Ingeniería del Software II
Parte II
Ingeniería del Software de los Sistemas Basados en Componentes Y de Distribución de Información

Tema1
Introducción al Desarrollo de Software Basado en Componentes

Índice

- **Conceptos del Software Basado en Componentes.**
- **Problemas del Diseño e Implementación de Componentes.**
- **Interconexión entre Componentes.**

Conceptos de los Sistemas Basados en Componentes (CBD)

- **¿Qué es el Desarrollo Basado en Componentes?**
  - ✓ Montaje de nuevos sistemas software (aplicaciones) a partir de elementos constructivos (componentes) y marcos de trabajo estándar y reusables.
  - ✓ En este sentido se puede buscar una analogía con el desarrollo de elementos o circuitos electrónicos complejos a partir de componentes o circuitos integrados base.
  - ✓ Los componentes deberían de representar un estado de maduración (industrialización) del proceso de generación del software.

Conceptos de los Sistemas Basados en Componentes (CBD)

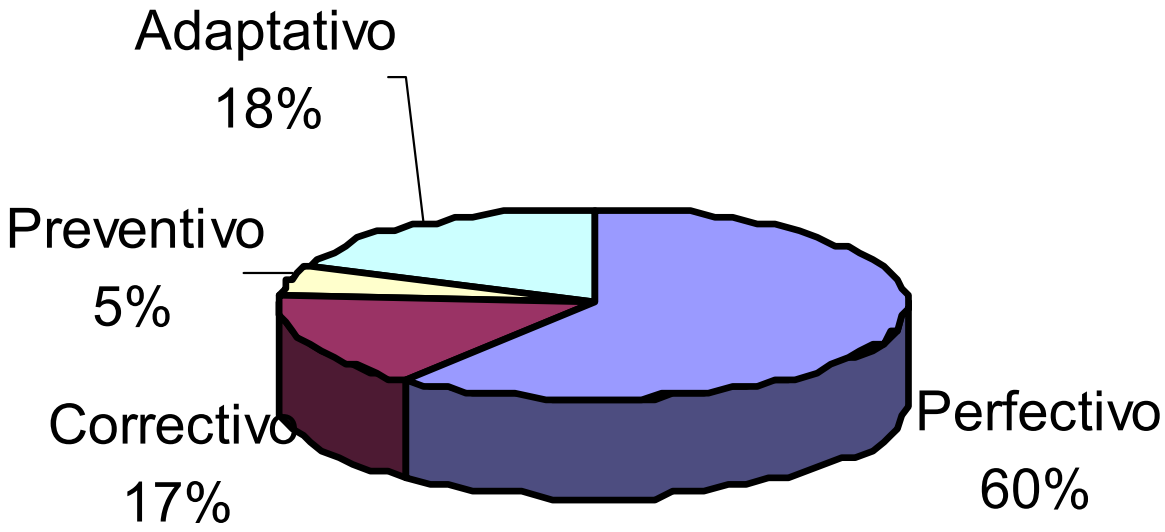
- **¿Son necesarios los CBD's?**
  - ✓ El desarrollo de sistemas software tiene ya una historia de casi 50 años.
  - ✓ En todo este tiempo las necesidades de desarrollo de nuevas aplicaciones han ido por delante de la aparición de las prácticas de desarrollo de software.
  - ✓ Como resultado la industria del software ha estado en permanente crisis todo este tiempo.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **¿Son necesarios los CBD's?**
  - ✓ Como consecuencia de esto los problemas típicos que nos encontramos son:
    - Los grandes sistemas software se vuelven extremadamente complejos.
      - Los sistemas monolíticos presentan problemas de diseño, desarrollo y son muy problemáticos de mantener.
      - La robusted y fiabilidad es un problema en estos grandes sistemas.
    - Es complicada la reutilización de activos adquiridos durante un proyecto en otro (ej. Conocimientos de diseño, software).
    - Adaptar el sistema software a los requerimientos específicos de distintos usuarios es problemático.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **¿Son necesarios los CBD's?**
  - ✓ Sin embargo si recordamos las acciones de mantenimiento perfectivo (adecuaciones a necesidades específicas del usuario) son los más solicitados.



Conceptos de los Sistemas Basados en Componentes (CBD)

- **¿Cómo puede el CBD paliar estos problemas?**
  - ✓ Los componentes puede ofrecer elementos constructivos robustos y reusables entre aplicaciones.
  - ✓ Los componentes puede provenir de fuentes distintas (fabricación propia, adquisición, etc.) y pueden catalogarse para su reutilización.
  - ✓ Los marcos de trabajo con componentes por su parte pueden ofrecer una aproximación estándar a la arquitectura de sistemas software complejos.
  - ✓ Los sistemas de componentes pueden ofrecer una base natural para la actualización, mantenimiento, particularización del software.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Pero cuidado, los componentes no son las respuesta a todos los problemas.**
  - ✓ A pesar de la tecnología de componentes, los grandes sistemas serán complejos, dificiles de diseñar, implementar y mantener.
  - ✓ El hecho de tener componentes no elimina la necesidad de utilizar buenas prácticas de ingeniería del software. En algunos casos incluso las hace más necesarias.
  - ✓ Es necesario tener en cuenta que muchas de las propiedades (reutilización, ensamblaje, etc) ya las propugnan las tecnologías de desarrollo de software orientadas a objetos.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Definición de Componente Software (Szypersky).**
  - ✓ Un componente software es una unidad de composición con interfaces especificados en forma contractual y con dependencias del contexto solamente explicitas. Un componente software debe poder ser desarrollado independiente y estar sujeto a su composición por terceras partes.
- **Una definición más intuitiva (D'Souza & Wills):**
  - ✓ Se trata de una unidad de software que puede ser entregada de forma independiente y encapsula su diseño e implementación, ofreciendo un interfaz al exterior a través de la cual puede ser compuesto con otros elementos para formar un todo mayor.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Algunos aspectos a tener en cuenta de las definiciones anteriores**
  - ✓ **Composición o montaje**:
    - Los componentes son bloque constructivos.
    - Los componentes son unidades de servicio en paquetadas.
    - Los componentes puede ser ensamblados entre sí para formar sistemas más complejos.
  - ✓ **Independencia**
    - Los componentes deben ser entregables de forma independiente:
      - Debe ser autocontenidos.
      - Deben encapsular complemente su comportamiento básico.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Algunos aspectos a tener en cuenta de las definiciones anteriores**
  - ✓ **Interfaces especificados de forma contractual:**
    - Los interfaces solamente pueden ser accedidos a través de su interfaz
    - El interfaz consistirá en un conjunto métodos u operaciones identificados.
    - La semántica (comportamiento) de cada operación deber ser especificad de forma precisa.
    - Los interfaces pueden ser especificados como un "contrato" entre el componente y el cliente.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Algunos aspectos a tener en cuenta de las definiciones anteriores**
  - ✓ Encapsulación
    - Todos los detalles de la estructura interna están ocultos
    - Los componentes son solamente accesibles a través del interface..
    - Los componentes son autocontenidos.
    - Visibilidad de Caja Blanca vs Visibilidad de Caja Negra
      - Caja Blanca. Usuario tiene acceso al código fuente del componente.
      - Caja Negra. El código del componente no está disponible.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Algunos aspectos a tener en cuenta de las definiciones anteriores**
  - ✓ Solamente dependencias del contexto explicitas.
    - Todos los requisitos acerca del entorno en el cual el componente debe funcionar deben estar especificados de forma explícita:
      - Sistema operativo.
      - Representación de datos.
      - Hardware.
      - ....
    - En este sentido algunos marcos de trabajo con componentes como CORBA, COM y JavaBeans ofrecer un entorno contextual que oculta el resto de dependencias.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Comparación entre Componentes y Clases (objetos)**
  - ✓ Granulariad.
    - Los componentes pretenden ser ofrecidos de forma independiente luego deben ser de grano más grueso.
    - La jerarquía de clases empaqueta grupos de unidades funcionales pequeños y fuertemente cohesionados.
  - ✓ Los componentes se enfocan a la definición de interfaces y el montaje en lugar de a la herencia y la jerarquía más propia de las clases.
  - ✓ Teniendo en cuenta el lenguaje y la localización en el caso de los componentes esta debería ser transparente.

