Diseño Modular Efectivo

- Beneficios:
  - Reduce la complejidad.
  - Facilita los cambios.
  - Produce una implementación mas sencilla.

- Tipos de Módulos:
  - **El Historial de incorporación:** Se refiere al momento en el que se incorpora el módulo al lenguaje fuente.
  - **El mecanismo de Activación:**
    - Mediante **Referencia**: llamada en forma directa.
    - Mediante **Interrupción**: Suceso exterior que produce un evento que produce el paso del control al modulo.

Diseño Modular Efectivo

- Tipos de Módulos (Continuación):
  - **El Camino de Control:** Describe la forma en la que el modulo se ejecuta internamente.
    - **Módulos de 1 Entrada 1 Salida**

      Es invocado por una rutina llamadora por vez.
    - **Módulos Re-entrantes.**

      Es invocado en **forma concurrente** por mas de una tarea.

Diseño Modular Efectivo

- Puede clasificarse como:
  - **Secuencial:** Se ejecuta sin interrupción.
  - **Incremental:** Puede ser interrumpido antes de que termine y posteriormente restablecida su ejecución en el punto en que se interrumpió.
  - **Paralelo:** Que se ejecuta simultáneamente en entornos de procesamiento paralelo.

Los módulos secuenciales son los más frecuentes y están caracterizados como subprogramas convencionales.

Los módulos incrementales y paralelos poseen una estructura de control no típica.

Diseño Modular Efectivo

- Independencia Funcional:
  - **Deriva directamente** de los conceptos de Modularidad, Abstracción y Ocultamiento de la Información.

La independencia funcional se adquiere desarrollando módulos con una función "clara" y con una "aversión" a una excesiva interacción con otros módulos.

La **independencia funcional** es la clave de un buen diseño que a su vez es la clave de la calidad del software.

Diseño Modular Efectivo

- Independencia Funcional:

  Se mide con 2 criterios cualitativos:

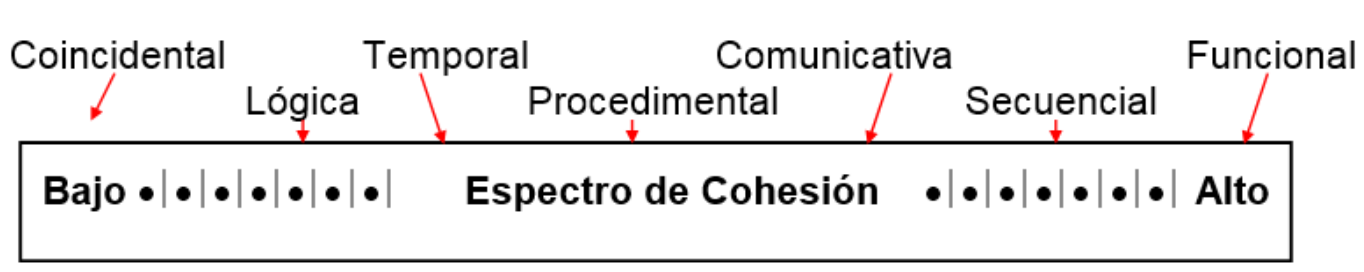
  - **Cohesión**

  Un módulo cohesivo ejecuta una tarea sencilla de un procedimiento de software y requiere poca interacción con procedimientos que ejecutan otras partes de un programa. (Lleva a cabo un única tarea).

  Buscamos la más alta cohesión posible.

Diseño Modular Efectivo

- Independencia Funcional:
  - Espectro de **Cohesión**



Coincidental: Modulo que realiza varias tareas débilmente relacionadas.

…

Secuencial: Algunas partes del modulo realizan mas de una tarea.

…

Funcional: Cada parte de un modulo es necesaria para realizar una única función.

Diseño Modular Efectivo

- Independencia Funcional:

  Se mide con 2 criterios cualitativos:

  - **Acoplamiento**

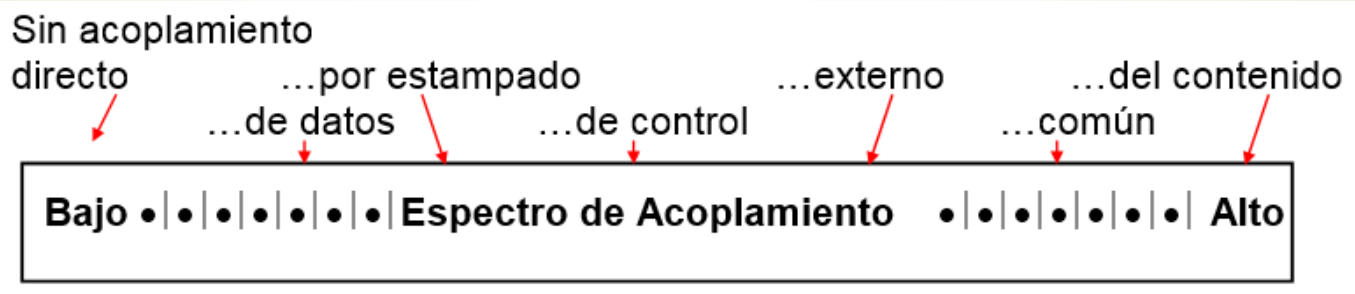
  Es una medida de la **interconexión** entre los módulos de una estructura de programa o bien de la **interdependencia** relativa entre los módulos.

