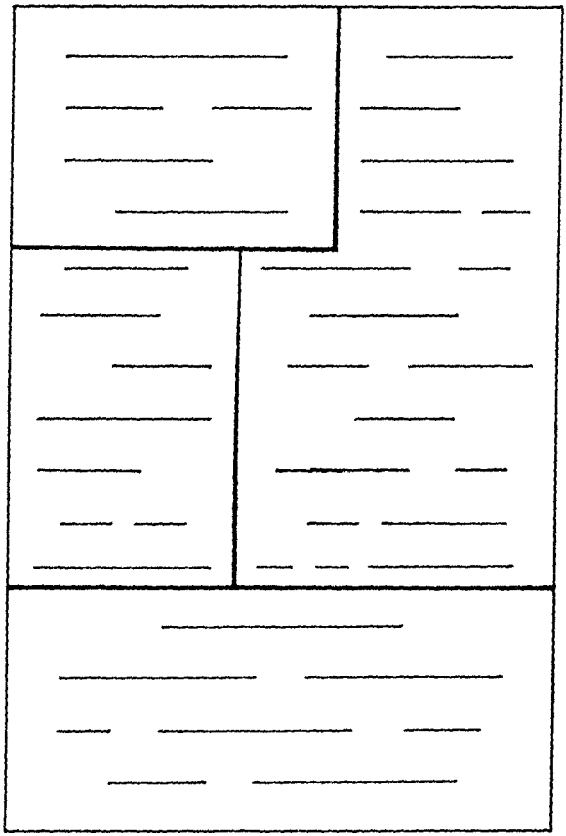


FIGURA 1-5

2. SISTEMAS MODULARES

2. SISTEMAS MODULARES.

En el capítulo anterior se mencionó que una buena forma de disminuir de manera notable los problemas que presentan los sistemas monolíticos es elaborar los sistemas con base en fragmentos pequeños, que permitan ocuparse de cada uno de ellos aisladamente sin verse en la necesidad de considerar la composición interna de los demás (Fig. 2-1).

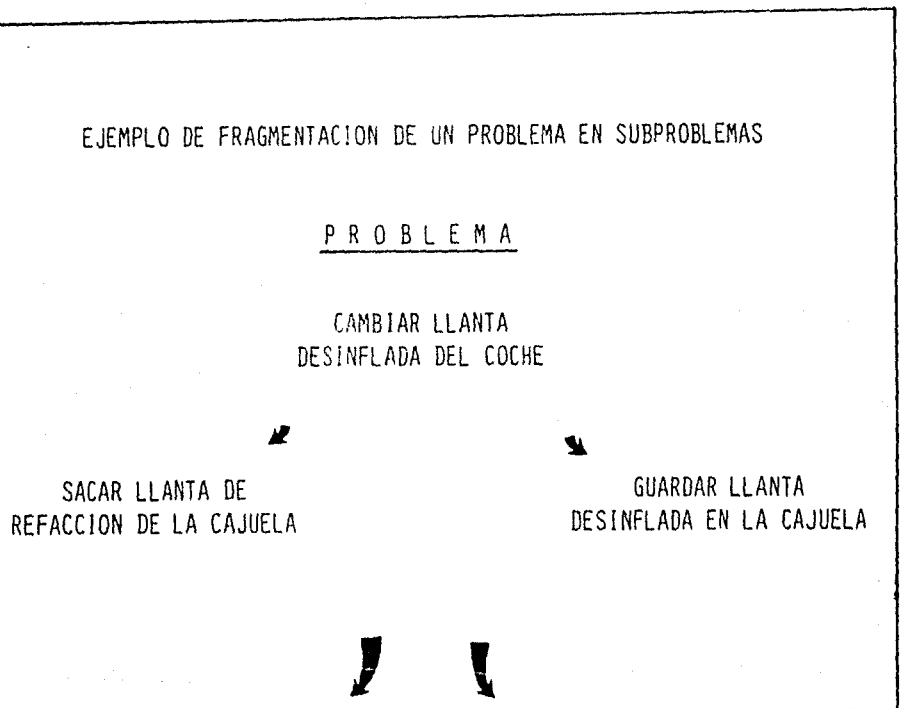
En este capítulo se formalizará esta idea y se harán notar algunas otras que también deben ser consideradas en la elaboración de los llamados sistemas modulares.

2.1. Un razonamiento básico en el desarrollo de sistemas.

Con base en la noción intuitiva de que en términos generales resulta más sencillo elaborar un sistema pequeño que uno grande, se ha desarrollado un razonamiento muy especial [Yourdon,1979,68-73] que a continuación se expondrá :

> Supóngase que se cuenta con un método M adecuado para determinar el tamaño o medida de un problema P, denotado como M(P).

EJEMPLO DE FRAGMENTACION DE UN PROBLEMA EN SUBPROBLEMAS



F I G U R A 2-1

Si el costo de resolver P es C(P). Entonces, intuitivamente, se espera que suceda lo siguiente

Sean P y Q dos problemas;

(a) Si M(P) > M(Q) entonces C(P) > C(Q)

Ahora supongamos que combinamos los dos problemas P y Q, formando (P+Q). La experiencia demuestra que los problemas combinados resultan más complicados que ambos contemplados de manera individual; lo cual es debido a que además de cada uno de los problemas, también es necesario considerar sus interacciones.

Tenemos entonces que,

(b) M(P+Q) > M(P) + M(Q)

y aplicando la relación (a) obtenemos

(c) C(P+Q) > C(P) + C(Q)

La conclusión es que resulta más sencillo y económico crear dos partes pequeñas que una grande, si ellas realizan en conjunto la misma función que la mayor.

La relación (c) puede interpretarse como

(d) $C(P) > C(\frac{1}{2}P) + C(\frac{1}{2}P)$

lo cual significa que es más sencillo resolver un problema si lo dividimos en subproblemas, pero de la relación (d), aplicada en forma repetida, se tiene

(e) $C(P) > C(\frac{1}{2}P) + C(\frac{1}{2}P) >$

$C(\frac{1}{4}P) + \ldots + C(\frac{1}{4}P) > \ldots$

y asi se podría continuar indefinidamente, reduciendo más y más el tamaño, y por tanto el costo de resolver el problema. Pero desde luego que este proceso debe tener un límite, y lo tiene : mientras más son los subproblemas generados a partir del problema original, más son las líneas de comunicación necesarias entre ellos. Al principio la complejidad agregada por las líneas de comunicación es despreciable, pero poco a poco va aumentando hasta que alcanza un nivel en donde

es mayor que la complejidad misma de los subproblemas. La situación anterior se puede apreciar en la figura 2-2. Consecuentemente, lo que se busca no es descomponer un problema en múltiples subproblemas, sino en el número y tipo de subproblemas que permitan resolverlos con el mínimo esfuerzo.

La exposición anterior señala en forma clara el camino a través del cual deben encauzarse los esfuerzos durante la fase de diseño para simplificar la construcción de los sistemas. Y una vez hecha esta necesaria introducción, se procederá a comentar las diversas formas de segmentar un sistema.

2.2. Módulo.

Una vez que se ha visto el razonamiento del punto anterior, se podría retomar el problema de segmentar un sistema de manera que facilite tanto su elaboración como su comprensión y, además, su modificación o mantenimiento.

Supongamos que se nos entrega un sistema monolítico y se nos solicita que lo segmentemos de manera conveniente. Si pensamos en el razonamiento expuesto, podríamos considerar que ocho es un buen número de partes para un sistema, e intentar dividirlo en ocho partes aproximadamente del mismo tamaño. Pero

COMPLEJIDAD DE LOS SISTEMAS

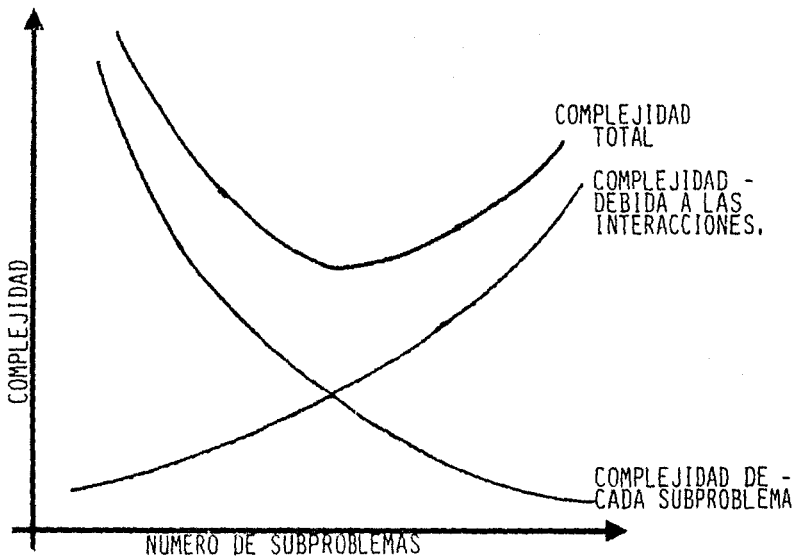


FIGURA 2-2

este procedimiento (Fig. 2-3) no nos llevaría a algo mejor que el sistema monolítico, y a continuación se verá por qué.

Como el lector ha de sospechar, además del número de fragmentos en que se divida el sistema, también resulta de gran importancia cómo sea cada uno de ellos. E. Yourdon [S.F.,93] afirma que un buen sistema es aquel en el cual puede modificarse cualquier porción lógica sin afectar el resto del sistema, y a este sistema le dá el nombre de modular.

El mismo autor en otro libro [1979,37] dice:

> Un módulo es una secuencia textualmente contigua de instrucciones, limitadas por elementos límite, que posee un identificador.

Lo cual quiere decir que un módulo (lo que hemos llamado segmento hasta ahora) está compuesto por instrucciones contiguas, que posee límites perfectamente definidos y que se le puede hacer referencia por medio de un nombre.

W. P. Stevens dice:

> El término módulo se utiliza para referirse a un conjunto de una o más instrucciones contiguas que posee un nombre por medio del cual otras partes

SEGMENTACION DE UN SISTEMA MONOLITICO EN PARTES IGUALES

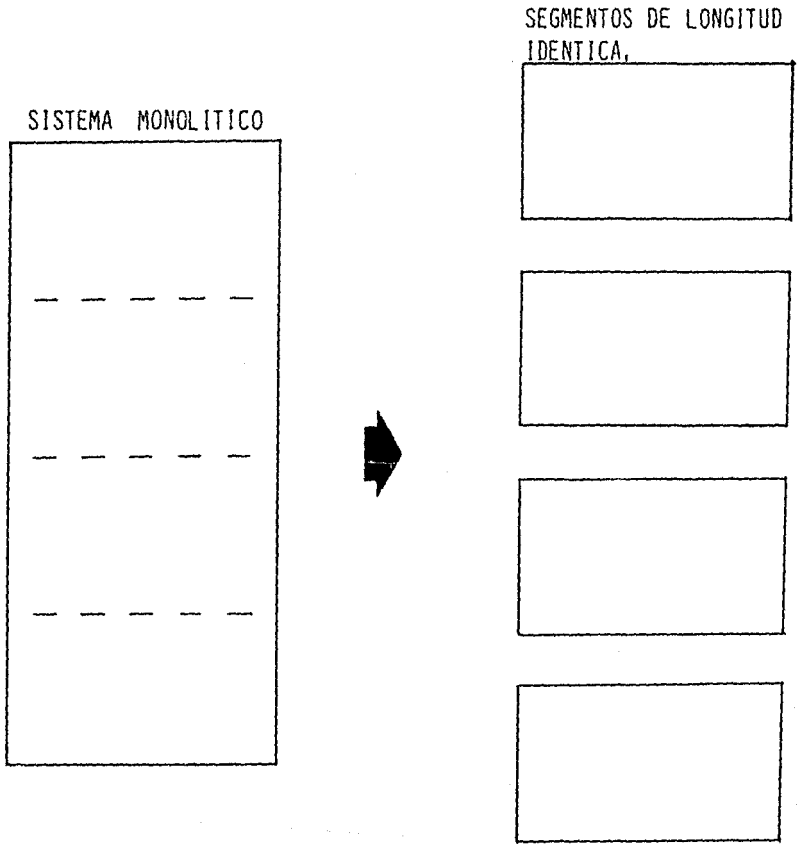


FIGURA 2-3

del sistema lo pueden llamar y preferentemente tiene un conjunto propio de nombres de variables.

