Diseño de Software y Principios del Diseño

- Ténicas avanzadas en diseño de Software -

- Dra. Marcela Capobianco -

Copyright

Qué es el diseño arquitectónico

- Las arquitecturas han existido desde el primer programa que usó módulos.
- Historicamente se han dado en forma implícita
- Concierne la organización del sistema como un conjunto de componentes, las propiedades de estos componentes y su interrelación.

Nivel del diseño arquitectónico

- Organización del sistema como un conjunto de componentes,
- control global,
- protocolos de comunicación,
- acceso a los datos,
- distribución física,
- escalabilidad,
- performance...

Contenidos

1) Arquitectura del software
2) Patrones
3) Distintos estilos arquitectónicos
4) Análisis de diseños alternativos
5) Conversión de los requisitos en una arquitectura de software

Utilidad del Diseño Arquitectónico

- Proporciona un panorama completo del sistema a construir.
- Permiten una descripción del sistema a través de constructores y patrones muchas veces no soportados
- Comienza con el diseño de datos y luego deriva una o mas representaciones alternativas

Cómo se realiza la descripción

- La descripción es típicamente informal
- Usa patrones de lenguajes que han evolucionado a través del tiempo
- Ejemplo: “Camelot está basado en el modelo *cliente servidor* y usa *llamadas de proceso remoto* en forma local y remota para proveer comunicación entre aplicaciones y servidores”
- Significa esto que no son útiles?

Por qué es importante

- Al nivel arquitectónico se pueden evaluar propiedades como capacidad, consistencia y compatibilidad entre los componentes
- Los elementos de la descripción son definidos en forma independiente para poder resusarlos en distintos contextos
- Permiten la resusabilidad en varios niveles
- Resulta crucial elegir el estilo correcto

Estilos arquitectónicos

- Se establece una analogía entre las construcciones y el desarrollo de software
- La descripción se realiza a nivel informal
- El comportamiento del sistema sigue un patrón idiomático bien reconocido entre los desarrolladores
- Se han ido generando a través de los años

Estilos arquitectónicos

- El uso de patrones y estilos de diseño se puede apreciar en múltiples disciplinas de ingeniería
- La estandarización de las formas comunes de diseño muestra la madurez del campo
- El software también posee estilos: client-server system, pipe-filter design, arquitectura por capas...

Estilos arquitectónicos

- Para analizar los distintos estilos veremos a la arquitectura como una colección de componentes junto con una descripción de su interacción (conectores)

clientes, servidores, filtros, niveles, bases de datos

llamadas a proc., broadcast de eventos, protocolos de BD, pipes

Qué define un estilo arquitectónico

- Vocabulario sobre los componentes y tipos de conectores,
- restricciones en la forma que pueden combinarse
- uno o más modelos semánticos que especifican como determinar las propiedades globales del sistema a partir de sus partes

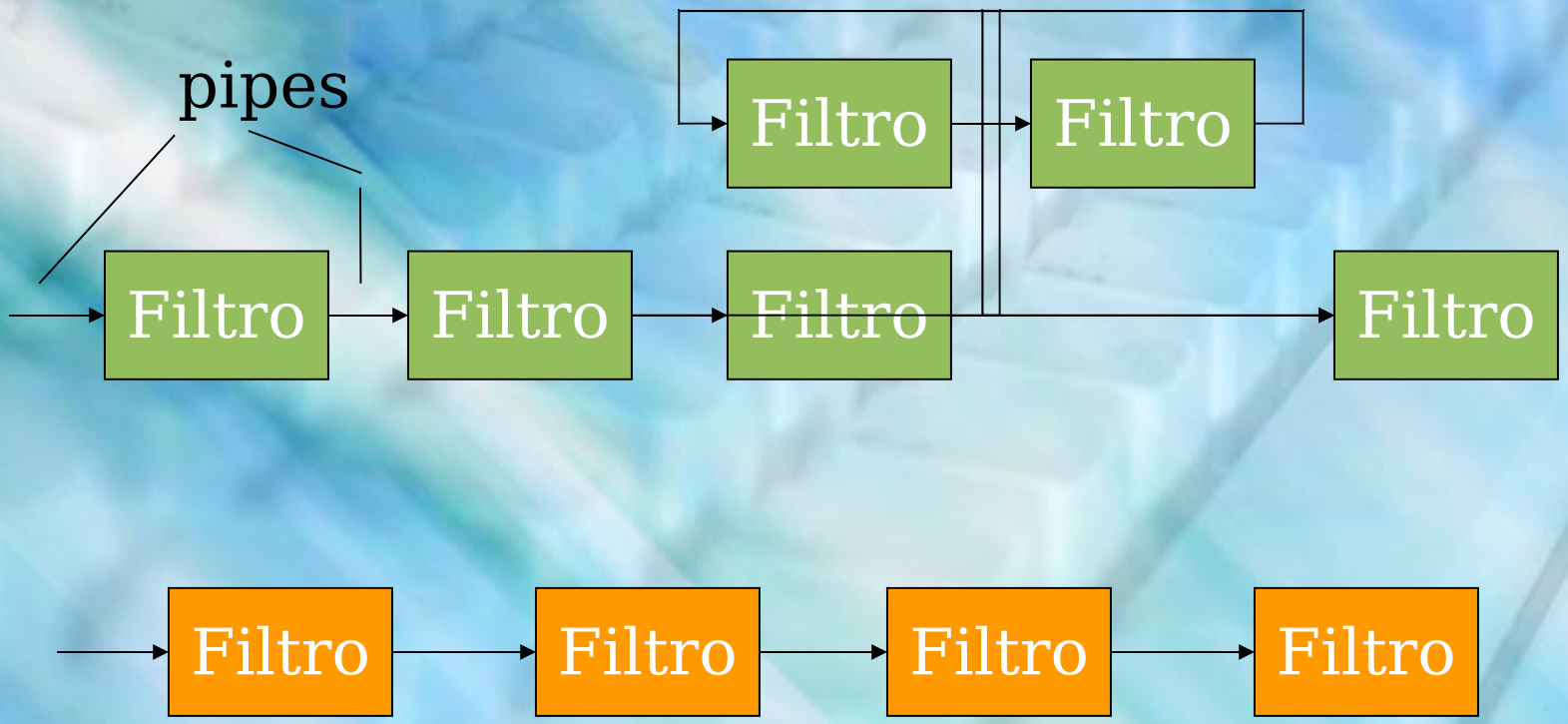
Estilos más comunes

- Sistemas de flujo de datos: batch secuencial, pipes y filtros
- Llamada y retorno: programa principal y subrutina, sistemas OO, jerárquicos
- Componentes indep.: sistemas de eventos, comunicación entre procesos
- Máquinas virtuales: intepretes, sistemas basados en reglas
- Repositiorios de datos: bases de datos, sistemas de pizarrón, hipertexto.

Pipes y filtros

- Cada componente tiene un conjunto de inputs y outputs
- La salida comienza antes de consumir la entrada (flitro)
- Los conectores sirven como conductos para los datos (pipes)
- No se conoce la identidad de los otros filtros.
- Ej: programas en unix shell, compiladores

Pipes y filtros

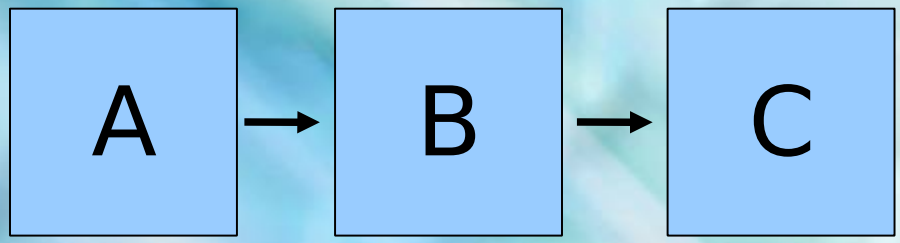


Sistemas OO

- Los componentes son objetos
- Los objetos interactuan a través de llamadas a funciones y procedimientos
- Cada objeto es responsable de mantener su integridad
- La representación está oculta entre los objetos
- Se debe conocer la identidad de los objetos

Sistemas OO

- Cuando la identidad de un objeto cambia se debe modificar a quienes lo invocan
- Puede también haber problemas con los efectos colaterales



Basados en eventos

- Usan técnicas de integración sin llamadas explícitas
- Invocación implícita, integración reactiva, broadcast selectivo
- Al producirse un evento el sistema invoca a todos los componentes registrados
- Quien anuncia los eventos no sabe a quien afectará

Basados en eventos

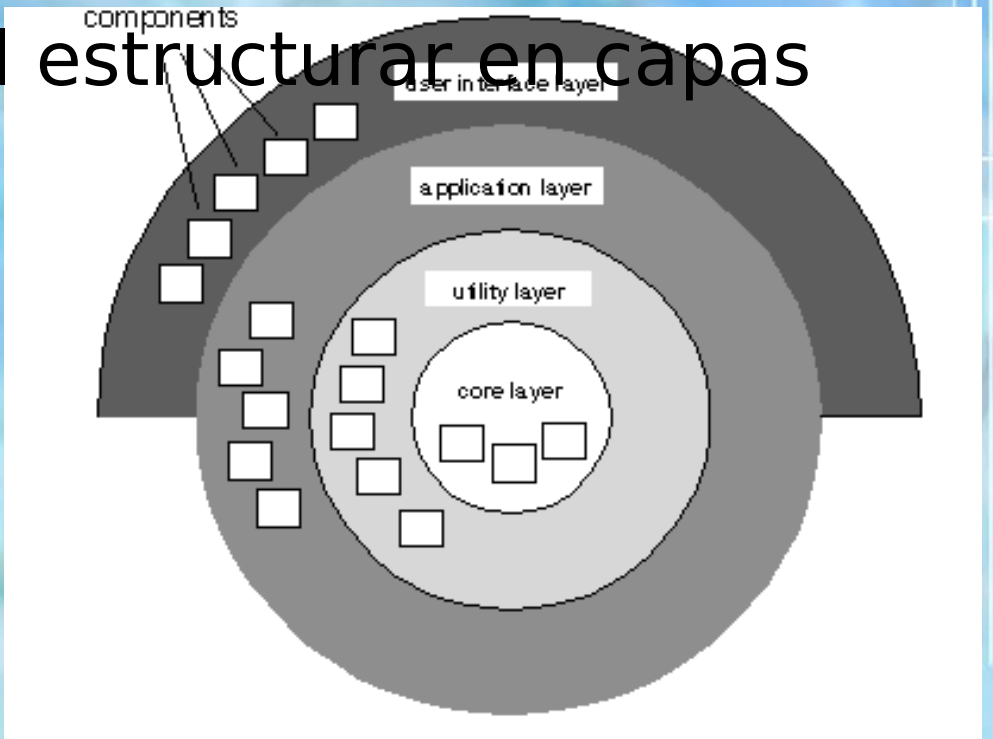
- No se pueden hacer suposiciones sobre el orden o el resultado del procesamiento
- Ej: interfases de usuarios para separar el contenido de la apariencia , DBMS para chequeos de consistencia
- Bueno para la reusabilidad
- Problemas: intercambio de datos, es difícil garantizar correctitud