Conceptos de los Sistemas Basados en Componentes (CBD)

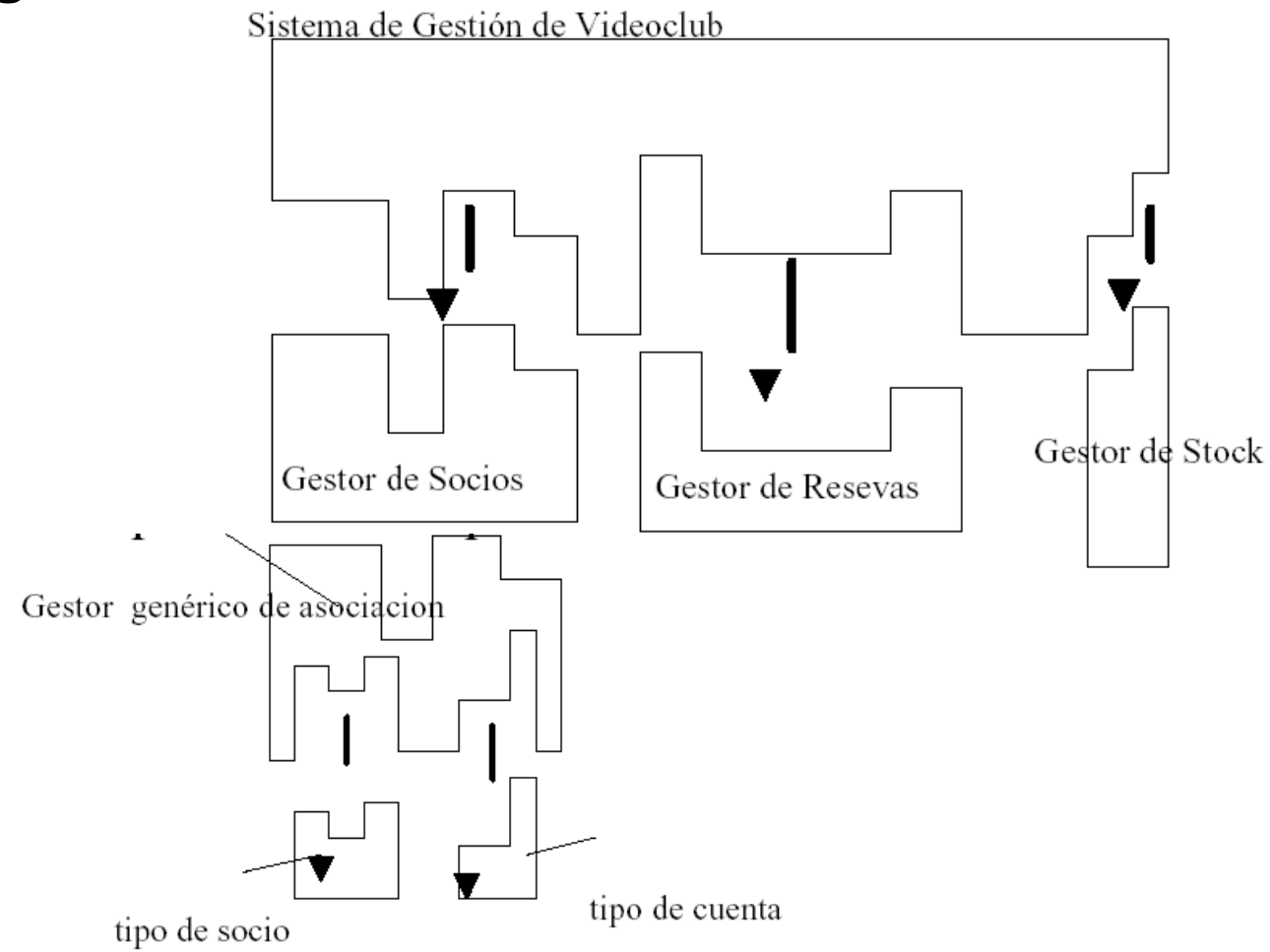
- **Comparación entre DOO y CBD**
  - ✓ OOD no conduce necesariamente a sistemas bien estructurados.
  - ✓ Las jerarquías de clases no necesariamente ofrecen una buena base para la reutilización del software.
  - ✓ CBD esta basado en una perspectiva arquitectónica, concebir un sistemas en términos de sus componentes 'físicos' y sus interconexiones.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Algo más acerca de la composición**
  - ✓ Los componente puede ser '*enchufados*' de diferentes formas:
    - Los componentes pueden ser integrados en grandes aplicaciones desarrollando software externo que utilice y coordine distintos servicios.
    - Un componente genéricos pude ser especializado 'enchufandole' subcomponentes que ofrecen distintas versiones de la funcionalidad básica del componente.

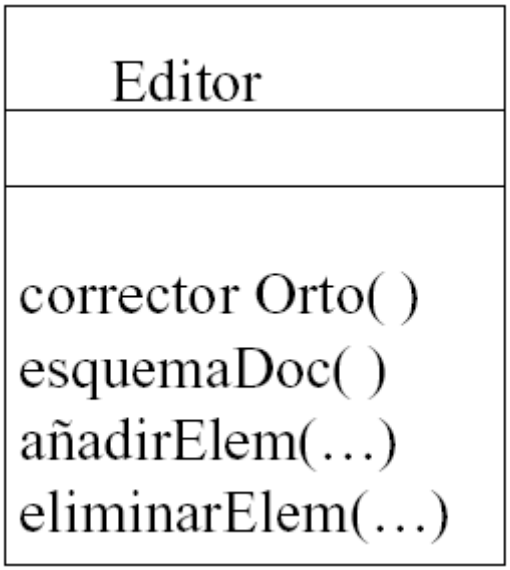
Conceptos de los Sistemas Basados en Componentes (CBD)