Myers dá los atributos básicos de un módulo (Fig. 2-4):

> Un módulo posee tres atributos básicos : realiza una o más funciones, posee alguna lógica y es utilizado en uno o más contextos. La función es una descripción externa del módulo; describe lo que el módulo hace cuando es llamado, pero no describe cómo lo hace. La lógica describe el algoritmo interno del módulo, en otras palabras, cómo realiza su función. El contexto describe una utilización particular de un módulo. [...] La función de un módulo puede verse como la suma de la lógica del módulo más las funciones de todos sus módulos subordinados.

Todas las opiniones y definiciones anteriores dan una idea bastante completa de lo que es un módulo. Ahora, corresponde encontrar métodos efectivos de modularización de los sistemas.

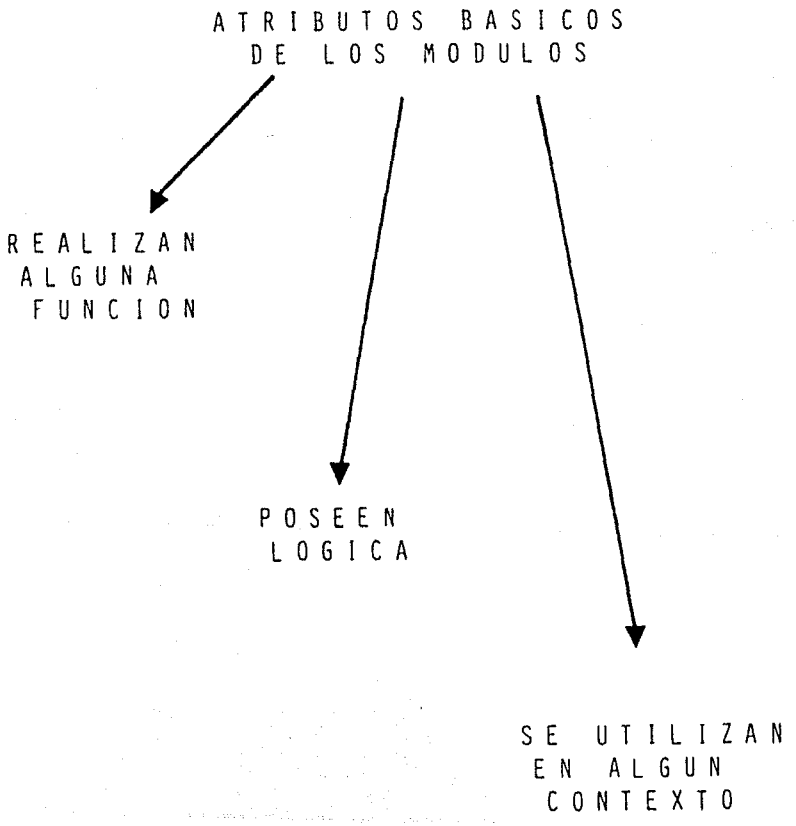

ATRIBUTOS BASICOS
DE LOS MODULOS
REALIZAN
ALGUNA
FUNCION
POSEEN
LOGICA
SE UTILIZAN
EN ALGUN
CONTEXTO

FIGURA 2-4

2.3. Modularidad de un sistema.

En este momento, nuestra principal preocupación consiste en hallar la manera de lograr sistemas modulares, partiendo del hecho de que se sabe qué es un módulo. Comencemos por las aproximaciones más sencillas y vayamos avanzando poco a poco hacia las que resultan más elaboradas.

Un método muy común para obtener sistemas "modulares" es restringir el tamaño de los módulos, desde cualquier segmento que quepa en N localidades de memoria hasta todo lo que un programador pueda elaborar y probar en un mes; pasando por limitar el número de instrucciones a algún número M (usualmente M $\leq$ 200).

En nuestro caso de programa monolítico se señaló que el tamaño de los módulos por sí solo no bastaría para lograr que un sistema fuera modular, y en los métodos anteriores toda la atención se centra en reducir o limitar el tamaño de los módulos; por lo tanto, de estos métodos podemos afirmar que no nos conducirán al logro de sistemas efectivamente modulares.

Como corroboración de la aserción anterior, tenemos las siguientes palabras de Parnas [1972b,1053] :

> La efectividad de una 'modularización' depende del criterio utilizado para dividir el sistema en módulos.

El mismo autor proporciona un criterio para lograr la modularización de los sistemas [1972b,1056] :

> [...] se efectuó [la modularización] utilizando el 'ocultamiento de información' como criterio. Los módulos ya no corresponden a pasos en el procesamiento [...]. Cada módulo [...] se caracteriza por su conocimiento de una decisión de diseño, la cual oculta de todos los demás. Su línea de comunicación o definición fué seleccionada para que revelara tan poco como fuese posible acerca de sus labores internas.

Lo que podemos sacar en claro de estas opiniones es que para obtener un sistema modular, no basta con dividirlo en módulos, sino que es necesario efectuar la división de acuerdo con algún criterio claramente definido; además hemos visto que dicho criterio no debe basarse únicamente sobre el tamaño de los módulos.

Debe considerarse de manera muy seria la afirmación de Parnas con respecto al ocultamiento de información, ya que este criterio redunda en beneficio del diseñador, pues poco a poco va disminuyendo la cantidad de detalles a los cuales se debe prestar atención de manera simultánea. De este último hecho, ya se han comentado los problemas que ocasiona en los sistemas monolíticos (Sección 1.3.2).

A todo lo dicho anteriormente se pueden agregar las observaciones de Liskov, respecto a algunos hechos que provocan que algunas modularizaciones no resulten efectivas como métodos para reducir la complejidad de los sistemas :

1) Los módulos son diseñados para realizar muchas funciones relacionadas, pero diferentes, lo cual dificulta la comprensión de su funcionamiento.

2) Las funciones comunes no son identificadas en el diseño, lo cual se traduce en su distribución entre muchos módulos diferentes, además de su construcción poco uniforme.

3) Los módulos actúan sobre información común en formas que resultan impredecibles.

El objetivo de esta sección es establecer de manera clara que el lograr un sistema modular es una labor que requiere

consideraciones de muy diversos tipos; desde luego, todas ellas buscando una división óptima del sistema en módulos.

2.4. Ventajas y desventajas.

Hasta ahora se ha visto qué es un módulo y cómo se busca la modularidad de los sistemas. Pero no se ha dicho nada respecto a qué se obtiene con un sistema modular y qué problemas presenta. Yourdon [S.F.,97-99] habla de las siguientes ventajas y desventajas de los sistemas modulares.

Ventajas.

1. La elaboración de un sistema modular es más sencilla que la de un sistema no modular. Lo cual resulta lógico (y sobre todo en sistemas de cierta magnitud), si se piensa que los diferentes componentes o módulos pueden ser creados de manera independiente. De esta forma es más simple poder calcular tiempos de elaboración y detectar oportunamente los retrasos.
2. La depuración de un sistema modular se simplifica; pues cada uno de los módulos puede probarse individualmente, aislando así las fallas dentro de pequeñas regiones.

3. Un sistema modular se mantiene y modifica de manera fácil. Los módulos pueden ser modificados, reescritos o reemplazados sin afectar otras partes del sistema.

Para apreciar la importancia de las dos primeras ventajas mencionadas, basta decir que el noventa por ciento del costo del desarrollo de los sistemas está asociado con los programas [Pressman,23]. Si además se observa que del cuarenta al setenta por ciento del presupuesto de muchas empresas en el área de sistemas se dedica al mantenimiento, se reconoce el beneficio que representa el tercer punto [Id.,19]

La magnitud de estas dos cifras provoca que cualquier esfuerzo -por pequeño que resulte- que se haga por reducirlas se considere importante, y es ahí precisamente donde la modularidad de los sistemas muestra su utilidad.

Desde luego, que no todo en la modularidad son ventajas, también existen algunas desventajas, de las cuales a continuación señalamos algunas. (Cabe aclara que Yourdon menciona otras desventajas que están relacionadas con espacio en memoria y tiempo de procesamiento, que aunque resultan significativas, comprenden aspectos de los que el presente trabajo no se ocupa)

Desventajas.

1. La mayoría de los programadores no comprenden lo que es la modularidad.
2. Para lograr la modularidad se requiere ser más meticuloso en la fase de diseño; es necesario trabajar más.

En este trabajo se busca ayudar a resolver el primer punto, aunque sea en un nivel elemental. Para el segundo, debe pensarse que todas las facilidades que se obtengan a causa de un diseño meticuloso se verán multiplicadas durante todo el tiempo que el sistema esté en uso.

2.5. Independencia Modular.

Un concepto muy importante dentro del tema que estamos tratando es el de la independencia modular. Al respecto, E. Yourdon [1979,84] dice : "Dos módulos son totalmente independientes si cada uno puede funcionar completamente sin la presencia del otro." Desde luego que a lo que se refiere es a la independencia absoluta, donde no existe ningún tipo de conexión, por leve que sea, entre ellos. Y también se comprende que, en general, mientras más interconexiones existan entre los módulos, menos independientes son (Fig. 2-5).

INTERCONEXION DE MODULOS

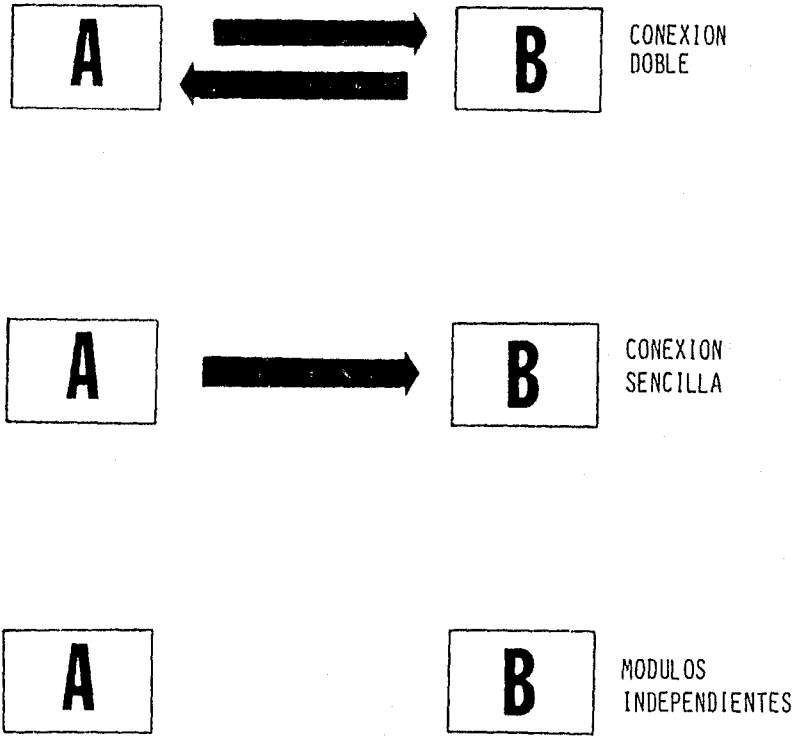


FIGURA 2-5

El objetivo que se persigue con la independencia modular es la posibilidad de realizar modificaciones a cualquier módulo sin afectar por ello a los demás.

