Ingeniería de Software
Unidad II

Diseño de Sistemas

Sergio Sánchez Rios.

Ingeniero en Informática – Licenciado en Informática

13 de abr de 2011

Diseño Conceptual y Técnico

Para transformar los requerimientos en un sistema que funcione, los diseñadores deben satisfacer tanto a los clientes como a los constructores de sistemas.

Los clientes deben comprender lo que el sistema debe hacer, y los constructores deben comprender cómo debe operar el sistema.

El diseño es un proceso iterativo que consta de dos partes: **Diseño conceptual o del sistema**, **Diseño técnico**.

¿Qué es el diseño?

El diseño es el proceso creativo de transformación del problema en una solución; la descripción de una solución también se denomina diseño.

La solución será la que satisface todos los requerimientos planteados en la especificación de requerimientos.

Las soluciones en muchos casos son ilimitadas.

Diseño Conceptual y Técnico

Diseño Conceptual o del Sistema

Se realiza primero, y le dice al cliente que es lo que hará realmente el sistema.

Una vez que se aprueba este diseño, se vuelca el trabajo a un trabajo de diseño más detallado.

El diseño conceptual describe el sistema en un lenguaje que el cliente pueda comprender, en lugar de usar términos técnicos o jergas de computación.

Diseño Conceptual y Técnico

Un buen diseño conceptual debería tener las siguientes características:

- Se escribe en el lenguaje del cliente.
- No contiene expresiones de jerga técnica.
- Describe claramente las funciones del sistema.
- Es independiente de la implementación.
- Esta vinculado con los documentos del requerimiento.

Diseño Conceptual y Técnico

Diseño Técnico

Permite que los constructores comprendan el hardware y el software concretos que se necesitan para resolver el problema del cliente.

Es decir este diseño describe:

- La configuración de hardware.
- Las necesidades de software.
- Las interfaces de comunicación.
- Las entradas y salidas del sistema.
- La arquitectura de red.
- Y cualquier otro aspecto que incida en la transformación de los requerimientos en una solución.

Diseño Conceptual y Técnico

El diseño técnico incluye los siguientes ítem:

- Una descripción de los principales componentes de hardware y de sus funciones.
- La jerarquía y funciones de los componentes de software.
- Las estructuras de datos y los flujos de datos.

Diseño Conceptual y Técnico

El diseño se describe en un único documento, pero muchas veces se vuelca en dos.

QUÉ

CÓMO

DISEÑO CONCEPTUAL

función

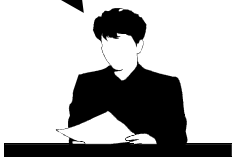
Diseñadores del Sistema

DISEÑO TÉCNICO

forma

Clientes

Constructores del Sistema

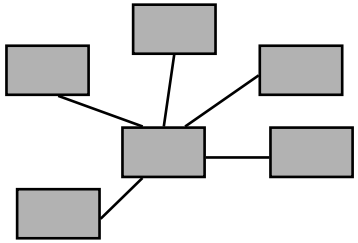


Diseño Conceptual y Técnico

Ejemplo de Diseño Conceptual y Técnico.

"El usuario podrá enviar mensajes a cualquier usuario en cualquier otra computadora en red"

DISEÑO CONCEPTUAL

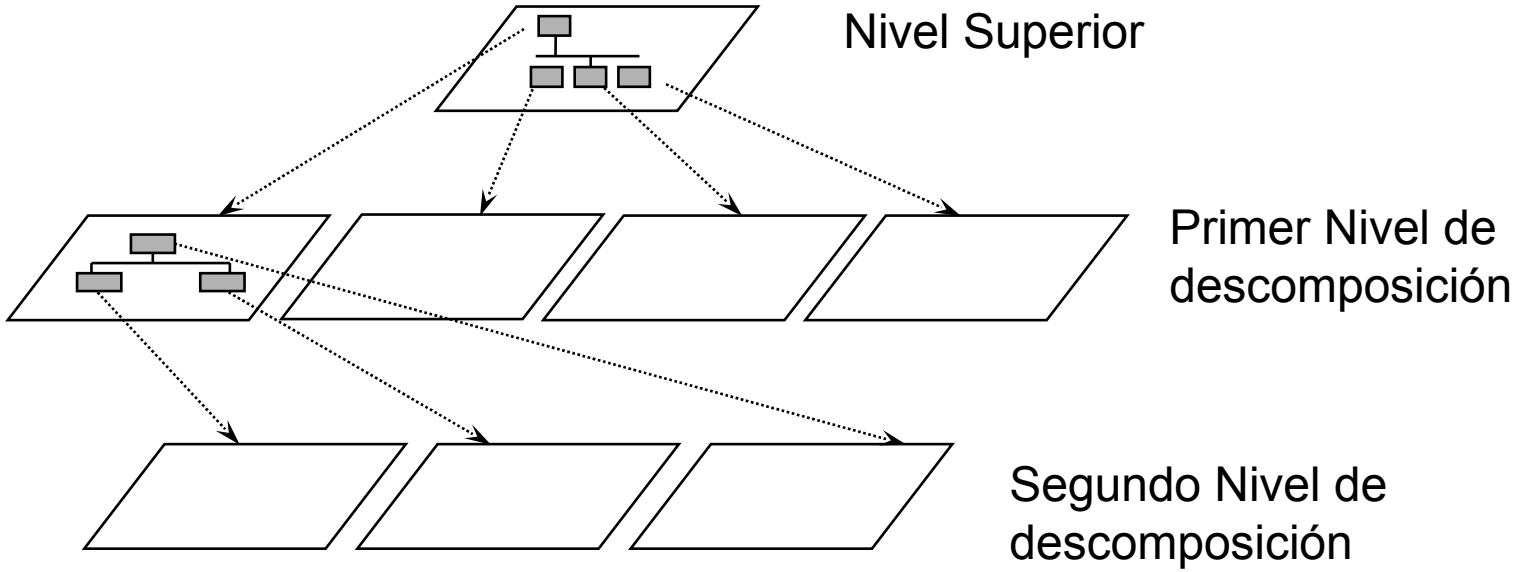


Topología de Red
Protocolo
Velocidad (bps)
. . .

DISEÑO TÉCNICO

Descomposición y Modularidad

Todo método de diseño abarca un tipo de descomposición a partir de una descripción de alto nivel de los elementos claves del sistema, y creando visiones a niveles inferiores para ver como las características y funciones del sistema se acomodaran en conjunto.



Descomposición y Modularidad

La descomposición separa el diseño en partes componibles denominadas **módulos o componentes**.

Se dice que un modulo está “bien definido” cuando todas las entradas en él son esenciales para su función y todas las salidas son generadas por una de sus acciones.

Niveles de Diseño

Shaw y Garlar (1996) sugieren tres niveles de diseño:

Arquitectura

Este nivel está asociado a las capacidades del sistema identificadas en la especificación de requisitos, con los componentes del sistema que la implementarán.

Por lo general estos componentes son módulos y la arquitectura también describe las interconexiones entre ellos.

La arquitectura define además operadores que crean sistemas a partir de los subsistemas.

Niveles de Diseño

Diseño de Código

Comprende algoritmos, estructuras de datos, y los componentes que son primitivas del lenguaje de programación, tales como números, caracteres, punteros, etc.

Diseño ejecutables

Se remite al diseño de código en un nivel de detalle todavía más bajo. Trata entre otros aspectos la asignación de memoria, los formatos de datos y la configuración de bits.

Etapas del proceso de diseño arquitectónico

Las siguientes etapas son comunes para todos los procesos de diseño arquitectónicos:

- **Estructuración del sistema**: se establecen los subsistemas y sus relaciones.
- **Modelo de control**: se establece un modelo general de las relaciones de control entre las partes del sistema.
- **Descomposición modular**: cada subsistema se descompone en módulos.

Etapas del proceso de diseño arquitectónico
Estructura del sistema

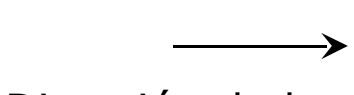
Está primera etapa, es la descomposición de un sistema en un conjunto de subsistemas que interactúan.

En el nivel más abstracto, un diseño se puede ver como un diagrama de bloques en el que cada cuadro representa un subsistema.

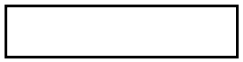
Un diagrama de bloques presenta un panorama general de la estructura del sistema. Por lo general, esto es comprensible por usuarios y desarrolladores.

Etapas del proceso de diseño arquitectónico
Estructura del sistema

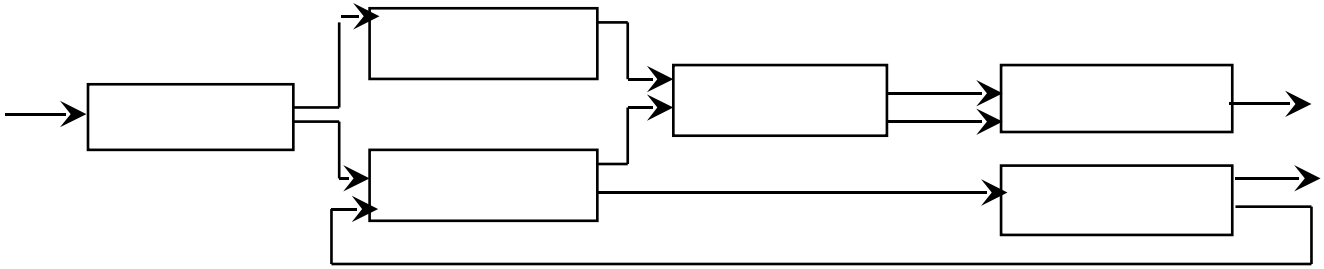
Ejemplo abstracto para un diagrama de bloques.