- **Algo más acerca de la composición**



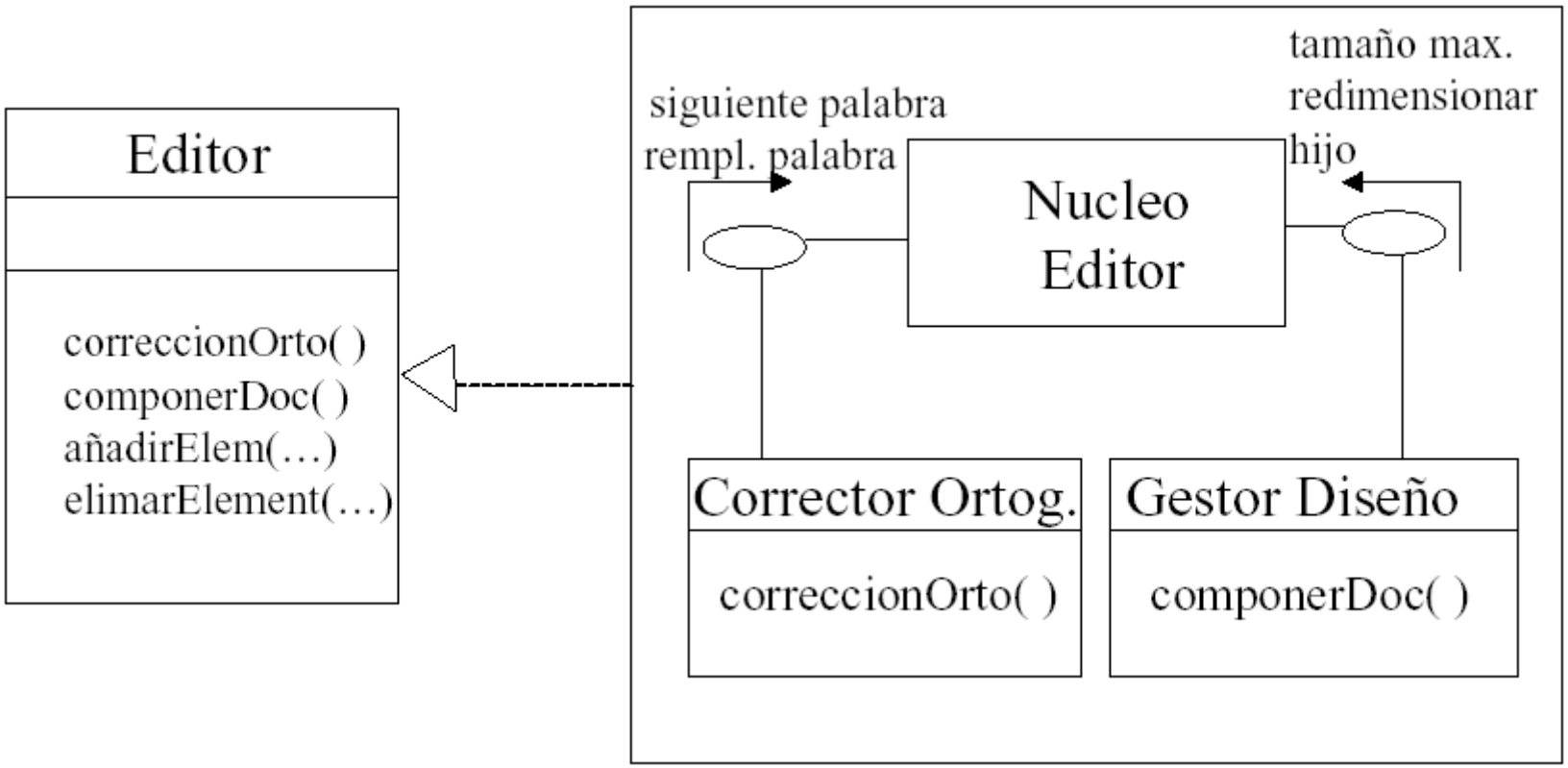
Conceptos de los Sistemas Basados en Componentes (CBD)

- **Otro ejemplo de Componente. Supongamos un editor de texto con las siguientes necesidades:**



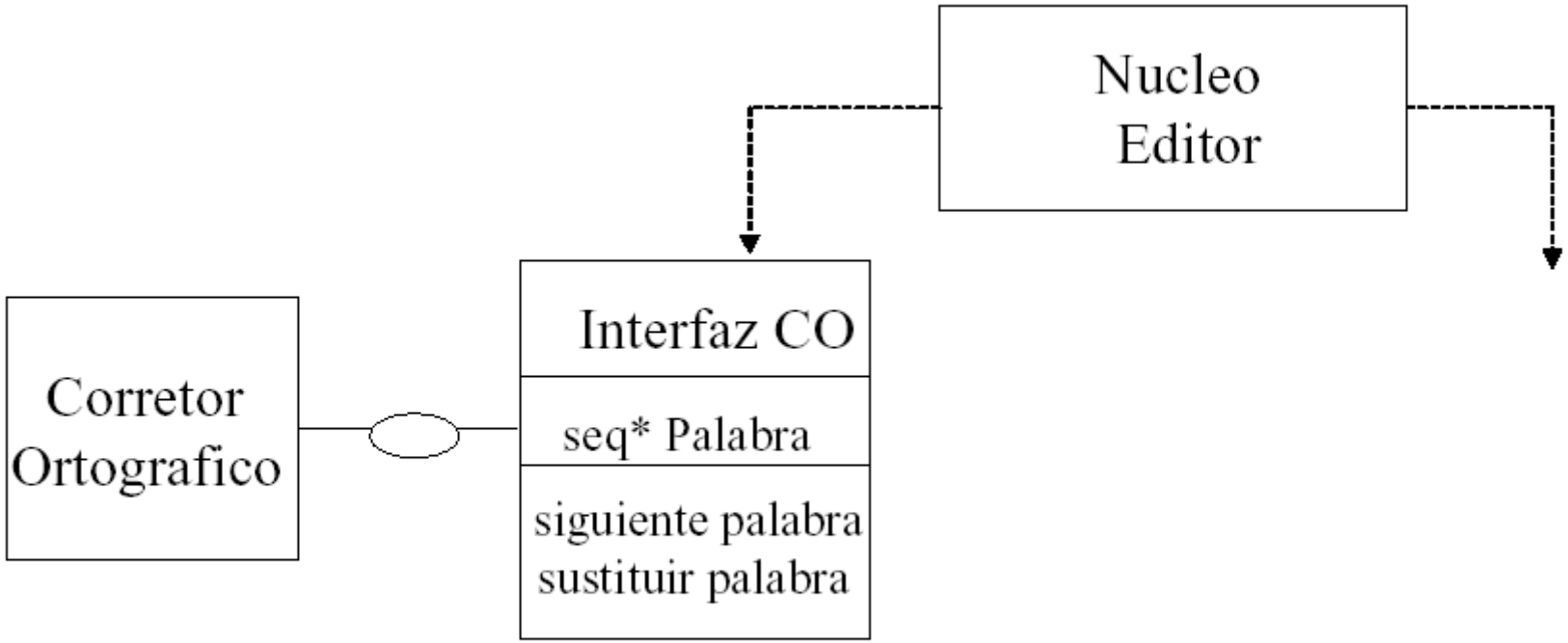
Conceptos de los Sistemas Basados en Componentes (CBD)

