DISEÑO ESTRUCTURADO

CONTENIDO

1. DISEÑO ESTRUCTURADO.

   Definición y Principios de Diseño
   Inicios del Diseño
   Efectividad del Diseño
      Modularidad
      Abstracción
      Refinamiento
   Factores de calidad: Acoplamiento y Cohesión
   Diseño de Calidad
   Resultados del Diseño
   Diagrama de Estructura

   Estrategias de Diseño

   Construcción del Diagrama de Estructura

CONTENIDO

2. DISEÑO ORIENTADO A OBJETOS.

   Diagramas UML

1. DISEÑO ESTRUCTURADO

DEFINICION Y PRINCIPIOS DE DISEÑO

- El Diseño es un proceso a través del cual los requerimientos establecidos en la fase de análisis deben traducirse en una representación modular del producto de software a construir.

- Cada módulo se acompaña de los procedimientos de acuerdo a los cuales debe llevar a cabo su tarea, y de las estructuras de datos que procesa.

DEFINICION Y PRINCIPIOS DE DISEÑO

- El Diseño Estructurado es un método de configuración de la organización modular del software que se desarrolla a partir de los Flujos de Datos que contiene la especificación de requerimientos obtenida en la fase de análisis bajo un enfoque de desarrollo estructurado (paradigma de desarrollo).
- Entonces, su punto de partida es la Especificación Estructurada.
- Puede decirse que consiste en el diseño de programas como estructuras de funciones únicas y de relativa independencia.

DISEÑO

El Diseño de un Sistema de Información corresponde a la fase que continúa al Análisis, y consiste principalmente en:

- la especificación de la estructura o arquitectura del producto de software a construir.
- la especificación de los procedimientos en virtud de los cuales cada componente estructural debe realizar su tarea.
- la especificación de las estructuras de datos, la cual tiene que responder a los requerimientos de tratamiento automatizado de datos y generación de información planteados en la especificación de requerimientos generada en la fase de análisis.

DISEÑO MODULAR EFECTIVO

Para poder evaluar la efectividad de una representación de diseño, es preciso establecer lo que se denomina en Ingeniería de Software, los CRITERIOS PARA UN BUEN DISEÑO:

• El Diseño debe exhibir una organización jerárquica con mecanismos de control que no atenten contra la independencia relativa de cada componente de la jerarquía.

DISEÑO MODULAR EFECTIVO

- El diseño debe ser modular, esto es, el software debe estar particionado lógicamente en elementos que ejecuten funciones y subfunciones específicas.

- El diseño debe generar módulos que exhiban niveles adecuados de independencia funcional.

- El diseño debe obtenerse a partir de la especificación de requerimientos generada durante la fase de análisis.

DISEÑO MODULAR EFECTIVO

Conceptos fundamentales que se han validado a lo largo del tiempo:

Modularidad

Abstracción

Refinamiento

MODULARIDAD

• Módulo: cada una de las unidades claramente definidas y manejables constituyentes del software.

• La modularidad consiste en el particionamiento del software en elementos con nombres y direcciones separadas que se denominan módulos, los cuales en su composición generan una totalidad que debe ser capaz de resolver el problema que da origen a la necesidad de construir un producto de software.

• Tiene que ver con la división de las funciones que en conjunto cumplen un objetivo mayor, esto es, responden a la idea de totalidades emergentes propia de la noción de sistemas.

MODULARIDAD

Beneficios de la Modularidad

- Programas más simples, ya que puede ser comprendido, verificado, programado, depurado, mejorado y alterado por partes.
- Módulos que pueden ser desarrollados con relativa independencia.
- Disminución de la posibilidad de errores al reducir la complejidad.

MODULARIDAD

Beneficios de la Modularidad

- Programas que pueden evaluarse por partes, por lo cual todo test se hace más fácil.

- Programas más fáciles de alterar ya que son menores las líneas de código a considerar para incorporar los cambios.

- Módulos de función única que pueden ser reutilizados.

MODULARIDAD

Beneficios de la Modularidad

- El programa puede ser comprendido por partes.
- Disminuye errores de programación. Son menos las líneas de código que deben enfrentar al mismo tiempo los programadores.
- Los efectos colaterales de los cambios que afectan al sistema son drásticamente reducidos.
- Rotación de personal menos crítica, ya que los programadores están involucrados en unidades de código más pequeñas por lo cual la sustitución resulta menos dificultosa.
- Responde al requerimiento de la división del código en segmentos de una página, como lo sugiere la programación estructurada.