  El acoplamiento depende de la complejidad de las interfaces entre los módulos.

  Buscamos el más bajo acoplamiento posible.

Diseño Modular Efectivo

- Independencia Funcional:
  - Espectro de **Acoplamiento**



Sin Acoplamiento directo: Los módulo no están relacionados.

…

Acoplamiento por estampado: Se pasa una porción de una estructura de datos (en vez de argumentos simples) a través de la interfaz del módulo.

…

Acoplamiento del contenido: Un módulo usa información de datos de control contenida dentro de los limites de otro módulo.

Diseño Modular Efectivo

- Diseño de Datos
  - Según Wasserman(80), el proceso de diseño puede resumirse como:

  La actividad principal durante el diseño de datos **es la selección** de las estructuras de datos, identificadas durante la fase de Análisis. **Éste proceso de selección** puede implicar un análisis algorítmico de dichas estructuras con el fin de determinar un diseño más eficiente que proporcione las operaciones deseadas sobre algún objeto.

  También es importante identificar los módulos de programa que deben operar directamente sobre las estructuras de datos lógicas.

Diseño Modular Efectivo

- Diseño de Datos
  - Además Wasserman(80), propone el siguiente conjunto de principios **para la especificación de datos**:

1. Los principios sistemáticos de análisis aplicados a la función y el comportamiento también deben aplicarse a los datos.

- Revisar flujos de datos y su contenido.

2. Deben identificarse todas las estructuras de datos y las operaciones que se han de realizar sobre cada una de ellas.

3. Debe establecerse y usarse un Diccionario de Datos para definir el diseño de datos y del programa.

Diseño Modular Efectivo

- Diseño de Datos
  - Además Wasserman(80), propone el siguiente conjunto de principios **para la especificación de datos**:

4. Se deben posponer las decisiones de diseño de datos de bajo nivel hasta más adelante en el proceso de diseño.

- Es la aplicación del proceso de refinamiento sucesivo.
- Durante el análisis de requisitos se define la organización global de los datos, se refina durante el diseño preliminar y se especifica en detalle durante el diseño detallado.

5. La representación de una estructura de datos sólo debe ser conocida por los módulos que hagan un uso directo de los datos contenidos en la estructura.

- Este principio alude a la importancia de los conceptos de ocultamiento de información

Diseño Modular Efectivo

- Diseño de Datos
  - Además Wasserman(80), propone el siguiente conjunto de principios **para la especificación de datos**:

6. Se debe desarrollar una biblioteca de datos útiles y de las operaciones que se les puede aplicar.

- Las estructuras de datos y operaciones deben verse como recursos reutilizables en el diseño de software.

7. El diseño de software, y el lenguaje de programación deben soportar la especificación y la realización de tipos abstractos de datos.

Diseño Modular Efectivo

- Diseño Arquitectónico

Su principal objetivo es **desarrollar una estructura de programa modular y representar las relaciones de control entre los módulos**.

Además, el diseño arquitectónico, mezcla las estructuras de programas y las estructuras de datos y define las interfaces que facilitan el flujo de los datos a lo largo del programa.

Diseño Modular Efectivo

Diseño Procedimental

El diseño procedimental se realiza después que se ha establecido la **estructura del programa y de los datos**. Idealmente, la especificación que define los detalles algorítmicos debería explicarse en lenguaje natural, el que entienden todos los miembros del equipo de desarrollo y los cliente o usuarios.

Pero la exigencia de que esta especificación se haga sin ambigüedades ha llevado a que se deriven **formas más restringidas** para la representación de los detalles procedimentales:

- **Diagrama de Flujo:** Notación gráfica ampliamente usada para diseño procedimental, pero también la más abusada.
- **Diagrama de caja:**
- **Tablas de decisión:** Notación que traduce las acciones y las condiciones a una forma tabular.
- **LDP (Lenguaje de diseño de programas):** Por todas sus características se hace la forma de diseño más usada actualmente.