Sistemas en capas

- Organización jerárquica, cada capa provee servicios a su superior y es un cliente para la inferior
- Las capas interiores pueden estar ocultas (restricciones)
- Ej: protocolos de comunicación
- Permiten particionar conceptualmente problemas complejos
- Favoren la reusabilidad

Sistemas en capas

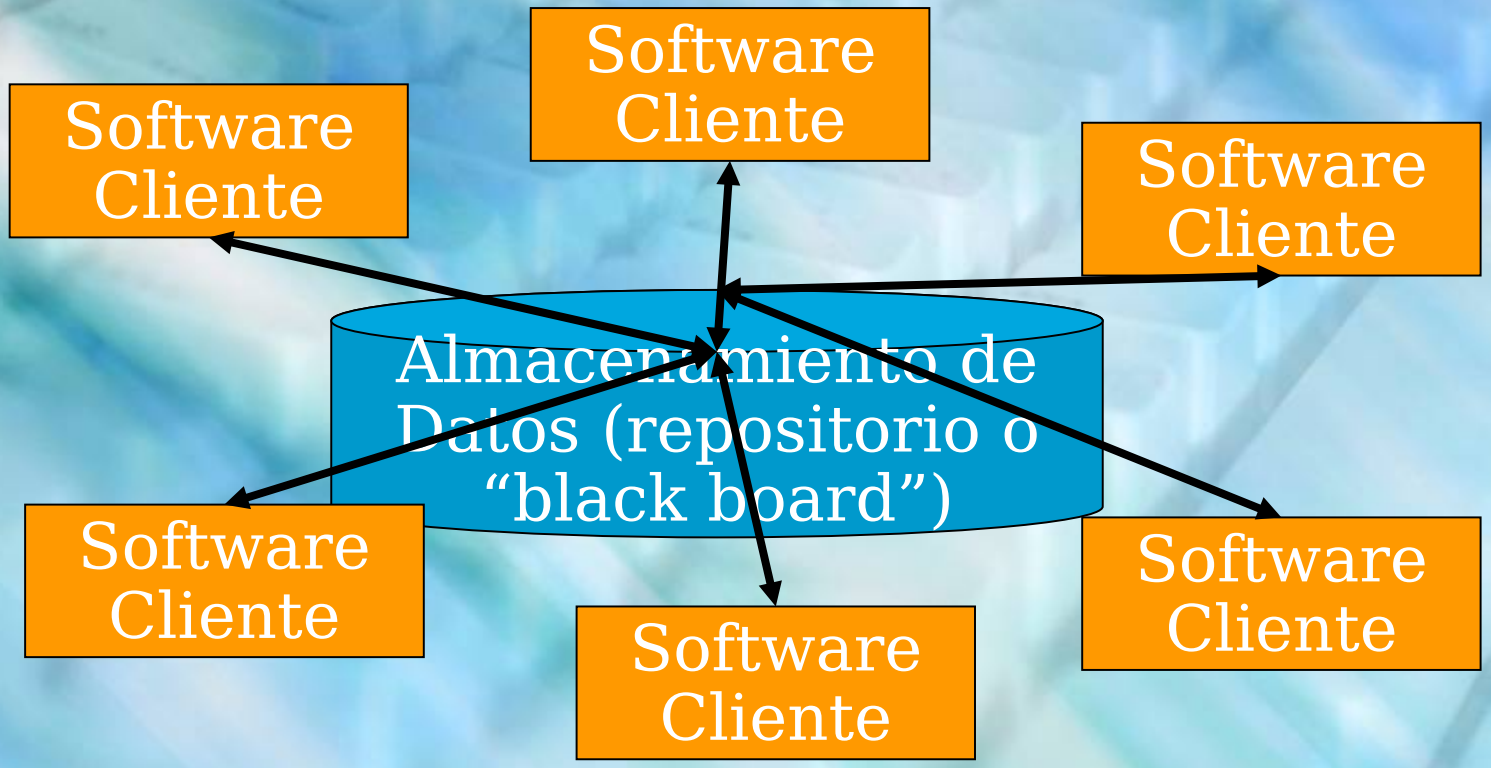
- Permiten definir interfases estándares
- Ej: OSI ISO y XWindow
- No siempre es fácil estructurar en capas



Repositorios

- Dos tipos de componentes: estructura central de datos y colección de componentes independientes
- La forma de interacción puede variar de acuerdo al sistema
- El repositorio central puede ser una base de datos tradicional o un blackboard dependiendo del tipo de control

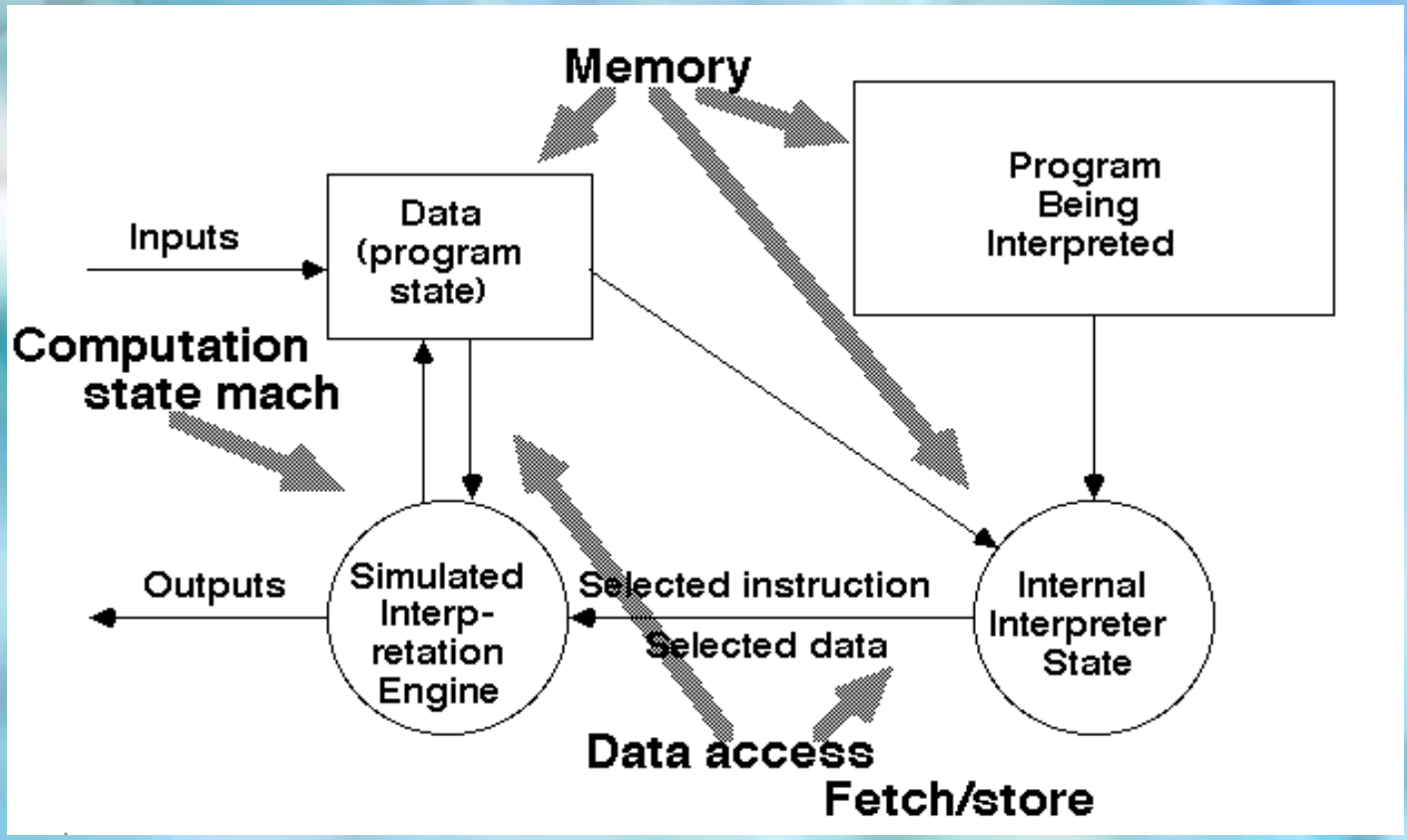
Repositorios



Intérpretes

- Se produce una máquina virtual por software
- Posee cuatro componentes:
  - máquina de interpretación,
  - memoria que contiene el pseuso código a ser interpretado
  - representación del estado de control de la máquina de interpretación
  - representación del estado de control del programa que está siendo simulado

Intérpretes



Otras arquitecturas...

- Procesos distribuidos
  - Ej: cliente/servidor
- Programa principal / subrutina
- De dominio específico
  - Ej: control de vehículos,
  - se pueden usar sistemas de generación semiautomáticos
- Sistemas de transición de estado (para sistemas reactivos)

Arquitecturas heterogeneas

- La mayoría de los sistemas poseen una combinación de estilos
- Combinaciones en forma jerárquica
  - A nivel de componentes
  - A nivel de conectores
- Permitir que un componente use distintos tipos de conectores
  - Ej: una base de datos e interacción a través de pipes

Cómo saber cual elegir

- No existen reglas fijas, la experiencia forma el juicio
- Es muy importante elegir un diseño adecuado
- Existen métodos para evaluar el diseño generado
  - Cualitativos
  - Cuantitavos (métricas)

Método de análisis para la arquitectura

1. Recopilación de escenarios (casos de uso)
2. Hacer aparecer todos los requisitos
3. Describir los patrones elegidos para derivar los escenarios y requisitos, usando
   1. vista de módulos
   2. vista de proceso
   3. vista de flujo de datos
4. Evaluar los atributos de calidad por separado (fiabilidad, seg., fac. de mantenimiento, etc)

Método de análisis para la arquitectura

1. Identificar la sensibilidad de los atributos de calidad con los diferentes atributos arq. en un estilo arquitectónico específico.
2. Análisis de las arquitecturas candidatas (paso 3) con la visión del paso 5.