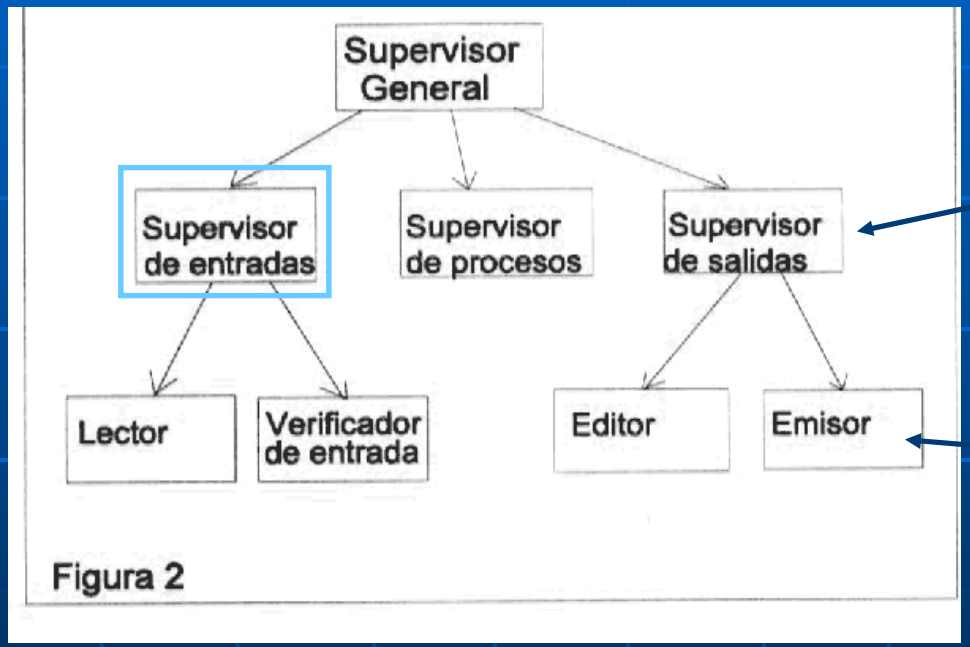
MODULARIDAD

El FAN OUT es una medida del número de módulos controlados directamente por otro módulo (número de subordinados inmediatos que posee).

El FAN IN indica cuántos módulos controlan directamente un determinado módulo (número de superiores inmediatos que posee).

Un módulo que controla a otro se dice que es "superordinado" a éste y, recíprocamente, un módulo controlado por otro se dice que es "subordinado".