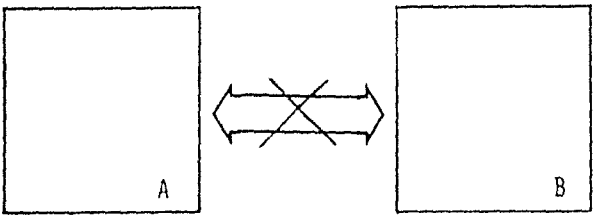
Otra forma de caracterizar la independencia modular es exigiendo que sea posible reemplazar cualquier módulo con otro que sea equivalente funcionalmente sin afectar a ninguno de los demás módulos del sistema [Yourdon,S.F.,96].

La opinión de R. S. Pressman[158] es la siguiente, "La independencia modular se logra al desarrollar módulos con funciones de 'un solo propósito' y con una 'aversión' a la interacción excesiva con otros módulos."

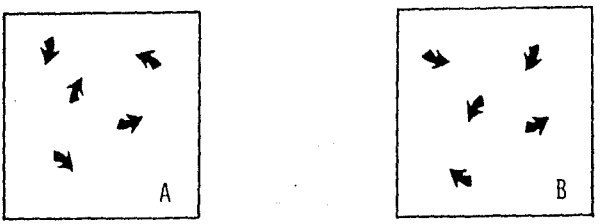
El estudio de la independencia modular ha llevado a los investigadores a identificar dos situaciones que destacan dentro del diseño de los módulos. Una de ellas consiste en hacer mínimas las relaciones entre módulos, y la otra en lograr que sean máximas las relaciones de los elementos dentro de cada módulo (Fig. 2-6). Para ello se han desarrollado los conceptos de cohesión y conexión*, los cuales son motivo de los siguientes dos capítulos, donde se describe cada uno de ellos y se muestran ejemplos de su aplicación.

(*) En los libros escritos en inglés llaman 'coupling' a lo que aqui se llama conexión.

DOS SITUACIONES IMPORTANTES EN EL DISEÑO DE LOS MODULOS



RELACION MINIMA ENTRE MODULOS



RELACION MAXIMA ENTRE ELEMENTOS DE CADA MODULO

FIGURA 2-6

Para terminar el presente capitulo conviene anotar las siguientes palabras [Pressman,158] : "... la independencia modular es la clave para un buen diseño, y el diseño es la clave para la programación de calidad."

3. CONEXION

3. CONEXION.

Como se ha mencionado, para lograr sistemas realmente modulares se requiere disminuir las relaciones existentes entre los módulos. Sin embargo, también deben considerarse los diferentes tipos de relaciones o asociaciones que hay entre módulos, ya que la complejidad de un sistema se ve afectada no solo por el número de enlaces, sino también por el grado en que cada uno de ellos asocia a dos módulos, haciéndolos interdependientes [Stevens,117] (Fig. 3-1). Para ello, existe un concepto de gran utilidad, llamado conexión. Veamos lo que dice Stevens sobre la conexión : "Conexión es la medida de la fuerza de asociación establecida por el enlace de un módulo con otro." Por su parte Pressman[161] dá esta definición : "Conexión es una medida del enlace entre módulos en una estructura de programación." Yourdon agrega a lo anterior :

> La conexión como un concepto abstracto -el grado de interdependencia de los módulos- puede ser manejado como la probabilidad de que al programar, depurar o modificar un módulo, un programador tendrá que considerar algo acerca de otro módulo.

Con las citas arriba transcritas, se obtiene una idea general de qué es el concepto de conexión. Fundamentalmente, se trata de una medida, la cual se aplica a la relación que guardan entre si los distintos módulos que integran un sistema. Además,

el hecho que motivó la aparición del concepto de conexión fué la necesidad de contar con un método que facilitara el manejo de las segmentaciones de los sistemas, asignándoles "calificaciones" que señalan qué tan buenas o eficientes resultan.

La conexión siendo una medida tiene varias intensidades; es decir, existe conexión leve, conexión media, conexión severa, etcétera. El objetivo en este capítulo es señalar los diversos tipos de conexión que existen, cuáles son sus principales causas y qué métodos existen para evitarla o disminuir su grado.

3.1. Factores que influyen en el grado de conexión.

Dado que una fuerte conexión entre módulos conduce a que un sistema resulte difícil de manipular (desde comprenderlo hasta modificarlo), lo que se debe buscar es que el grado de conexión entre los módulos de un sistema sea el mínimo posible. Para ello es conveniente identificar los diferentes aspectos que determinan o influyen en el grado de conexión.

En su artículo, Stevens[117] dice que el grado de conexión depende (1) de qué tan simple o complicado resulta el enlace entre los módulos, (2) de si el enlace hace referencia al módulo como tal o si la referencia es a algún elemento interno, y también (3) de lo que sea enviado y recibido por los módulos. A

PRINCIPALES FACTORES QUE AFECTAN LA COMPLEJIDAD DE UN SISTEMA.

- NUMERO DE ENLACES ENTRE MODULOS.
- INTENSIDAD DE LOS ENLACES ENTRE MODULOS.

FIGURA 3-1

TIPOS DE ENLACES ENTRE MODULOS

PRINCIPALES TIPOS DE ENLACES INTERMODULARES.

- MINIMO
  - A) PARAMETROS
  - B) ELEMENTO IDENTIFICADOR.
  - C) REGRESO AL PUNTO INMEDIATO.
- NORMAL
  - A) MAS DE UNA ENTRADA
  - B) REGRESO DISTINTO AL PUNTO INMEDIATO.
- PATOLOGICO
  - A) SE EVITAN LAS VIAS ESTABLECIDAS JERARQUICAMENTE.

FIGURA 3-2

los factores arriba mencionados, Yourdon[1979,86] agrega (4) el momento en que los enlaces intermodulares son establecidos. Es conveniente analizar cada uno de estos aspectos individualmente, además de mencionar algunos otros que también resultan de importancia.

3.1.1. Tipo de enlace entre los módulos.

Para analizar los diferentes tipos de enlaces que pueden existir entre los módulos de un sistema, debe establecerse que se llama enlace a la referencia que efectúa un elemento cualquiera a el nombre, la localidad o al identificador de algún otro elemento. Los enlaces pueden resultar dentro de un mismo módulo o pueden ser tales, que crucen sus límites; estos últimos enlaces son llamados intermodulares, mientras que a los primeros se les dá el nombre de intramodulares.

En este capítulo el interés está centrado en los enlaces intermodulares, de los cuales existen tres tipos principales : enlace mínimo, enlace normal y enlace patológico (Fig. 3-2). A continuación se mencionan las principales características que presenta cada uno de ellos.

- Enlace mínimo.

El enlace mínimo es el más simple de los tres, y el que menos complejidad aporta a un sistema. Este tipo de enlace está

caracterizado por las siguientes situaciones :

a) Toda la transmisión de información entre los módulos se efectúa por medio de parámetros; tanto la información de entrada como la de salida.

b) La transferencia inicial de control se efectúa al único elemento identificador y activador (es decir, al nombre) del módulo invocado.

c) La transferencia de control de regreso se efectúa al punto inmediato posterior a aquel en el que se realizó la transferencia inicial.

Los sistemas enlazados de este modo son mínimos porque contienen el menor número de elementos necesarios para establecer una transferencia bidireccional tanto de información como de control.

- Enlace normal.

Los enlaces normales tienen este nombre porque, al igual que los enlaces mínimos, permiten que el sistema mantenga un comportamiento normal (es decir, todo el contexto de información de un módulo y su regreso preciso son establecidos por el módulo activador), aunque no cumpla con todas las condiciones de los enlaces mínimos. Un enlace es normal siempre que no sea mínimo, debido a alguna de las siguientes razones :

a) Hay más de un punto de entrada para algún módulo, siempre que dicha entrada sea mínima con respecto a la transferencia de información.

b) El control es devuelto a un punto distinto al siguiente del de activación, siempre que los puntos alternativos de regreso sean definidos por el módulo activador como parte de su proceso de activación.

Los enlaces normales resultan más complicados que los enlaces mínimos, motivo por el cual su empleo debe efectuarse extremando las precauciones para que éste no resulte perjudicial.

- Enlace patológico.

Cuando un sistema no está enlazado mínima, ni normalmente, entonces lo que sucede es que algunos de sus módulos están enlazados patológicamente. Lo que esto quiere decir es que de alguna forma se ha establecido entre los módulos una comunicación que no sigue el camino usual, violando límites bien definidos en los módulos.

Una conexión patológica aparece cuando entre dos módulos se presenta una transmisión de información de manera tal que se evitan las vías que jerárquicamente han sido establecidas; es decir, la información no se transmite hacia los niveles jerárquicos superiores para que de allí sea llevada al módulo correspondiente, sino que se utiliza algún camino que en la práctica resulta más corto, ya sea directo o indirecto, con lo cual se trata de ahorrar esfuerzos. Aunque el "tratar de ahorrar esfuerzos" parece un buen propósito, en realidad lo que sucede con las conexiones patológicas es que se reduce la independencia

CONEXION PATOLOGICA DIRECTA

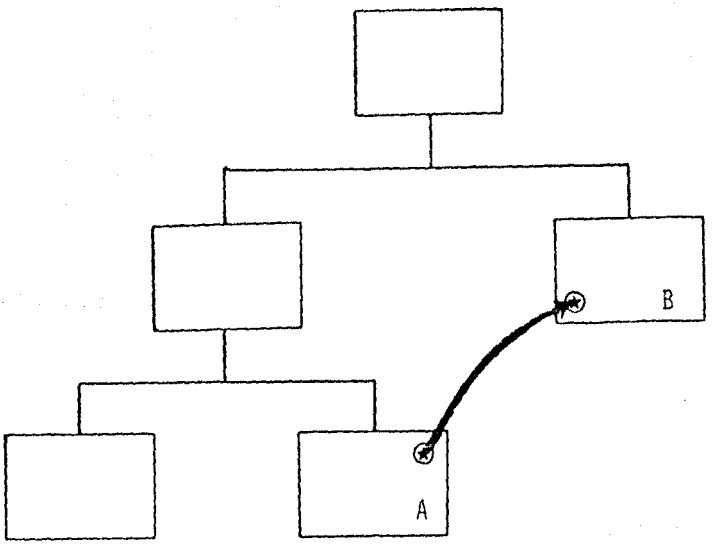


FIGURA 3-3

modular al establecer una especie de triangulación entre los módulos.