- **Otro ejemplo de Componente. Podríamos desarrollar este editor en función de varios compoentes:**



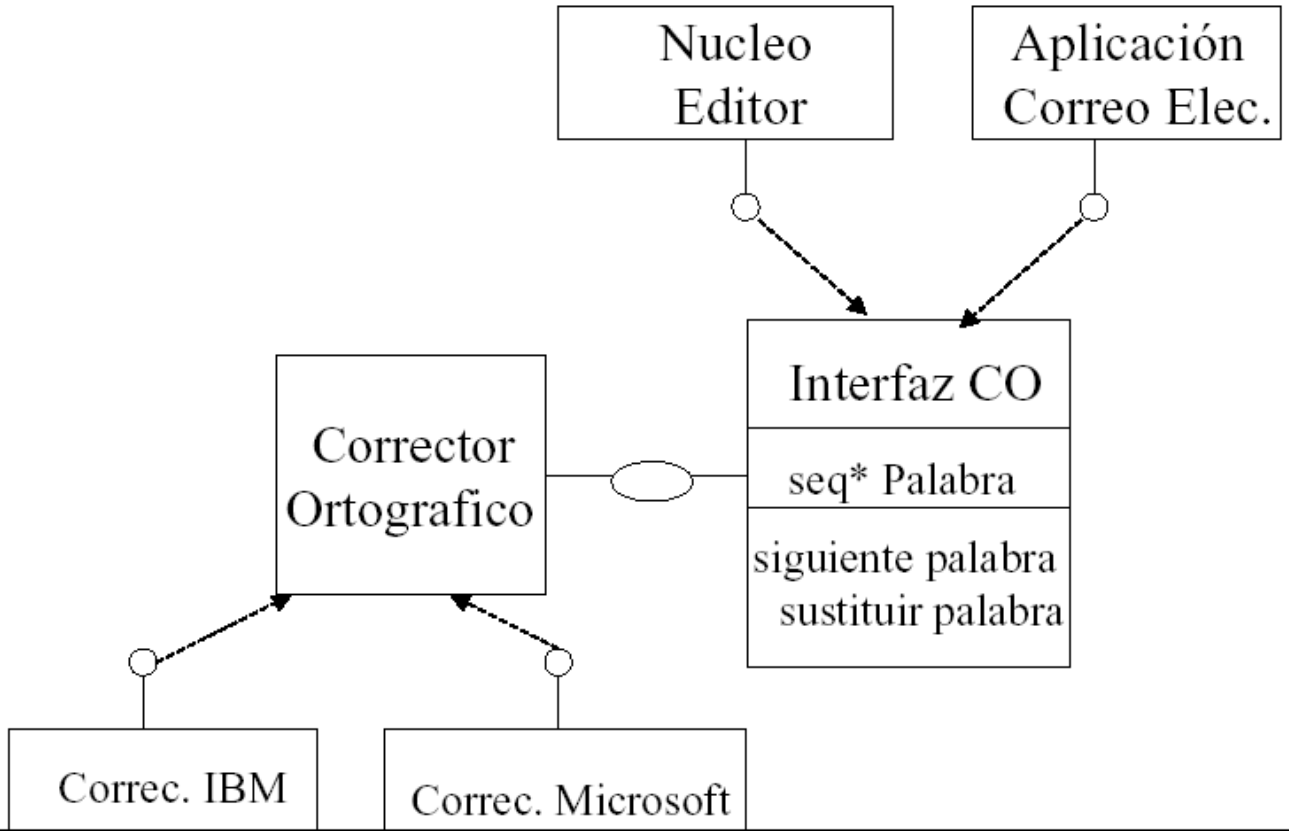
Conceptos de los Sistemas Basados en Componentes (CBD)

- **Otro ejemplo de Componente. La conexión entre el núcleo del editor y el corrector ortográfico se realiza a través de la interfaz del corrector.**



Conceptos de los Sistemas Basados en Componentes (CBD)

- **Otro ejemplo de Componente. Este modelo ofrece la ventajas de que podemos enchufar distintas aplicaciones o sustituir unos correctores por otros..**



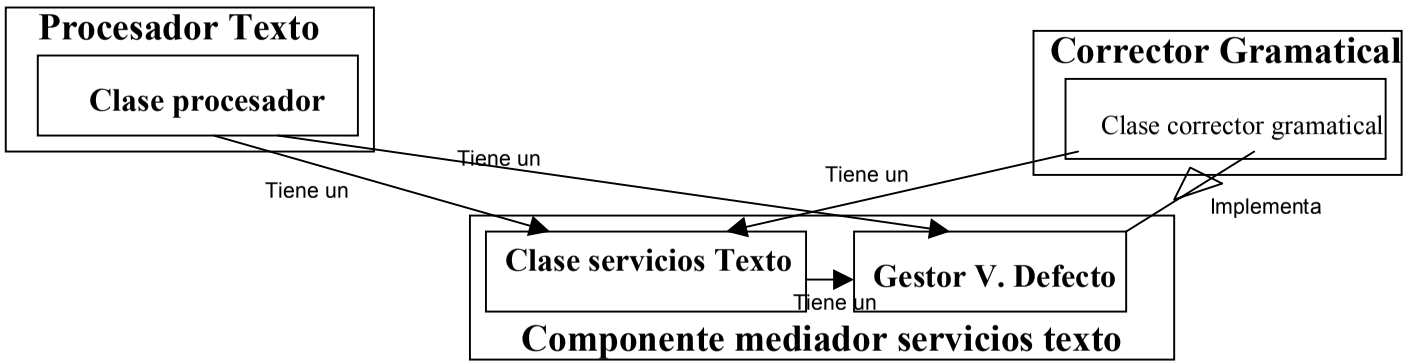
Conceptos de los Sistemas Basados en Componentes (CBD)

- **Los Interfaces y su especificación**
  - ✓ Como hemos visto uno de los elementos fundamentales de los CBD es la correcta especificación de los interfaces ya que son el único punto de conexión entre componentes.
  - ✓ El interfaz proporciona un conjunto de operaciones que pueden ser invocadas por los clientes.
  - ✓ Alguna terminología asociada a este concepto es la siguiente:
    - *Proveedor o Servidor*: es la entidad (componente) que implementa las operaciones de un interfaz.
    - *Cliente o usuario*: es la entidad (tal vez componente) que invoca los servicios ofrecidos por el interfaz.

