INGENIERÍA DE SOFTWARE

Diseño De Software

EN CLASES ANTERIORES VIMOS . . .

Conceptos generales

Modelos proceso

Metodologías agiles

Desarrollo de Software Dirigido por Modelos

Problemas de Comunicación

Elicitación de requerimientos

Técnicas de elicitación de requerimientos

Definición de Requerimientos

- Funcionales
- No Funcionales

Ingeniería de Requerimientos

Técnicas de especificación de requerimientos

Gestión de la Configuración del Software (GCS)

EN CLASES ANTERIORES VIMOS . . .

Definición de proyecto

- Características

Gestión de Proyecto

- Métricas / Estimaciones / Calendario temporal / Organización del personal /Análisis de riesgos / Seguimiento y control

Planificación

- Temporal
- Organizativa

Riesgos

- Definición /Estrategias de riesgos/Clasificación de riesgos / Proceso de Gestión de Riesgos

Métricas

- Definiciones / Producto - Líneas de Código / Control o predicción - Punto función / GQM

Estimaciones

- Definiciones / Juicio experto / Técnica Delphi División de trabajo / COCOMO I/II

Calidad de Software

- Definiciones/ Modelos holístico de la calidad Calidad de Producto / Calidad de Procesos /Estándares

DISEÑO DE SOFTWARE

- Representación significativa de ingeniería de algo que se va a construir.
- Es el proceso creativo de transformación del problema en una solución.
- A la descripción de la solución también se la denomina Diseño.
- Es el núcleo técnico de la ingeniería de software.
- Es independiente del modelo de proceso que se utilice.
- El diseño se centra en cuatro aéreas importantes:
  - Datos, Arquitecturas, Interfaces y Componentes.

Fuente: Pressman 5ta. Ed. Cap. 13

DISEÑO DE SOFTWARE - TIPOS

»Diseño de datos

- Transforma el modelo del dominio, obtenido del análisis, en estructuras de datos, objetos de datos, relaciones , etc.

»Diseño arquitectónico

- Define la relación entre los elementos estructurales más importantes del software, los estilos arquitectónicos, patrones de diseño, etc., para lograr los requisitos del sistema.
- La información para realizar el diseño puede derivarse de la especificación, del modelo de análisis y de la interacción de los subsistemas definidos.

Fuente: Pressman Cap. 13

DISEÑO DE SOFTWARE - TIPOS

»Diseño a nivel componentes

- Transforma los elementos estructurales de la arquitectura de software en una descripción procedimental de los componentes del software.
- La información obtenida de los modelos basados en clases, modelos de flujos, de comportamiento sirven como base.

»Diseño de interface

- Describe la forma de comunicación dentro del mismo sistema, con otros sistemas, y con las personas.
- Una interface implica flujo de información (datos o control) y comportamiento.

Fuente: Pressman Cap. 13

DISEÑO DE SOFTWARE

»Características de la evolución de un diseño

- El diseño deberá implementar todos los requisitos explícitos del modelo de análisis, y deberá ajustarse a todos los requisitos implícitos que desea el cliente.
- El diseño deberá ser una guía legible y comprensible para aquellos que generan código y para aquellos que comprueban y consecuentemente, dan soporte al software.
- El diseño deberá proporcionar una imagen completa del software, enfrentándose a los dominios de comportamiento funcionales y de datos desde una perspectiva de implementación.

Fuente: Pressman Cap. 13

DISEÑO DE LA INTERFAZ DEL USUARIO

DISEÑO DE LA INTERFAZ DEL USUARIO CONCEPTOS INICIALES

- Es la categoría de diseño que crea un medio de comunicación entre el hombre y la máquina.
- Con un conjunto de principios, crea un formato de pantalla.

Fuente: Pressman Cap. 15

DISEÑO DE LA INTERFAZ DEL USUARIO CONCEPTOS INICIALES

- De un buen diseño depende en parte el éxito de un sistema.

- Una interfaz difícil de utilizar provoca que los usuarios cometan errores o incluso que se rehúsen a utilizar el sistema.

- Partimos de la base de que personas diferentes pueden tener estilos diferentes de percepción, comprensión y trabajo.

- La interfaz debe contribuir a que el usuario consiga un rápido acceso al contenido de sistemas complejos, sin pérdida de la comprensión mientras se desplaza a través de la información.

Fuente: Conceptos iniciales

DISEÑO DE LA INTERFAZ DEL USUARIO CONCEPTOS INICIALES

- Variedad de tecnologías: hipertexto, sonido, presentaciones tridimensionales, video, realidad virtual, etc.

- Configuraciones de hardware: teclado, mouse, dispositivos de presentación gráfica, lápices, anteojos de realidad virtual, reconocimiento de voz, etc.

- Variedad de Dispositivos: PC, equipos específicos, celulares, televisores, etc.

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO

- Dar control al usuario
- Reducir la carga de memoria del usuario
- Lograr una Interfaz consistente
- Factores Humanos

Fuente: Pressman Cap. 12

DISEÑO DE LA INTERFAZ DEL USUARIO
REGLAS BÁSICA DEL DISEÑO

- Dar control al usuario
  - El usuario busca un sistema que reaccione a sus necesidades y lo ayude a hacer sus tareas.
  - Definir modos de interacción de forma que el usuario no realice acciones innecesarias
  - Proporcionar una interacción flexible
  - Incluir las opciones de interrumpir y deshacer
  - Depurar la interacción a medida que aumenta la destreza del usuario.
  - Ocultar al usuario ocasional los elementos técnicos internos
  - Diseñar interacción directa con los objetos que aparecen en pantalla

Fuente: Pressman Cap. 12

DISEÑO DE LA INTERFAZ DEL USUARIO
REGLAS BÁSICA DEL DISEÑO

- Reducir la carga de memoria del usuario
  - Reducir la demanda a corto plazo
  - Definir valores por defecto que tengan significado
  - Definir accesos directos intuitivos
  - El formato visual de la interfaz debe basarse en una metáfora de la realidad
  - Desglosar la información de manera progresiva

Fuente: Pressman Cap. 12

DISEÑO DE LA INTERFAZ DEL USUARIO
REGLAS BÁSICA DEL DISEÑO

» Lograr una interfaz consistente

- Permitir que el usuario incluya la tarea actual en un contexto que tenga algún significado
  - El usuario debe tener la capacidad de determinar de donde viene y hacia donde puede ir
- Mantener consistencia en toda la familia de aplicaciones
  - Utilizar las mismas reglas de diseño para las mismas interacciones
- Mantener modelos que son prácticos para el usuario, a menos que sea imprescindible cambiarlos

Fuente: Pressman Cap. 12

DISEÑO DE LA INTERFAZ DEL USUARIO
REGLAS BÁSICA DEL DISEÑO

»Factores Humanos:

- Percepción visual/auditiva/táctil
- Memoria humana
- Razonamiento
- Capacitación
- Comportamiento/Habilidad personales
- Diversidad de usuarios
  - Usuarios casuales: Necesitan interfaces que los guíen.
  - Usuarios experimentados: Requieren interfaces ágiles.

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO

- Interfaz de comandos
- Interfaz de menú simple
- GUI: Interfaz gráfica de usuarios
- Interfaz de reconocimiento de voz
- Interfaz Inteligente

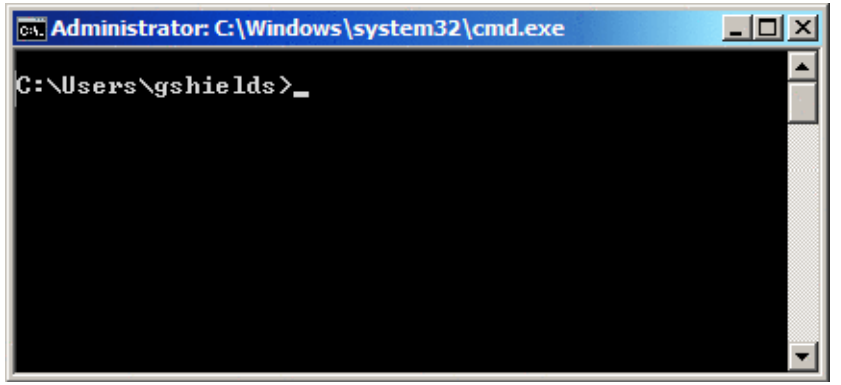
Fuente: Estilos de Interfaces

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES

»Es la interfaz mas elemental
- Solo se interactúa con texto

»Generalmente se interactúa desde una línea de comando de una consola de una aplicación en particular con el teclado

»Características:
- Poderoso y Flexible
- Administración de errores pobre
- Difícil de aprender