Cuando la conexión patológica es directa lo que ocurre es que el módulo A, por sí mismo, transmite o modifica información dentro del módulo B (Fig. 3-3). Si se trata de conexiones patológicas indirectas, entonces existe un tercer elemento que opera como intermediario en la transmisión de la información. Este tercer elemento puede ser otro módulo (subordinado tanto a A como a B), un ambiente común o algún tipo de dispositivo (Fig. 3-4).

3.1.2. Complejidad de la línea de comunicación.

Aquí aparece nuevamente el concepto de complejidad, que ya ha sido tratado en secciones anteriores, aunque siempre de manera no muy formal. Resulta que si la línea de comunicación entre dos módulos es "compleja", entonces el grado de conexión existente entre ellos es alto. La anterior no es ninguna afirmación asombrosa, sin embargo presenta un inconveniente, el cual radica en la manera de medir la complejidad de la línea de comunicación. De manera muy general, puede decirse [Yourdon,1979,86-90] que mientras más parámetros contiene la línea de comunicación entre dos módulos, mayor es el grado de conexión entre ellos.

CONEXION PATOLOGICA INDIRECTA

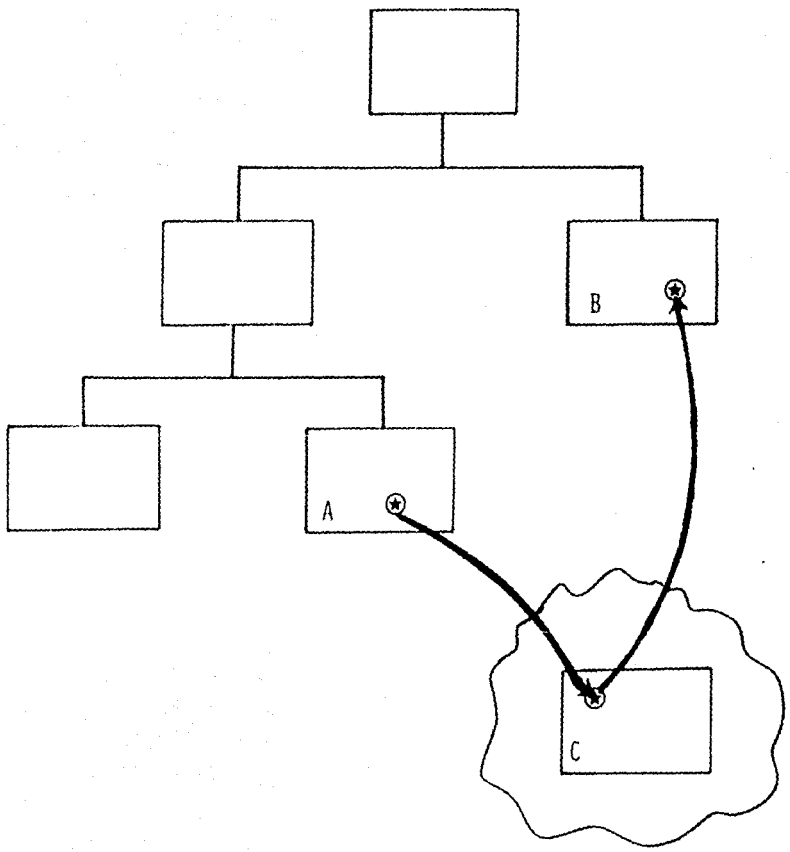


FIGURA 3-4

Al respecto Stevens[119] dice

> La complejidad de una línea de comunicación depende de qué tanta información se requiere para establecer o para comprender el enlace. Así, las relaciones obvias dan como resultado una conexión menor que las oscuras o supuestas. Mientras más unidades sintácticas (tales como los paramétros) haya en la instrucción del enlace, mayor será la conexión.

3.1.3. Flujo de información.

Para el funcionamiento de un sistema es indispensable que exista comunicación de información entre las diversas partes que lo constituyen. Desde luego que existen distintas clases de información, y como se verá, cada una de ellas afecta de forma distinta el grado de conexión de un sistema. A continuación se mencionan los principales tipos de conexiones que se producen según la clase de información que fluya dentro del sistema.

- Conexión de información o de entrada y salida.

Cuando a través del enlace establecido entre dos módulos fluye información del tipo de entrada y salida, el grado de conexión es mínimo, por lo que al flujo se refiere. La información de entrada y salida es aquella que es generada por un módulo y tomada por el siguiente, el cual la considera como

información externa (Fig. 3-5).

La conexión de información es la conexión de menor grado que puede tener un sistema; esto se debe a que ningún sistema puede funcionar si no existe transferencia de información entre los módulos que lo componen. Además, cualquier sistema puede construirse de manera que esta sea la única conexión que posea [Yourdon,1979,90].

- Conexión de control.

Existen algunos tipos de enlaces entre módulos donde los elementos que se transmiten no son solamente datos, sino que su propósito principal es afectar o dirigir la ejecución del módulo invocado. Cuando sucede esto, se está frente a una conexión que es denominada de control (Fig. 3-6).

El grado que se presenta con la conexión de control es mayor que el existente con la conexión de información, la causa es clara : desde fuera del módulo invocado se está teniendo injerencia en su comportamiento interno, lo cual va en contra de uno de los primeros planteamientos que con respecto a la modularidad se hicieron, es decir, el no preocuparse por la manera en que los módulos efectúen su función.

Dentro de esta misma clasificación existe un caso más grave de conexión, que aparece cuando el módulo invocado indica al que lo activó qué es lo que debe hacer. Es decir, el

CONEXION DE ENTRADA Y SALIDA

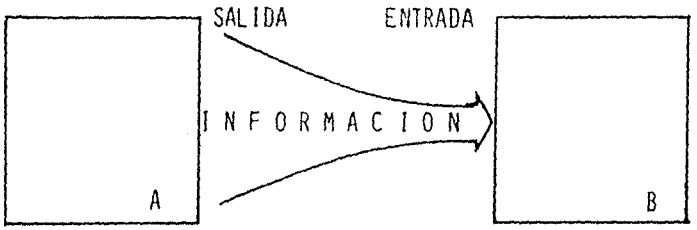


FIGURA 3-5

CONEXION DE CONTROL

FIGURA 3-6

comportamiento del módulo activador se ve afectado o resulta dirigido por el procesamiento interno del módulo subordinado, con lo cual se invierte el flujo de control, con respecto a las jerarquías (Fig. 3-7).

- Conexión híbrida.

Este tipo de conexión se presenta cuando el módulo invocador modifica el contenido procesal (código) del módulo subordinado. Se dice que esta conexión es híbrida porque el código modificado es considerado como información por el módulo invocador, sin embargo, el resultado de esta modificación es un aspecto de control para el módulo invocado.

En esta situación el grado de conexión es mayor que en el de información y que en el de control, debido a que cualquier modificación, por ligera que resulte, en alguno de los dos módulos enlazados, afecta de modo serio al otro. En resumen, el módulo subordinado depende en su comportamiento interno del módulo invocador, y éste se ocupa demasiado de la manera en que el primero efectúa su función.

En la actualidad resulta rarísimo encontrar un programador que utilice la técnica de modificar secciones de código durante la ejecución; es más, muy pocos lenguajes de programación contienen en su repertorio alguna instrucción que

CONEXION DE CONTROL GRAVE

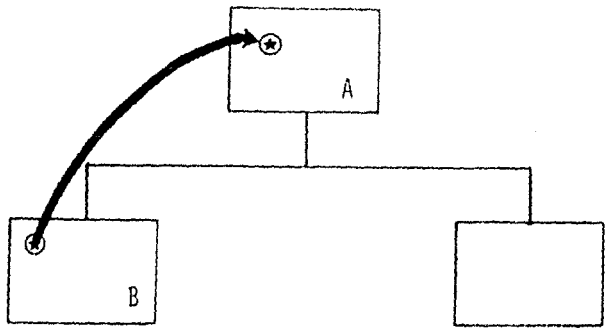


FIGURA 3-7

METODO PARA DISMINUIR LA CONEXION

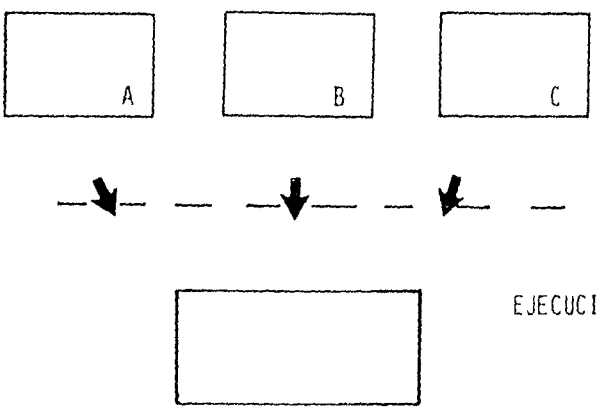


FIGURA 3-8

permita este tipo de manejos.

3.1.4. Momento en que los enlaces intermodulares son establecidos.

Dentro de toda la elaboración de un sistema, existen diferentes momentos durante los cuales se establecen los valores que se manejarán. Se tiene por ejemplo, que desde el primer diseño se puede establecer un límite de registros que se podrán manipular, aunque también se puede establecer este límite en el momento de efectuar la programación, o inclusive al momento mismo de la ejecución. Ahora, de manera general es preferible establecer los valores internos de un sistema, tan tarde como sea posible, porque de esta forma se facilita su modificación.

Situando este concepto en los enlaces intermodulares, resulta que el momento de establecer los enlaces intermodulares afecta en forma directa el grado de conexión entre los módulos. Mientras más temprano se establezca una referencia intermodular, mayor será el grado de conexión entre el módulo que efectúa la referencia y el módulo referido.

Se debe señalar que un método muy efectivo para disminuir el grado de este tipo de conexión lo constituye la posibilidad tanto de programar como de compilar los módulos de manera independiente. Por esta razón es que dicha facilidad se

encuentra cada vez con mayor frecuencia en los lenguajes de programación y en los sistemas operativos actuales (Fig. 3-8).

3.1.5. Conexión de contenido.

Este tipo de conexión es considerado por varios autores incluido dentro del caso anterior, pero aquí se ha preferido presentarlo como un caso aparte con el propósito de resaltar su importancia.

La conexión de contenido aparece cuando todo un módulo o alguna parte de él se encuentra incluida en otro módulo. En esta definición se distinguen dos casos de conexión de contenido. En el primer caso, que es el más simple, sucede que todo un módulo se halla contenido en algún otro (Fig. 3-9). En esta situación lo que usualmente se conoce por "llamada a subrutina" prácticamente no existe, pues la activación del módulo subordinado ocurre simplemente al llegar el control a donde él se encuentra. Este tipo de inclusión es sencillo de manejar, debido a que para hacerlos más independientes basta con separarlos. Esta separación es muy simple y directa, pues los límites de cada módulo se encuentran claramente definidos, y el que está incluido en el otro casi siempre se halla constituido por una sola pieza.