Conceptos de los Sistemas Basados en Componentes (CBD)

Los Interfaces y su especificación

- ✓ Se suele hablar de dos tipos principales de interfaces:
  - Interfaces Directos (procedurales). La implementación y el interfaz pertenecen al mismo objeto.
  - Interfaces Indirectos(objetos). Se decide que un único objeto haga de intermediario (mediador de servicios) entre varios interfaces y un cliente.
- ✓ Los interfaces indirectos son cómodos pero necesitan un buen control de versiones



Conceptos de los Sistemas Basados en Componentes (CBD)

- **Los Interfaces y su especificación**
  - ✓ También hemos indicado anteriormente que la semántica (el comportamiento) de cada interfaz viene dado por una ***especificación***.
  - ✓ Dicha especificación es usada tanto por el cliente como por el proveedor:
    - Servidor: define los requisitos para el comportamiento a ser implementado por el componente.
    - Cliente: Define el comportamiento que puede esperar del componente integrado.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Aspectos de la Especificación.**
  - ✓ Firma de una operación(*signature*): Define la sintaxis de una operación:
    - Nombre.
    - Parámetros.
    - Tipos.
  - ✓ Especificación Funcional. Define el procesamiento (comportamiento computacional) de la operación representada.
  - ✓ Especificación no Funcional. Define otros requisitos:
    - Prestaciones.
    - Almacenamiento
    - Otros requisitos.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Especificación por Contrato**
  - ✓ El concepto de contrato se entiende como una obligación entre el servidor y el cliente.
    - Especifica las reglas que restringen el uso de un servicio (u operación en un servicio) por parte del cliente. ***Precondiciones***
    - Especifica el resultado garantizado para una invocación legal de la operación. ***Postcondiciones.***
  - ✓ Entre versiones de un componente este no puede cambiar las condiciones del contrato. Como máximo podría pedir menos y dar más.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Especificación por Contrato**
  - ✓ Los contratos puede ser especificados a través de métodos de especificación formal o semiformal
    - Aserciones lógicas.
    - Invariantes
  - ✓ Derivados a partir de los conceptos asociados a los tipos abstractos de datos (ADT).

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Precondiciones y Postcondiciones en los contratos.**
  - ✓ Si consideramos un operación **A**
  - ✓ Una "formula correcta" para **A** se expresaría como una tripleta de "Hoare" (*Hoare triple*):
    - **{P}A{Q}**
    - **Lo que significa que: cada ejecución de A, donde al principio se cumpla P, terminará con un estado donde se cumplira Q.**
    - **En esta terminología a P y Q se les llama aserciones.**
      - **P son las llamadas precondiciones de A**
      - **Q son las llamados postcondiciones de A.**

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Precondiciones y Postcondiciones en los contratos.**
  - ✓ Algunos ejemplos de formulas en este formato pueden ser:
    - **{x>=9} x:=x+5{x>=13}**
    - **¿Por qué 13 y no 14?**
    - **Si pusiesemos {X>=14} diriamos que es una poscondición más fuerte.**
    - **Mientras que si pusiesemos {x>=8} hablaríamos de una precondición más débil.**
  - ✓ Esto enlaza con lo que antes decíamos de dar más por menos. Las precondiciones débiles y postcondiciones fuertes favorecen al cliente y obligan a hacer componentes más robustos y mejores.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Precondiciones y Postcondiciones en los contratos. Vemos un ejemplo**
  - ✓Consideremos una clase pila con las siguientes operaciones:
    - **int max(), capacidad máxima de la pila.**
    - **int numEle(), numero de elemntos**
    - **bool vacia(), no hay elementos.**
    - **bool llena(), alcanzada capacidad máxima.**
    - **item cima(), devuelve el elemento superior.**
    - **void apilar(item i), apila el elemento i.**
    - **void desapilar(), elimina el elemento de la cabeza.**

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Precondiciones y Postcondiciones en los contratos. Vemos un ejemplo**
  - ✓Precondiciones:
    - **Antes de apilar(), la pila no debe estar llena.**
    - **Antes de desapilar(), la pila no debe estar vacía.**
    - **Antes cima(), la pila no debe estar vacía.**
  - ✓ **Postcondiciones.**
    - **Después de apilar(item i), la pila no debe estar vacia; el elemento en la cima debe ser i; el numero de elementos se debe haber incrementado en uno.**
    - **Después de desapilar(), la pila no debe estar llena; el numero de elementos debe haberse decrementado en uno.**

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Precondiciones y Postcondiciones en los contratos. Vemos un ejemplo. Especificación semiformal**

```
interface Pila{
   int max();
   //  pre true
   // post result == maximo tamaño de pila.
   int numEle();
   // pre true;
   // post 0<=result<=this.max() y
   // result   == numero elemento en la pila.
   boolenan vacia()
   //pre true
   //post result ==(this.count()==0)
   boolean llena();
   //pre true
   //post result == this.max()
```

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Precondiciones y Postcondiciones en los contratos. Vemos un ejemplo. Especificación semiformal**

```
....
   item cima();
   //  pre !this.vacia();
   // post result == elemento cima de la pila
   void apila(item i);
   //[tam:int·
   // pre tam=this.numEle():tam<this.max();
   // post !this.vacia() y  this.numEle()== (tam+1)
   // y this.cima==i;
   //]
   void desapila()
   //[tam:int·
   //pre tam = this.numEle();!this.vacia();
   //post !this.numEle() == this.max() y this.numEle()== (tam-1)
   //]
```

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Los Invariantes.**
  - ✓ Los invariantes son afirmaciones que describen propiedades globales que deben ser ciertas para todo objeto o componente.
    - Por ejemplo si tienes una clase o Cuenta_Corriente que mantiene una lista_de_depositos, lista_de_retiradas y un balance.
    - Un invariante de esta clase seria:

```
Lista_de_depositos.total-lista_retiradas.total=balance.
```

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Los Invariantes.**
  - ✓ El problema en los sistemas de componentes es quién debe ser responsable de mantener los invariantes.
  - ✓ Esto puede resolverse reforzando tanto las precondiciones como las postcondiciones manera que nadie contribuya a romper la validez de un invariante:
    - {Inv y Pre}A{Inv y post}
  - ✓ Volviendo al ejemplo de la pila algunos invariantes podrian ser:
    - 0<= numEle
    - numEle<= máximo admitido en la pila
    - vacio ==(numEle==0)

Conceptos de los Sistemas Basados en Componentes (CBD)