MODULARIDAD



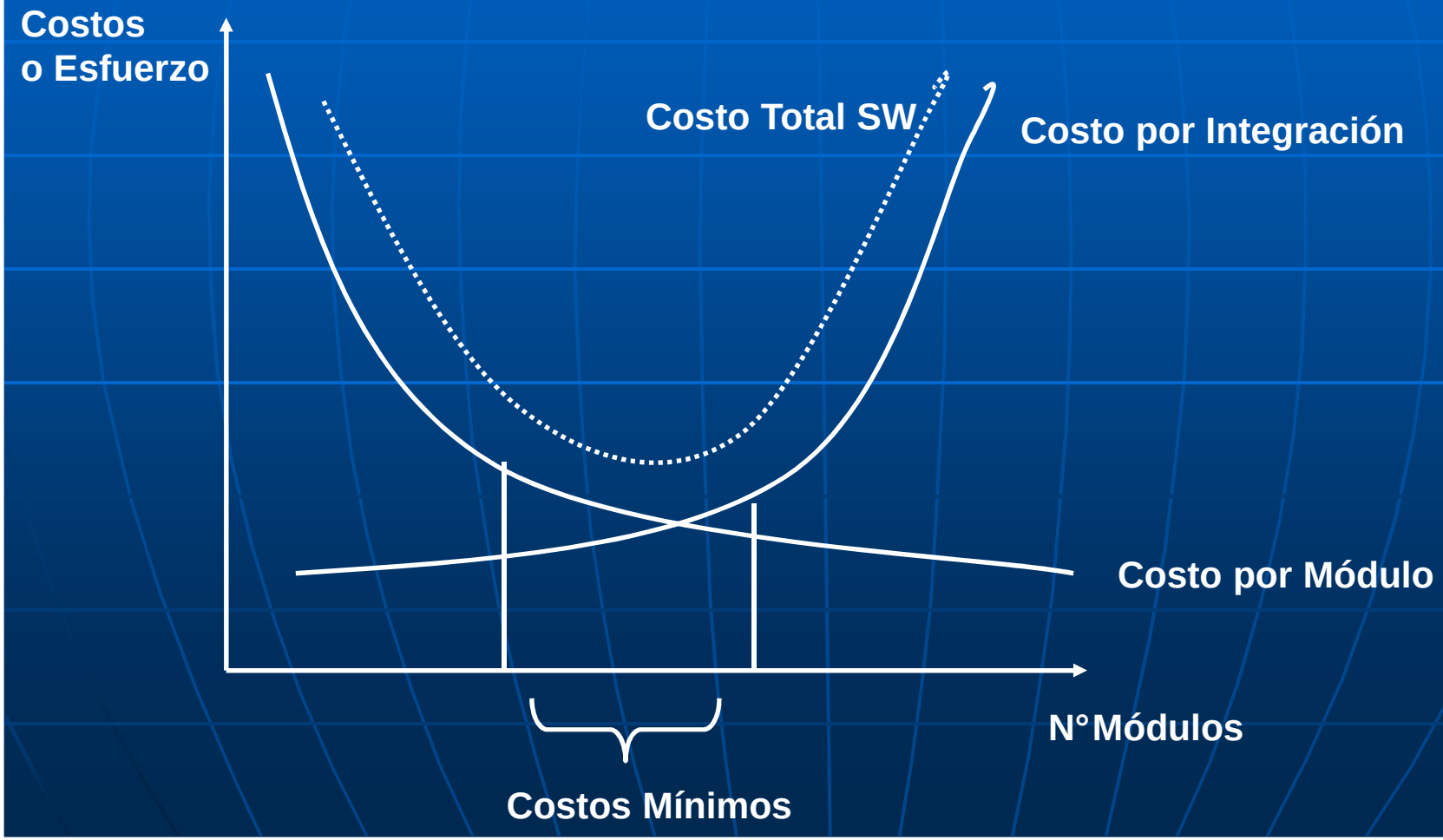
Módulo Superordinado

Módulo Subordinado

FAN OUT: 2 FAN IN : 1

MODULARIDAD

¿ Dividir hasta el infinito para que el Esfuerzo sea Cero ?



ABSTRACCION

Cuando se considera una solución modular para enfrentar un problema, se puede plantear en distintos niveles de abstracción.

Un nivel superior de Abstracción supone una solución en términos amplios, usando un lenguaje del entorno del problema.

A niveles más bajos, se toma una orientación más procedimental, se combina una terminología orientada al problema con una orientada a la implementación.

El nivel más bajo de abstracción permitirá que la solución pueda implementarse directamente.

REFINAMIENTO

El refinamiento sucesivo es una estrategia de diseño descendente (Top_Down) propuesta de Niklaus Wirth. Quien postuló que ”la arquitectura de un programa se desarrolla refinando sucesivamente los niveles de detalle de los procedimientos, lográndose una jerarquía de procedimientos al descomponer sucesivamente una sentencia global hasta alcanzar sentencias específicas a nivel de un lenguaje de programación ".

En “Ingeniería del Software: Un Enfoque Práctico” de Roger S. Pressman, se puede leer: “en cada etapa del refinamiento, se descomponen una o varias de las instrucciones del programa dado en instrucciones cada vez más detalladas. Esta descomposición o refinamiento sucesivo termina cuando todas las intrucciones están expresadas en términos de cualquier lenguaje básico de computador o de programación”.

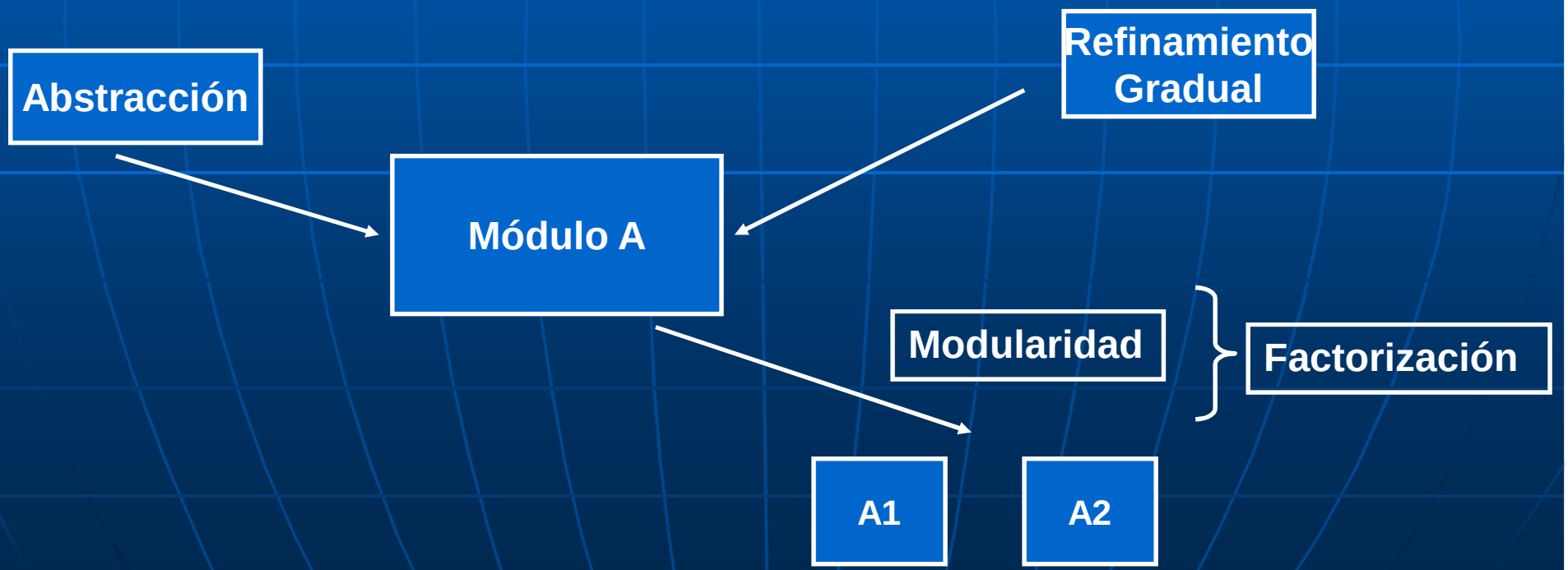
REFINAMIENTO

Proceso iterativo mediante el cual la arquitectura del software es el reflejo de la especificación de requerimientos del sistema, el cual se realiza sistemáticamente con el propósito de lograr una estructura modular, es decir, una representación que explicite las relaciones entre los principales elementos del software.