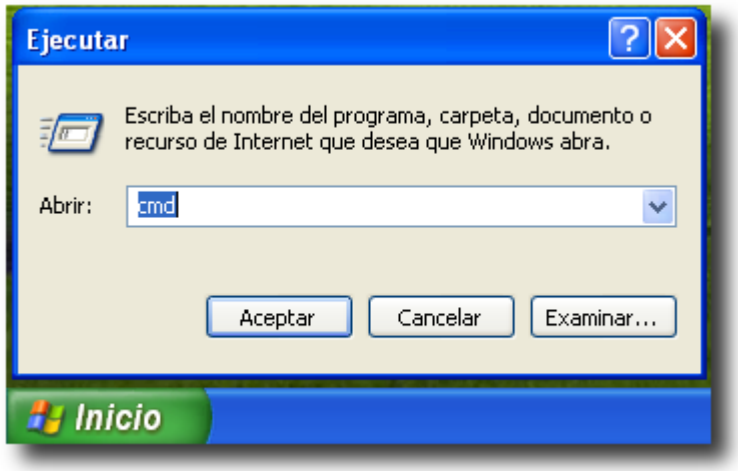


Fuente: Whitten Bentley

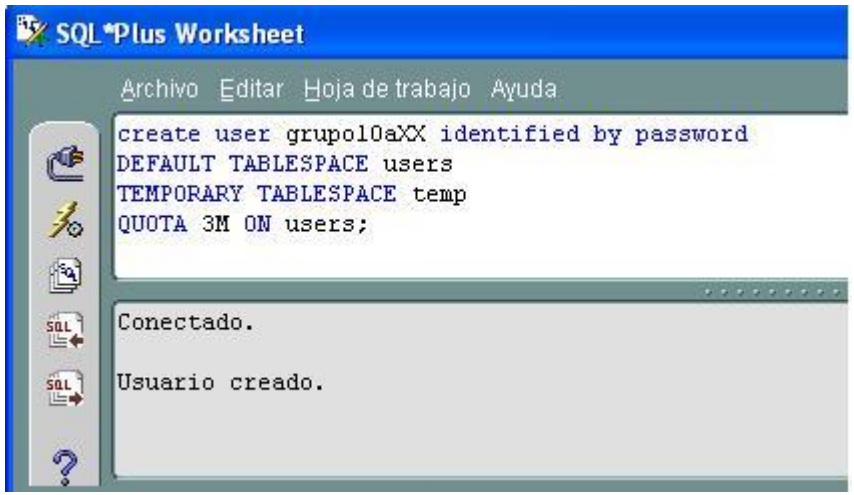
Consola del SO Windows

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES - INTERFAZ DE COMANDOS

»Interfaz de comando a través de una interfaz gráfica



Ejecutar comandos de Windows

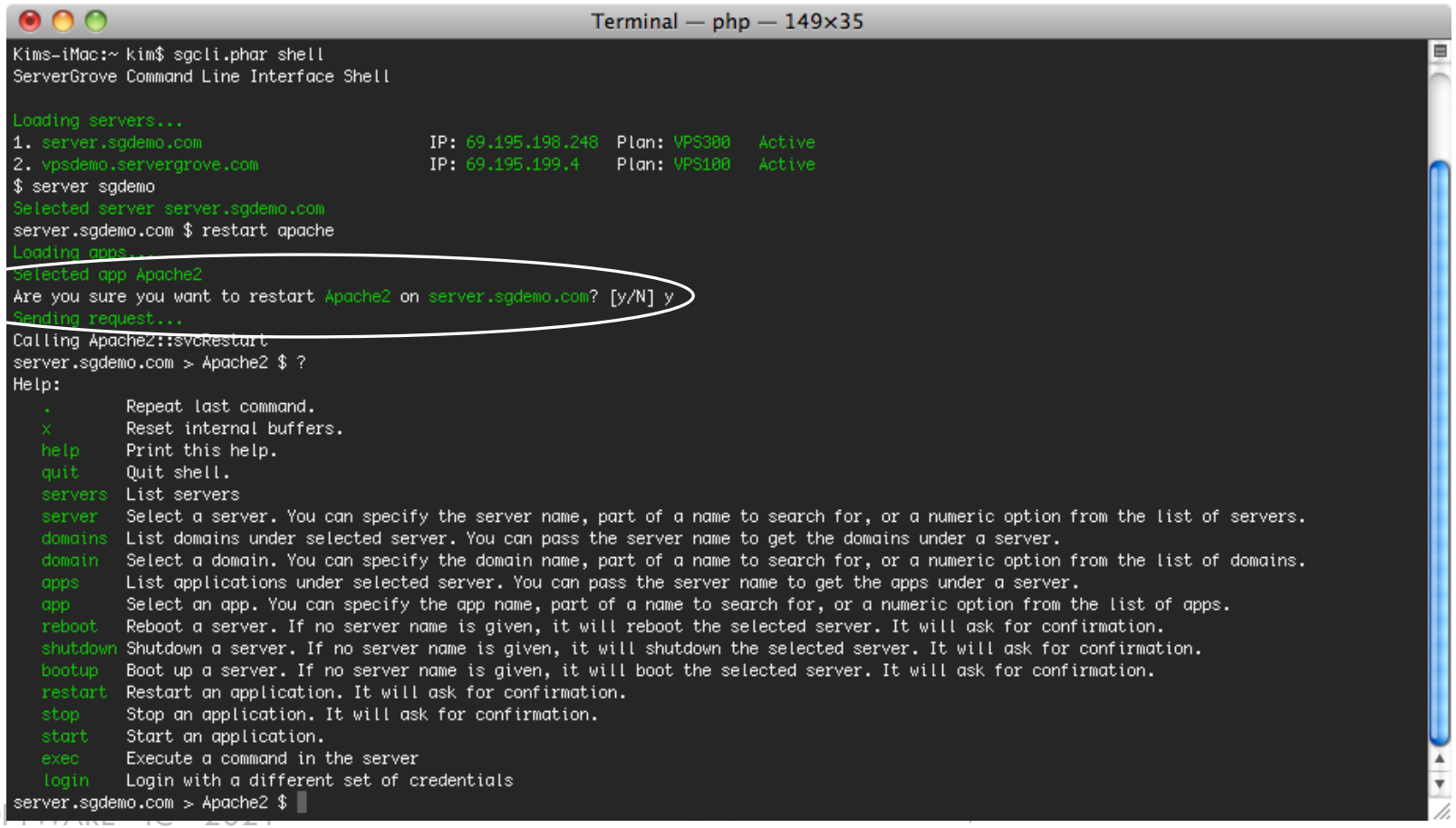


Ejecutar una consulta SQL utilizando la línea de comandos

Fuente: Whitten Bentley

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES - INTERFAZ DE COMANDOS

»Comandos del tipo pregunta respuesta



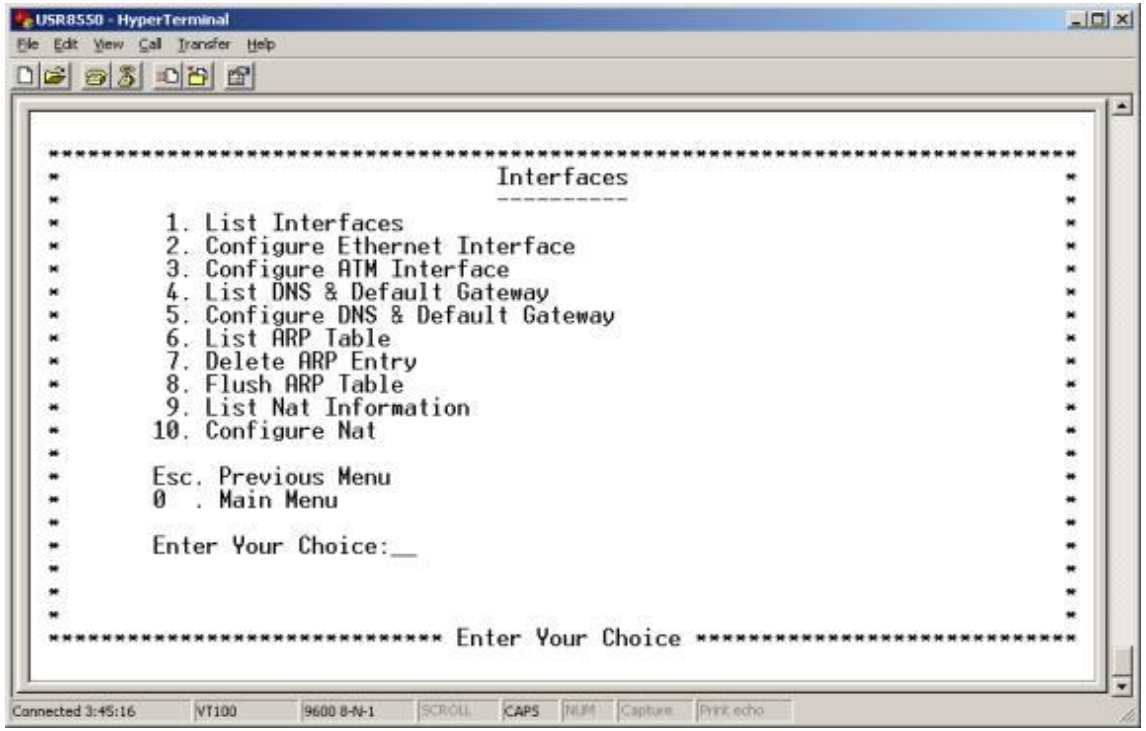
DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES - INTERFAZ DE MENÚ SIMPLE

»Se presentan un conjunto de opciones, que pueden ser seleccionadas por el usuario

»Solo se interactúa con los caracteres indicados

»Características:

- Evita errores del usuario.
- Lento para usuarios experimentados



Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES
INTERFAZ GRÁFICA DE USUARIOS

»Se caracterizan por la utilización de todo tipo de recursos visuales para la representación e interacción con el usuario.

»Ventajas:

- Son relativamente fáciles de aprender y utilizar.
- Los usuarios cuentan con pantallas múltiples (ventanas) para interactuar con el sistema.
- Se tiene acceso inmediato a cualquier punto de la pantalla.

Fuente:

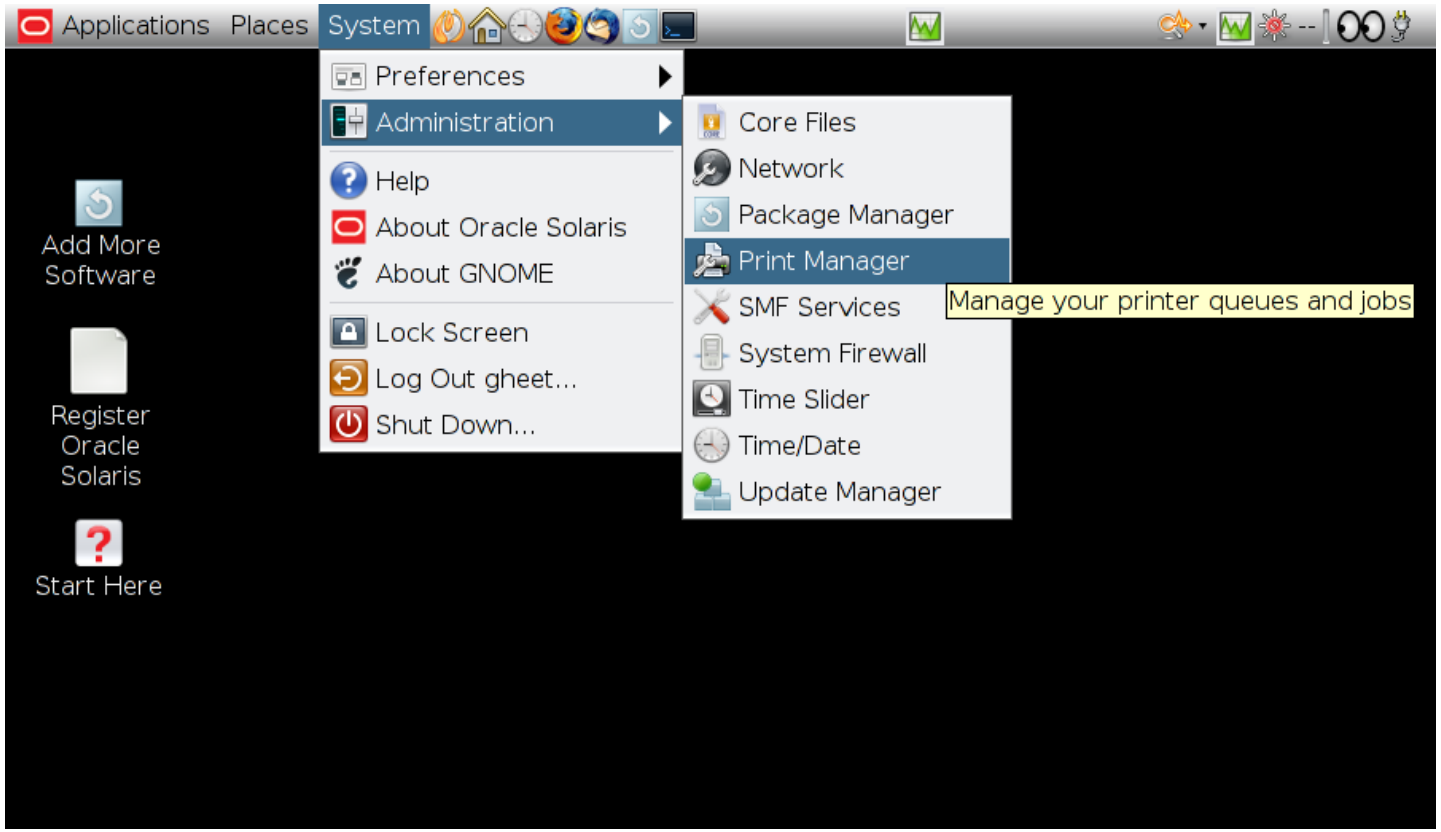
DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES -INTERFAZ GRÁFICA DE USUARIOS

»Ventanas

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES - INTERFAZ GRÁFICA DE USUARIOS

»Iconos y Menús



Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES - INTERFAZ GRÁFICA DE USUARIOS

»Interfaces de manipulación directa

Hardware Específico

Hardware Específico y evolución a la pantalla táctil

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES - INTERFAZ GRÁFICA DE USUARIOS

»Interfaces de manipulación directa táctil

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES - RECONOCIMIENTO DE VOZ

»Comunicación con los dispositivos a través de la voz

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES
INTERFACES PARA DIFERENTES DISPOSITIVOS

Interface Web adaptable a cada dispositivo

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
ESTILOS DE INTERFACES

»Interfaces Inteligentes

- Tienen la capacidad de captar la secuencia de acciones que el usuario repite con frecuencia para luego adelantarse y brindar la posibilidad de completar la secuencia de acciones en forma automática.

»Dentro de este tipo se encuentran las

- Adaptativas: Brindan diferentes modos de interacción que se pueden seleccionar automáticamente de acuerdo al tipo de usuario en cuestión. Son sensibles a los perfiles individuales de los usuarios y a sus estilos de interacción.
- Evolutivas: Tienen la propiedad de cambiar y evolucionar con el tiempo, junto con el grado de perfeccionamiento que el usuario va adquiriendo con el sistema.
- Interfaces Accesibles: Son las interfaces que respetan las normas del diseño universal para que puedan ser accedidas por cualquier usuario independientemente de sus condiciones físicas mentales.