El caso en que solo una porción de un módulo está contenida dentro de otro resulta más complicado (Fig. 3-10). La

CONEXION DE CONTENIDO SIMPLE

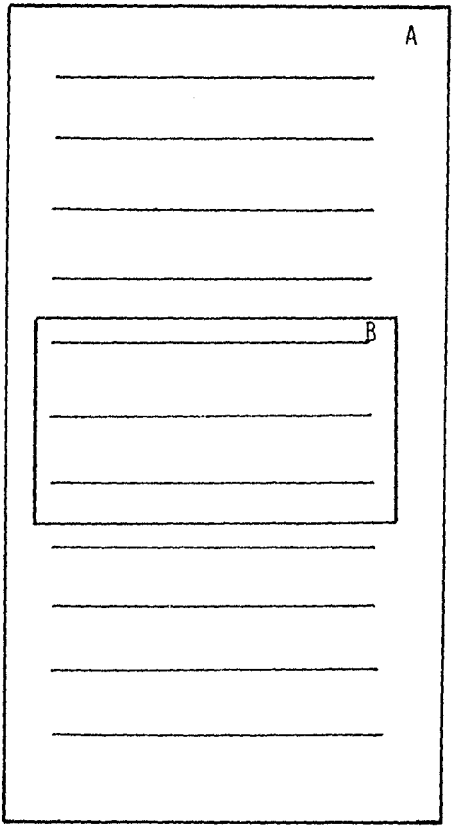


FIGURA 3-9

CONEXION DE CONTENIDO COMPLEJA

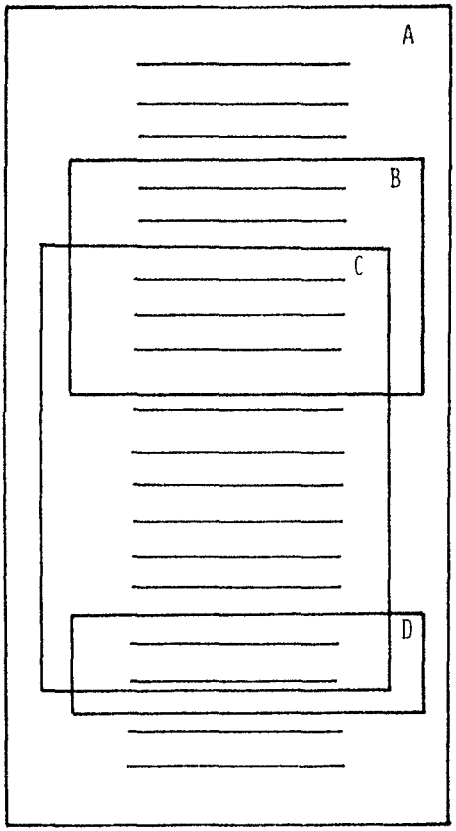


FIGURA 3-10

situación más común que presenta este tipo de inclusión aparece en los módulos con entradas múltiples, donde es típico que códigos de funciones distintas se encuentren superpuestos o enlazados. La dificultad para separar este tipo de módulos es bastante mayor que en el caso antes mencionado, lo cual se debe a que aquí no siempre existe una clara demarcación de los límites de cada módulo, y cuando ésta se presenta, puede ser necesario duplicar secciones de código para poder tener cada módulo completo además de independiente.

3.1.6. Conexión de ambiente común.

Para quienes manejan el lenguaje de programación FORTRAN es muy conocida la situación en la que por el tedio que significa anotar una larga lista de parámetros en algunos llamados a subrutinas, se prefiere englobarlos en un área común de datos (COMMON). Y aunque aparentemente de esta forma las cosas se han simplificado, la realidad está muy lejos de ser así; pues el hecho de que dos módulos completamente ajenos tengan acceso a la misma área común, basta para que entre ellos surja una conexión que de otra forma no existiría, y esta es la llamada "conexión de ambiente común".

Veamos lo que dice al respecto Stevens[1974]

> Cada elemento en el ambiente común, ya sea que se use por algunos módulos en particular o no, constituye una trayectoria independiente a lo largo de la cual los errores y los cambios se pueden propagar.

y es precisamente allí donde radica el núcleo del problema con un ambiente común: cada elemento (ya sea dato, variable, parámetro, etcétera) dentro del ambiente común, se convierte en un canal a través del cual pueden fluir situaciones, tanto positivas como negativas. Además, este número de vías se multiplica por el número de parejas de módulos que tengan acceso al ambiente común, sin importar si los módulos guardan, modifican o toman la información, es suficiente con que exista el acceso para que los módulos quedan conectados.

Quizá resulte revelador el imaginar que nos vemos en la necesidad de examinar un sistema en el cual cinco módulos tienen acceso a un ambiente común que contiene tres elementos (nótese que en la realidad no es raro que se presenten situaciones mucho más complicadas que ésta), lo cual automáticamente hace que existan

$$E \cdot M \cdot (M-1) \text{ vías,}$$

donde

E = número de elementos en el ambiente común.

M = número de módulos que tienen acceso al ambiente común.

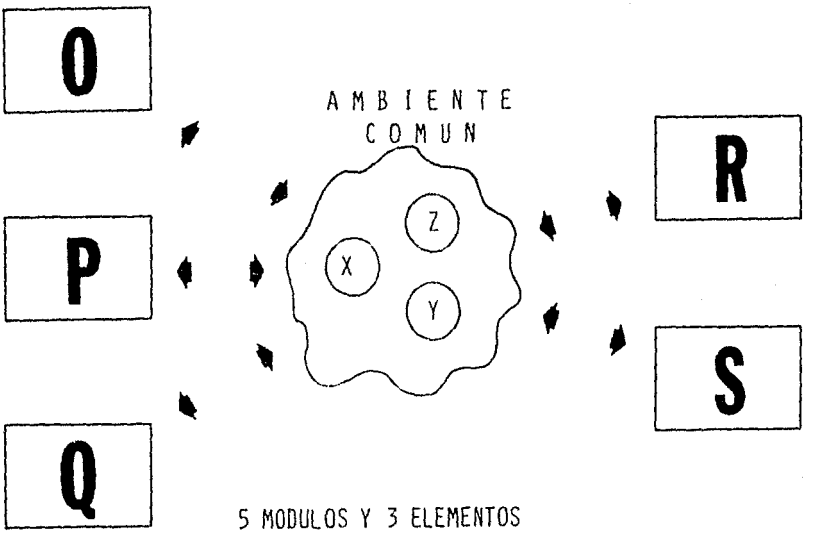
es decir, existen 3·5·4 = 60 vías que sería necesario considerar como posibles trayectorias de error (Fig. 3-11). Y lo que es muy importante es que entre los cinco módulos pueden existir parejas de ellos que no tuvieran entre sí ninguna relación, sin embargo, después de tener ambos acceso al ambiente común quedan conectados.

3.2. Recomendaciones para la disminución del grado de conexión de los sistemas.

Una vez analizadas las principales causas de la conexión modular, resulta oportuno elaborar algunas recomendaciones sobre la manera de evitar o disminuir la conexión en los sistemas, tomando como base las conclusiones obtenidas en el punto anterior.

Para facilitar la exposición, se efectuará la presentación de las recomendaciones siguiendo el orden en que se

CONEXION DE AMBIENTE COMUN



5 MODULOS Y 3 ELEMENTOS
GENERAN 3.5.4 = 60 VIAS

FIGURA 3-11

analizaron los factores en el punto anterior.

3.2.1. Tipo de enlace.

En el caso del tipo de enlace, se debe procurar el enlace mínimo y evitar tanto como sea posible los enlaces patológicos. Aquí cabe hacer la aclaración de que cuando por necesidad se decida establecer un enlace patológico, éste debe documentarse de manera muy especial para facilitar su identificación y comprensión. Esta aclaración puede generalizarse del siguiente modo : siempre que se utilice una construcción que genere un grado de conexión alto, debe ponerse un cuidado muy especial en hacer evidente y documentar dicha construcción o situación.

A continuación se enuncian las sugerencias correspondientes al tipo de enlace

- Toda comunicación entre módulos debe efectuarse a través de parámetros.
- Entrar a los módulos siempre por su parte inicial o encabezado. Ahora, cuando esto no sea posible, y la entrada a algún módulo sea por su parte media, debe efectuarse, como se dijo en el punto anterior, con los parámetros correspondientes.
- Evitar los regresos de control múltiples. Cuando exista la

necesidad de algún regreso múltiple, establecer desde la invocación todos los posibles puntos de regreso.

En lo que respecta a los enlaces de tipo patológico, que representan un caso que debe manejarse con muchísima precaución, se tiene

- Siempre respetar los niveles jerárquicos y los caminos que éstos determinan. Lo anterior debe hacerse aunque ello implique una cantidad mayor de trabajo en la elaboración o diseño del sistema.

3.2.2. Complejidad de la línea de comunicación.

Debe procurarse que las líneas de comunicación sean lo más trasparentes que sea posible. Además, el número de parámetros transmitidos debe ser tan pequeño como lo permita la aplicación en que se trabaje.

3.2.3. Flujo de información.

Debido a que la conexión de entrada y salida es la de menor grado y además es indispensable en todo sistema, los

esfuerzos durante la fase de diseño deben estar dirigidos a buscarla por medio de las siguientes acciones :

- Procurar diseñar los módulos de manera que entre ellos se transmita solo información que se pueda considerar como datos.
- Evitar desde la fase del diseño que algún módulo sea responsable del funcionamiento interno de otro, para que de esta manera no se requiera la transmisión de elementos de control entre los módulos.
- Concebir la estructura interna de los módulos de tal forma que ninguno de ellos modifique las instrucciones que conforman a alguno de los demás.

3.2.4. Momento en que los enlaces intermodulares son establecidos.

En este punto la recomendación es muy directa y simple:

- Establecer los enlaces o referencias intermodulares lo más tarde que lo permita el entorno en el cual está inmerso el sistema.

3.2.5. Conexión de contenido.

Como se mencionó, en la conexión de contenido existen dos casos principales, la contención parcial y la total. Para la contención parcial se tiene que la situación que más comunmente se presenta es la de módulos con varias entradas, por lo tanto una posible solución es :

- Evitar los módulos con entradas múltiples. Lo cual se obtiene definiendo claramente cada módulo y repitiendo líneas de código, si ello es necesario, para que cada módulo sea individual.

Para el caso de la contención total la solución resulta clara :

- Hacer que la referencia a un módulo por otro sea explícita. Es decir, que resulte aparente el cambio de control, y que no se produzca porque repentinamente el flujo del sistema "pasa por allí".
- Para lograr el punto anterior, es de gran ayuda que el módulo subordinado no esté contenido materialmente dentro del módulo activador. En caso de que esto suceda, debe separarse escribiendo sus instrucciones en forma de subrutina (o construcción equivalente) independiente.

3.2.6. Conexión de ambiente común.

Esta conexión aparece muchas veces por que existe pereza durante el diseño o durante la programación, pero aunque resulte tedioso, la forma de evitarla es :

- Anotar explícitamente en las referencias intermodulares los parámetros que se manejan, con lo cual disminuye la necesidad de trabajar con áreas comunes.

4. COHESION

4. COHESION.

Se ha visto que reducir el grado de conexión, esencialmente significa disminuir las relaciones que puedan existir entre elementos del sistema que se encuentran situados en módulos distintos. Hasta el momento sólo se ha tratado el aspecto referente a las relaciones intermodulares; en este capítulo se desarrolla el tema de las relaciones intramodulares.

Cuando elementos de un sistema que están situados en módulos distintos llegan a relacionarse, ello puede deberse a que ambos forman parte de una misma función y durante la fase de diseño no fueron situados en el mismo módulo. Además, puede suceder que alguno de esos elementos no tenga relación con los demás elementos del módulo en el cual se encuentra, o que su relación con ellos sea muy somera.