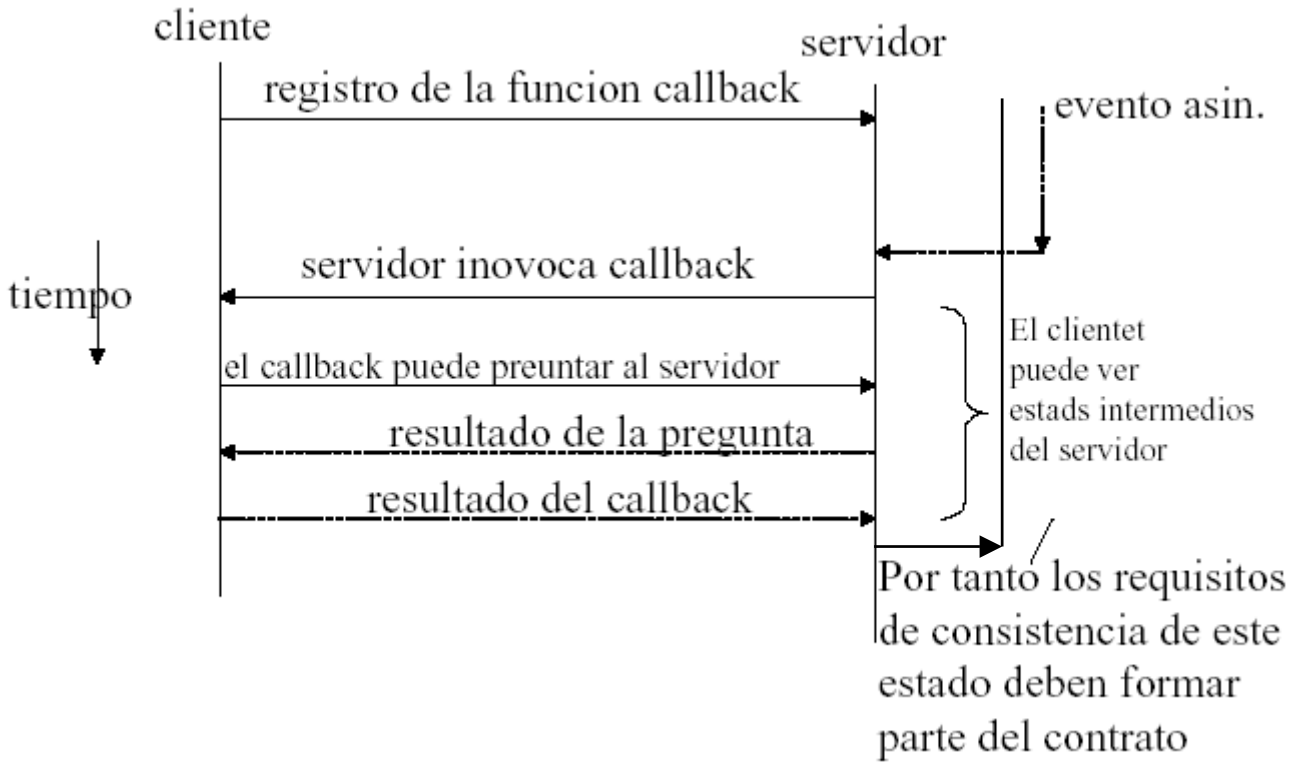
- **Algunos problemas con los Contratos**
  - ✓ Aunque los interfaces contractuales son sencillos para bibliotecas procedurales y bibliotecas de clases simples algunas estructuras que se utilizan en los marcos de trabajo con componentes puede complicar el asunto:
    - Funciones de *Callback*.
    - Re-entrada y recursividad.
    - Concurrencia
    - Mecanismos de registro y notificación de objetos.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Funciones de Callback**
  - ✓ Se trata de referencias a un procedimiento que se pasa (registrada en) un componente (servidor) por parte del cliente. Normalmente la asociación se produce a través de algún evento reconocido por parte de la entidad suministradora.
  - ✓ De esta manera el procedimiento es activado cada vez que el evento se produce.
  - ✓ Este tipo de mecanismos son comunes en los sistemas de desarrollo de IPC's (X-windows/Motif).

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Funciones de Callback.**
  - ✓ Diagrama de ejecución de una función de llamada .



Conceptos de los Sistemas Basados en Componentes (CBD)

- **Funciones de Callback.**
  - ✓ En una función de callback el servidor realmente llama al cliente.
    - Esto viola la noción de interacción cliente-servidor que es vista siempre vista como un evento instantáneo (indivisible) por parte del cliente.
    - En este proceso el estado del servidor puede ser visto en cualquier instante arbitrario por parte del cliente.
    - Por tanto los contratos deben garantizar la validad de los estados observables del servidor durante el tiempo que el callback esta activo.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Veamos un Ejemplo de este problema sobre un interfaz sencillo para manejar operaciones sobre cadenas de texto.**
  - ✓ Las operaciones principales del interfaz serian:
    - int max(), longitud maxima de un elemento de texto
    - int longitud(), longitud del elemento de texto.
    - char leer(int pos), devuelve el carácter el la posición indicada
    - void inserta(int pos, char ch)
    - void borrar(int pos)
    - void registrarObs(observadorTexto x)
    - void eliminarObs(observadorTexto x)

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Veamos un Ejemplo de este problema sobre un interfaz sencillo para manejar operaciones de texto.**

```
interface cadenaTexto{
        int max();
        //  pre true
        // post result == maximo tamaño de elem texto
        int longitud();
        // pre true;
        // post 0<=result<=this.max() y
        // result      == longitud del elemento de texto.
        char leer(int pos)
        //pre 0<=pos<this.longitud();
        //post result ==carácter en la posición pos
```

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Continua el interfaz**

```
void insertar(int pos, char ch);
//[lon:int;txt:vector de char·
//  pre lon = this.longitud();(∀i;i<=lon:txt[i]=this.leer(i))
// lon<this.max() y 0<=pos<=lon;
// post this.longitud()==lon+1 y
// (∀i ;0<i<pos:txt[i]=this.leer(i)==txt[i])
// y this.read(pos)=ch;
// (∀i ;pos<i<this.longitud():this.leer(i)==txt[i])
//]
void borrar(int pos);
//[lon:int;txt:vector de char·
//  pre lon = this.longitud();(∀i;i<=lon:txt[i]=this.leer(i))
//y 0<=pos<=lon;
// post this.longitud()==lon-1 y
// (∀i ;0<i<pos:txt[i]=this.leer(i)==txt[i])
// y (∀i ;pos<=i<this.longitud():this.leer(i)==txt[i])
//]
void registrarOb(observadorTexto x);
void eliminarOb(Observador Texto x);
}
```

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Continua el Ejemplo**
  - ✓ El interfaz anterior tiene un par de cosillas al final que nos hemos dejado a posta, estas sirven para registrar objetos de tipo observadores de texto.
  - ✓ Estos observadores tienen el privilegio de ser informados cada vez que se produce un cambio en el texto.

```
interface observadorTexto{
        void NotificacionInserccion(int pos);
        //  pre carácter en la posición pos ha sido insertado
        // post result == maximo tamaño de elem texto
        void NotificacionEliminacion(int pos)
        //pre carácter que estaba en la posición pos ha sido
        //eliminado
}
```

  - ✓ Cualquier objeto que implemente este interfaz podrá registrase en el gestor de cadena de texto.

Conceptos de los Sistemas Basados en Componentes (CBD)

Continua el Ejemplo

- ✓ Un ejemplo de este tipo de objetos observadores puede ser un objeto Vista responsable de mostrar en pantalla el texto.

```
interface VistaTexto extends observadorTexto{
        cadenaTexto texto();
        //  pre true
        // post result != nulo
        int posicionCInsercion();
        //pre true;
        //post 0<=pos<=this.texto().longitud()
        void establecerCInserccion(int pos);
        //pre 0<=pos<=this.texto().longitud();
        //post this.posicionInsercion()==pos;
         int posEnCoordX(int pos);
        //pre 0<=pos<=this.texto().longitud();
        //post result == coordenada x correspondiente al cara pos.
```

Conceptos de los Sistemas Basados en Componentes (CBD)

Continua el Ejemplo

- ✓ Un ejemplo de este tipo de objetos observadores puede ser un objeto Vista responsable de mostrar en pantalla el texto.