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
PRINCIPIOS DE NIELSEN

- Existen ciertos principios de diseño que enuncian el diálogo correcto que debe proveer una interfaz de usuario.
- Estos principios fueron desarrollados por Jacob Nielsen y son utilizados para el diseño de nuevas interfaces y, como métricas de evaluación de interfaces ya desarrolladas.
- Aunque estos principios fueron pensados inicialmente para interfaces textuales, sirven de base para el diseño preliminar de cualquier otro tipo de interfaz.

Fuente: Principios de Nielsen

DISEÑO DE LA INTERFAZ DEL USUARIO
PRINCIPIOS DE NIELSEN

»Los Principios de Nielsen constan de 10 normas

»1.- Diálogo simple y natural: Forma en que la interacción con el usuario debe llevarse a cabo.

- Realizar una escritura correcta, sin errores de tipeo.
- No mezclar información importante con la irrelevante.
- Distribución adecuada de la información.
- Prompts lógicamente bien diseñados.
- Evitar el uso excesivo de mayúsculas y de abreviaturas.
- Unificar el empleo de las funciones predefinidas.

»2.- Lenguaje del usuario: Emplear en el sistema un lenguaje familiar para el usuario

- Usar el lenguaje del usuario.
- No utilizar palabras técnicas ni extranjeras.
- Evitar el truncamiento excesivo de palabras.
- Diseñar correctamente las entradas de datos.
- Emplear un grado adecuado de información (ni excesivo ni escaso).

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO PRINCIPIOS DE NIELSEN

»3.- Minimizar el uso de la memoria del usuario: Evitar que el usuario esfuerce su memoria para interactuar con el sistema.

- Brindar Información de contexto.
- Brindar información de la navegación y sesión actual.
- Visualización de rangos de entrada admisibles, ejemplos, formatos.

»4.- Consistencia: Que no existan ambigüedades en el aspecto visual ni tecnológico en el diálogo o en el comportamiento del sistema. La consistencia es un punto clave para ofrecer confiabilidad y seguridad al sistema.

- Debe existir una consistencia terminológica y visual.

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
PRINCIPIOS DE NIELSEN

»5.- Feedback: Es una respuesta gráfica o textual en la pantalla, frente a una acción del usuario. El sistema debe mantener al usuario informado de lo que está sucediendo.

- Brindar información de los estados de los procesos.
- Brindar información del estado del sistema y del usuario.
- Utilización de mensajes de aclaración, validaciones, confirmación y cierre.
- Realizar validaciones de los datos ingresados por el usuario.

»6.- Salidas evidentes: Que el usuario tenga a su alcance de forma identificable y accesible una opción de salida.

- Brindar salidas de cada pantalla.
- Salidas para cada contexto.
- Salidas para cada acción, tarea o transacción.
- Brindar salidas en cada estado.
- Visualización de Opciones de Cancelación, Salidas, de Suspender, de Deshacer y Modificación.

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
PRINCIPIOS DE NIELSEN

»7.- Mensajes de error: Feedback del sistema ante la presencia de un error. De qué forma se ayuda al sistema para que salga de la situación en la que se encuentra.

- Deben existir mensajes de error para ser usados en los momentos que corresponda.
- Brindar Información del error, explicar el error y dar alternativas a seguir.
- Se deben categorizar los diferentes tipos de mensajes.
- No deben existir mensajes de error intimidatorios.
- Manejar adecuadamente la forma de aparición de los mensajes.

»8.- Prevención de errores: Evitar que el usuario llegue a una instancia de error.

- Brindar rangos de entradas posibles para que el usuario seleccione y no tipee.
- Mostrar ejemplos, valores por defecto y formatos de entrada admisibles.
- Brindar mecanismos de corrección automática en el ingreso de los datos.
- Flexibilidad en las entradas de los usuarios

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO PRINCIPIOS DE NIELSEN

»9.- Atajos: La interfaz debería proveer de alternativas de manejo para que resulte cómodo y amigable tanto para usuarios novatos como para usuarios experimentados.

- Brindar mecanismos alternativos para acelerar la interacción con el sistema.
- Brindar la posibilidad de reorganizar barras de herramientas, menús, de acuerdo a la necesidad del usuario.
- Brindar mecanismos de Macros, atajos, definición de teclas de función.

»10.- Ayudas: Componentes de asistencia para el usuario. Un mal diseño de las ayudas puede llegar a entorpecer y dificultar la usabilidad.

- Deben existir las ayudas.
- Se deben brindar diferentes tipos de ayuda : generales, contextuales, específicas, en línea.
- Las ayudas deben proveer diferentes formas de lectura.
- Se deben brindar diferentes mecanismos de asistencia como búsquedas, soporte en línea, e-mail del soporte técnico, acceso a las preguntas frecuente.

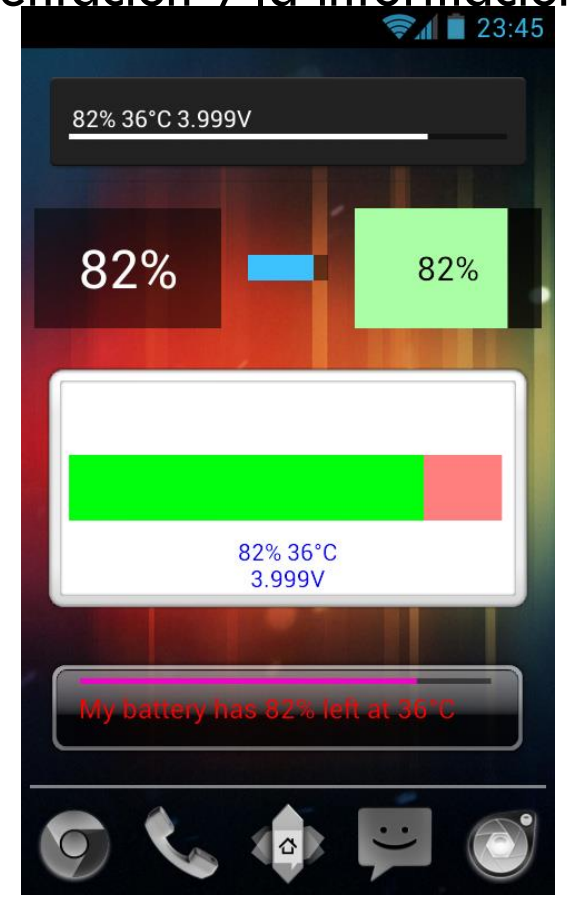
Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
PRESENTACIÓN DE LA INFORMACIÓN

» Mantener separada la lógica del software de la presentación y la información misma
(enfoque MVC)

- Presentación de la Información de manera Directa
- Presentación de la Información de manera Gráfica

82 %



Fuente: Presentación de la información

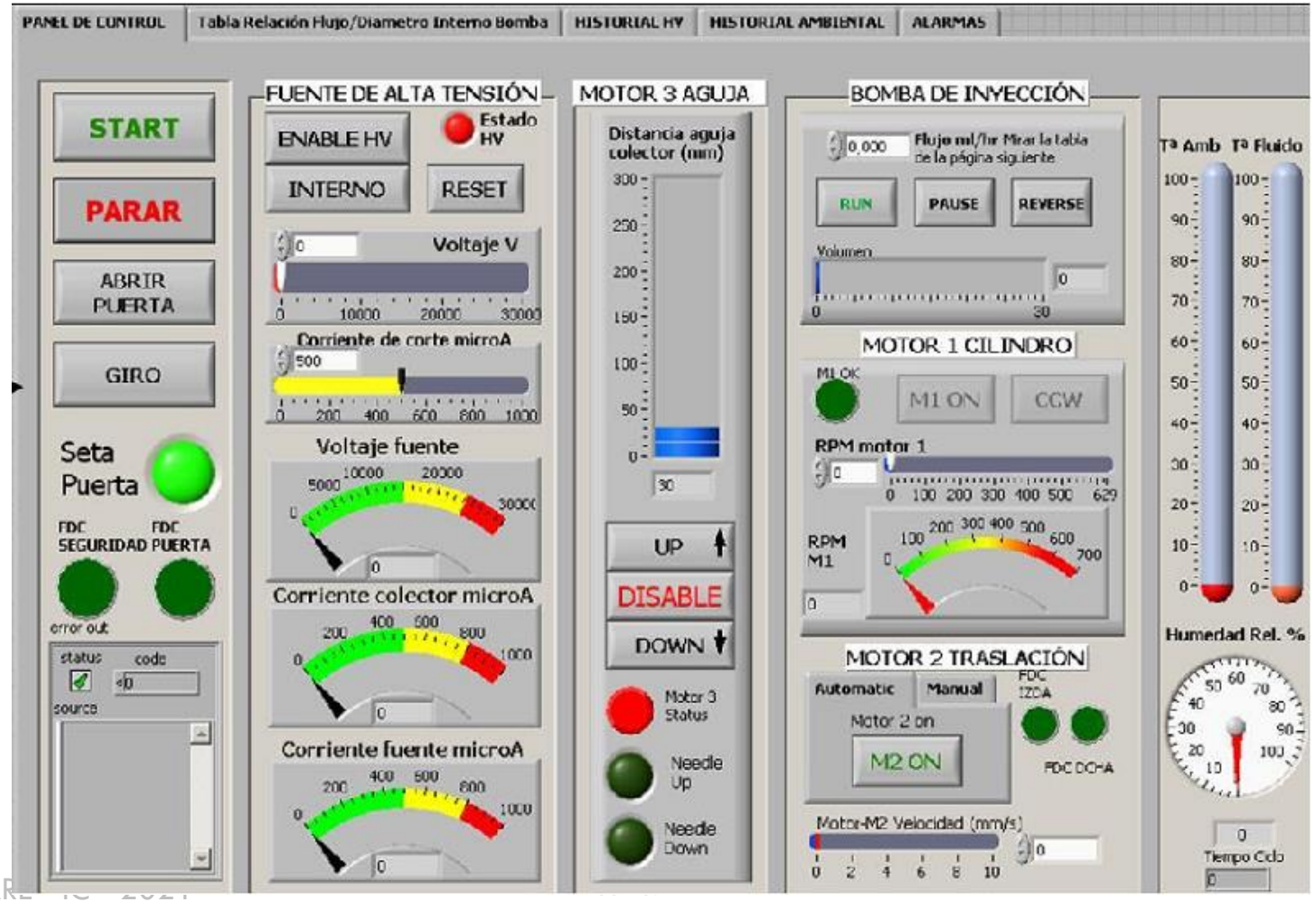
DISEÑO DE LA INTERFAZ DEL USUARIO
PRESENTACIÓN DE LA INFORMACIÓN

- Se deben conocer los usuarios y como utilizarán el sistema.
- ¿Información precisa o relación entre los valores?
- ¿Es necesario presentar inmediatamente los cambios?
- ¿El usuario realiza acciones en función de los cambios?
- ¿Información textual o numérica?
- ¿Información estática o dinámica?