Dirección de la relación



Subsistemas



Se pueden desarrollar modelos más específicos de la estructura, que muestren cómo los subsistemas comparten datos, cómo están distribuidos y como se conectan entre ellos.

Etapas del proceso de diseño arquitectónico

Estructura del sistema – Modelo de Depósito

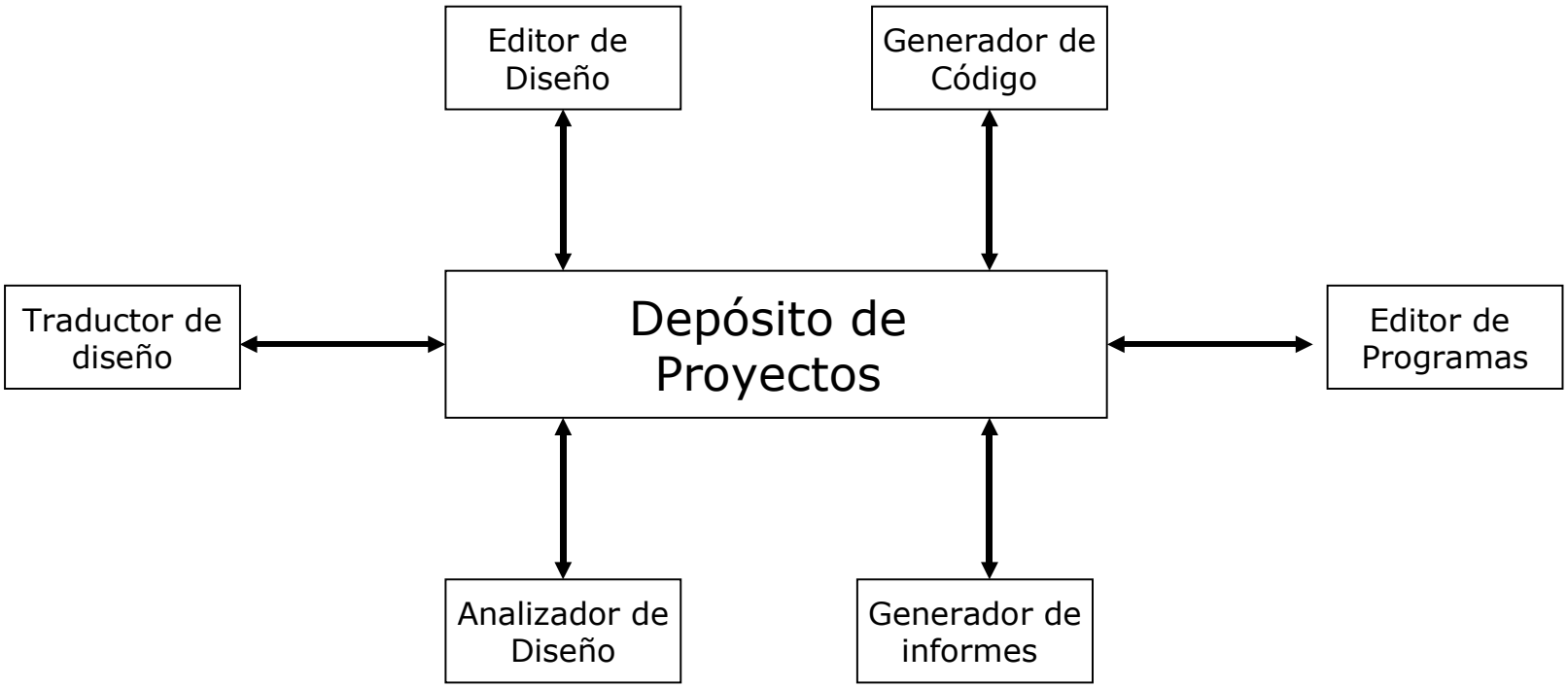
Los subsistemas que componen un sistema deben intercambiar información con el fin de que puedan trabajar de forma conjunta y efectiva.

Existen dos formas:

- Todos los datos compartidos se ubican en una base de datos central que pueda ser accedida por todos los subsistemas (se denomina modelo de deposito).
- Cada subsistema tiene su propia base de datos. Los datos se intercambian con otros subsistemas pasando mensajes entre ellos.

Etapas del proceso de diseño arquitectónico
Estructura del sistema – Modelo de Depósito

Ejemplo Modelo Depósito: La arquitectura de un conjunto integrado de herramientas case.



Etapas del proceso de diseño arquitectónico

Estructura del sistema – Modelo de Depósito

Ventajas

- ✓ Eficiente para compartir grandes cantidades de datos.
- ✓ Los subsistemas están acordes al modelo de depósito de datos.
- ✓ Las actividades de respaldo, seguridad, control de acceso y recuperación de errores están centralizadas. Son responsabilidad del deposito.

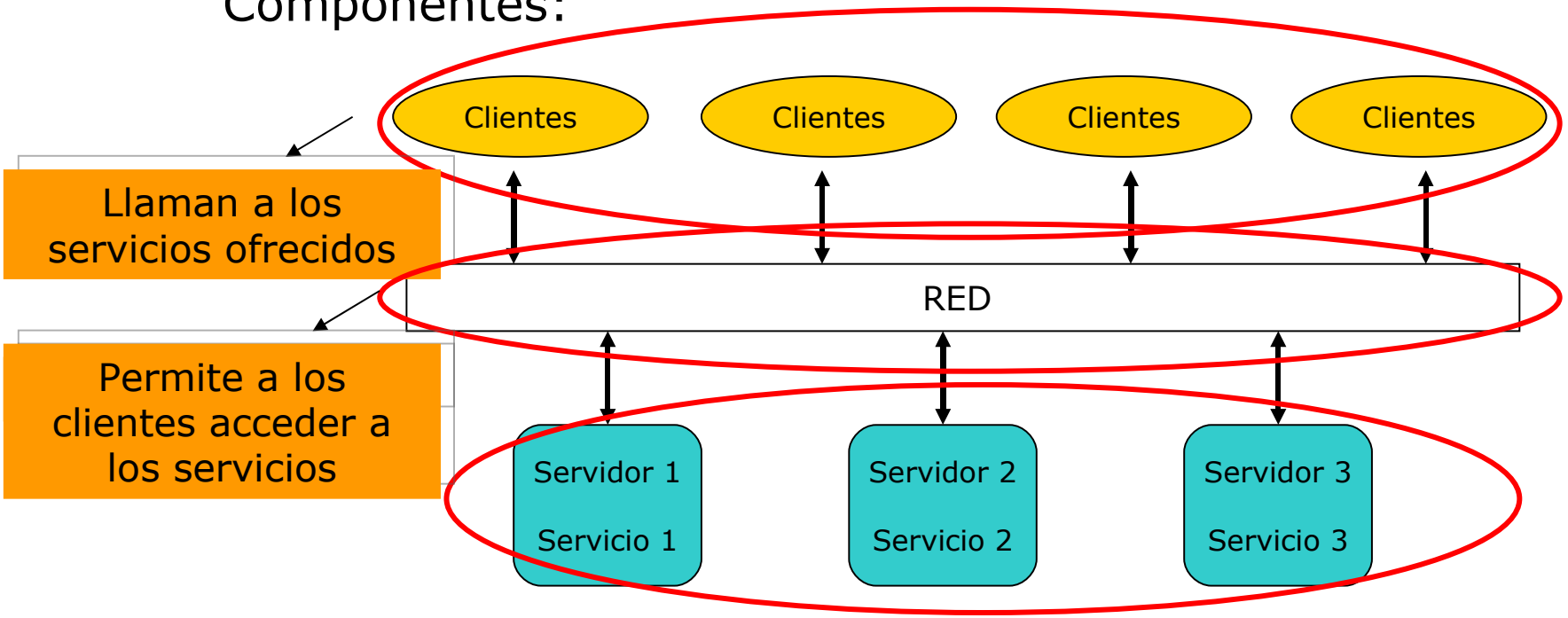
Desventajas

- ○ Si se genera un gran volumen de información, será difícil evolucionar si se a acordado un modelo de datos. Traducir esto en un nuevo modelo será costoso, puede ser difícil, e incluso imposible.
- ○ El modelo de depósito fuerza a los subsistema a las mismas políticas de seguridad.

Etapas del proceso de diseño arquitectónico
Estructura del sistema – Modelo Cliente - Servidor

Es un modelo de sistemas distribuidos que muestra como los datos y el procesamiento se distribuyen a lo largo de varios procesadores.