Estado de la arquitectura de software

- Lenguajes de descripción arquitectónicos
- Codificación del conocimiento experto de los arquitectos
- Frameworks para dominios específicos
- Formalismos para razonar y evaluar las arquitecturas

Conclusiones

- La arquitectura de software es una disciplina que se encuentra en pleno desarrollo
- El desarrollo de un modelo arquitectónico presenta múltiples beneficios
- Es importante conocer los distintos patrones existentes y formar criterio con la práctica

Conversión de los requisitos en una arquitectura de SW

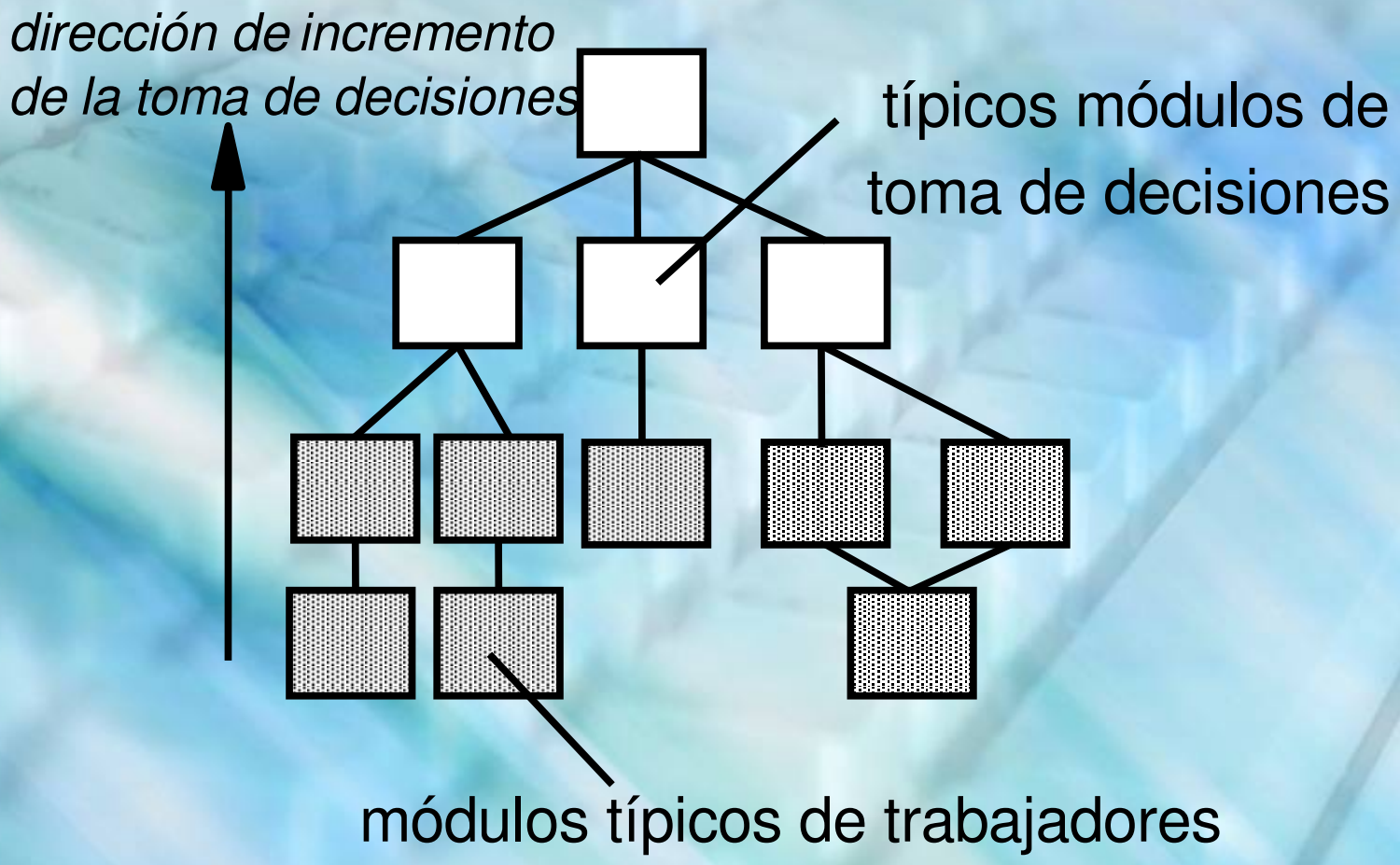
Conversión

- Los modelos arquitectónicos son muy diferentes
- No existe un único método que conduzca del análisis al diseño
- Para algunos modelos no existe un método de conversión y el diseñador debe realizarla en forma ad-hoc.
- Para ilustrar un enfoque de conversión usaremos la arquitectura de llamada y retorno.
- Esta técnica es también llamada diseño estructurado.

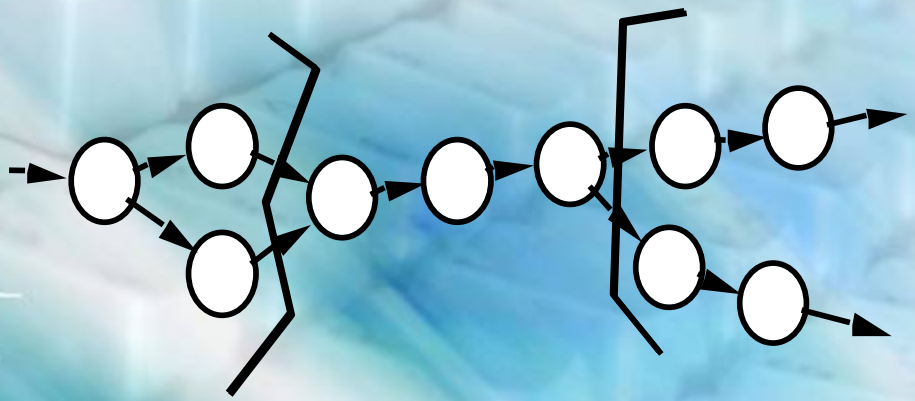
Diseño estructurado

- Objetivo: derivar una arquitectura de llamada y retorno que este particionada en forma adecuada.
- El DFD es mapeado en una arquitectura del sistema
- El DTS y la especificación de proceso son usados para indicar el contenido de cada módulo
- Notación: diagramas de estructuras

Diseño Estructurado



Flujo de transformación vs transaccional



Flujo de transformación

Flujo transaccional

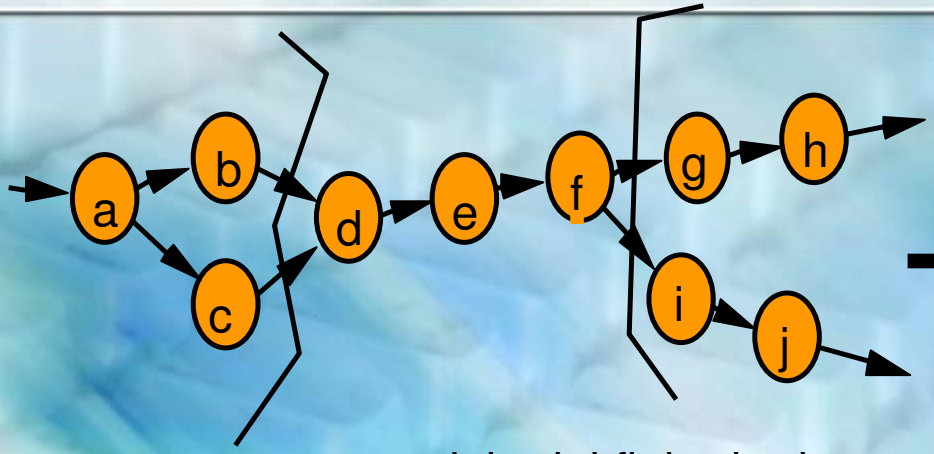
Análisis de las transformaciones

- Es un conjunto de pasos de diseño que permite convertir un DFD transformacional en una arq. de llamada/retorno
- El ejemplo está basado en el sistema Hogar Seguro (pág. 185, Pressman).

Pasos del diseño

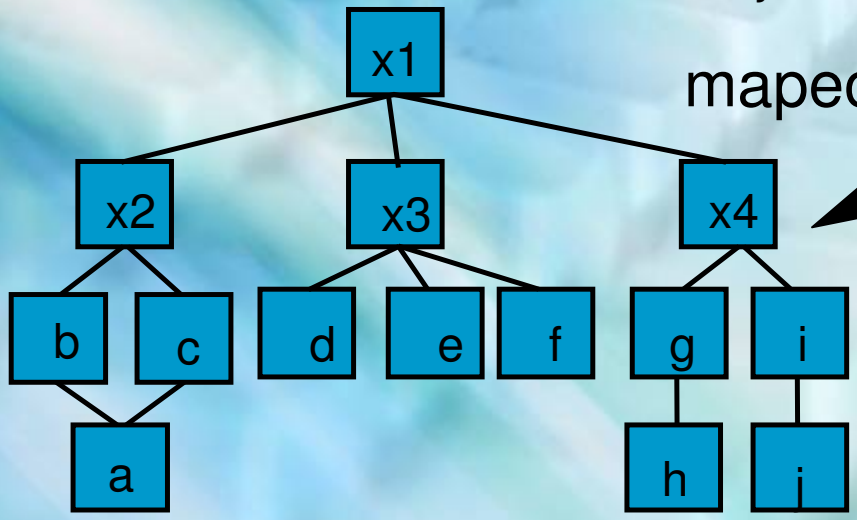
- Paso 1: revisar el modelo fundamental del sistema
- Paso 2: revisar y refinar los DFD del SW a fin de que las burbujas tengan una cohesión alta
- Paso 3: Determinar si el DFD es transformacional o transaccional
- Paso 4: aislar el centro de transformación especificando los límites de los flujos de entrada y salida
- Paso 5: realizar una “descomposición de primer nivel”