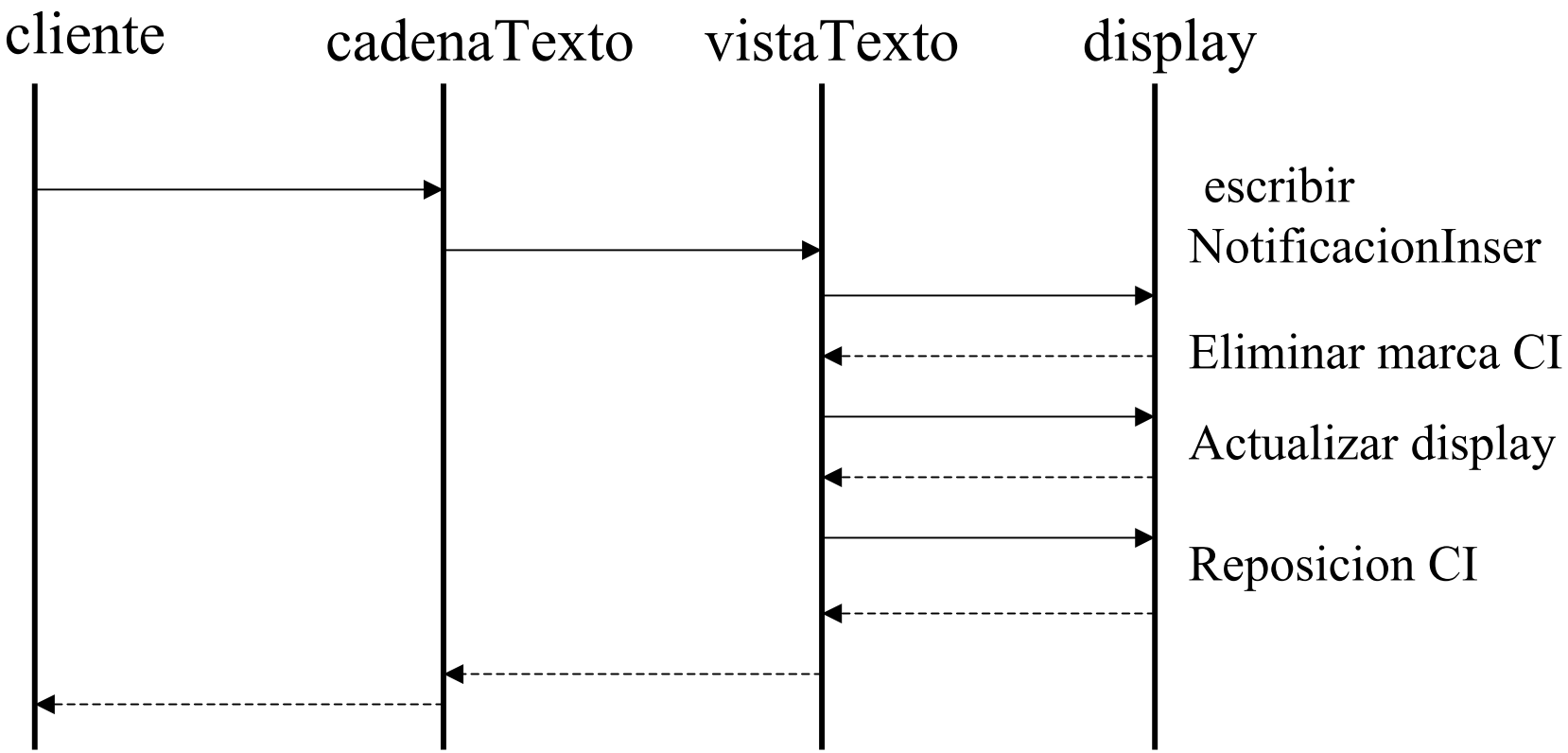
```
....
    int posEnCoordY(int pos);
   //pre 0<=pos<=this.texto().longitud();
   //post result == coordenada y correspondiente al cara pos.
   int posDesdeCoord(int x, int y);
   //pre (x,y) coordenadas validas.
   //post this.posEnCoordX(result)== x
   //       this.posEnCoordY(result)== y.
   void tecleaC(char ch);
   //[posCI:int ·
   //pre posCI=this.PosicionCInserccion():
   //this.texto().longitud()<this.texto().max()
   //equiv this.texto().inserta(posCI, ch)
   //post this.posicionCInserrcion()==PosCI+1
   ]
}
```

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Continua el Ejemplo.**
  - ✓ El componente vistaTexto se invoca para actualizar las vista después de que el texto ha cambiado.
  - ✓ vistaTexto necesita realizar varios pasos para completar la actualización.
  - ✓ Durante este tiempo pueden producirse inconsistencias entre el estado dibujado y el estado real del texto.
  - ✓ Que versión del estado debería mostrar vistaTexto a quienes estuviesen utilizando sus métodos.