Componentes:



Etapas del proceso de diseño arquitectónico
Estructura del sistema – Modelo Cliente - Servidor

Los clientes necesitan conocer los nombres de los servidores y los servicios que suministran. En cambio, los servidores no requieren conocer la identidad de los clientes o cuantos clientes existen.

La ventaja más importante del modelo cliente – servidor, es su arquitectura distribuida. Ya que, es fácil agregar o modificar un nuevo servidor sin afectar a otras partes del sistema.

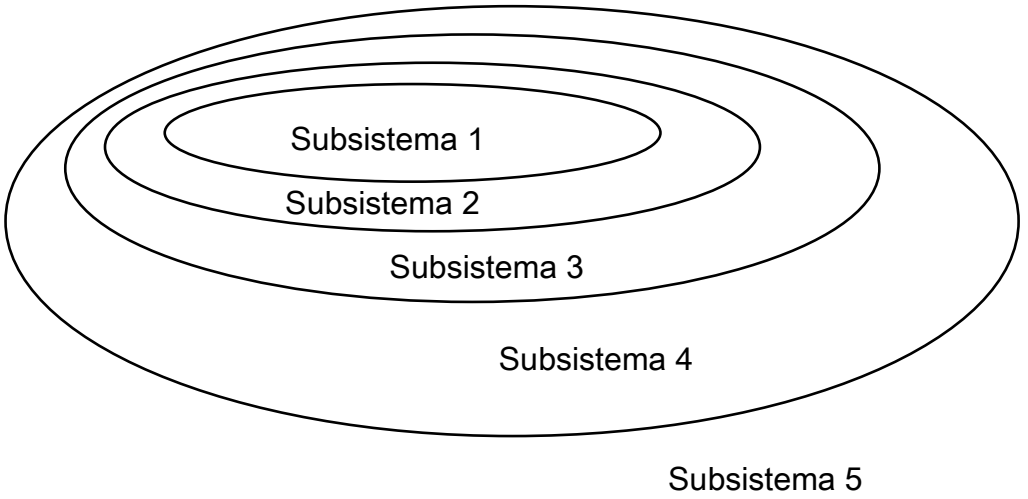
El lado negativo es que por lo general necesitan seguridad más elaborada, administración del sistema y desarrollo de aplicaciones. Así que necesitan de más recursos para ser implementados y darles soporte.

Etapas del proceso de diseño arquitectónico
Estructura del sistema – Modelo de Capas

Modela la interacción entre los subsistemas.

Organiza el sistema en una serie de capas, cada una de las cuales suministra un conjunto de servicio.

Un modelo bien conocido para este enfoque es el modelo de referencia ISO/OSI.



Etapas del proceso de diseño arquitectónico
Estructura del sistema – Modelo de Capas

Este modelo es una arquitectura cambiable y portable. Cuando se cambia o elimina una capa, solo se verán afectadas sus capas adyacentes.

Una desventaja del enfoque, es que estructurar los sistemas de está forma es difícil.

El desempeño también puede ser un problema debido a los múltiples niveles de interpretación de ordenes que algunas veces se requieren.

Etapas del proceso de diseño arquitectónico
Modelos de Control

Los modelos para estructurar un sistema se refieren a la manera en que un sistema se descompone en subsistemas. Esto no incluye información de control.

El diseñador debe organizar los subsistemas acorde a un modelo de control que complemente el modelo estructural que se utiliza.

Se identifican dos enfoques:

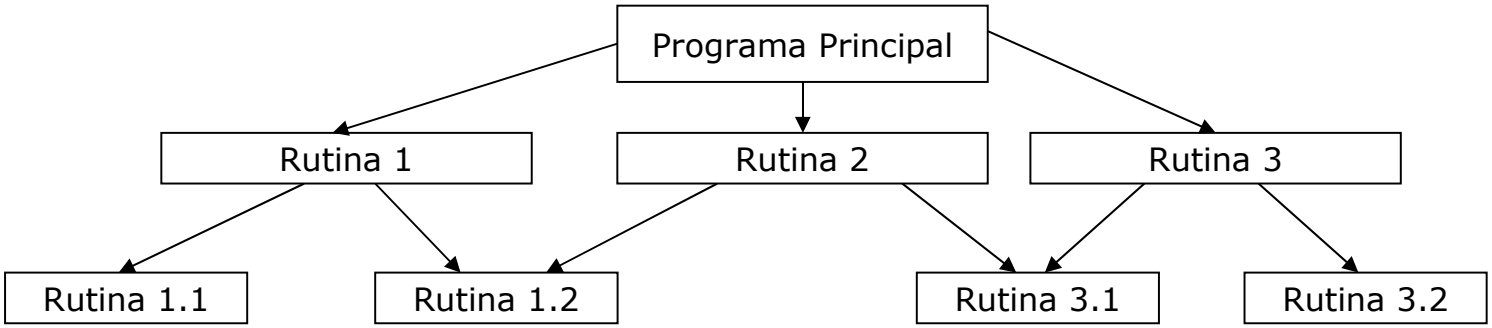
- **Control Centralizado**: un subsistema tiene completa responsabilidad para controlar, iniciar, y detener a otros subsistemas. Puede entregar el control, pero se lo deben devolver.
- **Control basado en eventos**: cada subsistema puede responder a eventos generados en el exterior.

Etapas del proceso de diseño arquitectónico
Modelos de Control – Control Centralizado

Un subsistema se designa como controlador del sistema y tiene la responsabilidad de administrar la ejecución de otros subsistemas.

Existen dos enfoques:

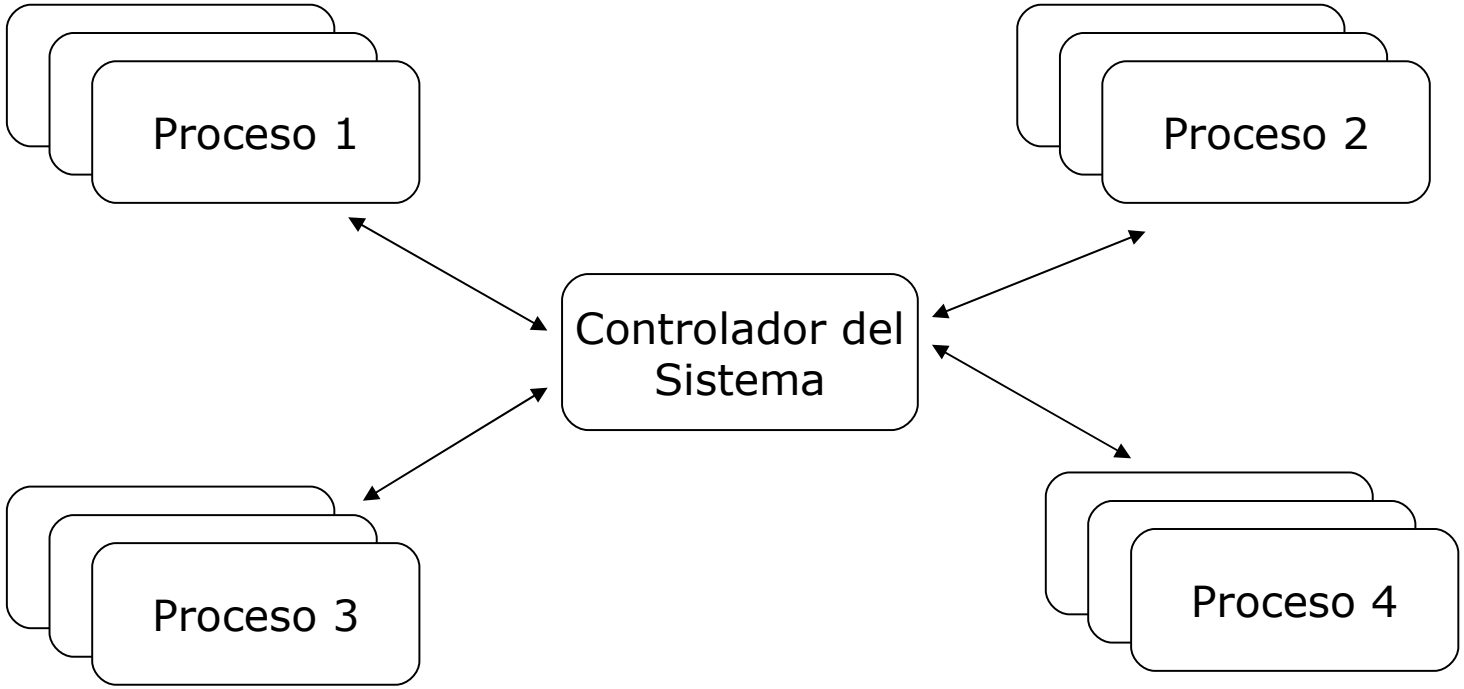
- **Modelo de llamada - retorno:** éste es el modelo familiar de subrutinas descendentes, donde el control se inicia en la parte superior de una jerarquía y por medio de llamadas a las subrutinas pasa a los niveles inferiores del árbol.



Aplicable solo a sistemas secuenciales

Etapas del proceso de diseño arquitectónico
Modelos de Control – Control Centralizado

- **Modelo del administrador:** se aplica a los modelos concurrentes. Un componente del sistema se designa como un sistema administrador y controla el inicio, detención y la coordinación de otros procesos del sistema.