De la misma manera como para las relaciones intermodulares se desarrolló el concepto de conexión, para las relaciones intramodulares se ha desarrollado el concepto de cohesión o firmeza modular. Myers al respecto dice : "La firmeza modular es una medida de las relaciones en el interior de un módulo." Vista la cohesión de esta forma, existe una correlación entre ella y la conexión : en general, el aumentar el grado de cohesión de los módulos de un sistema tiende a disminuir el grado de conexión entre ellos. Lo cual es bastante razonable, pues que el grado de cohesión de un módulo sea elevado significa que se ha

logrado reunir en él elementos que están muy intimamente relacionados entre sí, de tal forma que no será necesario que se comuniquen entre ellos rebasando fronteras modulares; es decir, existe menos conexión entre módulos.

Debe aclararse que la cohesión se refiere a todo el módulo; esto es, a todos los pares de elementos de procesamiento contenidos en el módulo. Entendiendo por elemento de procesamiento no solo las instrucciones, sino todas aquellas acciones que deban efectuarse dentro del módulo, ya sea que estén codificadas o no, e inclusive todas las instrucciones que se encuentren fuera del módulo, pero que se ejecutan por efecto de él (aquí se incluyen las llamadas a otros módulos).

4.1. Grados de cohesión.

Dado que la cohesión es una medida, existen diferentes grados o niveles de ella, los cuales guardan entre sí un orden, a manera de escala. Sin embargo, dicha escala no está elaborada en forma lineal.

En esa escala existen siete grados de cohesión, los cuales son, de menor a mayor :

- Coincidental.
- Lógica.
- Temporal.
- Procesal.
- Comunicacional.
- Secuencial.
- Funcional.

Se dice que esta escala no es lineal debido a que la cohesión temporal resulta casi tan mala como la coincidental, y la comunicacional es casi tan buena como la funcional.

A continuación se analizará cada uno de los grados de cohesión de manera individual, procurando hacer evidente la no linealidad de la escala.

4.1.1. Cohesión coincidental.

Este grado representa el nivel inferior dentro de la escala de la cohesión. El cual aparece cuando existe muy poca relación estructural entre los elementos de un módulo, o ni

siquiera existe. Cuando esto sucede, parece que dichos elementos han sido reunidos en el módulo debido a la casualidad. Y frecuentemente así es; de repente se descubre que en diferentes módulos existen secuencias idénticas o muy parecidas de código, y entonces se decide agruparlas en un solo módulo que podrá ser invocado por todos los módulos que contenían la secuencia. El principal problema que esta agrupación presenta aparece cuando las mencionadas secuencias de código tienen diferente significado para cada uno de los módulos donde originalmente se encontraban. Entonces el nuevo módulo, con cohesión coincidental, si se modifica, en favor de alguno de sus superiores puede ocasionar problemas para los demás, al no considerar la manera en que afectarán esas modificaciones a su secuencia de instrucciones.

Por fortuna, en la práctica no son muchos los módulos con cohesión coincidental que se pueden encontrar, y esto se debe en buena medida a la manera en que son creados, casi por casualidad y sin pensar en ellos como módulos con una función propia bien definida.

4.1.2. Cohesión lógica.

Un módulo posee cohesión lógica cuando ha sido creado para realizar varias operaciones similares o que se pueden considerar relacionadas, como por ejemplo las de impresión, o de cálculo de funciones, o de lectura de registros de distintos

tipos, etcétera. Usualmente el módulo superior selecciona por medio de algún parámetro en la llamada, cuál de las múltiples operaciones que realiza el módulo con cohesión lógica es la que desea que éste efectúe.

En general, la cohesión lógica es más firme que la coincidental; es decir, dá al módulo mayor consistencia, y por ello resulta preferible. En este tipo de cohesión comienza a observarse que los elementos del módulo son ordenados o dirigidos por la estructura del problema a resolver. El principal inconveniente que aquí se presenta es que el módulo no está diseñado para realizar una sola función especifica, lo cual propicia que las distintas funciones compartan instrucciones o elementos, complicando tanto la comprensión de la operación del módulo, como su modificación. La comprensión se dificulta, pues no es posible identificar de manera clara la ubicación de los elementos que conforman cada función particular de las que el módulo efectúa. La modificación se torna complicada, pues al existir elementos que trabajan para varias funciones, el modificar alguno de ellos, adaptándolo a la evolución de una de las funciones, puede alterar seriamente el comportamiento que tenía para las demás.

4.1.3. Cohesión temporal.

La cohesión temporal aparece cuando se agrupan en un solo módulo operaciones que durante la ejecución del sistema ocurren en un mismo período de tiempo. Aquí aparecen principalmente todas aquellas funciones de establecimiento de condiciones iniciales, apertura de archivos, identificación de dispositivos, limpieza de áreas de trabajo y demás acciones que tienen como propósito preparar el entorno para el verdadero procesamiento.

Este tipo de cohesión es más firme que la cohesión lógica. Sin embargo, también presenta algunos problemas, y uno de los más serios y comunes radica en que aunque los elementos con cohesión temporal son ejecutados en el mismo período de procesamiento, no por ello guardan entre sí alguna relación de funcionalidad, entonces si en el código resultan ligados al compartir variables o parámetros, esto complica, como en los casos anteriores, la modificación de cualquiera de ellos, ya que los demás pueden resultar afectados.

4.1.4. Cohesión procesal.

El siguiente nivel de cohesión es el llamado procesal. Los elementos que se encuentran asociados por estar ubicados en la misma unidad procesal se dice que poseen cohesión procesal. Se

entiende por unidad procesal toda aquella construcción del tipo de iteraciones, decisiones, secuencias de instrucciones que se consideran como un todo, etcétera.

Una característica importante de este tipo de cohesión es el orden en que deben ser obedecidas las instrucciones que la poseen. Al respecto tenemos la opinión de Pressman

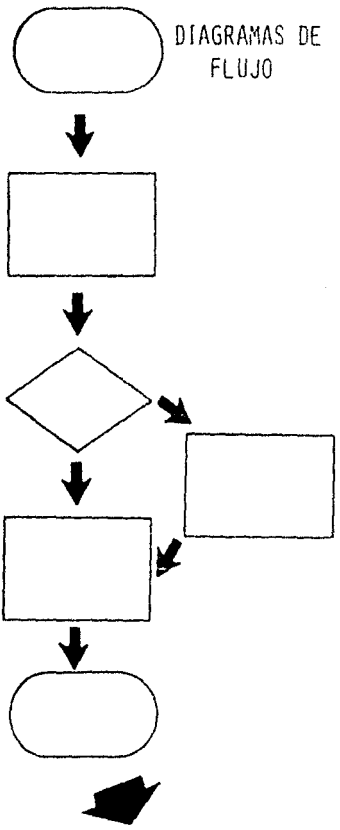
> Cuando los elementos de procesamiento de un módulo están relacionados y deben ejecutarse en un orden específico, allí existe cohesión procesal.

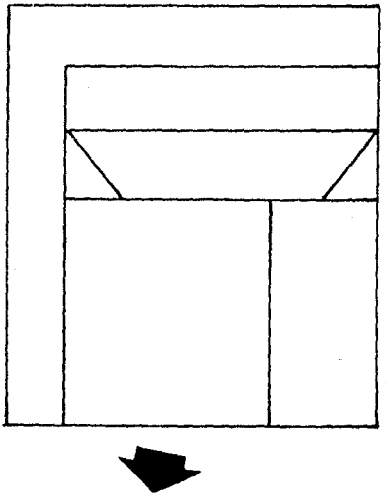
La práctica ha hecho ver que al trasladar a código los diseños plasmados en diagramas de flujo o diagramas de Chapin, es frecuente que se obtengan módulos con cohesión procesal (Fig.4-1).

Aunque pudiera confundirse la cohesión procesal con la cohesión temporal, existe entre ellas una diferencia básica : en la cohesión temporal aunque los elementos de proceso se ejecutan en un período de tiempo definido, entre ellos no existe relación de precedencia, se pueden ejecutar en cualquier orden; mientras que en la cohesión procesal esta relación si existe, aún dentro de la misma unidad procesal.

Se sabe que la cohesión procesal es preferible a la temporal. Sin embargo, presenta un problema principal : que resulta común fragmentar las funciones en diferentes módulos, o

COHESION PROCESAL





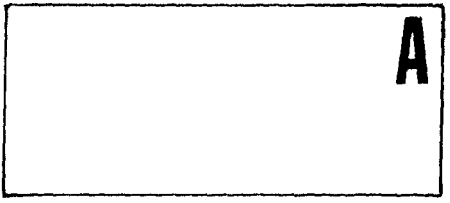


FIGURA 4-1

situar varias funciones en un mismo módulo, mezclando el código de cada una de ellas. Esta situación aleja la forma de solución de la estructura real del problema, lo cual, desde luego, es poco deseable.

4.1.5. Cohesión comunicacional.

Un módulo posee cohesión comunicacional cuando todos sus elementos de procesamiento actúan sobre un mismo conjunto de información, ya sea ésta de entrada o de salida.

Esta clase de módulos padecen el mismo problema señalado para los módulos con cohesión procesal; es decir, se tiende a mezclar en un módulo código correspondiente a varias funciones. Sin embargo, en la cohesión comunicacional aparece una situación que no se tiene en ninguno de los demás niveles de cohesión analizados hasta ahora, y es, precisamente, que los elementos de procesamiento poseen una relación que depende de la forma del problema original. Este hecho es el que hace que la cohesión comunicacional resulte preferible que la cohesión procesal. La cohesión comunicacional resulta menos fuerte cuando la relación entre los elementos se establece tanto en la entrada como en la salida y es más fuerte cuando la relación es solo en una dirección (Fig. 4-2).

COHESION COMUNICACIONAL

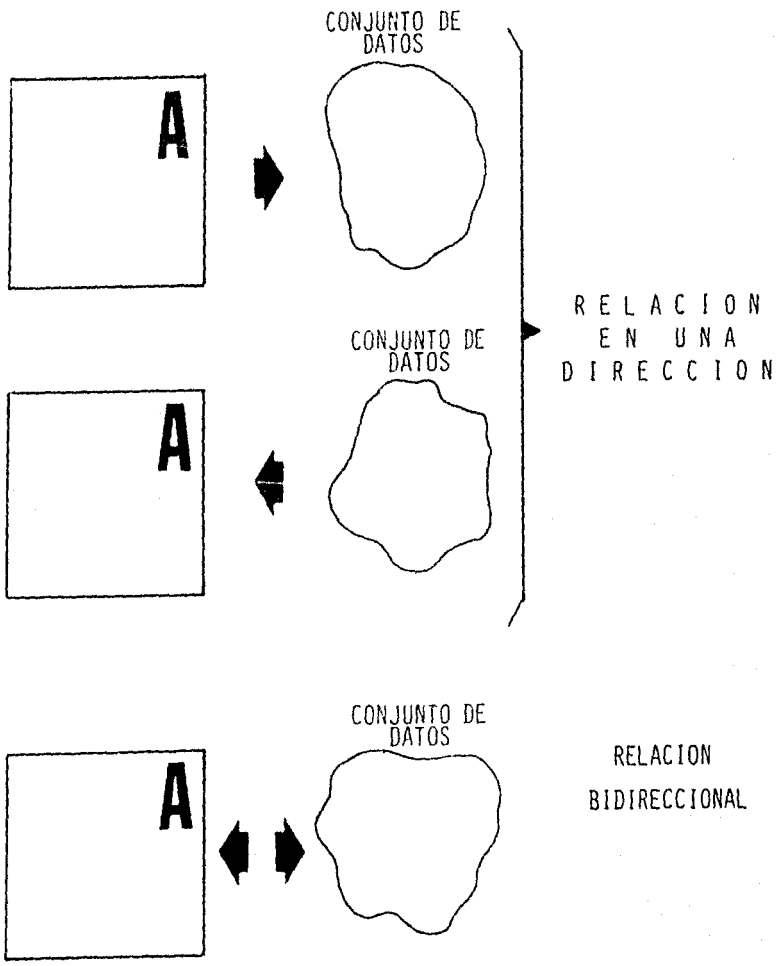


FIGURA 4-2

4.1.6. Cohesion secuencial.

La cohesión secuencial se presenta cuando los resultados generados por un elemento de procesamiento son tomados y utilizados por el siguiente como datos de entrada. En general, esta situación se representa como una cadena lineal de transformaciones sucesivas (secuenciales) del conjunto de datos (Fig. 4-3).