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
PRESENTACIÓN DE LA INFORMACIÓN

Simulador de una Central Hidroeléctrica



DISEÑO DE LA INTERFAZ DEL USUARIO PRESENTACIÓN DE LA INFORMACIÓN

»Manejo de los colores

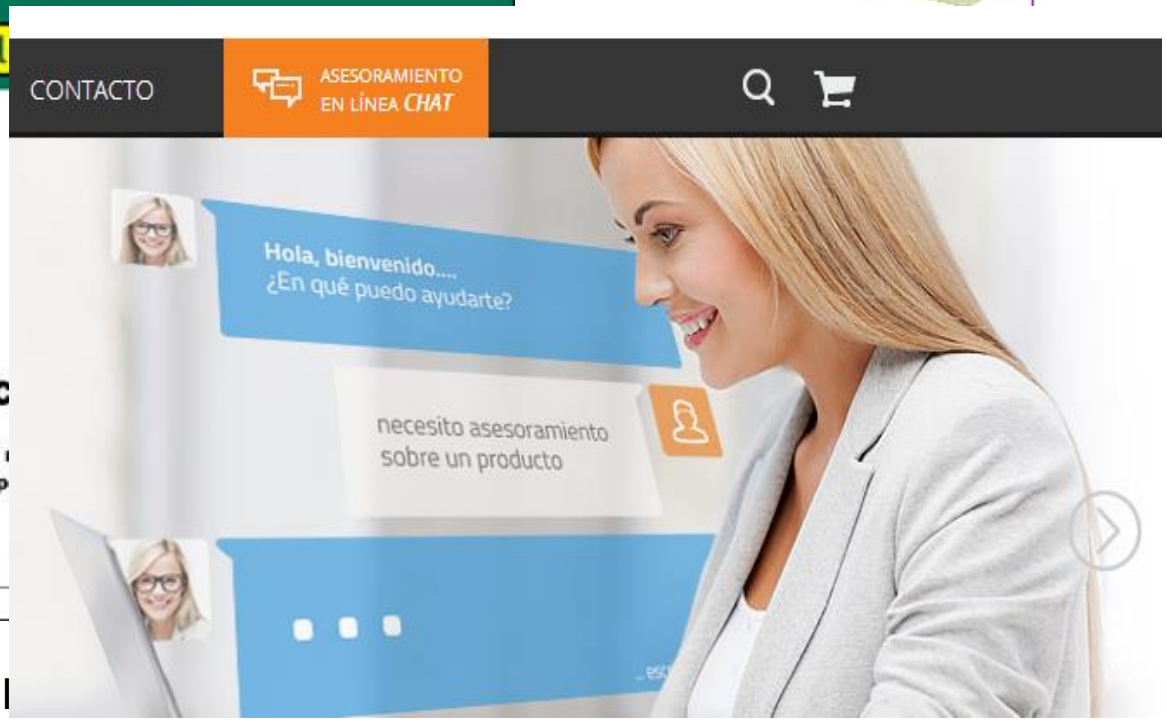
- Limitar el número de colores utilizados.
- No asociar solamente colores a significados.
  - 10% de los humanos no perciben el color.
  - Acompañarlos de algún otro tipo de identificación
- Usar los colores consistentemente.
- Usar cambio de color para mostrar cambios en el estado del sistema.
- Combinar los colores cuidadosamente.

Fuente:

DISEÑO DE LA INTERFAZ DEL USUARIO
SOPORTE AL USUARIO

- »Mensajes del sistema por acciones del usuario.
- »Ayudas en línea
- »Docur

Mensaje de error personal



Fuente: Ayuda en Línea

CONCEPTOS DE DISEÑO

CRITERIOS TÉCNICOS PARA UN BUEN DISEÑO

- Un diseño deberá presentar una estructura arquitectónica.
- Un diseño deberá ser modular, el software deberá dividirse lógicamente en elementos que realicen funciones y sub-funciones específicas.
- Un diseño deberá contener distintas representaciones de datos, arquitectura, interfaces y componentes (módulos).
- Un diseño deberá conducir a estructuras de datos adecuadas para los objetos que se van a implementar y que procedan de patrones de datos reconocibles.
- Un diseño deberá conducir a componentes que presenten características funcionales independientes.
- Un diseño deberá conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y con el entorno externo.
- Un diseño deberá derivarse mediante un método repetitivo y controlado por la información obtenida durante el análisis de los requisitos del software.

Fuente: Pressman Cap. 13

PRINCIPIOS DEL DISEÑO

- En el proceso de diseño se deben tener en cuenta enfoques alternativos.
- El diseño deberá poderse rastrear hasta el modelo de análisis.
- El diseño deberá minimizar la distancia intelectual entre el software y el problema.
- El diseño deberá presentar uniformidad e integración.
- El diseño deberá estructurarse para admitir cambios.

Fuente: Pressman Cap. 13

PRINCIPIOS DEL DISEÑO

- El diseño deberá estructurarse para degradarse poco a poco, incluso cuando se enfrenta con datos, sucesos o condiciones de operaciones aberrantes.
- El diseño no es escribir código y escribir código no es diseñar.
- El diseño deberá evaluarse en función de la calidad mientras se va creando, no después de terminado.
- El diseño deberá revisarse para minimizar los errores conceptuales.

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

» "El comienzo de la sabiduría para un ingeniero de software es reconocer la diferencia entre hacer que un programa funcione y conseguir que lo haga correctamente".

» M. A. Jackson

» Los conceptos de diseño de software fundamentales proporcionan el marco de trabajo necesario para conseguir que lo haga correctamente.

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Cada concepto proporciona al diseñador una base para aplicar los métodos de diseño

»El diseñador debe saber:

- ¿Qué criterios se podrán utilizar para la partición del software en componentes individuales?
- ¿Cómo se puede separar la función y la estructura de datos de una representación conceptual del software?
- ¿Existen criterios uniformes que definen la calidad técnica de un diseño de software?

»Los conceptos van a ayudar al diseñador a responder esas preguntas

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

- Abstracción
- Refinamiento
- Modularidad
- Arquitectura
- Jerarquía de control
- División estructural
- Estructuras de datos
- Procedimiento de software
- Ocultamiento de información

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Abstracción

- La noción de abstracción permite concentrarse en un problema a un nivel de generalización sin tener en cuenta los detalles irrelevantes de bajo nivel
- Nivel de abstracción
  - A medida que profundizamos en la solución del problema se reduce el nivel de abstracción.
  - Desde los requerimientos (abstractos) hasta llegar al código fuente.

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Tipos de abstracción

- De procedimiento
  - Secuencia “nombrada” de instrucciones que tienen una funcionalidad especifica.
  - Ej.: Módulos (procedimientos, funciones, unidades, etc.)
- De datos
  - Colección “nombrada” de datos que definen un objeto real
  - Ej.: un registro que representa una persona con sus datos, el objeto persona en POO
- De control
  - Mecanismo de control del programa sin especificar los mecanismos internos
  - Ej.: Mecanismos de concurrencia del S.O., Semáforos, monitores, etc.

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Refinamiento

- Cada paso se descompone en una o varias instrucciones más detalladas
- El refinamiento es un proceso de elaboración.
  - Se comienza con una descripción de información de alto nivel de abstracción, sobre una funcionalidad puntual, sin conocer las características del funcionamiento, se va trabajando sobre la funcionalidad original proporcionando en cada iteración un mayor nivel de detalle hasta obtener todos los detalles necesarios para conocer su funcionamiento.

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Modularidad

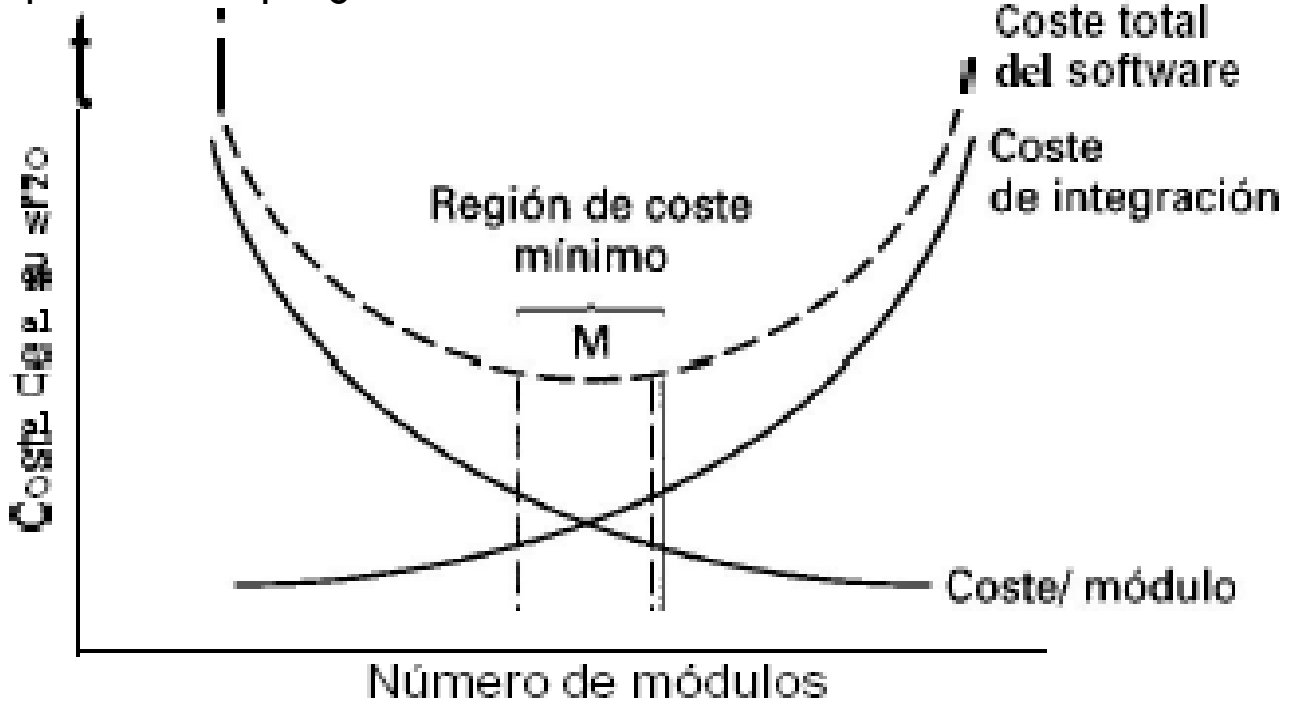
- El software se divide en componentes nombrados y abordados por separado, llamados frecuentemente módulos, que se integran para satisfacer los requisitos del problema.
- En un diseño modular, las componentes tienen entradas y salidas claramente definidas y cada componente tiene un propósito claramente establecido.
- Los componentes modulares están organizados en una jerarquía, como resultado de la descomposición o abstracción, de modo que puede investigarse el sistema de a un nivel por vez.

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Códigos Monolíticos (un único modulo) y Modularización excesiva (a nivel de instrucciones)

- ¿Cuantos módulos tiene que tener un programa?



Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Arquitectura del software

- Es la estructura jerárquica de los componentes del programa, la manera en que los componentes interactúan, y la estructura de datos que van a utilizar los componentes.
- Más adelante se estudiarán las arquitecturas con más detalle

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

- Jerarquía de Control (Estructura de Programa)
  - Representa la organización de los componentes.
  - No representa los aspectos procedimentales del software, ni se aplica en todas las arquitecturas

-

a,b,c son controlados por M

Ámbito de control
Profundidad
Grado de Salida
Grado de Entrada
Anchura

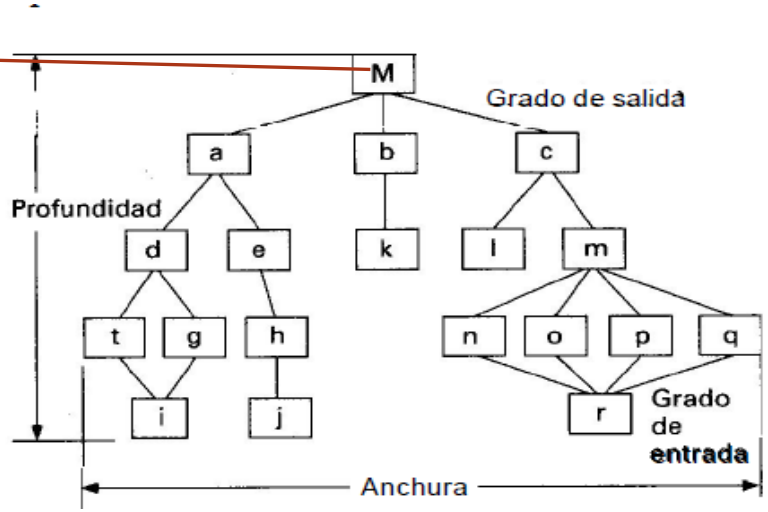


FIG minologías de estructura para un estilo uitectónico de llamada y retorno.