Etapas del proceso de diseño arquitectónico
Modelos de Control – Dirigidos por eventos

Los modelos de control dirigidos por eventos se rigen por eventos generados en el exterior.

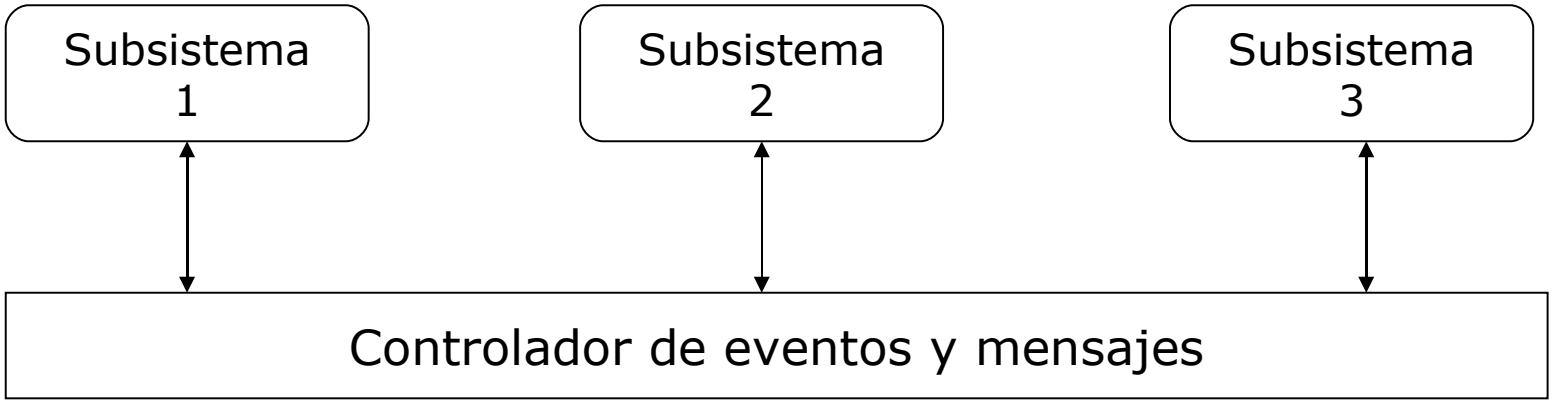
Existen varios sistemas dirigidos por eventos que se pueden desarrollar. Ej.: las hojas de calculo, en la que el valor cambiante de las celdas provoca que otras celdas se modifiquen.

Se discuten solo dos modelos de este tipo:

- **Modelo de Transmisión**: en estos modelos, un evento se transmite, en principio a todos los subsistemas. Cualquier subsistema que pueda manejar ese evento responde a él.

Etapas del proceso de diseño arquitectónico

Modelos de Control – Dirigidos por eventos



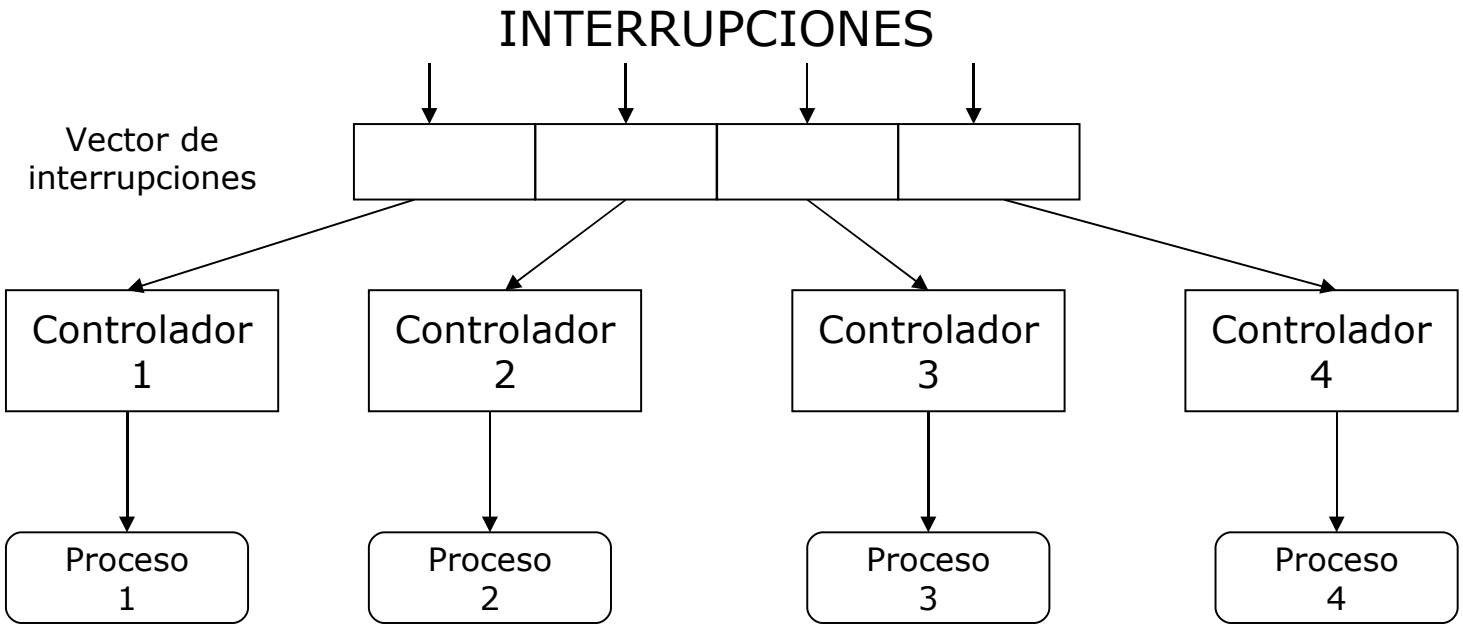
La ventaja de este modelo, es su evolución relativamente sencilla. Un nuevo subsistema que maneje clases particulares de eventos, se puede integrar registrando sus eventos en el controlador de eventos.

La desventaja es que los subsistemas no saben si los eventos se manejarán, ni cuando lo harán.

Etapas del proceso de diseño arquitectónico

Modelos de Control – Dirigidos por eventos

- **Modelo dirigido por interrupciones**: Estos se utilizan exclusivamente en sistema de tiempo real, donde las interrupciones externas son detectadas por un controlador de interrupciones. Después éstos se pasan a algún componente para su procesamiento.



Etapas del proceso de diseño arquitectónico
Descomposición modular

Después de diseñar una arquitectura estructural, la siguiente etapa del proceso de diseño arquitectónico es descomponer los subsistemas en módulos.

Se consideran dos modelos que son de utilidad cuando se descompone un subsistema en módulos:

- **Modelo orientado a objetos:** el sistema se descompone en un conjunto de objetos que se comunican entre ellos.
- **Modelo de flujo de datos:** el sistema se descompone en módulos funcionales que afectan entradas de datos y las transforman, de alguna manera, en datos de salida.

En el modelo objetual, los módulos son objetos con estado privado y con operaciones definidas sobre ese estado.

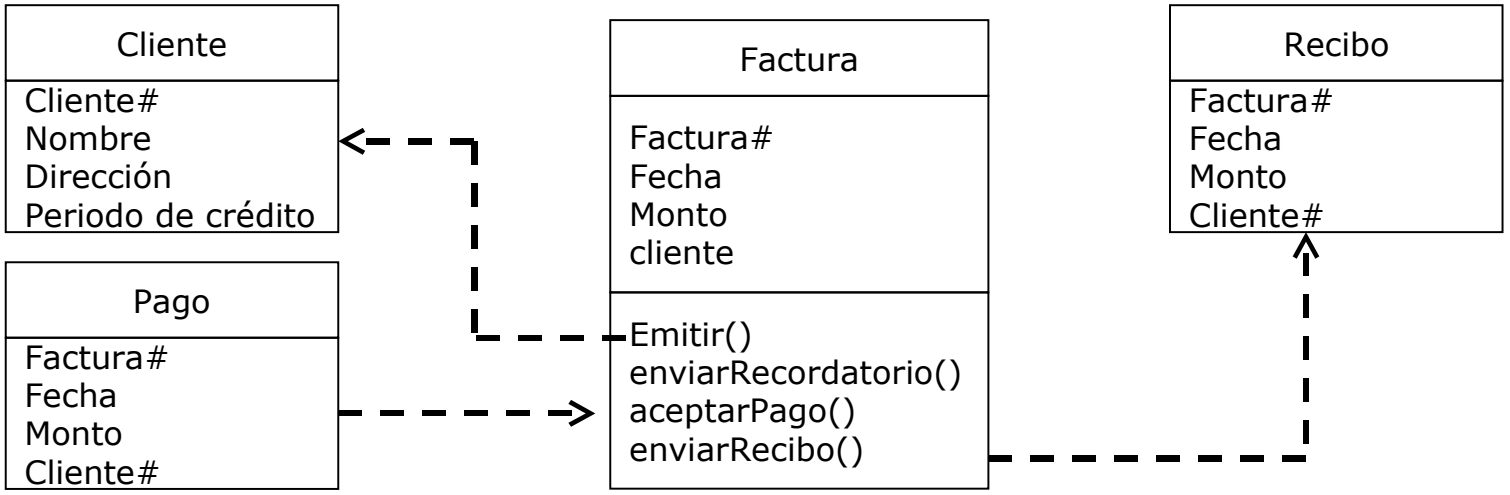
En el modelo de flujo de datos, los módulos son transformaciones funcionales.