La cohesión secuencial produce menos relaciones dentro de los módulos y éstas son más sencillas que las que se obtienen a partir de la cohesión comunicacional. Este hecho es el que ha llevado a situar en la escala a la cohesión secuencial arriba de la comunicacional.

Aún cuando este tipo de cohesión es buena, no se libra del problema que hemos comentado con todos los niveles anteriores de cohesión : el contener un mismo módulo varias funciones e inclusive que estas funciones se hallen mezcladas. Situación que puede resultar en ciertas complicaciones, principalmente para las modificaciones.

4.1.7. Cohesión funcional.

Finalmente, se ha llegado a la cohesión funcional, que es la cota máxima a que puede aspirar un módulo. En un módulo completamente funcional, cada elemento de procesamiento es

COHESION SECUENCIAL

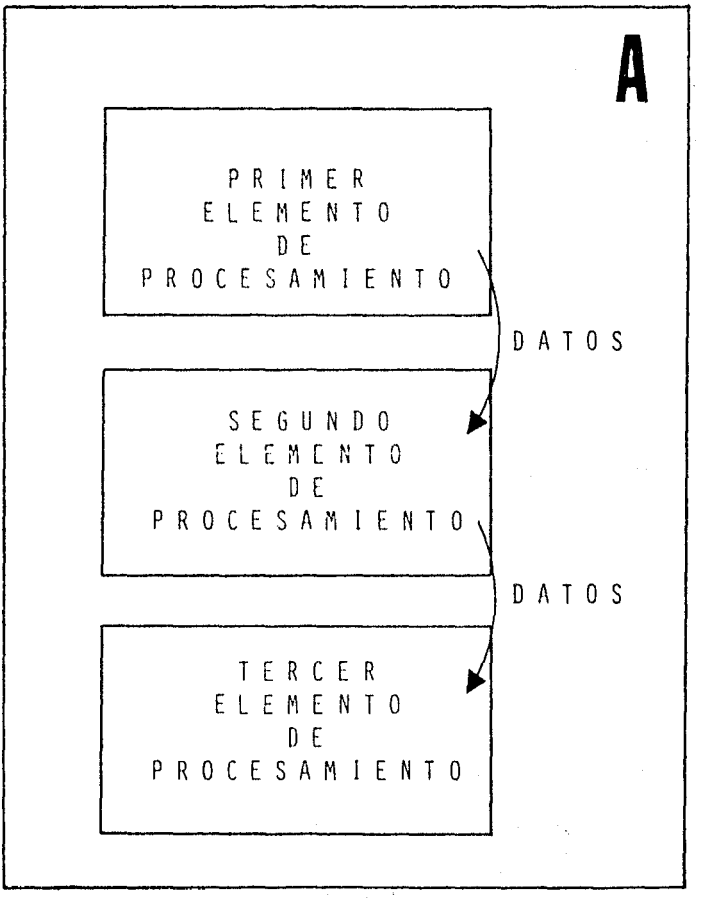


FIGURA 4-3

una parte de, y es esencial para, la realización de una sola función.

En general, resulta difícil identificar los módulos con cohesión funcional. Si se puede hacer ver que un módulo tiene cohesión mejor que coincidental, mejor que lógica, mejor que temporal, etcétera, entonces posee cohesión funcional.

Lo que realmente caracteriza la cohesión funcional es que los módulos que la poseen realizan una sola función. De esta forma queda resuelto el problema que se ha acarreado durante el estudio de todos los demás niveles de cohesión.

Como se dijo, es difícil identificar la cohesión funcional; sin embargo, es sencillo identificar los módulos que no son funcionales. Para ello existe la siguiente receta (Yourdon, Pressman):

I Describase completa y precisamente la función del módulo por medio de una oración.

II Si el módulo posee cohesión funcional, debe ser posible describir su operación en una oración imperativa de estructura sencilla, usualmente con un solo verbo transitivo y un objeto específico no plural.

III Analícese la oración creada y compárese con lo siguiente

a) Si la oración tiene que ser compuesta, contiene alguna coma o contiene más de un verbo, entonces es

probable que el módulo realice más de una función. Es decir, su cohesión no es funcional. Puede que sea secuencial, comunicacional o lógica.

b) Si la oración contiene palabras relacionadas con el tiempo, tales como "primero", "siguiente", "entonces", "después", "cuando", "inicio", "hasta", "para todos", etcétera, entonces el módulo probablemente posee cohesión temporal o procesal; e inclusive dichas palabras pudieran ser indicio de cohesión secuencial.

c) Si el predicado de la oración no contiene después del verbo un objeto único y específico, entonces es probable que el módulo posea cohesión lógica.

d) La existencia de palabras tales como "iniciar", "limpiar", "establecer condiciones", etcétera, dentro de la oración, implican cohesión temporal.

4.2. Medición del nivel de cohesión.

Hasta el momento se han señalado siete puntos en la escala de los niveles de cohesión, sin embargo ello no debe hacernos pensar que dicha escala es discreta; lo correcto es imaginar la escala como continua, y esos siete puntos como representativos de ciertas regiones contiguas.

Por lo anterior es raro encontrar un módulo que posea solo y claramente uno de los niveles mencionados. Resulta más común hallar módulos que posean una mixtura de niveles de cohesión entre sus diferentes elementos de procesamiento.

Para manejar la determinación del nivel de cohesión entre elementos de procesamiento se utiliza la siguiente definición :

> Cuando existe más de una relación entre dos elementos de procesamiento, se considera la de nivel de cohesión mayor.

En el caso de cohesión de módulos decimos que

> La cohesión de un módulo es aproximadamente el nivel mayor de cohesión que es aplicable a todos los elementos de procesamiento dentro del módulo.

4.3. Recomendaciones para incrementar el grado de cohesión de los módulos.

Se han descrito siete grados o niveles que representan los tipos más comunes de cohesión. Ahora, corresponde identificar

las acciones que deben seguirse para lograr sistemas con un nivel de cohesión elevado. Las recomendaciones que aquí se dan son de carácter eminentemente práctico, de manera que puedan aplicarse directamente en la creación de sistemas, y que propicien que éstos últimos resulten de calidad.

El objetivo que siempre deberá tenerse en mente es obtener sistemas cuyos módulos posean en su mayoría cohesión funcional, y en caso de no poseerla, cuando menos que se encuentren cerca de ella.

Con lo dicho en los párrafos anteriores se presentan tres recomendaciones muy directas y que son consideradas de gran importancia :

a) La primera, y más importante, recomendación es que, al igual que se mencionó con respecto a la conexión, no se economice tiempo ni esfuerzo durante la fase de diseño de los sistemas; debe tomarse todo el tiempo que sea necesario para identificar de manera clara todas y cada una de las funciones que contendrá el nuevo sistema, para posteriormente situarlas dentro de éste en los lugares que resulten más apropiados.
b) Otro aspecto de gran importancia es que una vez que se han identificado las distintas funciones que realizará el sistema, se les ubique en módulos individuales, es decir, una sola función por módulo, con lo cual se estará

buscando que posean cohesión funcional.

c) Como tercera recomendación principal, siempre debe lograrse que los resultados que arroja cada elemento de procesamiento dentro del módulo, sean tomados por el siguiente elemento o por algún otro, tan pronto como sea posible; es decir, no debe permitirse que existan resultados obtenidos por algún módulo que se encuentren "flotando" dentro del módulo durante mucho tiempo. Si esto se logra, el módulo poseerá casi seguramente cohesión secuencial, que como se ha mencionado se encuentra en la parte superior de la escala, siendo por ello una de las más firmes.

CONCLUSIONES

C O N C L U S I O N E S.

- Los sistemas poseen distintos tipos de estructuras.

- La estructura de control es la que más se maneja en el ámbito de diseño de sistemas.

- Un sistema monolítico está formado por piezas tan altamente interrelacionadas que se comportan como si se tratara de una sola pieza.

- El diseño de los sistemas monolíticos resulta muy sencillo.

- Es muy difícil identificar las distintas funciones que existen dentro de un sistema monolítico.

- Al modificar un sistema monolítico es común que se presenten efectos laterales.

- La causa principal de la complejidad de un sistema monolítico está representada por el número de situaciones distintas que deben considerarse de manera simultánea.

- Un buen sistema es aquel en el cual puede modificarse cualquier porción lógica sin afectar el resto del sistema, a este sistema se le dá el nombre de "modular".

- Limitar el tamaño de los módulos, por si solo no basta para lograr sistemas modulares.

- La efectividad de una modularización depende del criterio utilizado para dividir el sistema en módulos.

- Un criterio efectivo para lograr la modularización de los sistemas es el "ocultamiento de información".

- La depuración de un sistema modular es más simple que la de uno monolitico, debido a que los módulos pueden probarse individuamente. Lo anterior también se aplica al mantenimiento y modificación.

- La independencia modular se logra con módulos cuya función sea de "un solo propósito" y con poca interacción con otros módulos.

- La complejidad de un sistema se ve afectada no solo por el número de enlaces, sino también por el grado en que cada uno de ellos asocia a los módulos.

Las siguientes acciones ayudan a lograr módulos de baja conexión:

- Toda comunicación entre módulos debe efectuarse a través de parámetros.

- La entrada a los módulos siempre debe ser por su encabezado.

- Deben evitarse los regresos de control múltiples.

- Siempre respetar los niveles jerárquicos.

- Las líneas de comunicación deben ser lo más transparentes que sea posible, y el número de parámetros tan pequeño como lo permita cada aplicación particular.

- Diseñar los módulos de manera que entre ellos solo se transmita información ,que se pueda considerar como datos.

- Evitar que algún módulo sea responsable del funcionamiento interno de otro, o que modifique sus instrucciones.

- Establecer las referencias intermodulares tan tarde como sea posible.

- Evitar los módulos con entradas múltiples.

- Hacer que toda referencia a un módulo por otro, sea siempre explícita.

- Anotar explícitamente los parámetros que se manejan en las referencias intermodulares.

La cohesión se incrementa si :

- Se identifican claramente todas las funciones del sistema y son situadas en lugares apropiados.

- Cada función es ubicada en un módulo individual.

- No se permite los resultados obtenidos por algún módulo se encuentren "flotando" dentro de él por mucho tiempo.

BIBLIOGRAFIA

B I B L I O G R A F I A.

Aron, J. D.
S. F.
"The Program Development Process; Part 1: The Individual Programmer"; Addison-Wesley 264 Págs.