Pasos del diseño



modelo del flujo de datos

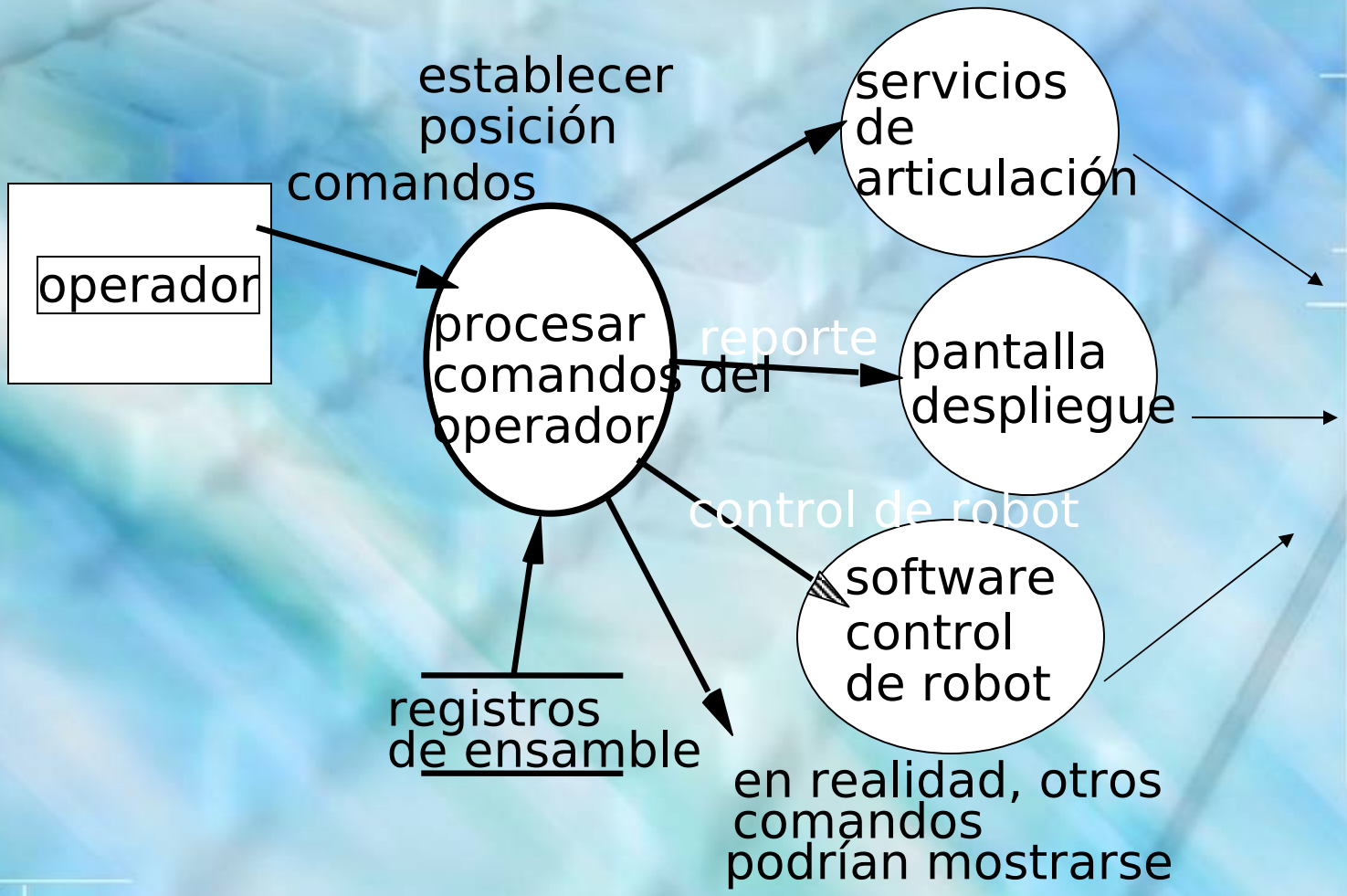
mapeo de transformación



Pasos del diseño

- Paso 6: realizar una descomposición de segundo nivel. Para cada módulo se debe escribir un breve texto que lo describa
- Paso 7: refinar la estructura inicial del programa usando heurísticas de diseño

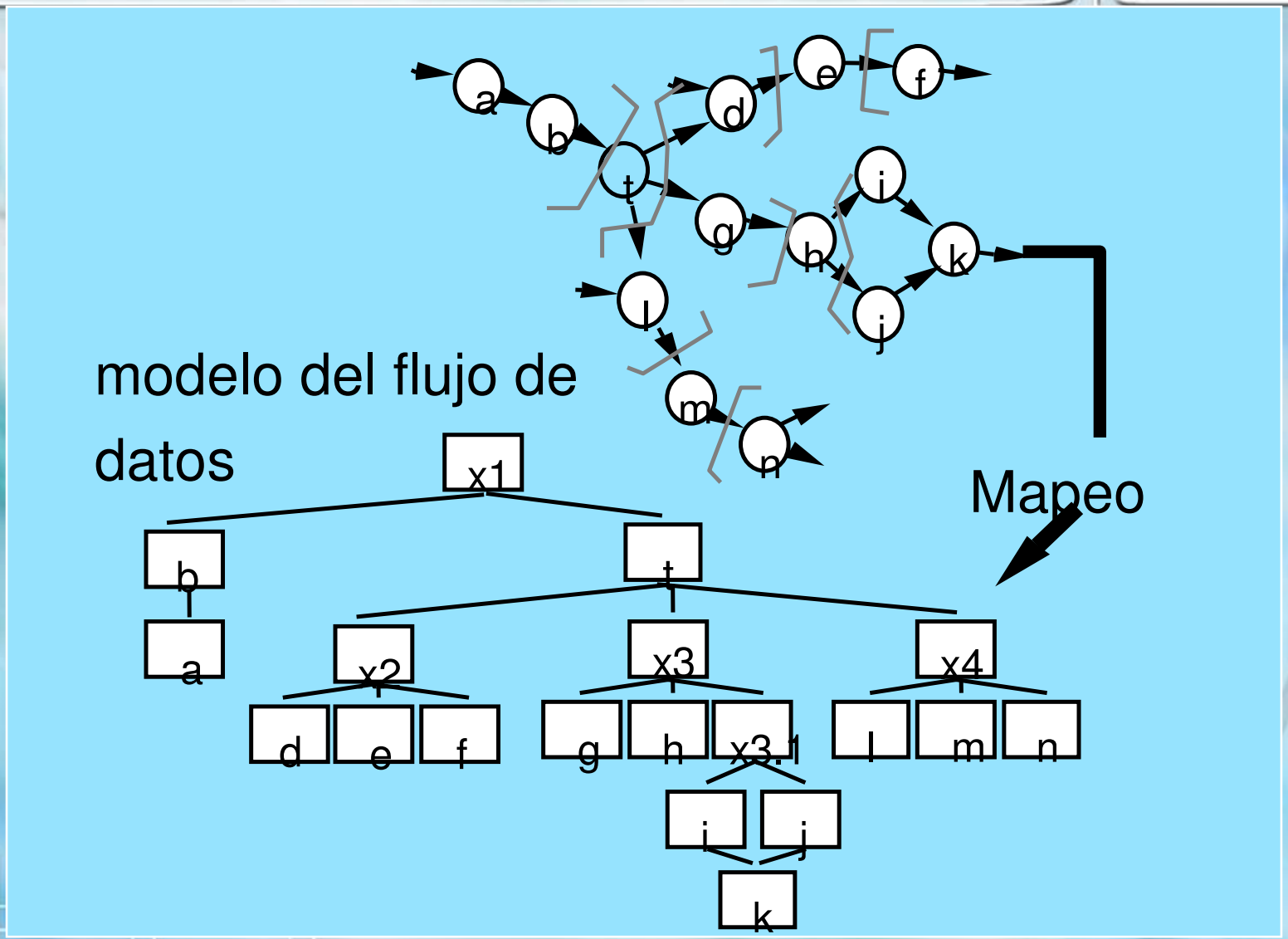
Análisis de las transacciones



Diferencia con transformacional

- Paso 1, 2 y 3 son iguales
- Paso 4: Identificar el centro de transacción y las características del flujo a lo largo de c/camino de acción
- Paso 5: transformar el DFD en una estructura de programa adecuada. Se usa un módulo de distribución que controla a todos los caminos subordinados a la transacción. Cada camino de acción del DFD se transforma separadamente.

Ejemplo de análisis transaccional



Diferencia con transformacional

- Paso 6: Descomponer y refinar la estructura de la transacción y la estructura de todos los caminos de acción
- Paso 7: refinar la arquitectura preliminar

Refinamiento

- Al finalizar se debe completar el resto de la información requerida:
  - Descripción del procesamiento de c/modulo
  - Descripción de la interfaz de c/modulo
  - Definición de las E de D
  - anotar limitaciones y restricciones del diseño
  - considerar otro paso de refinamiento
- Debe fomentarse el cambio en las primeras etapas

Bibliografía

- M. Shaw y D. Garlan: Software Architecture, perspectives on an emerging discipline
- R. Pressman: Software engineering

Casos de estudio

Cómo usar los estilos arquitectónicos

- Veremos distintos casos de estudio en los que se muestra como usar los estilos arquitectónicos para mejorar nuestra comprensión del SW
- Cada caso tiene distintos objetivos

Caso 1: KWIC

- El sistema de indexado “Key Word In Context” acepta un conjunto ordenado de líneas, cada línea es a su vez un conjunto ordenado de palabras y cada palabra un conjunto ordenado de caracteres. Cada línea puede sufrir un “shift circular”. El sistema debe devolver un listado de todos los “shift circulares” de las líneas en orden alfabético.

Caso 1: KWIC

- Veremos distintas formas de dividir el problema en módulos para ver las ventajas de cada una de ellas.
- El problema se puede implementar como un sistema pequeño pero no es de interés solamente teórico (pemuted index en Unix Man pages)
- Distintas descomposiciones modifican la capacidad de acomodar cambios del diseño.

Caso 1: KWIC

- Se consideran los siguientes cambios:
  1)cambios en el algoritmo de procesamiento
  2)cambios en la representación de los datos
  3)mejoras a la funcionalidad del sistema
  4)performance
  5)reusabilidad
- En base a estos se evalúan las soluciones

1.1 Programa ppal/ subrutina con datos compartidos

- Se descompone en base a 4 funciones: input, shift, alphabetize y output
- Estos componentes dependen de un programa principal
- Los datos se comunican mediante un almacenamiento común
- Se usa un protocolo irrestricto de lectura/escritura