Etapas del proceso de diseño arquitectónico
Descomposición modular – Modelo de Objetos

El modelo orientado a objetos de una arquitectura de sistema, estructura el sistema en un conjunto de objetos débilmente acoplados con interfaces bien definidas.

Los objetos llaman a los servicios ofrecidos por otros objetos.

Una descomposición orientada a objetos, comprende clases de objetos, sus atributos y operaciones.



Etapas del proceso de diseño arquitectónico
Descomposición modular – Modelo de Objetos

Las ventajas de este modelo son bien conocidas, puesto que los objetos están débilmente acoplados, la implementación de objetos se puede modificar sin afectar a otros objetos.

Además, los objeto se pueden reutilizar.

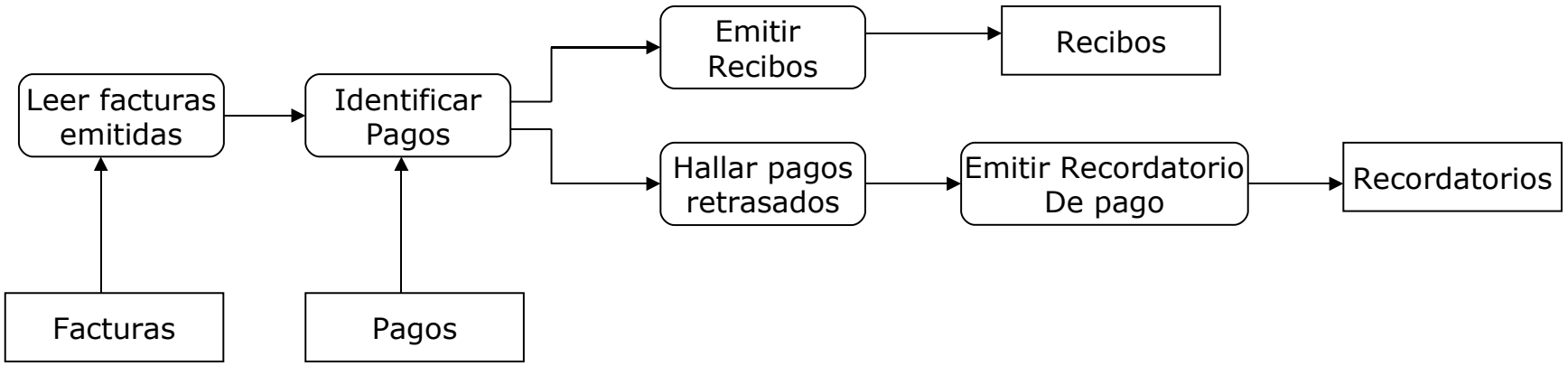
La desventaja, es que los objetos para utilizar los servicios deben hacer referencia explicita al nombre y la interfaz de otros objetos.

Etapas del proceso de diseño arquitectónico

Descomposición modular – Modelo de Flujo de datos.

En un modelo de flujo de datos, las transformaciones funcionales procesan sus entradas y producen sus salidas .

Los datos fluyen de un lugar a otro y se transforman durante este movimiento.



Etapas del proceso de diseño arquitectónico
Descomposición modular – Modelo de Flujo de datos.

Ventajas de esta arquitectura:

- ✓ Permite la reutilización de transformaciones.
- ✓ Es intuitivo.
- ✓ Permite que el sistema evolucione al agregar nuevas transformaciones.
- ✓ Sencilla de implementar, ya sea como un sistema concurrente o secuencial.

Desventajas

- ✘ Cada transformación debe estar acorde con las transformaciones que se comunica, en el formato de los datos a ser procesado.

Problemas en la creación del diseño
Modularidad y nivel de abstracción

En un diseño modular, los componentes tienen entradas y salidas bien definidas y cada componente tiene un propósito previamente establecido.

Los componentes modulares están organizados en una jerarquía, como resultado de la descomposición o abstracción.

A medida que nos desplazamos hacia abajo encontramos mayor nivel de detalle para cada componente.

El nivel más alto es el más abstracto.

Los niveles más abstractos se resuelven primero y la solución se alcanza al desarrollar el detalle.

Problemas en la creación del diseño
Modularidad y nivel de abstracción

La modularidad oculta los detalles. Una ventaja de este **"ocultamiento de información"**, es que cada componente oculta de los demás una decisión de diseño.

Por esto es probable que cambien las decisiones de diseño, pero el diseño como un todo puede permanecer intacto, solo cambia el diseño de componentes.

Características de un buen diseño

Los diseños de calidad superior deben tener características que dan como resultado productos de calidad: facilidad de comprender, de implementar, de probar, de modificar y la correcta traducción de la especificación de requerimientos.

Atributos que reflejan la calidad del diseño:

- Independencias de los componentes.
- Identificación y tratamiento de excepciones.
- Prevención de defectos y tolerancia de defectos.

Características de un buen diseño

Independencia de los componentes

En la mayoría de los diseños se trata de que los componentes sean independientes unos de otros. Ya que, un componente independiente es mucho más fácil de modificar.

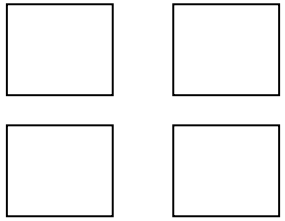
Para reconocer y medir el grado de independencia de los componentes de un diseño se aplican dos conceptos : **acoplamiento** y **cohesión** (Yourdon y Constantine 1978).

Características de un buen diseño

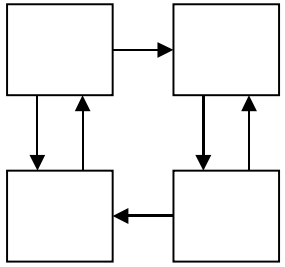
Independencia de los componentes - Acoplamiento

Se dice que dos componentes están "altamente acoplado" cuando existe mucha dependencia entre ellos.

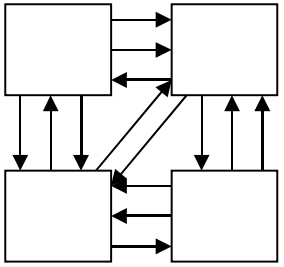
Los componentes "poco acoplado" tienen algunas dependencias, pero las interconexiones entre ellos son débiles.



No acoplado



Débilmente acoplado (pocas dependencias)



Fuertemente acoplado (muchas dependencias)

Características de un buen diseño

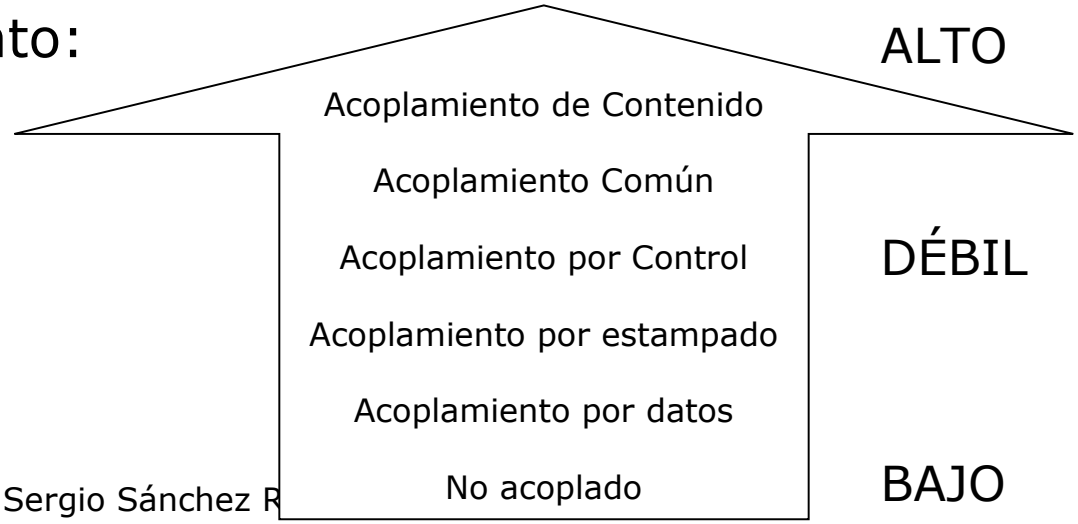
Independencia de los componentes - Acoplamiento

Se puede medir el acoplamiento en función de un rango de dependencia, que va de la dependencia completa a la completa independencia.

La meta no necesariamente es la completa independencia, sino mantener el menor grado de acoplamiento posible.

El bajo acoplamiento contribuye a minimizar el número de componentes que necesitan revisión.

Tipos de acoplamiento:



Características de un buen diseño

Independencia de los componentes - Acoplamiento

Acoplamiento de Contenidos

Es cuando un componente modifica a otro. Entonces el componente modificado es completamente dependiente del que lo modifica.