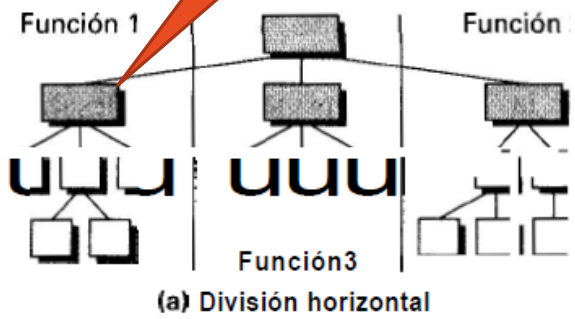
Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

Se utilizan para coordinar la comunicación entre ellos y la ejecución de las funciones

»División estructural

- Si el estilo arquitectónico de un sistema es jerárquico, la estr ´se de manera:
- Horizontal
  - Ramas separadas de la
  - jerarquía para cada
  - función principal



(a) División horizontal

- Ventajas:
  - proporciona software más fácil de probar
  - conduce a un software más fácil de mantener
  - propaga menos efectos secundarios
  - proporciona software más fácil de ampliar

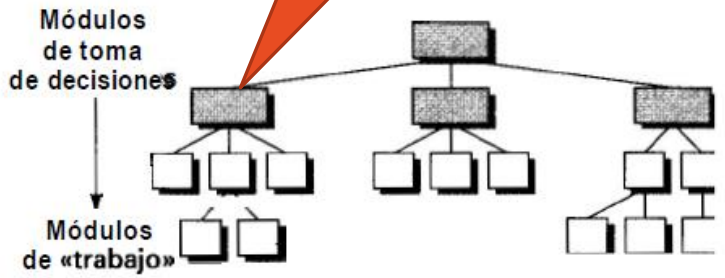
Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»División estructural

- Vertical
  - Presenta una división
  - entre los módulos de control
  - y los de trabajo

Deberán llevar a cabo funciones de control y no realizarán mucho trabajo de procesamiento.



(b) División Vertical

Fuente: Pressman Cap. 13

CONCEPTOS DE DISEÑO

»Estructuras de datos

- Es la relación de los datos del dominio y su representación en el sistema

»Procedimiento de software

- Se centra en el procesamiento de cada modulo individualmente, da un especificación precisa de la secuencia de sucesos, los puntos de toma de decisión, las iteraciones, etc.

»Ocultamiento de información

- Se consigue una modularidad efectiva definiendo un conjunto de módulos independientes que se comunican entre si intercambiando solo la información necesaria para su funcionalidad

Fuente: Pressman Cap. 13

DISEÑO MODULAR EFECTIVO

- Un diseño modular reduce la complejidad, facilita los cambios, y da como resultado una implantación mas fácil al fomentar el desarrollo paralelo de las diferentes partes del sistema.

Fuente: Pressman Cap. 13

DISEÑO MODULAR EFECTIVO

»Independencia Funcional

- Modularidad + Abstracción + Ocultamiento de Información
- Es deseable que cada modulo trate una subfunción de requisitos y tenga una interfaz sencilla para que sea mas fácil de desarrollar, mantener, probar y reusar
- Se mide mediante la cohesión y el acoplamiento entre los módulos
- Se busca una alta cohesión y bajo acoplamiento

Fuente: Pressman Cap. 13

DISEÑO MODULAR EFECTIVO

»Cohesión (Coherente)

- Se define como la medida de fuerza o relación funcional existente entre las sentencias o grupos de sentencias de un mismo módulo.
- Un módulo es altamente cohesivo cuando lleva a cabo solo una tarea dentro del procedimiento y requiere poca interacción con el resto de los procedimientos
- Un módulo es poco cohesivo cuando realiza tareas muy diferentes o sin relación entre ellas

Fuente: Pressman Cap. 13

DISEÑO MODULAR EFECTIVO

»Tipos de Cohesión

- Funcional → Cuando los módulos están relacionados en el desarrollo de una única función (la cohesión más alta)
- Comunicacional → Cuando los elementos de procesamiento se centran en los datos de entrada y salida
- Procedimental → Cuando los módulos relacionados tiene que ejecutarse en un orden especifico
- Temporal → Cuando los módulos se deben ejecutar en el mismo intervalo de tiempo
- Lógica → Cuando los módulos se relacionan lógicamente
- Coincidental → Cuando los módulos llevan a cabo un conjunto de tareas que no están relacionadas o tienen poca relación (la cohesión más baja)

Fuente: Pressman Cap. 13

DISEÑO MODULAR EFECTIVO

»Acoplamiento

- Es la medida de interconexión entre los módulos
- Punto donde se realiza la entrada o referencia y los datos que pasan a través de la interfaz.
- Una conectividad sencilla entre módulos da como resultado una conectividad mas fácil.

Fuente: Pressman Cap. 13

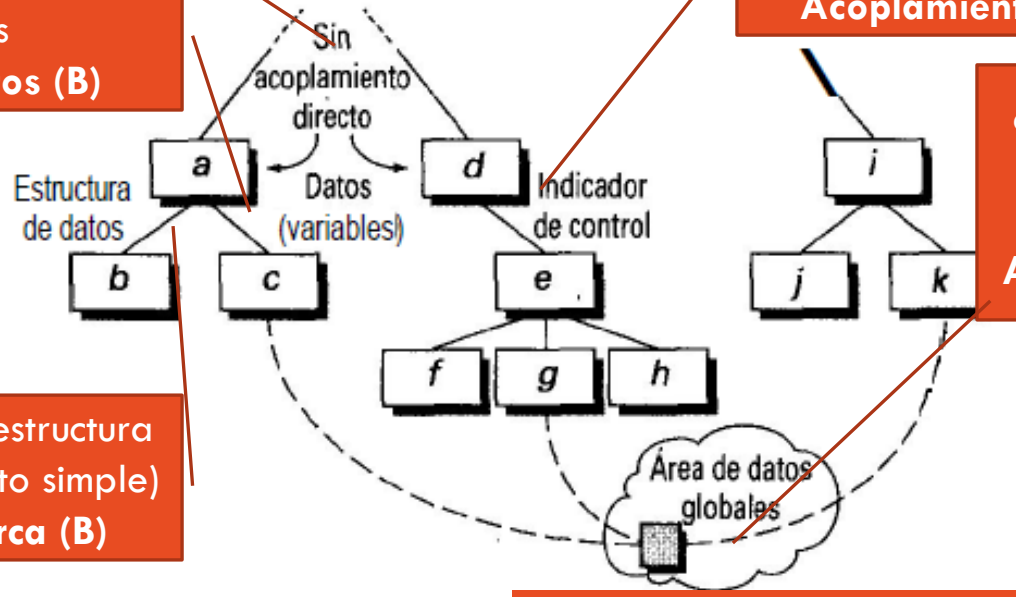
DISEÑO MODULAR EFECTIVO

»Press

a y d no hay acoplamiento

a y c lista de argumento s por la que pasan datos
Acoplamiento de datos (B)

d y e se pasa un indicador de control
Acoplamiento de control (M)



c ,g y k utilizan variables globales o compartidas (por ejemplo archivos)
Acoplamiento común (A)

a y b pasa parte de la estructura de datos (no un argumento simple)
Acoplamiento de marca (B)

Acoplamiento externo (A):
E/S protocolos de comunicación
Acoplamiento de contenido (A):
Un modulo hace usos de datos de control mantenido dentro de los limites de otro modulo

Fuente:

DISEÑO MODULAR EFECTIVO

- Niveles de Acoplamiento
  - Bajo :
    - Acoplamiento de datos
    - Acoplamiento de marca
  - Moderado :
    - Acoplamiento de control
  - Alto :
    - Acoplamiento común
    - Acoplamiento externo
    - Acoplamiento de contenido

Fuente: Pressman Cap. 13

DISEÑO DE DATOS

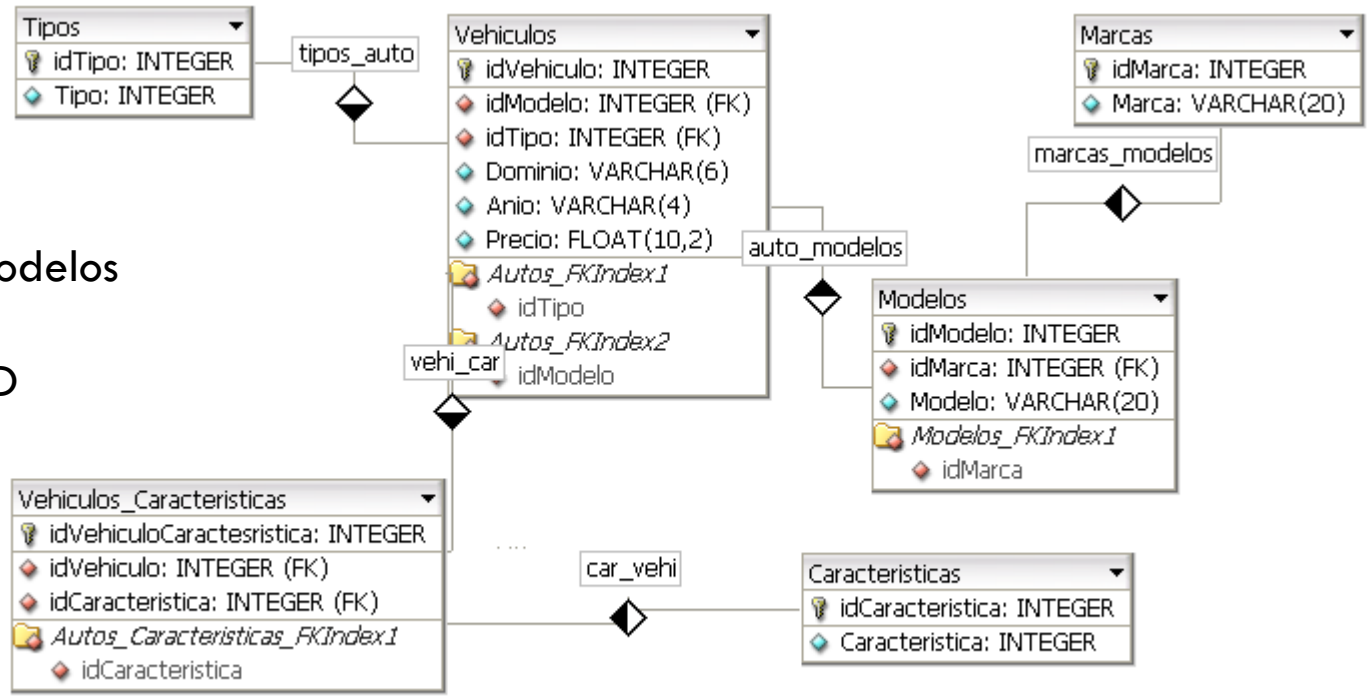
- Transforma el modelo del dominio obtenido del análisis en estructuras de datos, objetos de datos, relaciones

Fuente:

DISEÑO DE DATOS

»Los entidades y las relaciones definidas en el diagrama Entidad-Relación proporcionan la base de la actividad del diseño de datos

El diseño de los modelos de datos se ve en profundidad en BD



Fuente:

DISEÑO ARQUITECTÓNICO

Define la relación entre los elementos estructurales, para lograr los requisitos del sistema

DISEÑO ARQUITECTÓNICO

- Es el proceso de identificar los subsistemas dentro del sistema y establecer el marco de control y comunicación entre ellos.
- Los grandes sistemas se dividen en subsistemas que proporcionan algún conjunto de servicios relacionados

- Ventajas de un buen diseño
  - Comunicación con los stakeholders
  - Análisis del sistema
  - Reutilización a gran escala

- Arquitectura y requisitos no funcionales
  - La arquitectura afecta directamente a los requerimientos no funcionales
    - Rendimiento, Protección, Seguridad, Disponibilidad, Mantenibilidad

Fuente: Sommerville Cap 11

DISEÑO ARQUITECTÓNICO

»Arquitectura y requisitos no funcionales CRÍTICOS

- Rendimiento
  - Se deben agrupar las operaciones críticas en un grupo reducido de sub-sistemas (componentes de grano grueso, baja comunicación).
- Protección
  - Se debe utilizar una arquitectura en capas, protegiendo los recursos mas críticos en las capas mas internas.
- Seguridad
  - La arquitectura deberá diseñarse para que las operaciones relacionadas con la seguridad se localicen en único sub-sistema (o grupo pequeño), para reducir los costos por problemas de violaciones.
- Disponibilidad
  - La arquitectura se deberá diseñar con componentes redundantes para que sea posible el reemplazo sin detener el sistema, arquitectura muy tolerante a fallos.
- Mantenibilidad
  - La arquitectura del sistema debe diseñarse con componentes de grano fino que puedan modificarse con facilidad.