Esto significa que, inicialmente, se configura una representación que muestra una visión global tanto de las estructuras de datos como de las estructuras de los componentes, la cual tras un proceso de refinamiento sucesivo se transforma en una representación del diseño muy cercana al código fuente, en cuanto a los detalles de los procedimientos que han de regir las transformaciones que deben llevarse a cabo en cada módulo y en estructuras de datos detalladas.

REFINAMIENTO

En la Ingeniería de Software, la modularización se apoya en lo que se conoce como refinamiento sucesivo o gradual, para la configuración de la estructura del software.



FACTORES DE CALIDAD: ACOPLAMIENTO

Corresponde al grado de independencia entre dos módulos.

Minimizar el acoplamiento aparece entonces como un objetivo al configurar la estructura.

La obtención de módulos tan independientes como sea posible,se puede ser lograr principalmente de tres maneras:

- Eliminando relaciones innecesarias.
- Reduciendo el número de relaciones necesarias.
- Debilitando la dependencia de las relaciones necesarias.

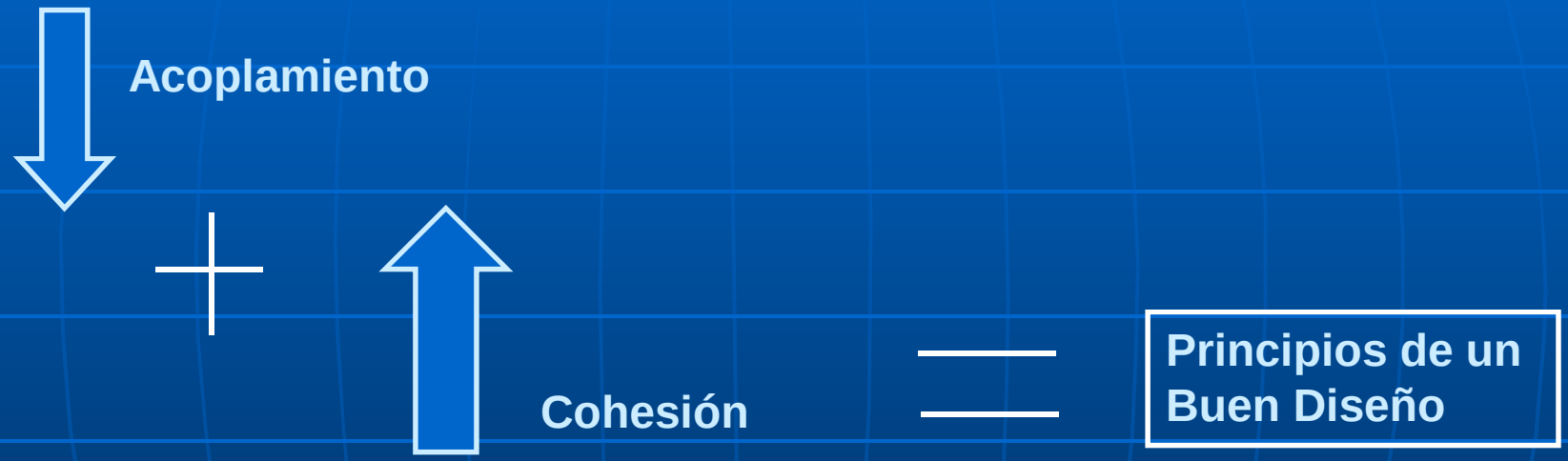
FACTORES DE CALIDAD: COHESION

Corresponde a la medida de relación funcional de los elementos en un módulo.

Los elementos de un módulo corresponden a instrucciones, definiciones de datos, o llamadas o otros módulos.

La idea es organizar estos elementos de tal manera que tengan una mayor relación entre ellos al momento de cumplir su tarea.