Situaciones en las que se da:

- Un módulo modifica algún elemento de otro modulo.
- Un módulo modifica una variable local a otro.
- Desde un módulo se pasa a otro por puntos distintos de los de la entrada (Ej.: goto).

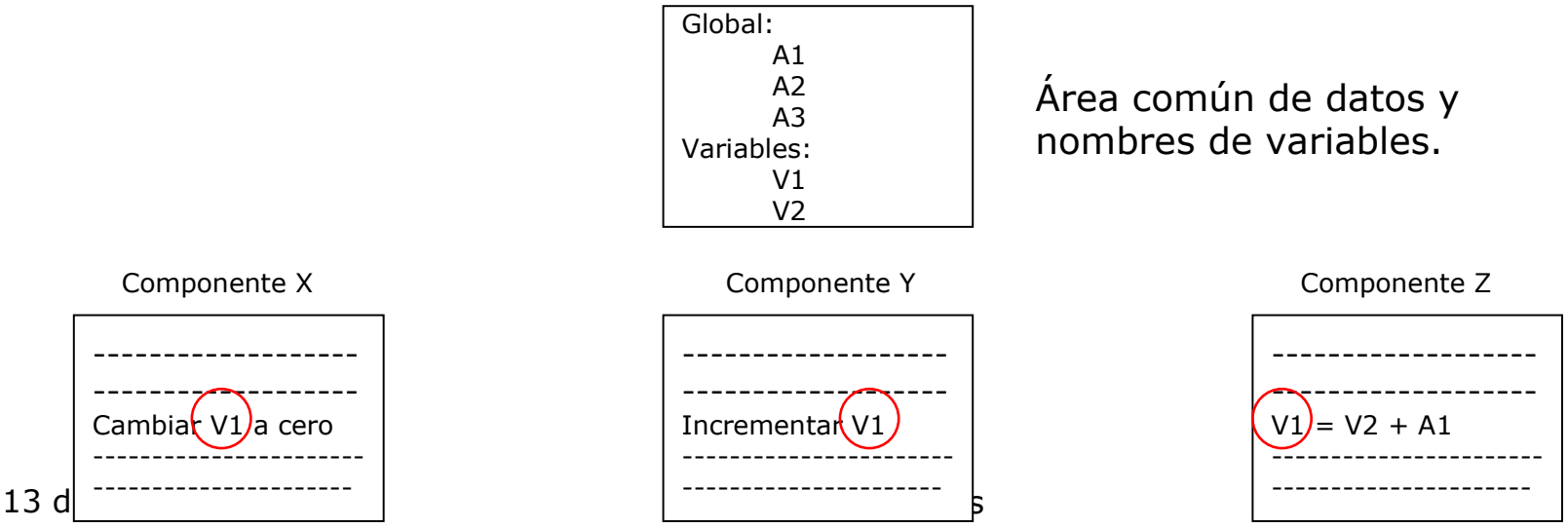
Características de un buen diseño

Independencia de los componentes - Acoplamiento

Acoplamiento Común

Los datos son accesibles desde un almacenamiento común de datos.

La dependencia todavía existe, dado que hacer un cambio a los datos comunes significa rastrear en todos los componentes que tienen acceso a estos datos, a fin de evaluar el efecto del cambio.



Características de un buen diseño

Independencia de los componentes - Acoplamiento

Acoplamiento de Control

Se produce cuando un componente pasa parámetros para controlar la actividad del otro.

Para el componente controlado es imposible funcionar sin la dirección del que lo controla.

Acoplamiento por estampado

Se produce cuando se usa una estructura de datos para pasar información de un componente a otro, y es la estructura misma la que se pasa.

Características de un buen diseño
Independencia de los componentes - Acoplamiento

Acoplamiento por Datos (deseable)

Es el más simple, es solo el paso de datos a otro componente.

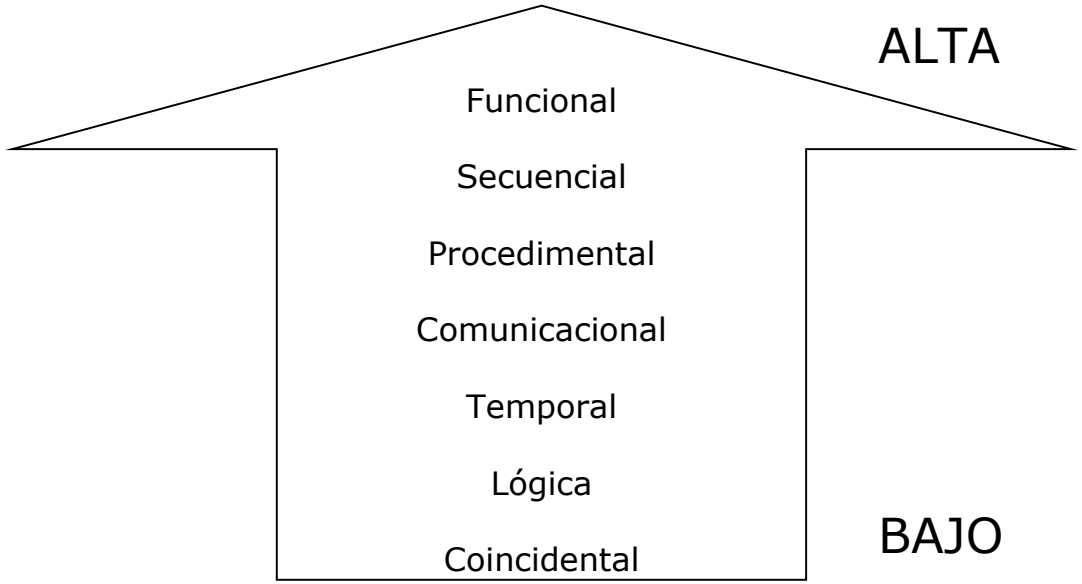
Lo aceptable para un buen diseño es poseer un acoplamiento débil (lo ideal es bajo).

Características de un buen diseño

Independencia de los componentes - Cohesión

Un componente es cohesivo si todos sus elementos están orientados a la realización de una única tarea y son esenciales para llevarla a cabo.

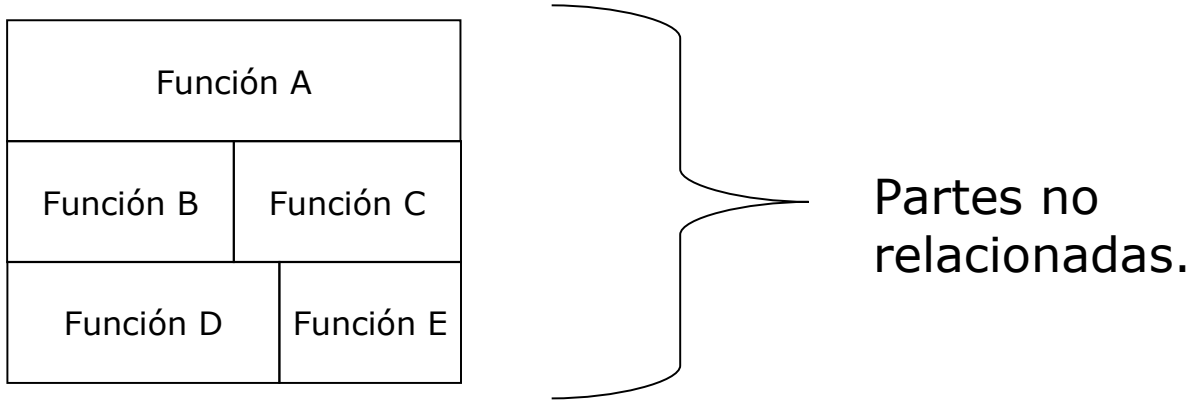
Tipos de cohesión:



Características de un buen diseño
Independencia de los componentes - Cohesión

Cohesión Coincidental

Ocurre cuando las partes de un componente no tienen relación alguna entre si.(Cohesión baja)



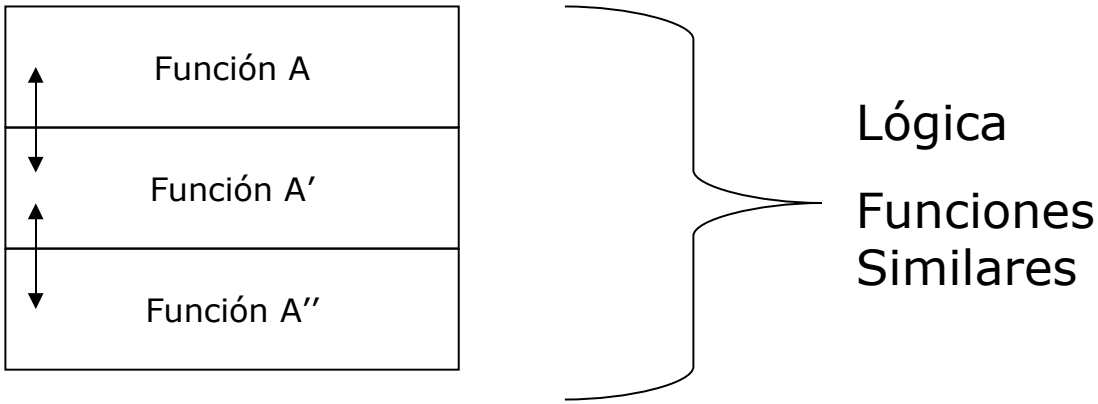
Características de un buen diseño

Independencia de los componentes - Cohesión

Cohesión Lógica

Algunas funciones o elementos de datos relacionados lógicamente están puestos en el mismo componente.