1.1 Programa ppal/ subrutina con datos compartidos

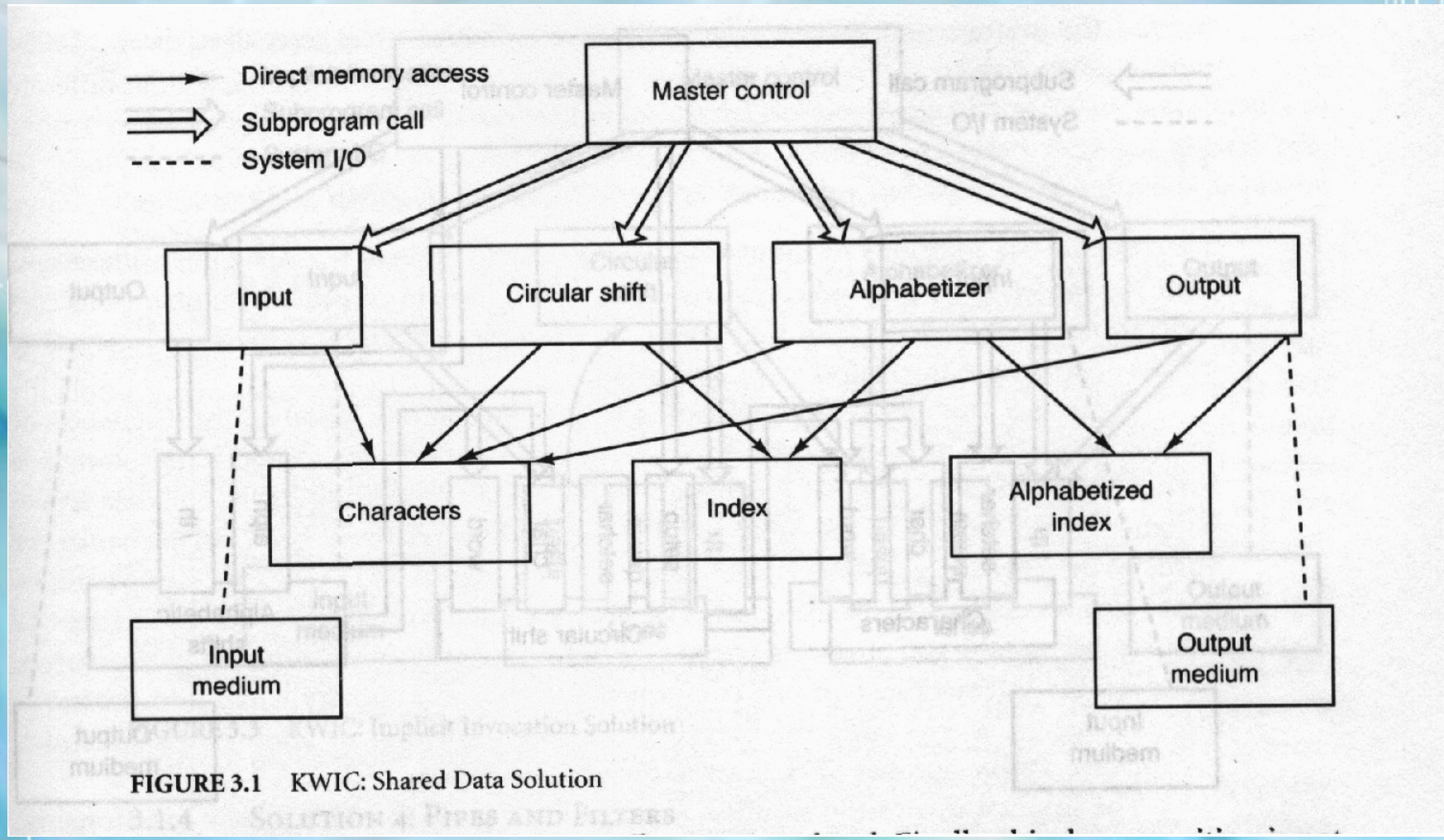


FIGURE 3.1 KWIC: Shared Data Solution

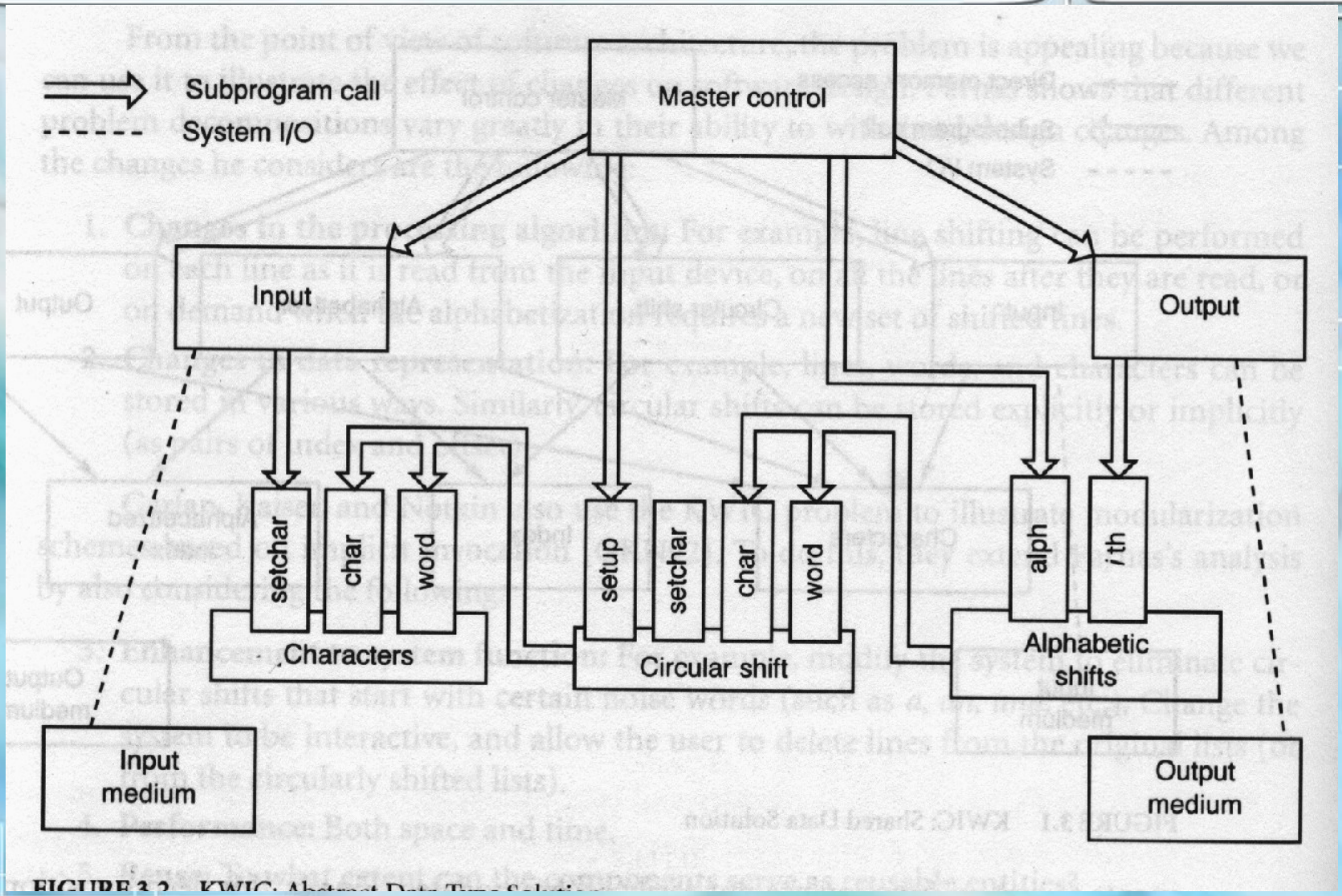
1.1 Programa ppal/ subrutina con datos compartidos

- Ventajas: Distinto procesamiento, distintos módulos. Los datos se pueden representar en forma eficiente, se comparte el mismo almacenamiento.
- Desventajas:Problemas para manejar los cambios
- Cambios en la representación de los datos afectan TODOS los módulos
- Tampoco es facil incorporar (1) o (3) o lograr módulos reusables

1.2 Tipos de datos abstractos

- La descomposición es similar pero ahora cada módulo provee una interfase.
- Tanto los algoritmos como las representaciones de cada módulo pueden cambiarse sin afectar a los otros.
- Los módulos son más resusables porque hacen menos suposiciones sobre el funcionamiento de los otros.
- Problema: agregar nuevas funciones!

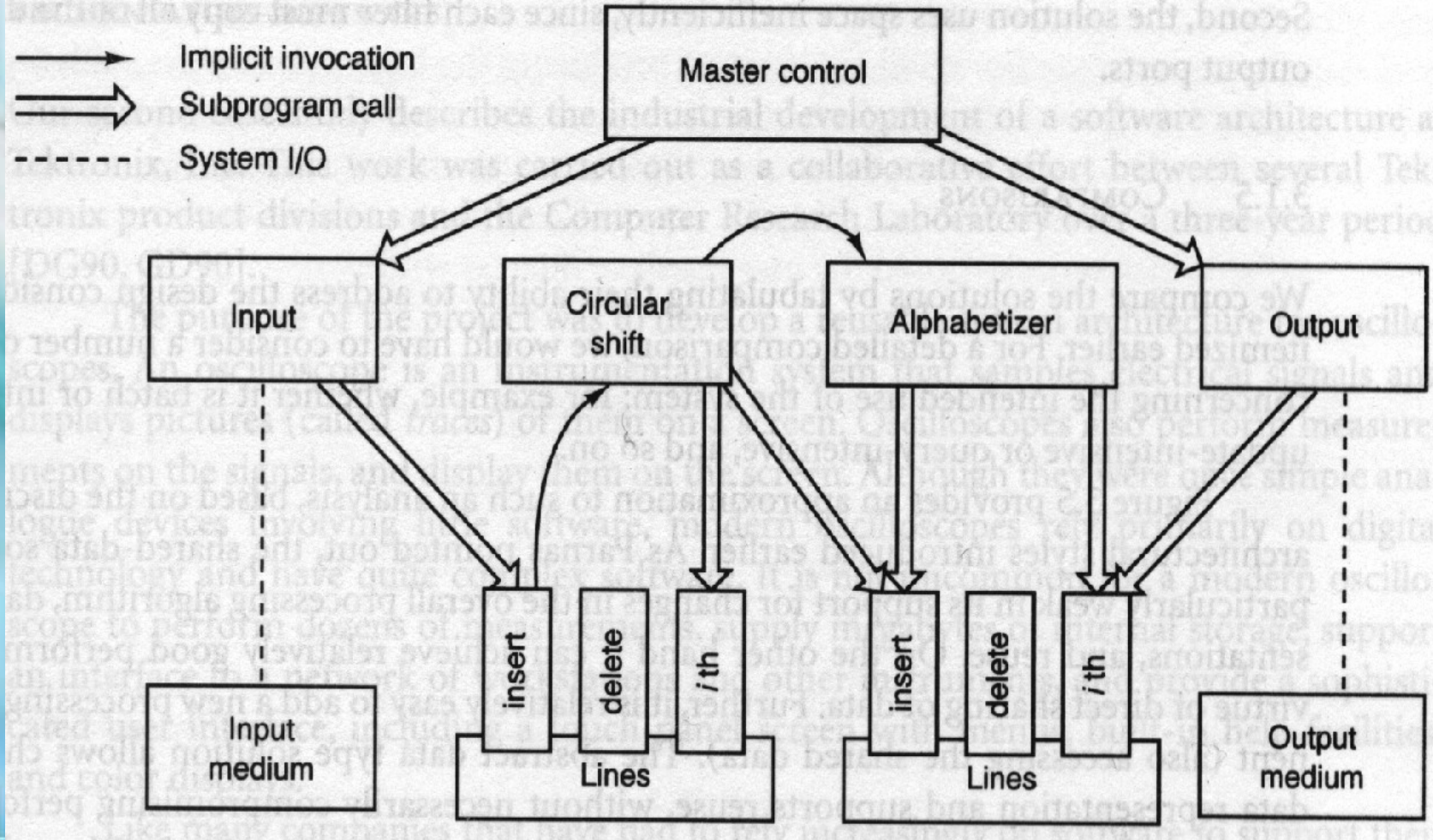
1.2 Tipos de datos abstractos



1.3 Invocación implícita

- Usamos una integración de componentes basada en datos compartidos
- Diferencias con la 1er solución:
  - Interfase abstracta a los datos
  - Las invocaciones se realizan en forma implícita cuando se accede a los datos
- Agregar una línea en storage dispara un evento que provoca que se realicen un shift circulares...