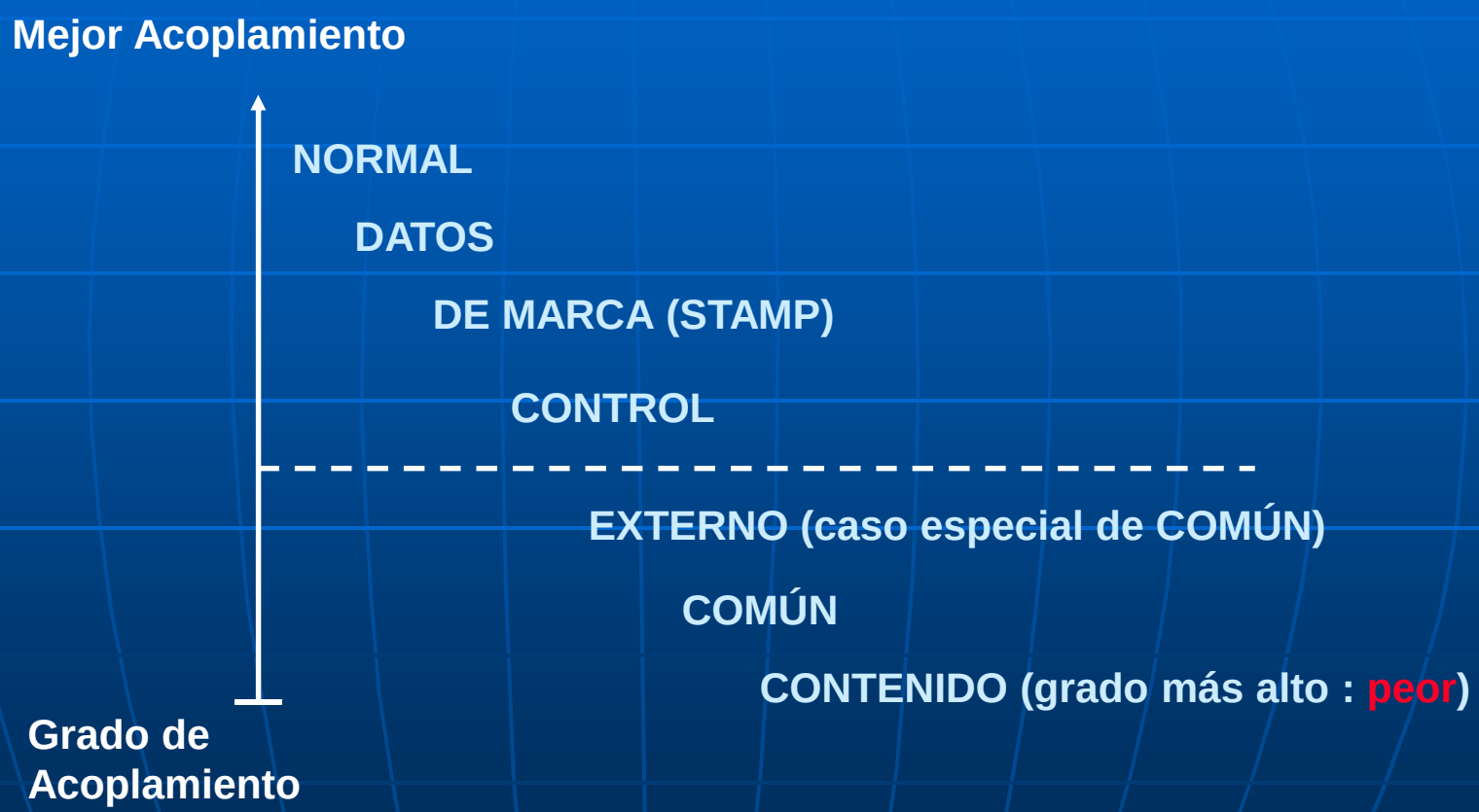
FACTORES DE CALIDAD: ACOPLAMIENTO Y COHESION



Tipos de Acoplamiento

1. **Acoplamiento Normal**
2. **Acoplamiento de Datos**
3. **Acoplamiento de Marca (Stamp)**
4. **Acoplamiento de Control**
5. **Acoplamiento Común**
6. **Acopalmiento Externo**
7. **Acoplamiento de Contenido**

Tipos de Acoplamiento

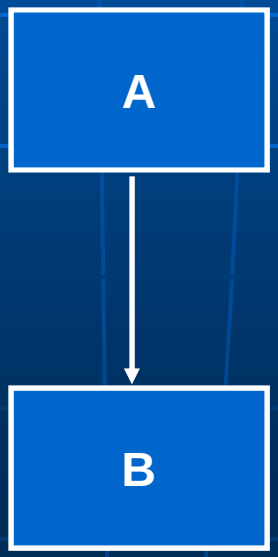


Acoplamiento Normal

Dos Módulo A y B están Normalmente Acoplados si:

- Un Módulo A llama a otro B
- B retorna el control a A

No se produce traspaso de parámetros entre ellos, sólo existe la llamada de uno a otro.