Ej: Componente que lee todo tipo de entradas (cinta, disco, puertos, etc.) existe una relación lógica, pero no todas las formas son iguales

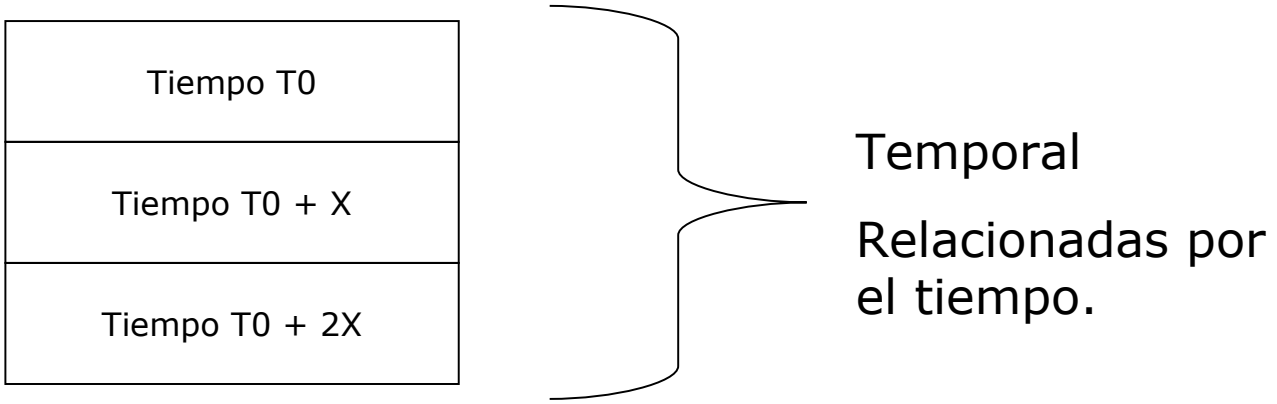


Características de un buen diseño

Independencia de los componentes - Cohesión

Cohesión Temporal

A veces un componente se utiliza para inicializar un sistema o un conjunto de variables. Este componente realiza varias funciones en secuencia, pero las funciones en si solo se realizan en el momento que ocurren.



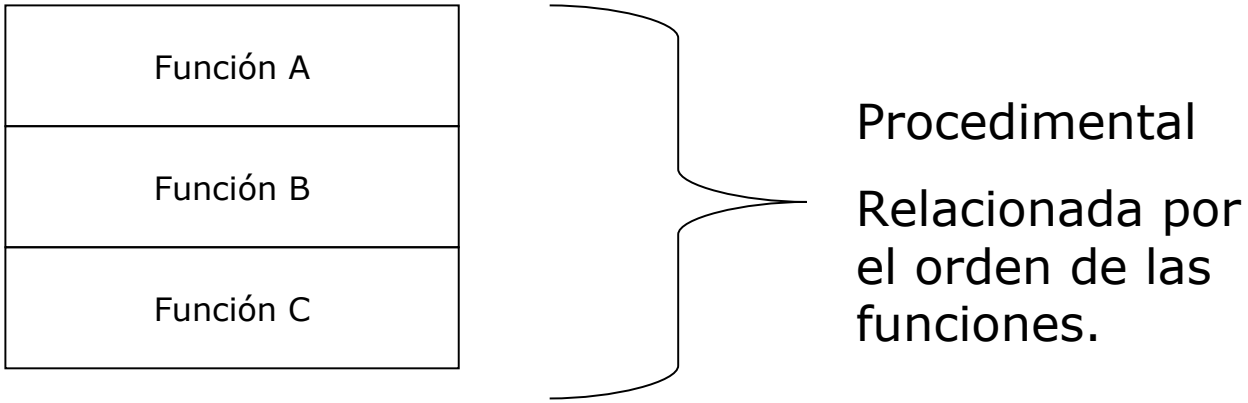
Características de un buen diseño

Independencia de los componentes - Cohesión

Cohesión Procedimental

Es cuando las funciones se agrupan en un mismo componente para asegurar un orden previsto.

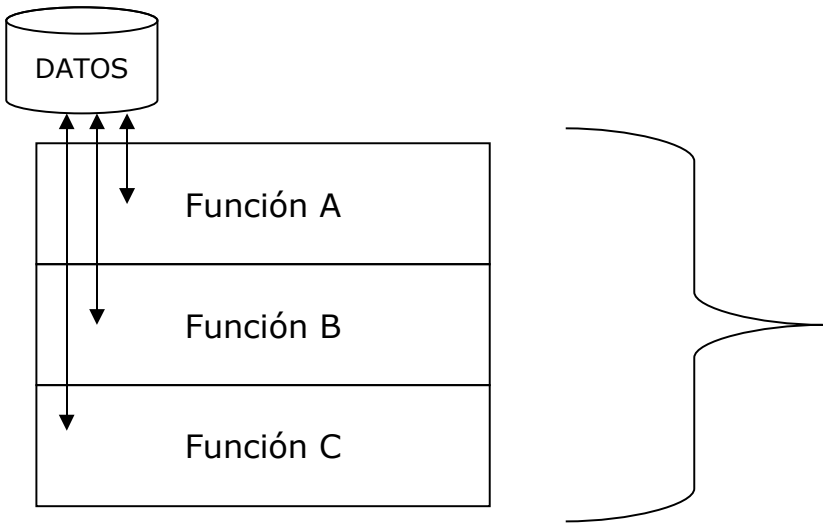
Por Ejemplo: los datos deben ser ingresados antes de chequearlos o de manipularlos, tres funciones en una secuencia específica.



Características de un buen diseño
Independencia de los componentes - Cohesión

Cohesión Comunicacional

Es cuando en un componente se asocian funciones que acceden a un mismo conjunto de datos.

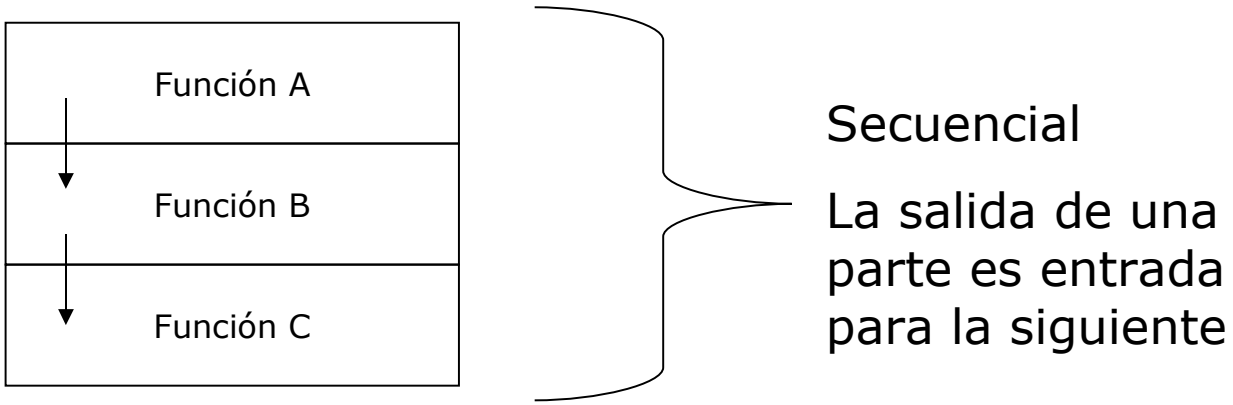


Características de un buen diseño

Independencia de los componentes - Cohesión

Cohesión Secuencial

Se produce cuando la salida por una parte del componente actúa como entrada para la parte que le sigue.

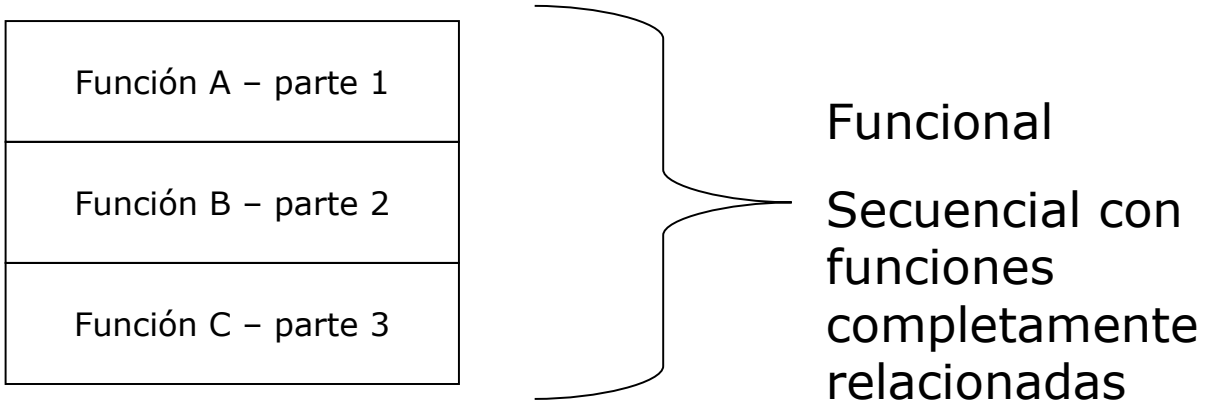


Características de un buen diseño

Independencia de los componentes - Cohesión

Cohesión funcional (deseable)

Es la ideal, donde cada elemento de proceso es esencial para la realización de una única función y todos los elementos esenciales están contenidos en un único componente.