Fuente: Sommerville Cap 11

DISEÑO ARQUITECTÓNICO

- Organización del sistema
- Descomposición modular
- Modelos de control
- Arquitectura de los Sistemas Distribuidos
- Arquitectura de Aplicaciones

Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

- La organización del sistema representa la estrategia básica usada para estructurar el sistema
- Los subsistemas de un sistema deben intercambiar información de forma efectiva
  - Todos los datos compartidos, se almacenan en una base de datos central
  - Cada subsistema mantiene su información y los intercambia entre los subsistemas
- Estilos organizacionales
  - Repositorio, Cliente Servidor, Capas o combinaciones entre ellos

Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

»Modelo de repositorio

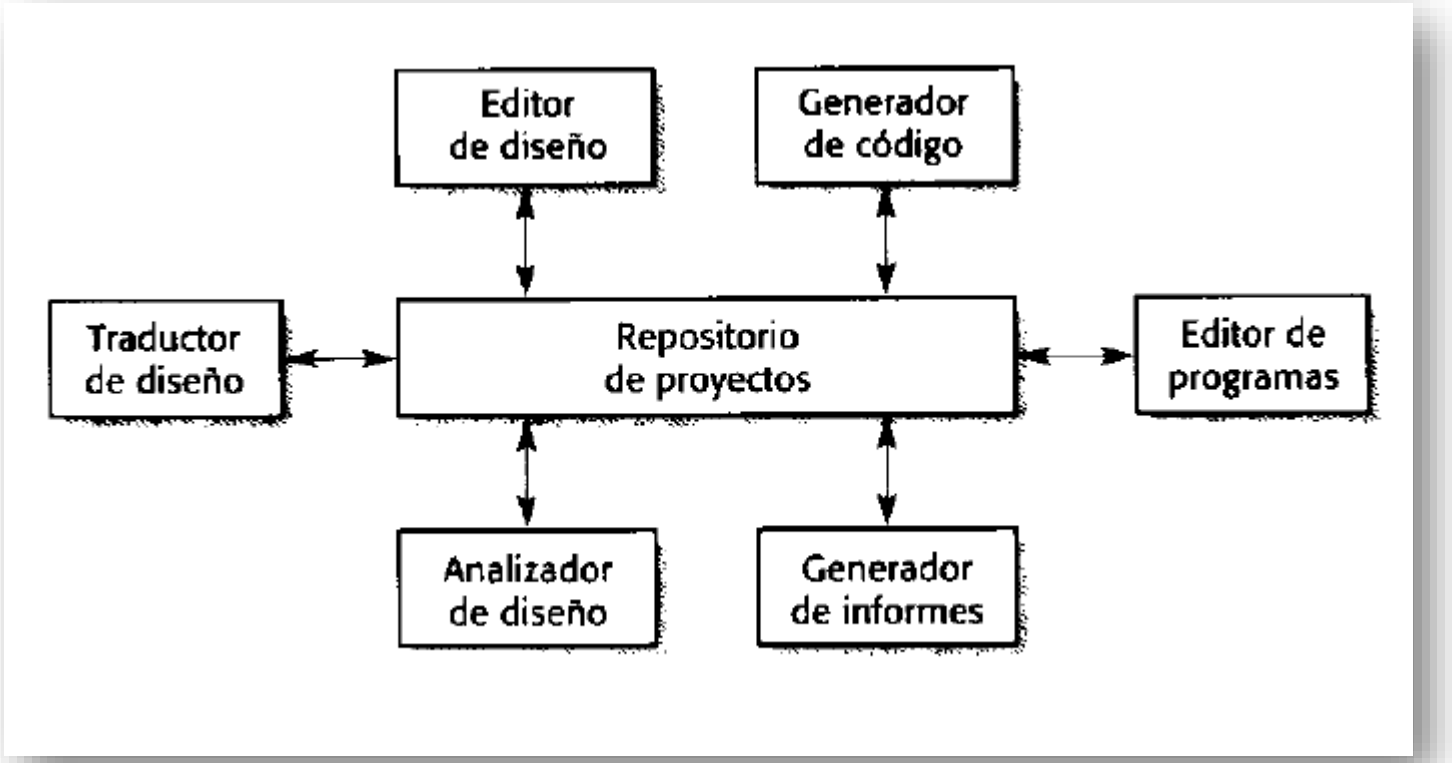
- La mayoría de los sistemas que usan grandes cantidades de datos se organizan alrededor de una base de datos compartida (repositorio)
- Los datos son generados por un subsistema y utilizados por otros módulos

- Ejemplo
  - Sistemas de gestión, sistemas CAD, Herramientas Case, etc.

Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

- Modelo de repositorio



Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

»Modelo de repositorio

- Ventajas
  - Forma eficiente de compartir grandes cantidades de datos, no hay necesidad de transmitir datos de un subsistema a otro
  - Los subsistemas que producen datos no deben saber como se utilizan
  - Las actividades de backup, protección, control de acceso están centralizadas.
  - El modelo compartido es visible a través del esquema del repositorio. Las nuevas herramientas se integran de forma directa, ya que son compatibles con el modelo de datos

Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

»Modelo de repositorio

- Desventajas
  - Los subsistemas deben estar acordes a los modelos de datos del repositorio, esto en algunos casos puede afectar el rendimiento
  - La evolución puede ser difícil a medida que se genera un gran volumen de información de acuerdo con el modelo de datos establecido, la migración de estos modelos puede ser muy difícil en alguno casos imposible
  - Diferentes subsistemas pueden tener distintos requerimientos de protección o políticas de seguridad y el modelo de repositorio impone las mismas para todos
  - Es difícil distribuir el repositorio en varias máquinas, existen repositorios centralizados lógicamente pero pueden ocasionar problemas de redundancia e inconsistencias

Fuente: Sommerville Cap 11

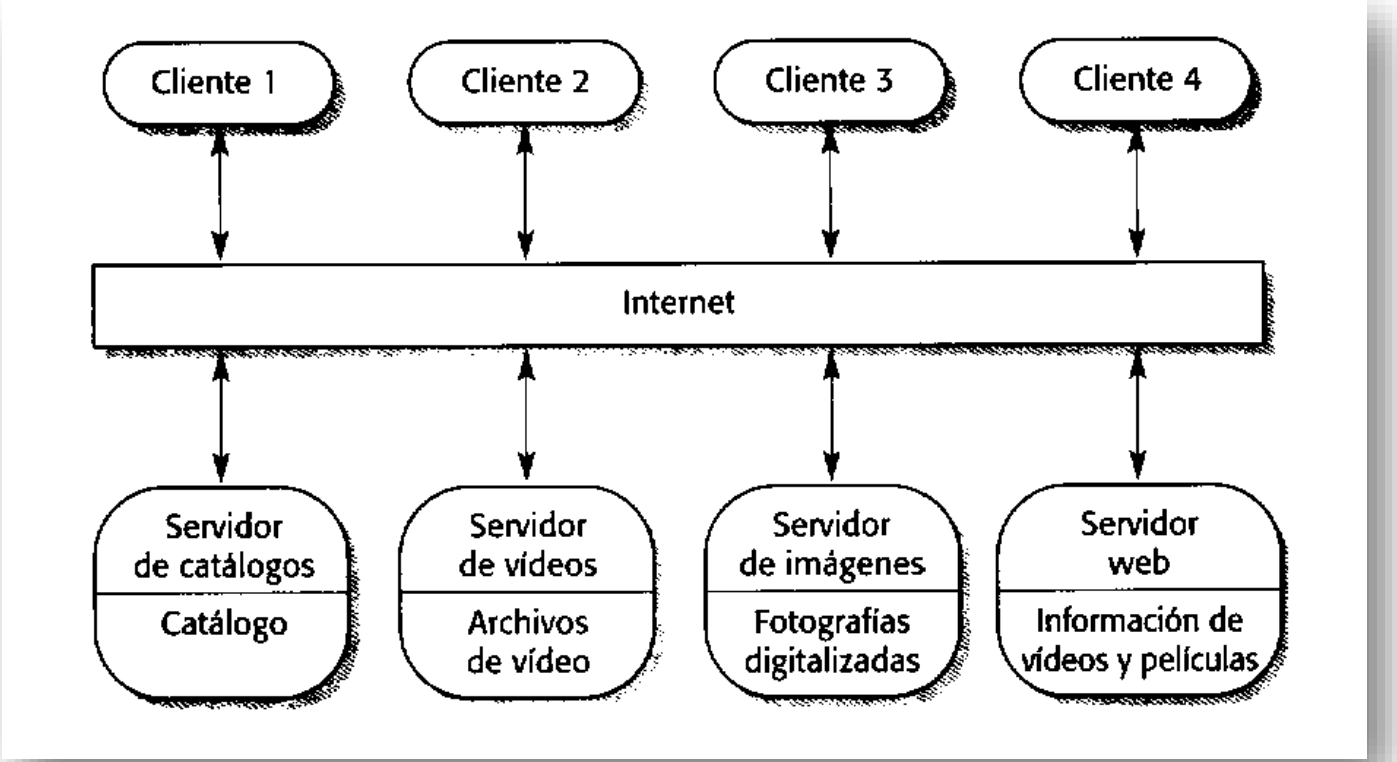
ORGANIZACIÓN DEL SISTEMA

- Modelo cliente servidor
- Es un modelo donde el sistema se organiza como un conjunto de servicios y servidores asociados, mas unos clientes que utilizan los servicios
- Componentes
  - Un conjunto de servidores que ofrecen servicios
  - Un conjunto de clientes que llaman a los servicios
  - Una red que permite a los clientes acceder a los servicios
    - Caso particular cuando los servicios y el cliente corren en la misma máquina
- Los clientes conocen los nombres de los servidores y los servicios que brinda, pero el servidor no necesita conocer a los clientes

Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

»Modelo cliente servidor



Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

»Modelo de capas

- El sistema se organiza en capas, donde cada una de ellas presenta un conjunto de servicios a sus capas adyacentes

- Ventajas
  - Soporta el desarrollo incremental
  - Es portable y resistente a cambios
  - Una capa puede ser reemplazada siempre que se mantenga la interfaz, y si varía la interfaz se genera una capa para adaptarlas
  - Permite generar sistemas multiplataforma, ya que solamente las capas mas internas son dependientes de la plataforma (se genera una capa interna para cada plataforma)

Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

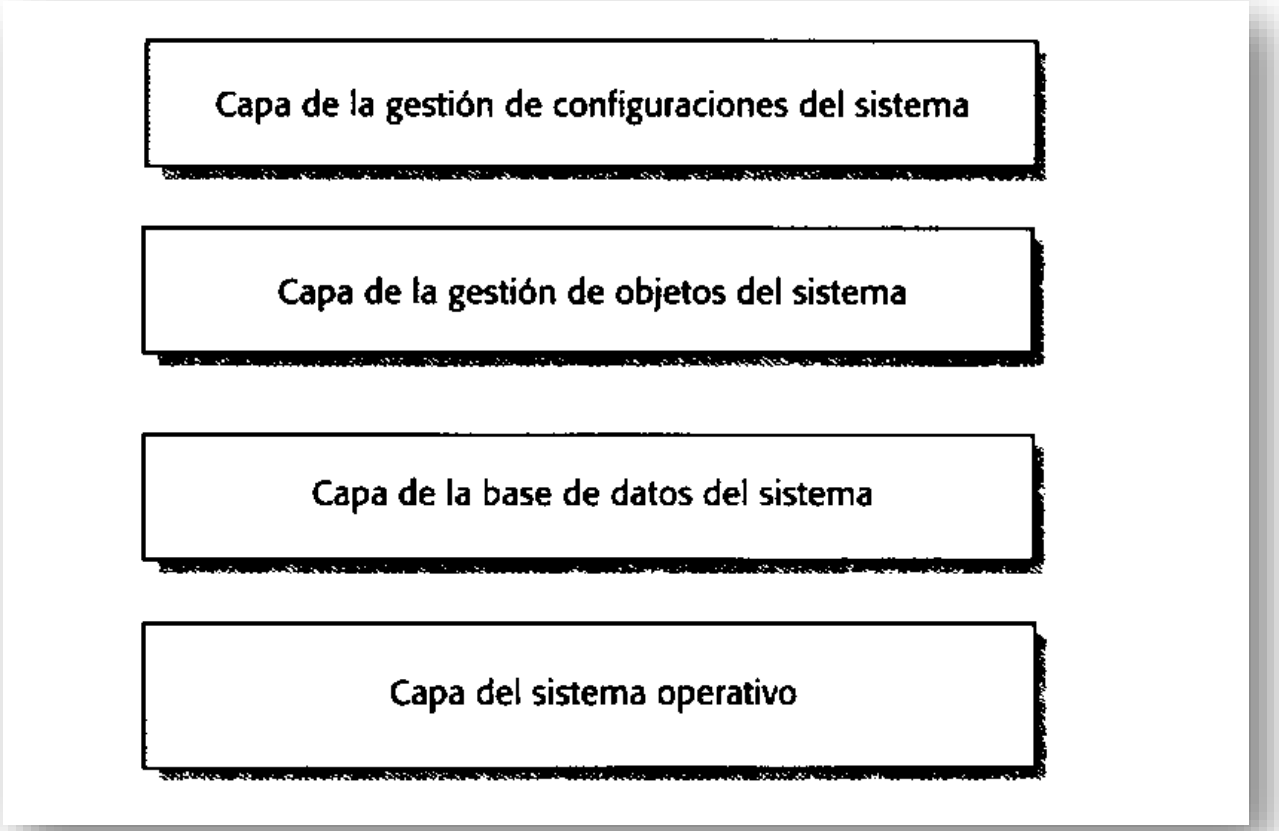
»Modelo de capas

- Desventajas
  - Difícil de estructurar
  - Las capas internas proporcionan servicios que son requeridos por todos los niveles
  - Los servicios requeridos por el usuario pueden estar brindados por las capas internas teniendo que atravesar varias capas adyacentes
  - Si hay muchas capas un servicio solicitado de la capa superior puede tener que ser interpretado por varias veces en diferentes capas

Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

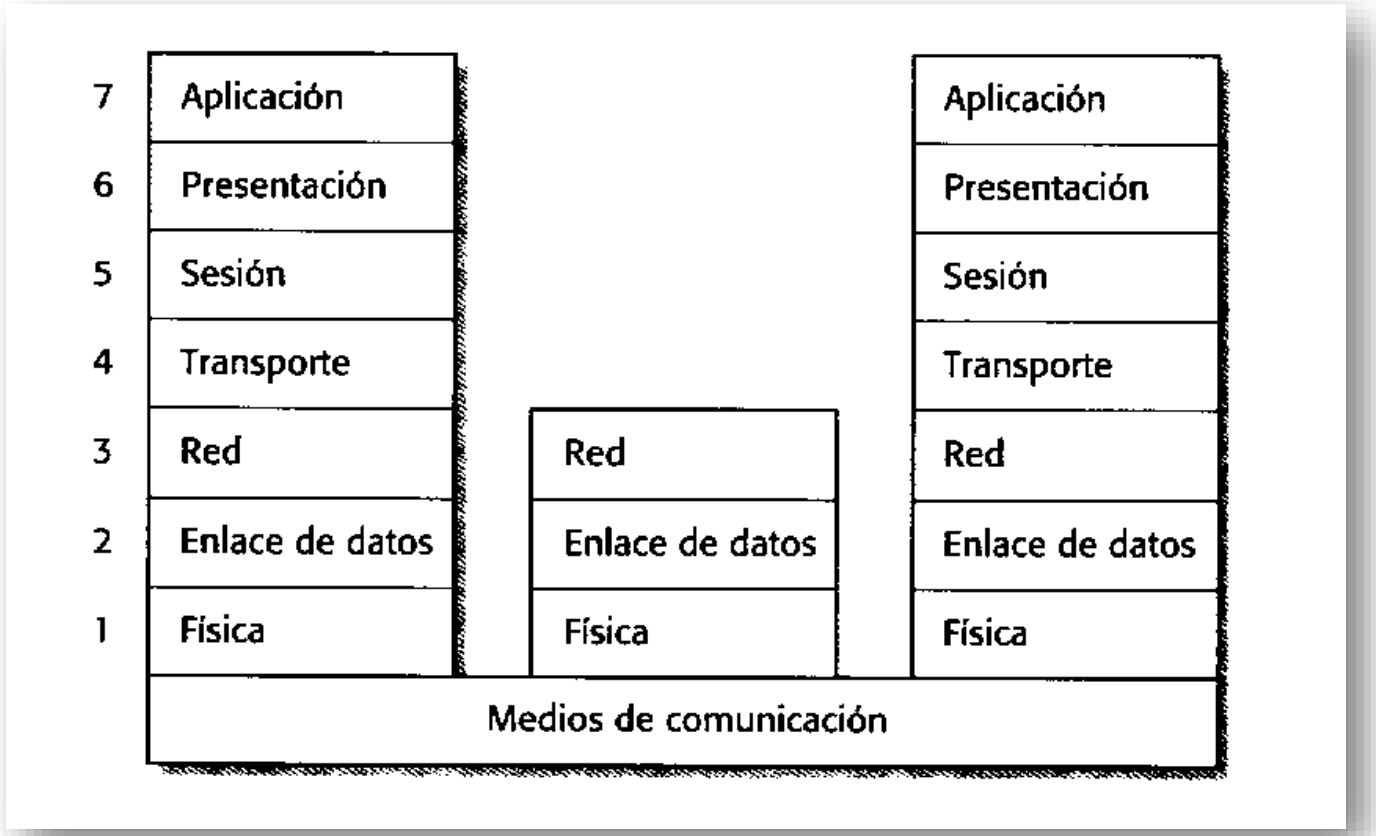
»Modelo de capas



Fuente: Sommerville Cap 11

ORGANIZACIÓN DEL SISTEMA

»Modelo OSI (Open System Interconection)



Fuente:

DESCOMPOSICIÓN MODULAR

»Una vez organizado el sistema, a los subsistemas los podemos dividir en módulos, se puede aplicar los mismos criterios que vimos en la organización, pero la descomposición modular es mas pequeña y permite utilizar otros estilos alternativos.

»Estrategias de descomposición modular

- Descomposición orientada a flujo de funciones
  - Conjunto de módulos funcionales (ingresan datos y los transforman en salida).
- Descomposición orientada a objetos
  - Conjunto de objetos que se comunican.

Fuente: Sommerville Cap 11

DESCOMPOSICIÓN MODULAR

»Definiciones

- Subsistema
  - Es un sistema en si mismo cuyo funcionamiento no depende de los servicios proporcionados por otros, los subsistemas se componen de módulos con interfaces definidas que se utilizan para comunicarse con otros subsistemas.

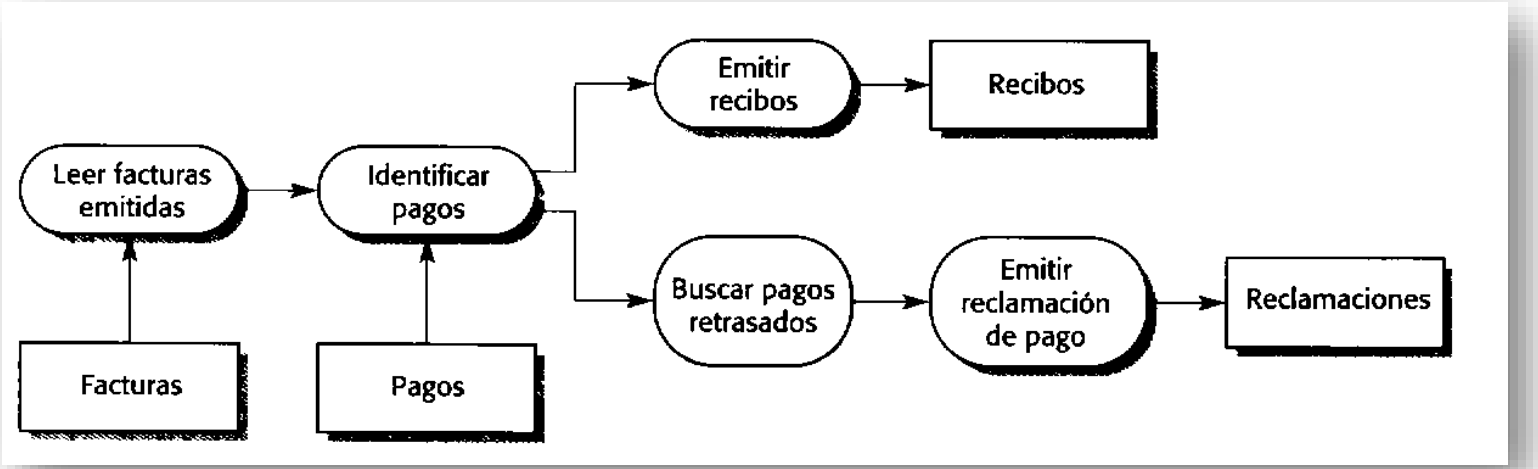
- Módulo
  - Es un componente de un subsistema que proporciona uno o mas servicios a otros módulos. A su vez utiliza servicios proporcionados por otros módulos. Por lo general no se los considera un sistema independiente.

Fuente: Sommerville Cap 11

DESCOMPOSICIÓN MODULAR

»Descomposición orientada a flujo de funciones

- En un Modelo orientado a flujo de funciones , los datos fluyen de una función a otra y se transforman a medida que pasan por una secuencia de funciones hasta llegar a los datos de salida. Las transformaciones se pueden ejecutar en secuencial o en paralelo.

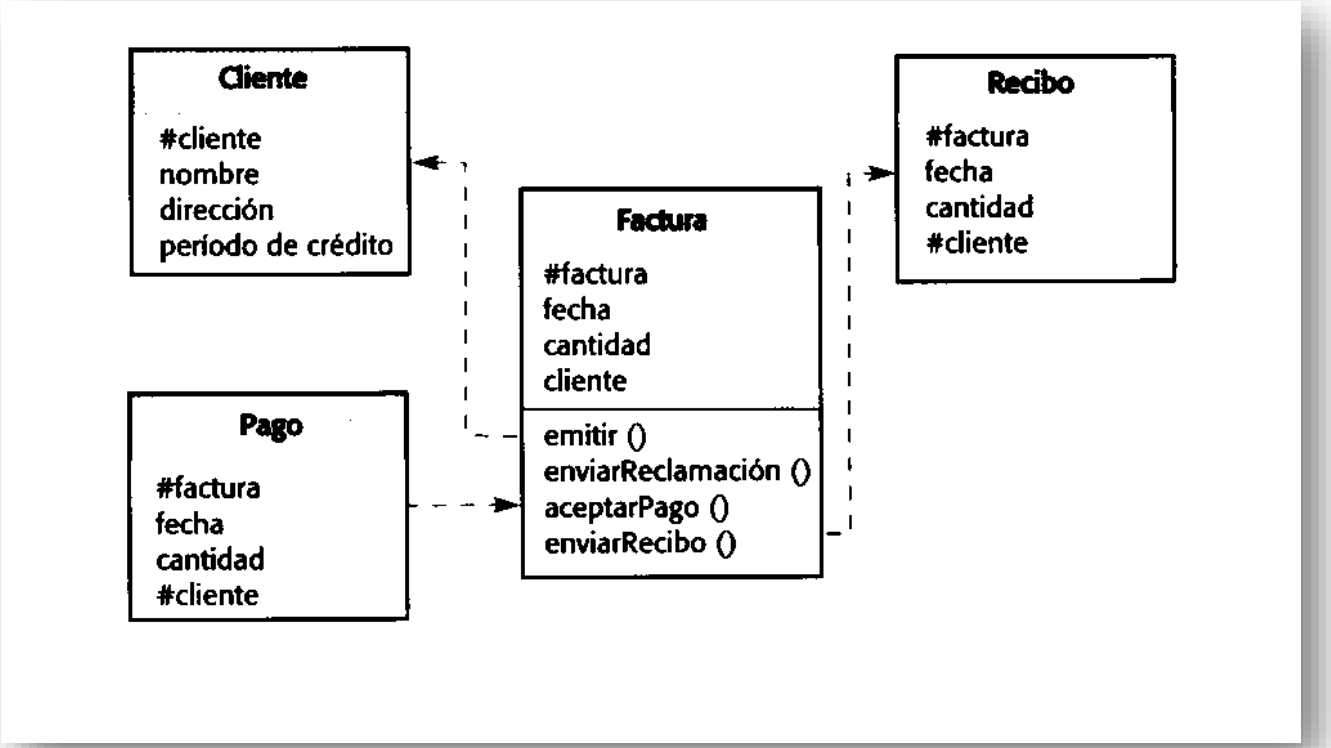


Fuente:

DESCOMPOSICIÓN MODULAR

»Descomposición orientada a objetos

- Un modelo arquitectónico orientado a objetos, estructura el sistema en un conjunto de objetos débilmente acoplados y con interfaces bien definidas.