Brooks, Frederick P. Jr.
1975
"The Mythical Man-Month, Essays on Software Enginnering", Addison-Wesley, 195 Págs.

Chand, D. R./
S. B. Yadav
1980
"Logical Construction of Software", Communications of the ACM, Vol. 23, No. 10, Págs. 564-555.

Chapin, Ned
1974
"New Format for Flowchart", Software-Practice and Experience, Vol. 4, Págs. 341-357.

Chapin, Ned
1979
"Flowcharting With the ANSI Standard : A Tutorial", Computing Surveys, Vol. 2, No. 2, Págs. 119-146.

Dahl, O. J./
E. W. Dijkstra/
C. A. R. Hoare
1976
"Programación Estructurada", Ed. Tiempo Contemporáneo.

Kernighan, B. W./
P. J. Plauger
1978
"The Elements of Programming Style", McGraw-Hill, 168 Págs.

Kernighan, B. W./
P. J. Plauger
1976
"Software Tools", Addison-Wesley, 338 Págs.

Hartman, W./
H. Matthes/
A. Proeme
1984
"Manual de los Sistemas de Información", Ed. Paraninfo, 382 Págs.

Hice, G. F./
W. S. Turner/
L. F. Cashwell
1974
"System Development Methodology",
Pandata B. V., 370 Págs.

Maurer, H. A./
M. R. Williams
1972
"A Collection of Programming Problems and
Techniques", Prentice-Hall, 256 Págs.

Meek, B. L./ P. M. Heat/
N. J. Rushby
1982
"Guide to Good Programming Practice",
Ellis horwood, 192 Págs.

Myers, Glenford J.
S. F.
"Software Reliability. Principles and
Practices", John Wiley and Sons, 360 Págs.

Parnas D. L.
1972a
"A Technique for Software Module
Specifications with Examples",
Communications of the ACM, Vol. 15, No. 5,
Págs. 330-336.

Parnas D. L.
1972b
"On the Criteria To Be Used in Decomposing
Systems into Modules", Communications of
the ACM, Vol. 15, No. 12, Págs. 1053-1058.

Peters, L. J./
L. L. Tripp
1977
"Comparing Software Design Methodologies",
Datamation, Noviembre, 1977.

Pressman, Roger S.
1982
"Software Engineering : A Practitioner's
Approach", McGraw-Hill, 352 Págs.

Stay, J. F.
1976
HIPO and Integrated Progran Design",
IBM Systems Journal, No. 2, Págs. 143-154.

Stevens, W. P./
G. J. Myers/
L. L. Constantine
1974
"Structured Design",IBM Systems Journal,
1972, No. 2, Págs. 115-139.

Stone, Harold S./
Daniel P. Sewiorek
1975
"Introduction to Computer Organization and
Data Structures : PDP-11 Edition",
McGraw-Hill, 368 Págs.

Van Tassel, Dennie
1978
"Program Style, Design, Efficiency, Debugging and Testing", Prentice-Hall, 323 Pàgs.

Yourdon, E.
S. F.
"Techniques of Program Structure and Design", Prentice-Hall, 364 Pàgs.

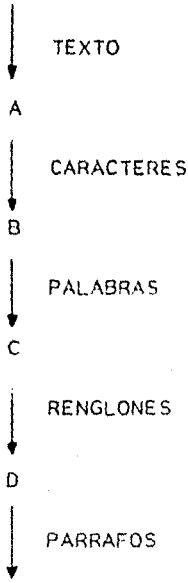
Yourdon E./
L. L. Constantine
1979
"Structured Design : Fundamentals of a Discipline of Computer Program and Systems Design", Prentice-Hall 473 Pàgs.

A N E X O I

E J E M P L O S

CONEXION DE ENTRADA Y SALIDA.

En el procesamiento o análisis de textos, éstos son descompuestos paso a paso hasta identificar algún constituyente básico (párrafo, renglón, palabra o carácter). Los módulos que trabajan uno tras otro, manejando, cada uno de ellos, componentes de distinto nivel del texto, representan un ejemplo de conexión de entrada y salida.



CONEXION DE CONTROL

Considérense los siguientes módulos A y B:

MODULO A

INICIA A

_____

_____

_____

(1) EJECUTA B (Parámetro: X)

_____

_____

_____

TERMINA A

MODULO B

INICIA B

_____

_____

(2) Si X = 0, entonces

_____

_____

OTRO CASO

_____

_____

FIN SI

TERMINA B

En este caso, el módulo A activa al módulo B (1), incluyendo en la transferencia de control el parámetro "X". La ejecución de B depende del valor del parámetro recibido "X" (2). Por lo anterior, A dirige el comportamiento interno de B a través de "X", lo cual genera la conexión de control.

CONEXION HIBRIDA.

Considérense los módulos A y B siguientes:

MODULO "A"

INICIA A

_____

_____

_____

(1) Si condición1 entonces

(2) instruccion = imprime en papel.

En otro caso

(3) instrucción = despliega en pantalla.

Fin si

(4) Carga instrucción, dirección1

(5) EJECUTA B

TERMINA A

MODULO B

INICIA B

_____

_____

_____

dirección1 : instrucción nula; espacio reservado
para ser llenado por el Módulo A.

TERMINA B

El módulo A de acuerdo a la condición1(1), (2) selecciona si el módulo B imprimirá en papel sus resultados, o (3) los desplegará en la plantalla; después (4) modifica el código de B con la función seleccionada, para finalmente (5) transferirle el control. Así, B ha sufrido modificaciones en su contenido procesal (código), efectuadas por su módulo invocador. Por lo tanto, nos encontramos ante dos módulos con conexión híbrida.

CONEXION DE CONTENIDO TOTAL (SIMPLE).

Considérense los Módulo A y B siguientes:

MODULO A

Inicia A /Calcula volúmenes de prismas triangulares o cuadrangulares./

Variables: tipo, /tipo de prisma/

b, h, /base y altura de tapas del prisma./

a, /área de la base del prisma/

H, /altura del prisma./

V, /volumen del prisma./

Lee tipo, B, h, H

MODULO B Calcula el área de la base del prisma.

SI tipo = triangular, entonces

a = (b*h)/2

SI NO

a = b*h

FIN SI

```
TERMINA B

/Cálculo del volumen/

    V = a ^ H

TERMINA A
```

Aquí, el Módulo B se encuentra totalmente contenido dentro del Módulo A.

CONEXION DE AMBIENTE COMUN.

Considérense los Módulos A, B y C siguientes:

MODULO A

INICIA A

Variables comunes: x, y, u, v

_____

_____

_____

x = x + 1

Ejecuta B

_____

_____

y = y * 2

Ejecuta C

_____

_____

_____

TERMINA A

MODULO B

INICIA B

Variables comunes: x, u

```
Si x es par, entonces
        u = x/2
Si no
        u = (x + 1)/2
FIN SI
TERMINA B
```

MODULO C

```
INICIA C
Variables comunes: y, v
_____
_____
 y = y ** 2
_____
_____
TERMINA C
```

En este caso, los módulos B y C que de otra forma serían independientes, resultan conectados por el ambiente común a que ambos tienen acceso, aunque no compartan en él ninguna variable.

COHESION COINCIDENTAL.

Considérese el siguiente código:

(a) Poner títulos en pantalla.

(b) Situar cursor en renglón 5.

(c) Borrar el resto de la pantalla.

(d) Preparar tabla de colores, fondo azul.

(e) Dibujar mapa en la pantalla.

_____

_____

_____

(r) Desplegar opciones.

(s) Ubicar cursor en renglón 23.

(t) Borrar el resto de la pantalla.

(u) Preparar tabla de colores, fondo rojo.

(v) Leer opción seleccionada.

De inmediato se distinguen dos secuencias de instrucciones muy parecidas (b) - (d) y (s) - (u) y se decide modificar la programación como sigue:

Poner títulos en pantalla.

Ejecutar bd-su (5, azul).

Dibujar mapa en la pantalla.

Desplegar opciones.

Ejecutar bd-su (23, rojo).

Leer opción seleccionada.

Módulo bd-su (r,c)

Inicia bd-su

Ubicar cursor en el renglón r.

Borrar el resto de la pantalla.

preparar tabla de colores, fondo c.

Termina.

El módulo bd-su resultante posee cohesión coincidental debido a que las instrucciones que lo constituyen no tienen un significado concreto, ni el módulo tiene una función determinada; sino que fue creado para evitar la repetición de instrucciones.

COHESION LOGICA.

El siguiente módulo posee cohesión lógica, ya que realiza varias funciones similares.

```
Módulo ERROR (numerr)

Inicia

Caso Numerr = 1
     despliega mensaje 1
     desconecta archivos
     reinicia sistema

Caso Numerr = 2
     despliega mensaje 2
     borra último registro grabado

Caso Numerr = 3
     despliega mensaje 3
     detiene ejecución

Fin caso

Termina error
```

COHESION TEMPORAL.

El módulo COND-INIC posee cohesión temporal porque la -- única relación que guardan todas sus operaciones es la de ocurrir - en un mismo periodo de tiempo.

Módulo COND-INIC.

/Establece las condiciones iniciales de trabajo del sistema./

INICIA

Abrir archivo A.

Abrir archivo B.

Establecer enlace con impresora.

Limpiar área común C,

Leer fecha actual.

TERMINA.

CONEXION PROCESAL.

Considérese el siguiente fragmento de código:

```
_____________

_____________

_____________

Leer archivo de datos ARCHD

Ordenar entrada : ARCHD, salida: ARCH-O

Imprimir  ARCH-O

Borrar    ARCH-O

_____________

_____________

_____________
```

Este fragmento posee cohesión procesal, debido a que sus elementos mantienen una relación de precedencia. Nótese cómo en el ejemplo de la cohesión temporal el orden de las instrucciones resulta irrelevante, mientras que en éste es fundamental.

COHESION COMUNICACIONAL.

El módulo INTEGRA se encarga de armar cada registro de salida a partir de varios catálogos o archivos. Toda esta función lo define como un módulo con cohesión comunicacional.

Módulo INTEGRA (RFC)

Inicia

Obtener el registro del trabajador según RFC

Obtener salario diario según puesto del trabajador.

Calcular pago mensual.

Calcular impuesto.

Armar registro de salida con RFC, nombre, -- pago mensual e impuesto.

Termina

EFECTOS LATERALES.

Considérese un módulo encargado del calendario trimestral de pagos en un sistema de presupuesto. Dicho calendario tiene la siguiente estructura:

	Ene	Feb	Mar	T o t a l
Concepto 1	$	$	$	$
.				
.				
.				
Concepto N	$	$	$	$
T o t a l	$	$	$	$

Debido a los altos índices inflacionarios, las cifras manejadas son cada vez mayores; por lo tanto, se toma la decisión de manejar las cifras de este calendario a miles de pesos. Para ilustrar esta situación, se presenta una línea de dicho calendario antes y después de la modificación.

	Ene	Feb.	Mar.	T o t a l
Antes : Concepto	733,550	482,700	621,600	1'837,850
Despúes: Concepto	734	483	622	1'838

Nótese que la suma total manejada a miles de pesos se ob-- tiene $ 1'838 de la cantidad $ 1'837,850, mientras que de la suma - de las cantidades $ 734, $ 483 y $ 622 resulta $ 1'839. Esta dife-- rencia provocada involuntariamente, debido a ciertas modificaciones al sistema, representa un caso de efecto lateral.