1.3 Invocación implícita



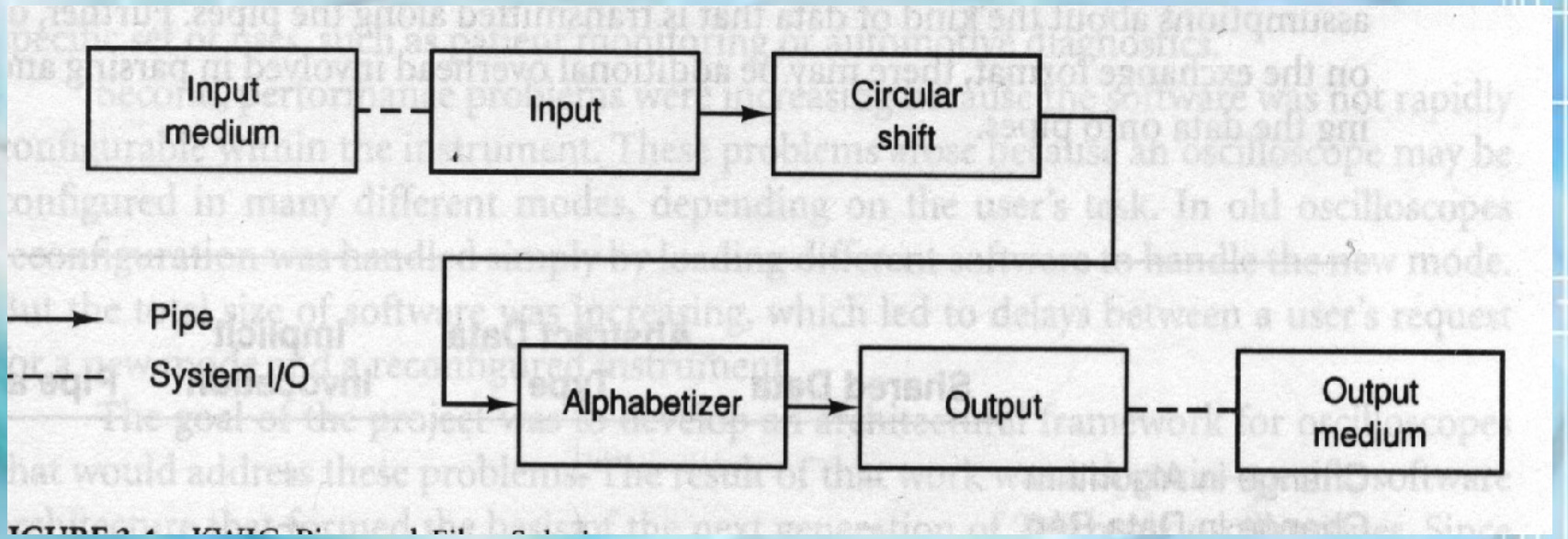
1.3 Invocación implícita

- Ventajas:
  - se pueden realizar mejoras fácilmente, basta registrar los nuevos módulos en los eventos correspondientes
  - la representación es independiente de la computación
  - soporta reusabilidad
- Desventaja: es difícil controlar el orden de procesamiento de los módulos que se invocan en forma implícita!

1.4 Pipes y Filtros

- Usa un acercamiento basado en un pipeline
- Tenemos cuatro filtros: entrada, shift, alfabetizar ysalida
- El control es distribuido: cada filtro ejecuta cuando tiene datos
- Solo se trasmiten datos por medio de los pipes

1.4 Pipes y Filtros



1.4 Pipes y filtros

- Mantiene el flujo de procesamiento intuitivo
- Soporta la reusabilidad, cada filtro debe cambiar (teniendo cuidado de respetar el formato de salida)
- Se agregan funciones nuevas facilmente

1.4 Pipes y filtros

- Problemas:
- Cómo modificar el diseño para que soporte un sistema interactivo?
- Uso ineficiente del espacio: cada filtro debe copiar todos los datos a sus puertos de salida

Comparación

	Datos compart.	TDA	Inv. Implícita	Pipes y flitros
Cambio algoritmo	-	-	+	+
Cambio Rep Datos	-	+	-	-
Cambio función	+	-	+	+
Performance	+	+	-	-
Reusabilidad	-	+	-	+

Caso 2: Instrumentation SW

- Desarrollo industrial de una arquitectura de SW en Tektronix
- Objetivo: desarrollar una arquitectura reusable para osciloscopios
- Actualmente:
  - Poca reusabilidad entre distintos productos
  - Siempre había que hacer cambios para nuevas interfaceso cambios en el HW
- Se busca lograr productos de propósito especifico.

Mas problemas

- Problemas de performance porque el software no era facilmente configurable para la máquina específica
- Los osciloscopios pueden configurarse en muchos modos diferentes dependiendo de la tarea del usr
- No es fácil ni eficiente cambiar de módulos porque el tamaño aumenta en los modelos nuevos

Objetivos

- Desarrollar una arquitectura para osciloscopios que solucione estos problemas
- Esta arquitectura debe ser adaptable para otros sistemas (reusable)

Un modelo OO

- Arquitectura que clarifica los TD usados:
  - Waveforms
  - Signals
  - Trigger modes
- No se pudo hallar un modelo que explicara la interacción entre los mismos
- Por ejemplo: las mediciones deben asociarse con los TD siendo medidos o representadas externamente?

Modelo OO

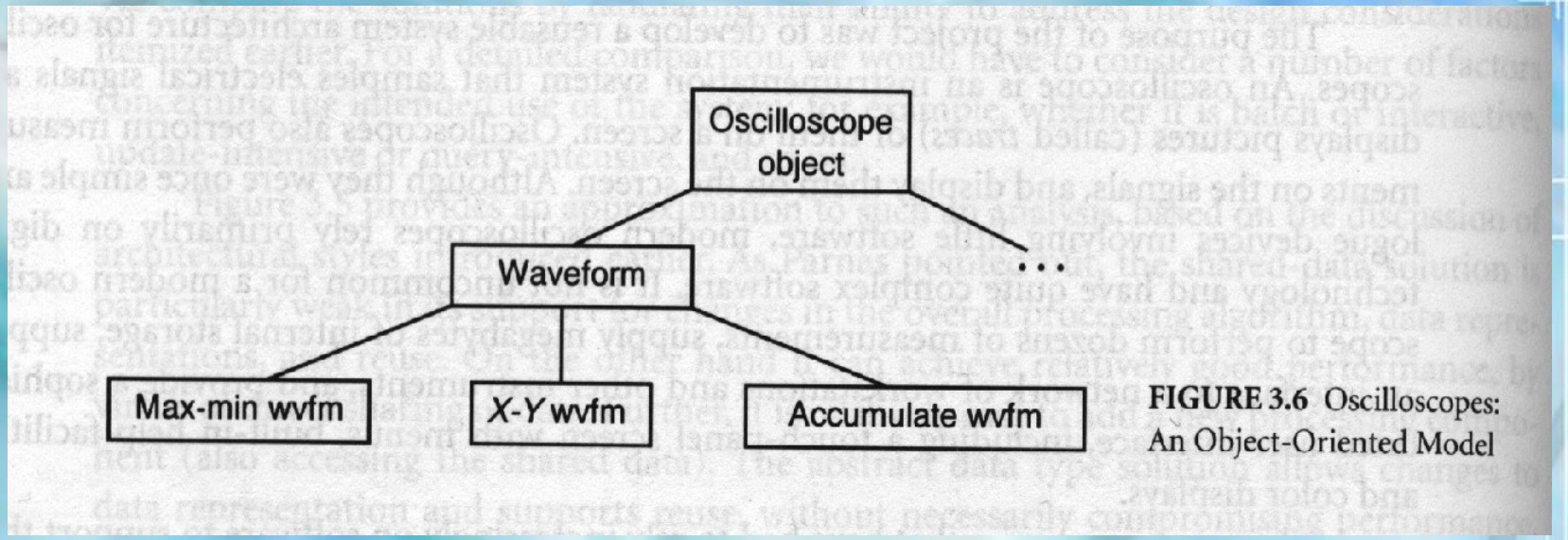


FIGURE 3.6 Oscilloscopes: An Object-Oriented Model

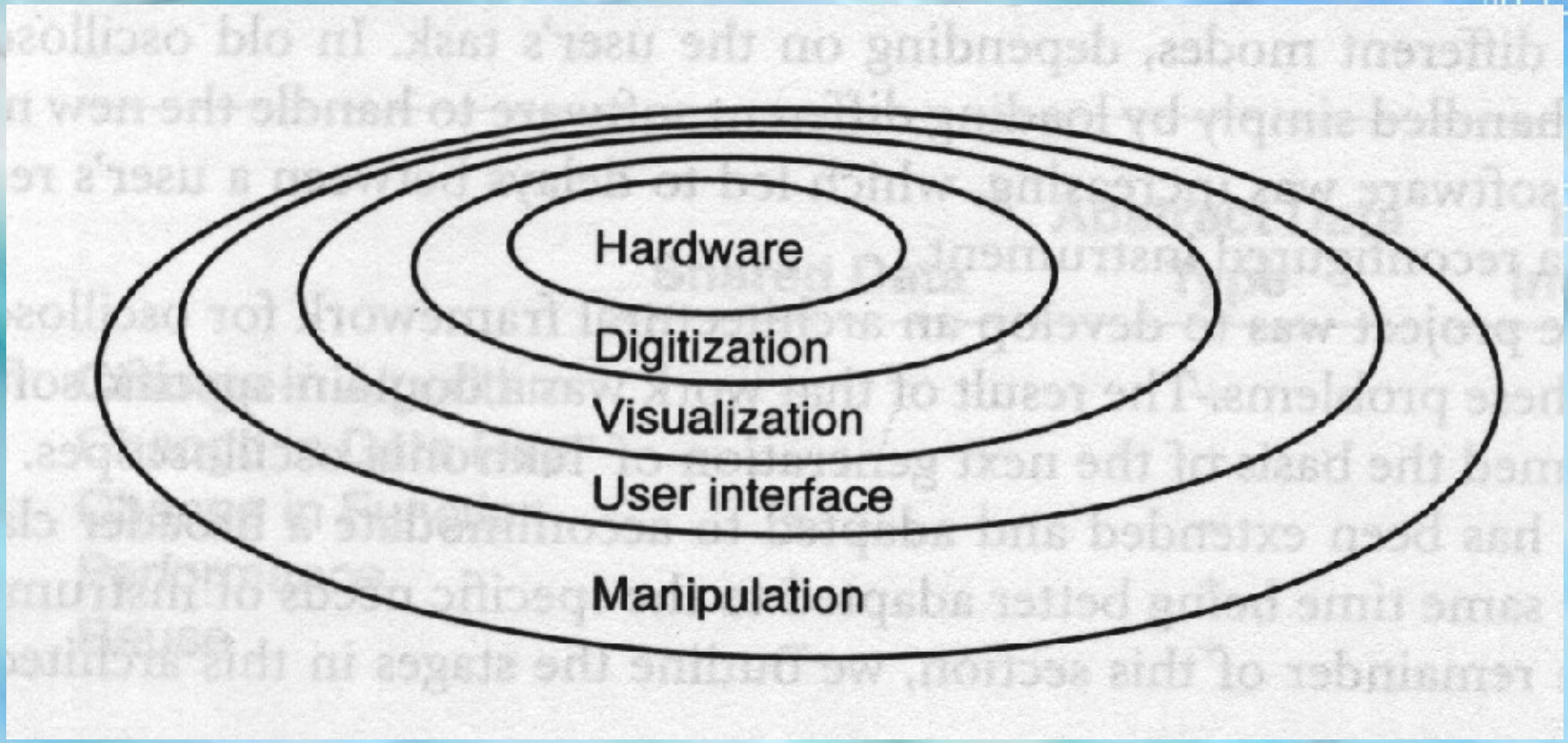
Modelo en capas

- El núcleo representa las funciones de manipulación de señales que filtran las señales cuando estas entran al osciloscopio
- Se implementan típicamente en HW
- Proximo: digitalización de las señales y almacenamiento de las mismas para después procesarlas
- Tercero: manipulación de estas señales

Modelo en capas

- Cuarto visualización de funciones
- Ultimo: interfaz del usr
- Problema: la abstracción propia del modelo no resulta adecuada para el nivel de interacción que la aplicación necesita entre las mismas