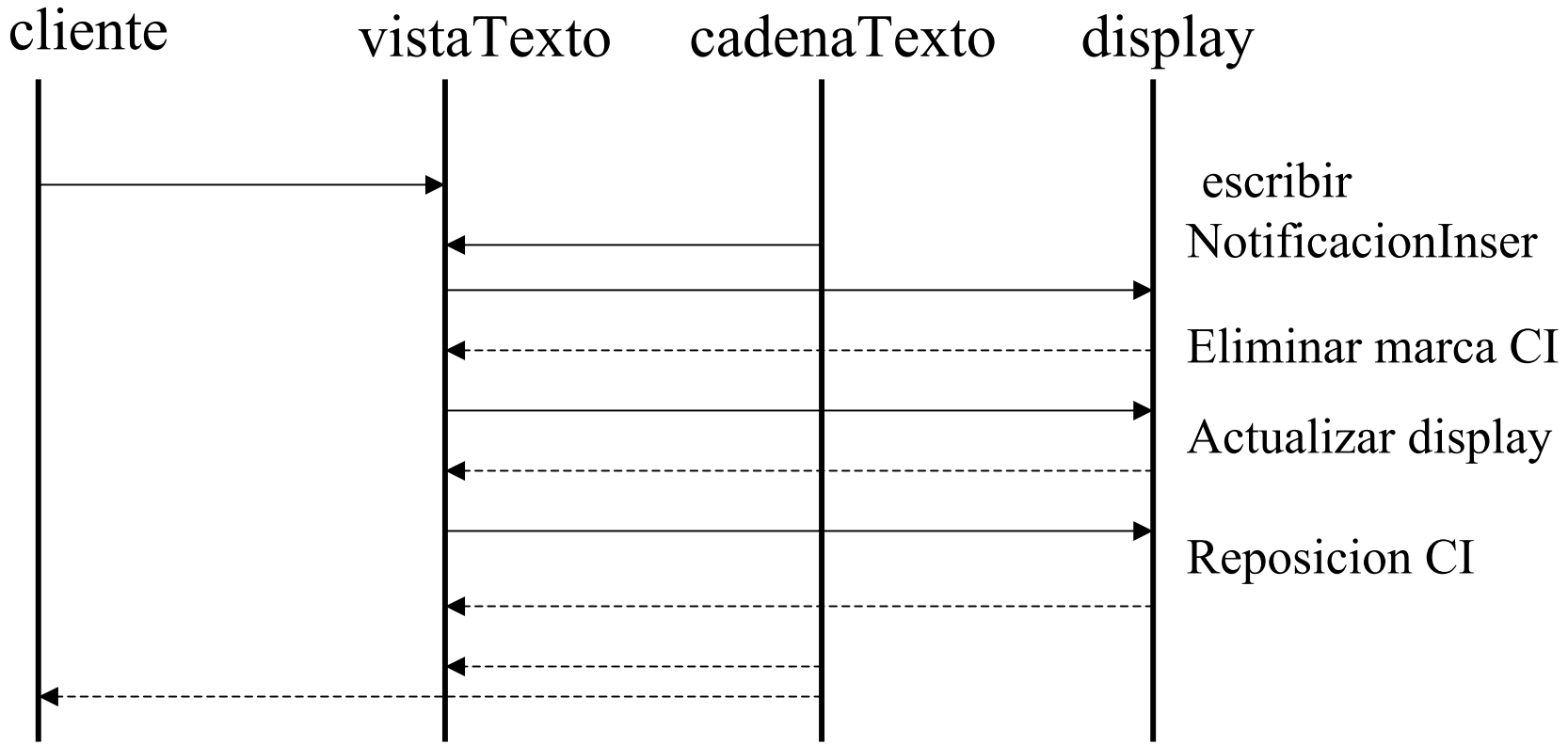
Conceptos de los Sistemas Basados en Componentes (CBD)

- **Continua el Ejemplo. Secuencia de ejecución sobre un cliente**



Conceptos de los Sistemas Basados en Componentes (CBD)

- **Continua el Ejemplo. La secuencia de ejecución mejor ordenada seria**



Conceptos de los Sistemas Basados en Componentes (CBD)

- **Criterios Adicionales sobre lo que esperamos de los sistemas CBD.**
  - ✓ Extensibilidad
    - Que nos sea fácil adaptar los productos a los cambios de especificación.
    - Simplicidad y descentralización.
  - ✓ Reusabilidad
    - La habilidad de los elementos software para construir diferentes aplicaciones.
  - ✓ Compatibilidad
    - Facilidad para combinar elementos software entre si.(CORBA, OLE-COM, JavaBeans).
  - ✓ Eficiencia
    - Capacidad de un sistema software de exigir los menores requerimientos de recursos hardware posible, memoria, etc. Ofreciendo las mayores prestaciones.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Criterios Adicionales sobre lo que esperamos de los sistemas CBD.**
  - ✓ Portabilidad
    - Que se sencillo transferir los productos software entre varios entornos hardware y software.
  - ✓ Facilidad de Uso
    - Que las especificaciones y formas de uso sean simples para distinto tipo de desarrolladores de software que sean capaces de integrar estas soluciones.
  - ✓ Costes Reducidos.
    - Minimizar los trabajos de integración.
    - Mediante la reutilización y uso de componentes externos reducimos los costes de mantenimiento.

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Todo lo anterior exige una cuidada Modularidad en el diseño:**
  - ✓ La modularidad ofrece posibilidades de reutilización cuando los **módulos** están **autocontenidos** y organizados en arquitecturas estables.
  - ✓ Los métodos de desarrollo de software deben producir un sistema hecho a partir de componentes autónomos **conectados** por una **estructura de interfaces** coherentes y simples.
  - ✓ Se debe tender a la **protección de componentes o módulos**. En la arquitectura que se produzca las condiciones de error en tiempo de ejecución deben permanecer confinadas al componente en el que suceden.
  - ✓ **Reducir el número de interfaces**: cada componente o módulo debería comunicarse con el mínimo numero de interfaces posible.
  - ✓ **Interfaces Reducidos**: Si dos componentes o módulos se comunican, deberían intercambiar la menor información posible (garantiza la continuidad y protección).

Conceptos de los Sistemas Basados en Componentes (CBD)

- **Todo lo anterior exige una cuidada Modulariad en el diseño:**
  - ✓ **Interfaces Explícitos**: siempre que dos componentes A y B se comunican esto debe estar recogido explícitamente en su interfaz.
  - ✓ **Ocultación de Información:** El diseñador debe seleccionar la mínima cantidad de información posible del módulo para hacerla disponible al resto de componentes. Además aunque estemos en un entorno de *caja de cristal o caja blanca* los clientes no deben ser escritos confiando en lo que sucede en la parte secreta del componente.
  - ✓ **Principio de Apertura-Cierre.** Los componentes deben ser abiertos y cerrados:
    - Abierto implica que es posible su extensión.
    - Cerrado significa que la conexión con otros componentes queda reducida a su interfaz.
    - Cerrado implica que es un elemento binario que ya está compilado.

Arquitecturas Cliente-Servidor base de los CBD

- Dentro de un arquitectura C/S se distinguen unos componentes principales, que se pueden distribuir entre varias máquinas:
  - Componente de Interfaz de Usuario y Presentación.
  - Componente de Aplicación.
  - Componente de Gestión de la Información.

- Las funcionalidades pueden recaer más en el cliente (*cliente principal o grueso*) o en el servidor (*servidor principal o grueso*) dependiendo del tipo de aplicación.
- Distinguiendo donde se concentra la carga de computación se habla de 5 configuraciones del modelo C/S:
  - Presentación Distribuida.
  - Presentación Remota.
  - Lógica Distribuida.
  - Gestión de datos Remota.
  - Bases de datos distribuidas.

Arquitecturas Cliente-Servidor base de los CBD

- Las reglas lógicas para la distribución de la carga entre cliente y servidor dicen que:
  - El componente de presentación/interacción debe ubicarse en el lado del cliente.
  - Si es necesario compartir la información entre varios usuarios, ésta deberá localizarse en el servidor.
  - Los datos estáticos deben localizarle en el lado del cliente para minimizar flujos innecesarios en las comunicaciones.

Arquitecturas Cliente-Servidor base de los CBD

Consideraciones de Ingeniería de Software sobre Arquitecturas Cliente/Servidor.

- Aplicables las mismas fases en el proceso de generación del software: análisis, diseño, implementación y prueba.
- Especial atención al diseño de los datos, que ahora pueden estar distribuido por el sistema.
- Tendencia a emplear modelos orientados a objetos para el diseño.
- Comprobaciones del sistema modulares (cliente, servidor, transacciones, comunicaciones, uso de perfiles funcionales)