Características de un buen diseño
Identificación y tratamiento de excepciones

El diseño debe realizarse defensivamente, tratando de anticiparse a situaciones que podrían derivar en problemas en el sistema.

No es fácil diseñar defensivamente, ya que la especificación dice lo que debe hacer el sistema no lo que no debe.

Para poder hacer un diseño defensivo se debe hacer un **manejo de excepciones**.

- **Excepción**: son situaciones que se saben contrarias a lo que se espera realmente que haga el sistema.

El manejo de excepciones debe permitir que el sistema dirija cada excepción en una forma satisfactoria que no degrade las funciones del sistema.

Características de un buen diseño
Identificación y tratamiento de excepciones

Excepciones más comunes:

- Fracaso al proporcionar un servicio.
- Proporcionar un servicio o dato erróneo.
- Corrupción de datos.

Las excepciones se pueden manejar según uno de los siguientes criterios:

- **Reintentar**: se restaura el sistema a su estado previo y se intenta realizar el servicio utilizando una estrategia diferente.
- **Corregir**: se restaura el sistema a su estado previo y se intenta realizar el servicio usando la misma estrategia.
- **Informar**: se restaura el sistema a su estado previo, se informa el problema a un componente de tratamiento de error y no se proporciona el servicio.

Características de un buen diseño
Prevención de defectos y tolerancia de defectos

La meta es hacer un código tan libre de defectos como sea posible incorporando al sistema la prevención y la tolerancia de defectos.

Una de las características de un buen diseño es la forma en que previene y tolera los defectos.

En lugar de esperar hasta que el sistema falle para corregir el problema, los diseñadores pueden anticipar lo que puede ocurrir y construir el sistema para que reaccione de manera aceptable.

Características de un buen diseño
Prevención de defectos y tolerancia de defectos

Detección activa de defectos

Es la forma de controlar periódicamente la presencia de síntomas de defectos, se intenta anticipar la producción de fallas.

Ejemplo:

- **Sospecha Mutua**: cada componente del sistema supone que los otros componentes contienen defectos. Cada componente controla la presición y consistencia de sus entradas. Este manejo inmediato del defecto limita el daño que produce, evita que se transforme en una falla.

- **Redundancia**: según los resultados obtenidos por dos procesos se comparan si son idénticos. Ej.: En un programa de contabilidad primero se suman las filas, y luego se suman las columnas para ver si son iguales.

Características de un buen diseño
Prevención de defectos y tolerancia de defectos

Corrección de defectos

Una vez descubierto el defecto se debe corregir.

Por lo general, la corrección del defecto subsana el daño producido, así como modifica el producto para eliminar tal defecto.

Los diseñadores deben incluir una estrategia para los defectos a medida que son encontrados.

Se puede optar por detener el sistema cuando lo afecta el defecto de algún modo o simplemente se registra la existencia de la falla, apuntar el estado del sistema en el momento de producirse la falla y arreglar el daño después.

La criticidad del sistema determinara la estrategia a usar.

Características de un buen diseño
Prevención de defectos y tolerancia de defectos

Tolerancia a defectos

Muchas veces la corrección de un defecto es demasiado costosa, riesgosa e inconveniente.

La tolerancia de fallas en el aislamiento del daño producido por una falla y es conveniente y hasta deseable en muchas circunstancias.

Para poder aplicar tolerancia debe existir una anticipación de los defectos, lo que es difícil sobre todo en sistemas complejos.

Revisiones del diseño

Cuando el diseño se completa se mantienen reuniones con los clientes para revisarlo antes de avanzar al desarrollo.

El proceso de revisión se realiza en tres etapas en correspondencia con los pasos del proceso de diseño:

1. Revisión del diseño preliminar.
2. Revisión crítica del diseño.
3. Revisión del diseño de programas.

Revisiones del diseño
Revisión del diseño preliminar

Los clientes y usuarios se reúnen para validar el diseño conceptual.

Se asegura que todos los aspecto relativos a los requerimientos han sido apropiadamente contemplados en el diseño.

Se invita a participar a ciertas personas claves:

- Cliente (s), quien ayuda a definir los req. del sistema.
- Analista (s), quien colabora para definir los req. del sistema
- Usuario (s), potenciales del sistema.
- Diseñador (es) del sistema.
- Un moderador (solo coordina), un secretario (no se involucra).
- Otros desarrolladores (entregan perspectiva externa)

Revisiones del diseño
Revisión del diseño preliminar

Se recomienda que el número de integrantes sea pequeño de modo que facilite las discusiones.

Durante la revisión se presenta a la audiencia el diseño conceptual. Al hacerlo, se demuestra que el sistema tiene la estructura requerida, las funciones y las características especificadas por los documentos de análisis.

Todos los participantes, en conjunto, verifican que el diseño propuesto incluya el hardware necesario, interfaces con otros sistemas, entradas y salidas.

Los clientes aprueban los diálogos y menús, los formatos de los informes y el tratamiento de defectos propuestos.

Revisiones del diseño
Revisión crítica del diseño

Realiza una revisión crítica del diseño, donde se presenta una vista general del diseño técnico.

Integrantes:

- Analista (s), quien colabora para definir los req. del sistema.
- Diseñador (es) del sistema.
- Un moderador (solo coordina), un secretario (no se involucra).
- Diseñador (es) de programas para este proyecto.
- Probador del sistema.
- Analista que escribirá la documentación del sistema.
- Otros desarrolladores (entregan perspectiva externa).

Revisiones del diseño
Revisión crítica del diseño

Este grupo es más técnico que el anterior. Ya que la revisión trata de aspectos técnicos.

El moderador conduce la reunión para que se traten dos puntos: si el diseño implementa todos los requerimientos y si es un diseño de calidad.

Usando diagramas, datos o ambas cosas, se explican las estrategias de diseño alternativa y como y porque se han tomado las principales decisiones de diseño.

Si se identifican problemas mayores el diseño se rehace.

Revisiones del diseño
Revisión del diseño de programas

Cuando el diseño técnico resulta satisfactorio, los diseñadores de programas estarán en posición de interpretarlo como el conjunto de descripciones de diseño para los componentes reales, que deben ser codificados y probados.

Después de completar los diseños de programas, pero antes de comenzar la codificación, presentan sus planes.

Integrantes:

- Analista (s), que produjeran los req. del sistema.
- Diseñador (es) del sistema.
- Diseñador (es) del programa.
- Un moderador (solo coordina), un secretario (no se involucra).
- Diseñador (es) de programas para este proyecto.
- Probador del sistema.
- Analista que escribirá la documentación del sistema.
- Otros desarrolladores (entregan perspectiva externa).

Revisiones del diseño
Revisión del diseño de programas

Este proceso se centra en la detección de defectos más que en su corrección. Además se esta evaluando el diseño no a los diseñadores.

El proceso beneficia a todos al encontrar defectos y problemas cuando aún son fáciles y poco costosos de corregir.

Documentando el diseño

Una parte de la documentación esta dirigida a clientes y usuarios, en lenguaje natural para describir que es lo que el sistema hará.

La segunda parte usa la terminología técnica para escribir la estructura del sistema, datos y funciones.

Pauta de los informes:

- **Justificación racional del diseño**: donde se delinean las cuestiones criticas y compromisos que fueron considerados en la generación del diseño.

- **Descripción de los componentes del sistema:** una de las secciones debería indicar como interactúan los usuarios con el sistema incluyendo:

Documentando el diseño

- Menús y otros formatos de presentación en pantalla.
- Interfaces hombre – maquina.
- Formatos de los informes.
- Entrada: sobre los datos.
- Salida: sobre los datos.
- Características funcionales generales.
- Exigencias de perfomance.
- Procedimientos de archivos.
- Enfoque del tratamiento de defectos.

Por lo general, un conjunto de diagramas o de notaciones formales describe la organización y estructura global del sistema, incluyendo todos los niveles de abstracción.

Documentando el diseño

3. Finalmente se realiza un referencia cruzada del diseño y los requerimientos.

Bibliografía

Guía del Tópico:

- Software Engineering 6a. ed.– Ian Sommerville – Pearson Education – 2000.
- Ingeniería de Software Teoría y Práctica – Shari Lawrence Pfleeger – Pearson Education – 2002.

Solo referencial:

- Ingeniería de Software: Un enfoque práctico - Roger S. Pressman - Mc Graw Hill – 2002.