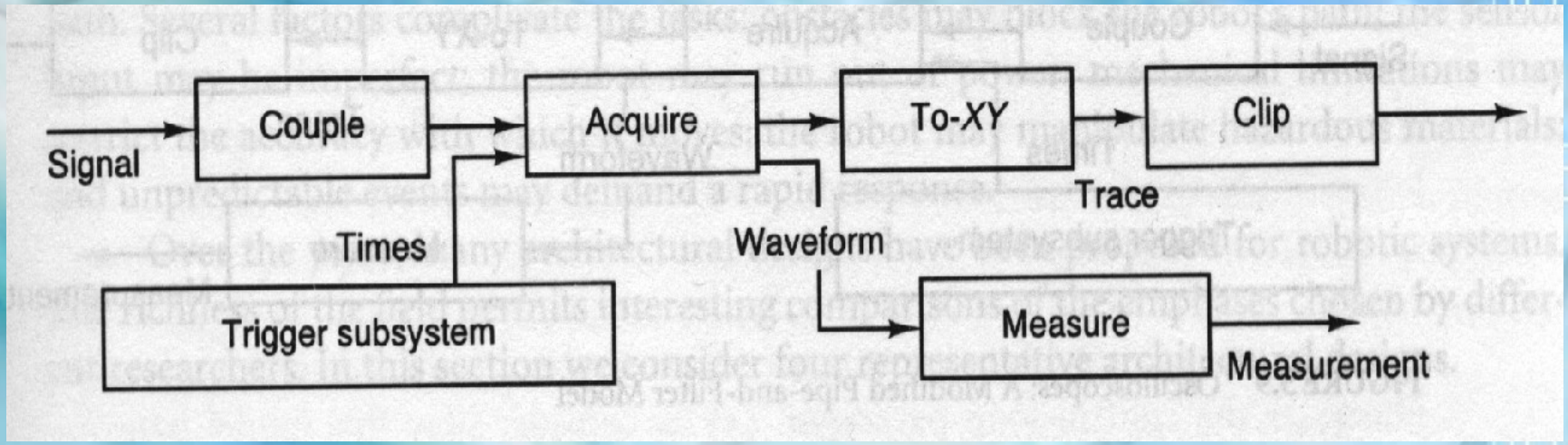
Modelo en capas



Modelo de pipes y filtros

- Las funciones del osciloscopio se modelan como una transformación incremental de los datos
- Los transformadores de señales se usan para condicionar las señales externas
- Los tranformadores de la adquisición derivan formas de onda digitalizadas de esas señales
- Los transformadores visuales permiten visualizarlas gráficamente

Modelo de pipes y filtros



Modelo de pipes y filtros

- Ventaja: no se aislan las funciones en particiones separadas
- Por ejemplo, los datos de las señales se pueden alimentar directamente a los filtros de display
- El modelo se coresponde con la visión del procesamiento de señales como un modelo flujo de datos
- Problema: interacción con el usuario!

Pipes Modificado

- Asociar a cada filtro una interfaz de control
- Esto permite setear los parámetros de operación del filtro
- El filtro de adquisición puede tener parámetros que determinan el sample rate
- ESto provee una colección de settings que determinan que aspectos pueden ser modificados dinámicamente

Pipes modificado

- Separa las funciones de procesamiento de señales de la interfaz
- El Sw de procesamiento de señales no hace suposiciones sobre como el usr comunica los cambios a los parametros de control
- Se puede cambiar la implementación del procesamiento de señales sin cambiar la interfaz (si la interfaz de control permanece sin cambiar)

Pipes modificado

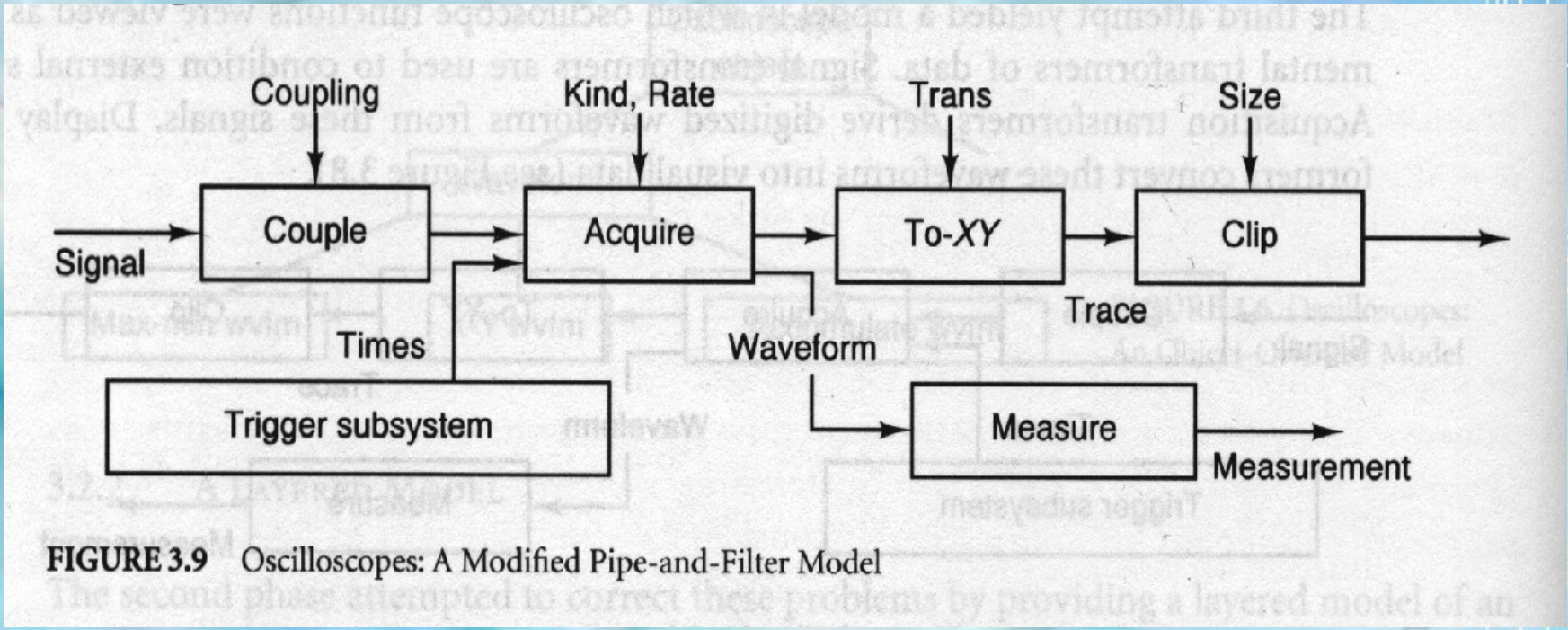


FIGURE 3.9 Oscilloscopes: A Modified Pipe-and-Filter Model

De que se trata

- El diseño de interfaz del usuario. crea un medio de comunicación entre el hombre y la maquina
- El Ing. de SW crea el diseño de la pantalla mediante un proceso iterativo
- Objetivos: evitar la frustración y mejorar la facilidad de uso
- Importante: la interfaz es la que da forma a la percepción que tiene el usuario del SW

Cuales son los pasos

- Identificación de los requisitos del usuario, de la tarea y del entorno
- Se crean y analizan los escenarios
- Se define el conjunto de objetos y acciones de la interfaz
- Creación de un formato de pantalla
- Se usan las herramientas para generar prototipos
- Producto obtenido: formatos de pantalla + prototipo + escenarios de usuarios

Contenido

- Reglas de oro
- Diseño de la interfaz de usuario
- Análisis y modelado de tareas
- Actividades del diseño de la interfaz
- Herramientas de implementación
- Evaluación de diseño

Diseño de interfaces

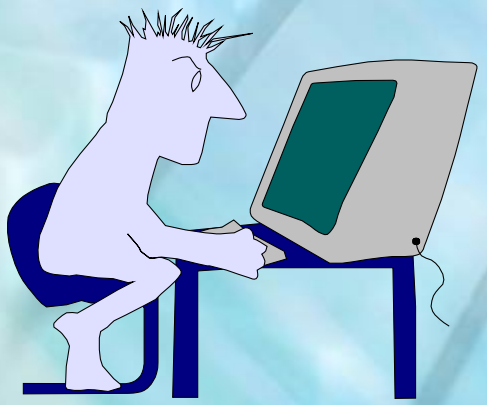
¿fácil de aprender?

¿fácil de utilizar?

¿fácil de entender?

Errores de Diseño Típicos

escasez de consistencia
mucha memorización
sin guía / ni ayuda
sin sensitividad de contexto
pobre respuesta
Inflexible/no amigable



Reglas de oro

1. Dar control al usuario
2. Reducir la carga de memoria del usuario (no abrumarlo con información)
3. Construir una interfaz consecuente

1. Dar el control al usuario

- “un sistema que lea mi mente”
- las limitaciones y restricciones impuestas por el diseñador simplifican el modo de interacción ¿pero para quien?
- Conjunto de principios para mantener el control del lado del usuario:
  - No obligar al usuario a realizar acciones innecesarias y no deseadas
  - Tener en cuenta una interacción flexible

1. Dar el control al usuario

- Permitir que la interacción del usuario se pueda interrumpir y deshacer
- Aligerar la interacción a medida que aumenta el nivel de conocimiento y permitir personalizar la interacción
- Ocultar al usuario ocasional los elementos técnicos
- Diseñar la interacción directa con los objetos que aparecen en la pantalla

2. Reducir la carga de memoria

- Reducir la demanda de memoria a corto plazo
- Establecer valores por defecto útiles
- Definir shortcuts útiles
- Basar el formato visual en una metáfora del mundo real
- Desglosar la información en forma progresiva

3. Construcción de una interfaz consistente

- Toda información visual debe estar organizada de acuerdo con un diseño estandar
- Los mecanismos de entrada se reducen a un conjunto limitado
- los mecanismos para ir de tarea a tarea deben definirse e implementarse coherentemente

3. Una interfaz consistente – Principios -

1. Permitir que el usuario realice una tarea en el contexto adecuado
2. Mantener la consistencia en toda la familia de aplicaciones
3. No realizar cambios con respecto a los modelos interactivos comunes a menos que exista una buena razón