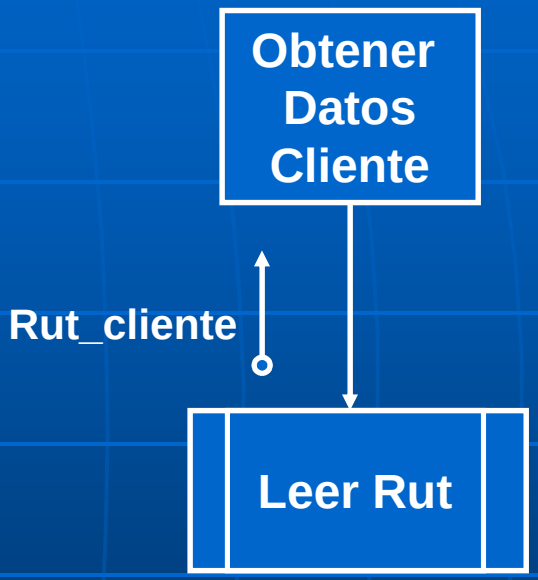


Acoplamiento de Datos

Dos módulos están acoplados por datos si ellos se comunican por parámetros.

Siendo cada parámetro una unidad elemental de datos

El acoplamiento por datos corresponde a la comunicación de datos necesaria entre módulos. Toda vez que los módulos tienen que comunicarse entre sí, la ligazón por datos es inevitable y serán adecuadas si se mantienen a niveles mínimos.

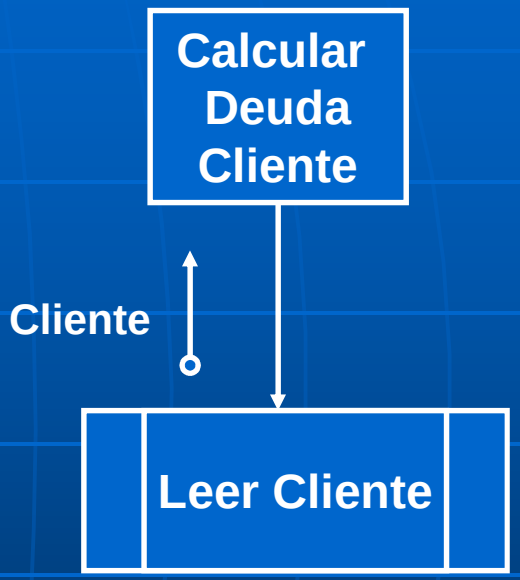


Acoplamiento de Marca (Stamp)

Dos módulos aparecen "acoplados de marca" si ellos se refieren a la misma estructura datos local.

Por estructura de datos se debe entender un grupo compuesto de datos en vez de argumentos simples.

Por ejemplo un Registro.



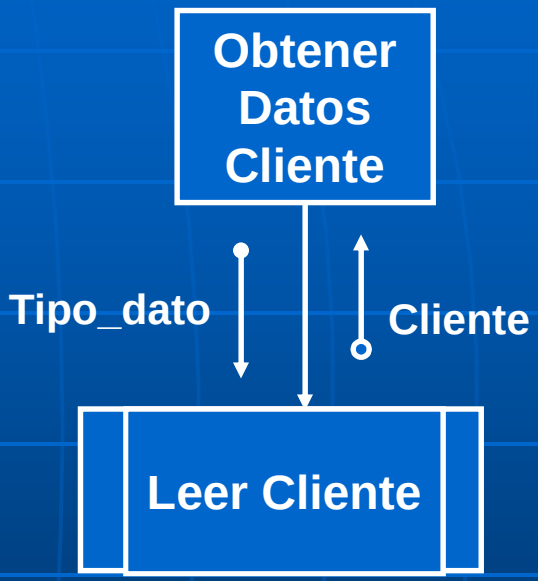
Cliente= rut+nombres+apellido_paterno+ apellido_materno+dirección+fono+e_mail

Acoplamiento de Control

Dos módulos están acoplados por control cuando uno de ellos pasa al otro módulo indicadores de control (flag, switch).

Provoca dependencia de ejecución entre un módulo y otro.

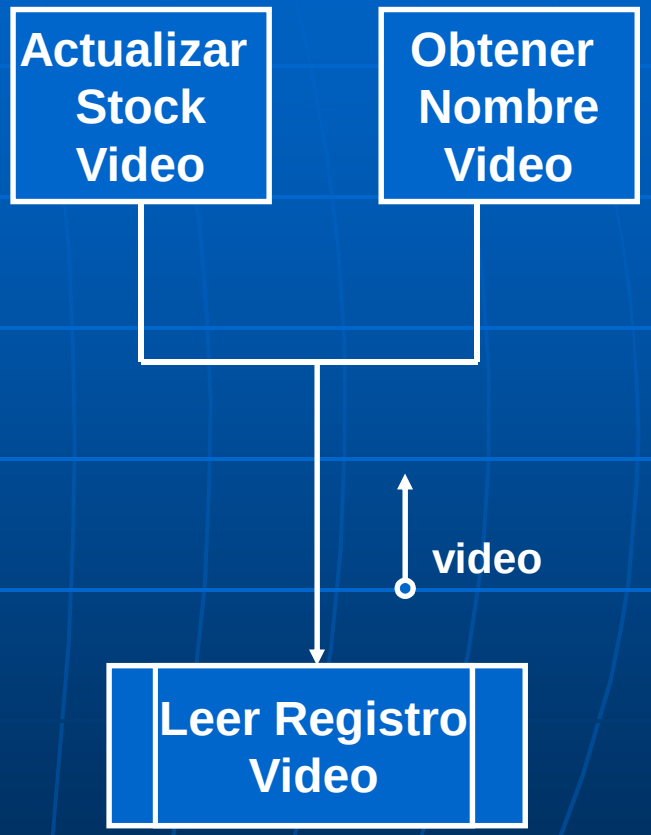
No es muy recomendable. Tratar de utilizarlo moderadamente.