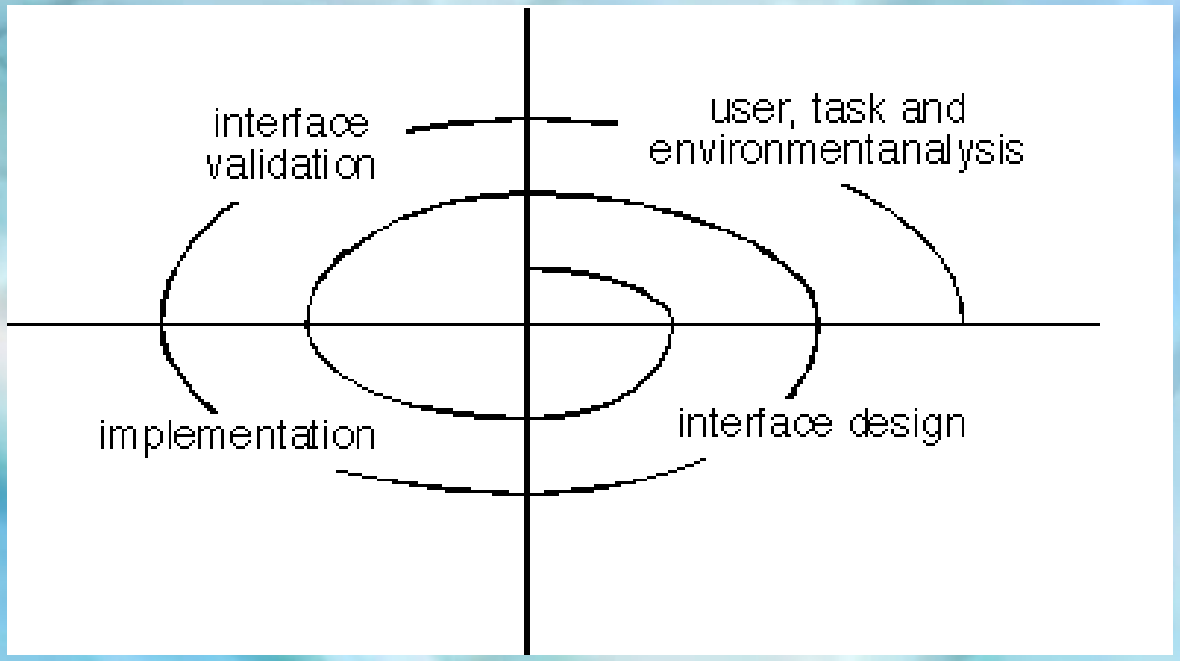
Modelos de Diseño de la Interfaz de Usuario

- Percepción del sistema — imagen mental del usuario de cuál es la interfaz del sistema
- Modelo de usuario — un perfil de todos los usuarios finales del sistema
- Imagen del sistema — la “presentación” del sistema proyectada por la intefaz completa
- Modelo de diseño — representaciones de los datos, arquitectura, interfaz y procedural.

Modelos de Diseño de la Interfaz de Usuario

- Se debe comenzar por conocer los usuarios
- Principiantes: no tienen conocimientos sintácticos ni semánticos del sistema
- Esporádicos y con conocimiento
- Frecuentes y con conocimiento
- Imagen del sistema: fachada + libros de soporte
- Los usuarios se sienten a gusto cuando su percepción coincide con la imagen

Proceso de diseño de la interfaz del usuario



Objetivo del diseño de interfaz

- El objetivo es definir un conjunto de objetos y acciones de interfaz
- La actividad de implementación comienza con un prototipo (a medida que avanza se puede definir un kit de herramientas de usuario)
- La validación se centra en:
  - (1) la habilidad de la interfaz para implementar correctamente las tareas
  - (2) la facilidad de uso
  - (3) la aceptación por parte del usuario como una herramienta útil en su trabajo

¿Cómo hacer el diseño?

- Análisis y modelado de tareas
- Un sistema interactivo puede reemplazar una actividad manual
- Se deben entender las tareas que se realizan manualmente y establecer una correspondencia
- ...O estudiar la especificación y extraer un modelo
- Independientemente del enfoque se debe entender, definir y clasificar las tareas

Ejemplo

- Sistema de diseño asistido para un diseñador de interiores
- Observando al diseñador se identifican:
  - (1) diseño del mobiliario
  - (2) selección de telas y materiales
  - (3) presentación al cliente
  - (4) costo y compras

Ejemplo

- El diseño del mobiliario puede refinarse en las tareas siguientes:
  - dibujar los planos de los ambientes
  - ubicar las puertas y ventanas
  - usar plantillas para dibujar los muebles
  - mover el dibujo del mobiliario
  - etiquetarlo
  - darle distintas perspectivas al dibujo
- El diseño de la interfaz se debe adaptar a estas tareas de acuerdo con el tipo de usuario.

Actividades del diseño

- Al disponer de una lista de tareas quedan definidos los objetos y acciones
- Pasos del diseño de la interfaz:
  1. Establecer los objetivos e intenciones para cada área
  2. Hacer corresponder cada objetivo/intención con una secuencia de acciones específica
  3. Especificar la secuencia de acciones de tareas y subtareas (escenario del usuario) de la forma en que se ejecutarán en la interfaz

Actividades del diseño

1. Indicar el aspecto que tiene la interfaz cuando se esta realizando el escenario de usuario
2. Definir los mecanismos de control
3. Mostrar la forma en que los mecanismos de control afectan al estado del sistema
4. Indicar la forma en que el usuario interpreta el estado del sistema a través de la información dada por la interfaz

Definición de objetos y acciones

- Se analiza sintácticamente el escenario de usuario, aislando los sustantivos y verbos
- En base a estos se crear una lista de objetos y acciones
- Se clasifican los objetos en origen, destino y de aplicación
- Finalmente se realiza el formato de pantalla

Problemas del diseño

1. Tiempo de respuesta del usuario
2. servicios de ayuda
3. manipulación de información de errores
4. etiquetado de ordenes

- Problema: muchos diseñadores no abordan estos temas hasta que es relativamente tarde
- Resultado: frustración del usuario y demoras en la entrega

Tiempo de respuesta

- Tiempo desde que el usr. realiza la acción hasta que el sistema responde
- Duración y variabilidad
- Es conveniente usar indicadores de progreso

Servicios de ayuda

- Tipos de ayuda: integradas y complementarias
- En la construcción de un servicio de ayuda se debe tener en cuenta:
  - ¿Se necesitará disponer de ayuda para todas las funciones en todo momento?
  - ¿De que forma solicitará ayuda el usuario?
  - ¿Cómo se representará la ayuda?
  - ¿Cómo regresará el usuario de la ayuda?
  - ¿Cómo se estructurará la información sobre la pantalla?

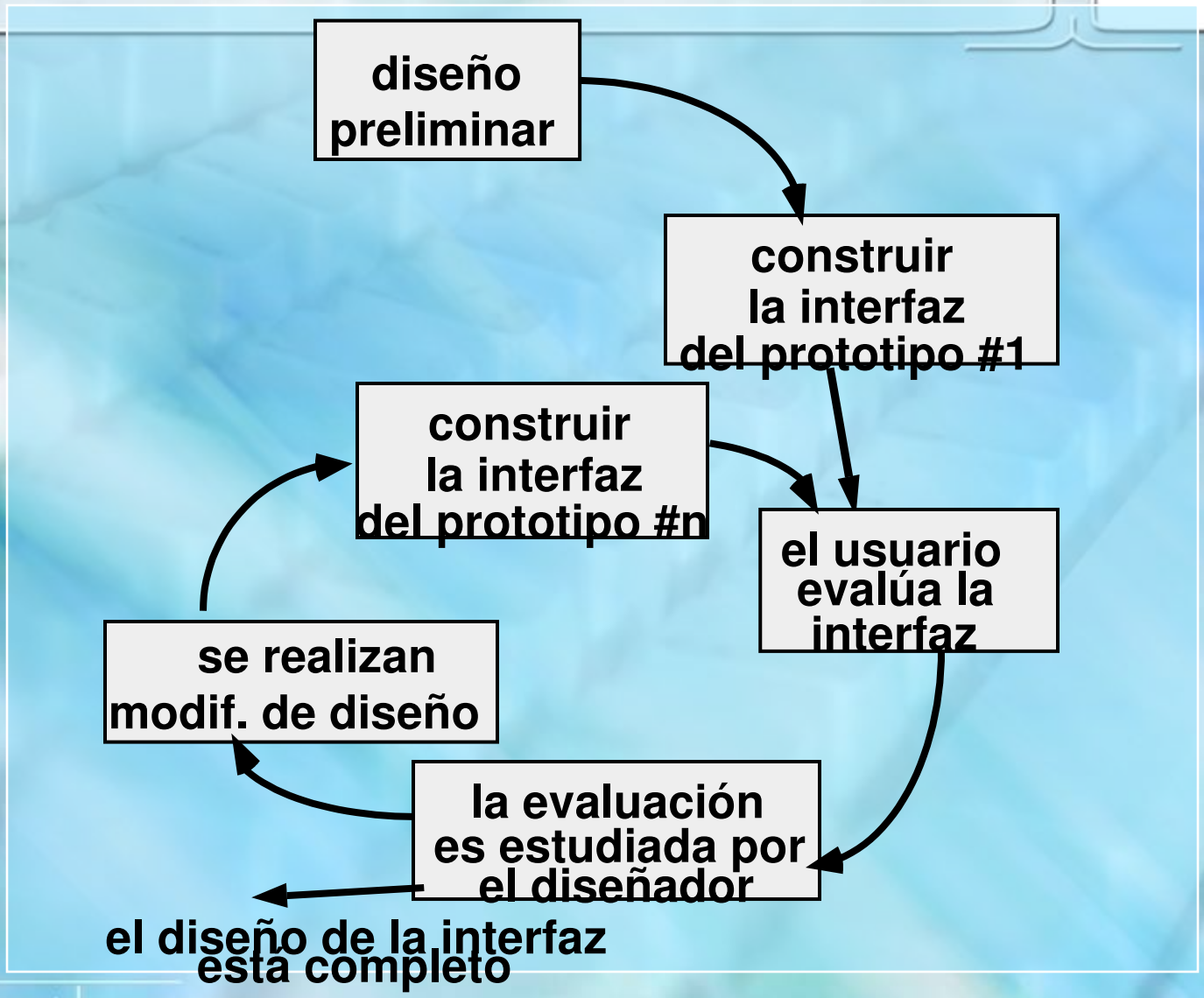
Mensajes de error

- Algunos no sirven para mas que aumentar la frustración
- El mensaje debe describir el problema en un vocabulario que el usuario pueda entender
- Proporcionar consejos para recuperarse
- Indicar cualquier consecuencia negativa del error
- El mensaje debe estar acompañado de una clave visual y/o auditiva
- Nunca culpar al usuario

Herramientas de implementación

- Una vez creado el modelo se implementa como un prototipo
- Se han desarrollado herramientas de diseño de interfaz y elaboración de prototipos denominadas SDUI
- Estos proporcionan mecanismos para:
  (1) gestionar los dispositivos de salida,
  (2) validar la entrada, manipular errores,
  (3) proporcionar ayuda e indicaciones,
  (4) manipular ventanas,
  (5) establecer conexiones entre el SW y la interfaz,
  (6) permitir que el usuario personalice la interfaz

Evaluación del diseño



Criterios para el diseño

1)Se pueden aplicar antes de la creación del prototipo

2)La complejidad de la especificación proporciona una idea de la cantidad de aprendizaje

3)La cantidad de tareas y de acciones por tarea da una idea del tiempo

4)La cantidad de acciones, tareas y estados de sistema indican la carga de memoria

5)El estilo de la interfaz, las funciones de ayuda y la solución de los errores dan una idea de la aceptación por parte del usuario

Recopilar datos para la evaluación

- Cuestionarios a los usuarios del prototipo
- Observar a los usuarios durante la interacción con el prototipo
- Evaluar el tiempo que pasan “mirando la pantalla”

Resumen

- La interfaz es el nexo de comunicación entre el usuario y el sistema
- Una interfaz débil puede llevar al fracaso de una aplicación
- Un proceso de diseño organizado permite que el sistema siga las “reglas de oro”