Acoplamiento Común o Global

Dos módulos presentan acoplamiento común, si ellos se refieren a la misma área global de datos (archivo o área de memoria).

Programas con muchos datos globales son difíciles de entender por los programadores de mantención, porque no es fácil saber cuáles son los datos usados por un cierto módulo.

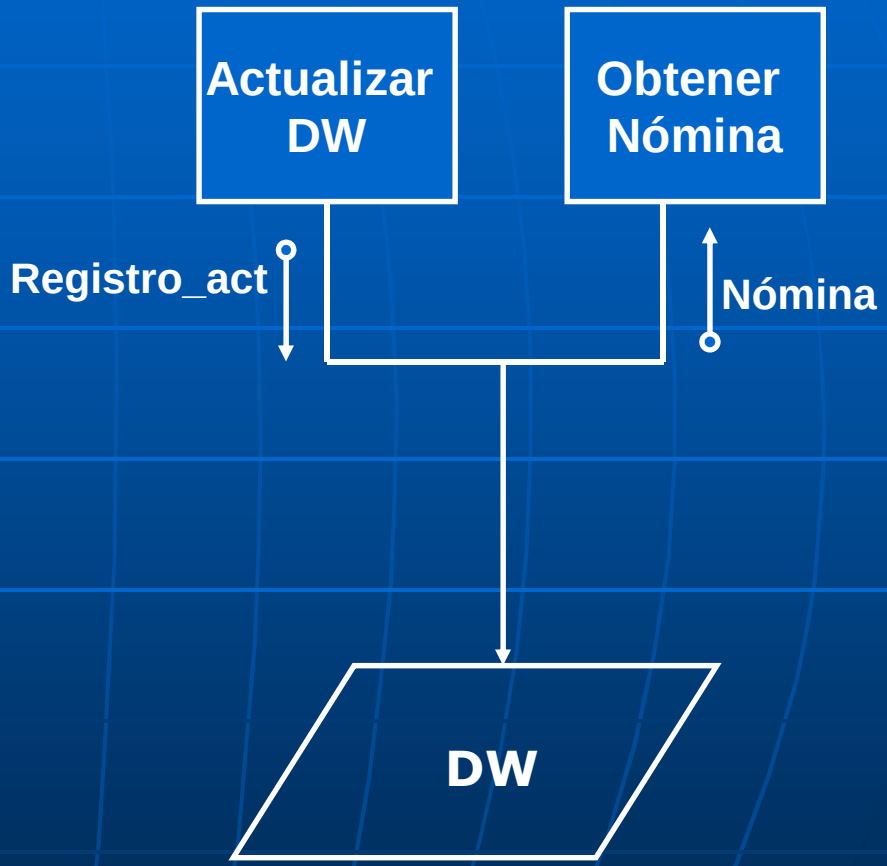


Acoplamiento Externo

Cuando los módulos están atados a un entorno externo al software se dan niveles relativamente altos de acoplamiento.

Por ejemplo, la E/S acopla un módulo a dispositivos, formatos y protocolos de comunicación.

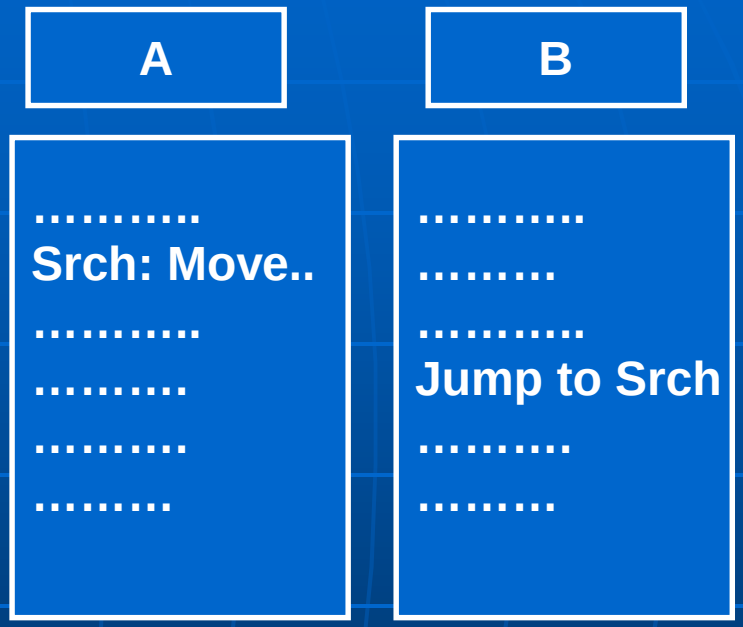
El acoplamiento externo debe limitarse a unos pocos módulos en la estructura.



Acoplamiento de Contenido

Este es un tipo de Acoplamiento indeseable.

Dos módulos presentan acoplamiento de contenido (o patológico) si uno hace referencia al interior del otro. Esto ocurre si por ejemplo, en un módulo se desvía la secuencia de instrucciones al interior de otro o si un módulo altera un comando de otro.



Tal acoplamiento rompe el concepto de módulos configurados bajo el criterio de la caja negra. Forzando a un módulo conocer explícitamente los contenidos y la implementación de otro.

Tipos de Acoplamiento

Dos módulos pueden estar relacionados por más de un tipo de acoplamiento. Si esto ocurre, el acoplamiento que caracteriza la relación entre ellos queda definido por el peor tipo que presenten.

Por ejemplo, si dos módulos están ligados por acoplamiento de marca y acoplamiento común a la vez, se dirá que los módulos están ligados por acoplamiento común.

¿Cómo Analizar el Tipo de Acoplamiento?

Imaginar el Módulo como una Biblioteca

Cada Módulo es codificado por un programador diferente

“Imaginar el Módulo como una Biblioteca”.

- Cómo el módulo podría ser más fácilmente entendido.
- Cómo el módulo podría ser más utilizado por otros sistemas o invocado por otros módulos del mismo sistema

“Cada Módulo es codificado por un programador diferente”.

- ¿Qué tan independientes pueden trabajar los programadores?
- ¿Existe algún supuesto, convención o decisiones de implementación a los cuales más de un módulo deba prestar atención?
- ¿Cuáles son las posibilidades de cambio que existen en relación a los supuestos, convenciones o a la implementación?
- ¿Existe alguna manera de aislar aquellos cambios y situarlos en un sólo módulo?

Ideas Centrales

- Sistemas altamente acoplados conducen a depurar verdaderas pesadillas. Evítelos
- Sistemas altamente acoplados, tienden a tratarse como una sola gran unidad. Y un sistema monolítico es la contrapartida a particionar.
- Hablar de módulos, cajas negras, es hablar de particionamiento.
- Particionar es la estrategia para abordar la complejidad.
- Las cajas negras se organizan jerárquicamente.

Tipos de Cohesión

Módulo como Caja Negra

Módulo Transparente

STEVEN, MYERS, CONSTANTINE y YOURDON (1974) establecieron "una escala de cohesión"

Cohesión Funcional

Un módulo con cohesión funcional es aquel que contiene elementos que contribuyen a la ejecución de una y sólo una tarea relacionada al problema objeto de diseño.

Ejemplos:

Calcular el coseno de un ángulo

Calcular el I.V.A. de una factura

Verificar el dígito de un RUT

Cohesión Secuencial

Un módulo secuencialmente cohesionado es aquel cuyos elementos están envueltos en actividades tales que los datos de salida de una actividad en general sirven como datos de entrada para la próxima actividad.

Ejemplo: Calcular Salario

1. Obtener sueldo base
2. Verificar número de cargas
3. Revisar días con permiso
4. Revisar días con licencia
5. Calcular horas de trabajo
6. Descontar horas de atraso
7. Agregar horas extras

....

Cohesión Comunicacional

Un módulo presenta cohesión comunicacional cuando sus elementos contribuyen a actividades que usan la misma entrada o la misma salida. No importa el orden secuencial

Ejemplo: Obtener datos Producto

1. Obtener nombre
2. Obtener stock
3. Obtener ubicación
4. Obtener precio

....

Cohesión Procedimental

Cuando sus elementos de procesamiento están relacionados y deben ejecutarse en un orden específico.

Cohesión Temporal

Un módulo con cohesión temporal es aquel cuyos elementos están envueltos en actividades que están relacionadas en función del momento en que se realizan.

Ejemplo: Actividades al iniciar el día

1. Apagar despertador
2. Tomar una ducha
3. Vestirse
4. Hacer la cama
5. Tomar desayuno

....

Cohesión Lógica

Un módulo tiene cohesión lógica, cuando existe alguna relación entre los elementos del módulo, contribuyendo al desarrollo de actividades de una misma categoría general, donde la actividad o las actividades a ser ejecutadas se seleccionan desde fuera del módulo.

Ejemplo: Registrar Pago

1. Registrar pago con tarjeta de crédito
2. Registrar pago con cheque
3. Registrar pago con efectivo

....

Cohesión Coincidental

Un módulo coincidentemente cohesionado es aquel cuyos elementos desarrollan actividades sin relación significativa entre sí.

Ejemplo:

1. Comprar un libro
2. Comer un trozo de torta
3. Ir al teatro
4. Lavar la ropa
5. Dormir

....