Fuente: Sommerville Cap 11

MODELOS DE CONTROL

»En un sistema, los subsistemas están controlados para que sus servicios se entreguen en el lugar correcto en el momento preciso.

»Los modelos de control a nivel arquitectónico

- Control Centralizado
  - Un subsistema tiene la responsabilidad de iniciar y detener otro subsistema.
- Control Basadas en Eventos
  - Cada subsistema responde a eventos externo al subsistema.

Fuente: Sommerville Cap 11

MODELOS DE CONTROL

»Control Centralizado

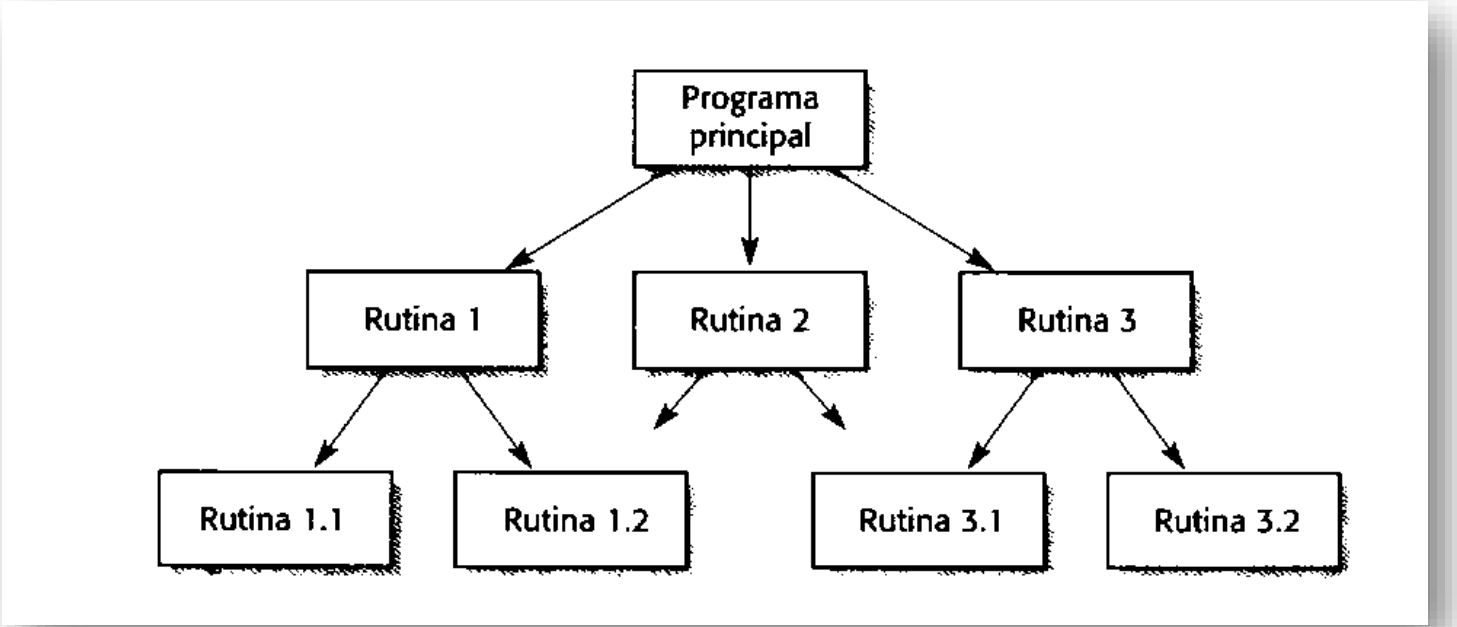
- Un subsistema se diseña como controlador y tiene la responsabilidad de gestionar la ejecución de otros subsistemas, la ejecución puede ser secuencial o en paralelo
  - Modelo de llamada y retorno
    - Modelo de subrutinas descendentes
    - Aplicable a modelos secuenciales
  - Modelo de gestor
    - Un gestor controla el inicio y parada coordinado con el resto de los procesos
    - Aplicable a modelos concurrentes

Fuente: Sommerville Cap 11

MODELOS DE CONTROL

»Control Centralizado

- Modelo de llamada y retorno

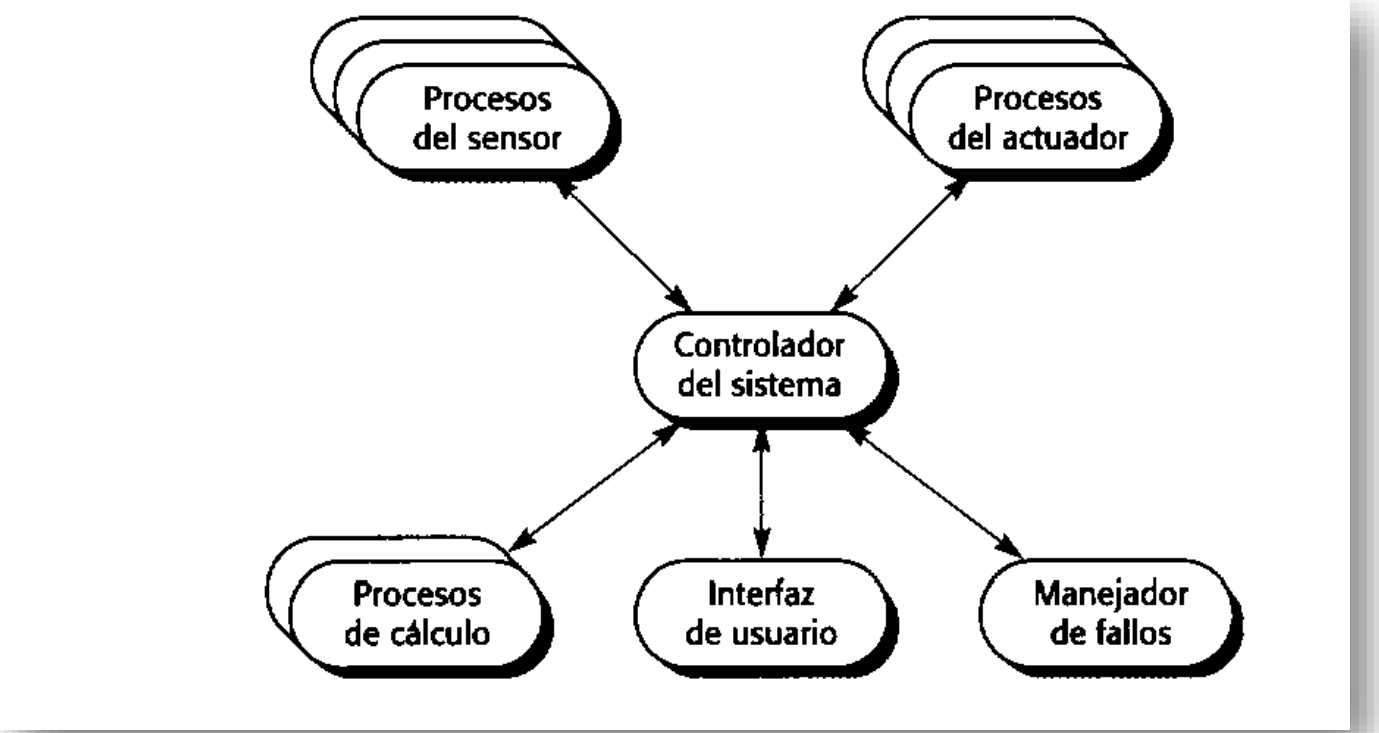


Fuente: Sommerville Cap 11

MODELOS DE CONTROL

»Control Centralizado

- Modelo de gestor



Fuente: Sommerville Cap 11

MODELOS DE CONTROL

»Sistema Dirigido Por Eventos

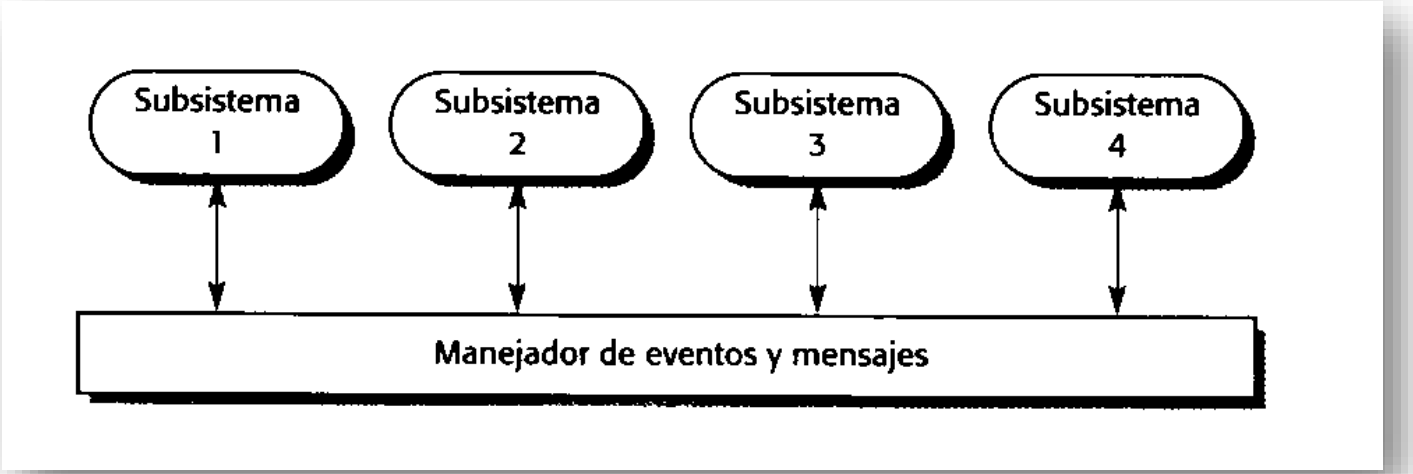
- Se rigen por eventos generados externamente al proceso
- Evento
  - Señal binaria
  - Un valor dentro de un rango
  - Una entrada de un comando
  - Una selección del menú
- Modelos de sistemas dirigidos por eventos
  - Modelos de transmisión (Broadcast)
  - Modelo dirigido por interrupciones

Fuente: Sommerville Cap 11

MODELOS DE CONTROL

»Sistema Dirigido Por Eventos

- Modelos de transmisión (Broadcast)
  - Un evento se transmite a todos los subsistemas, cualquier subsistema programado para manejar ese evento lo atenderá

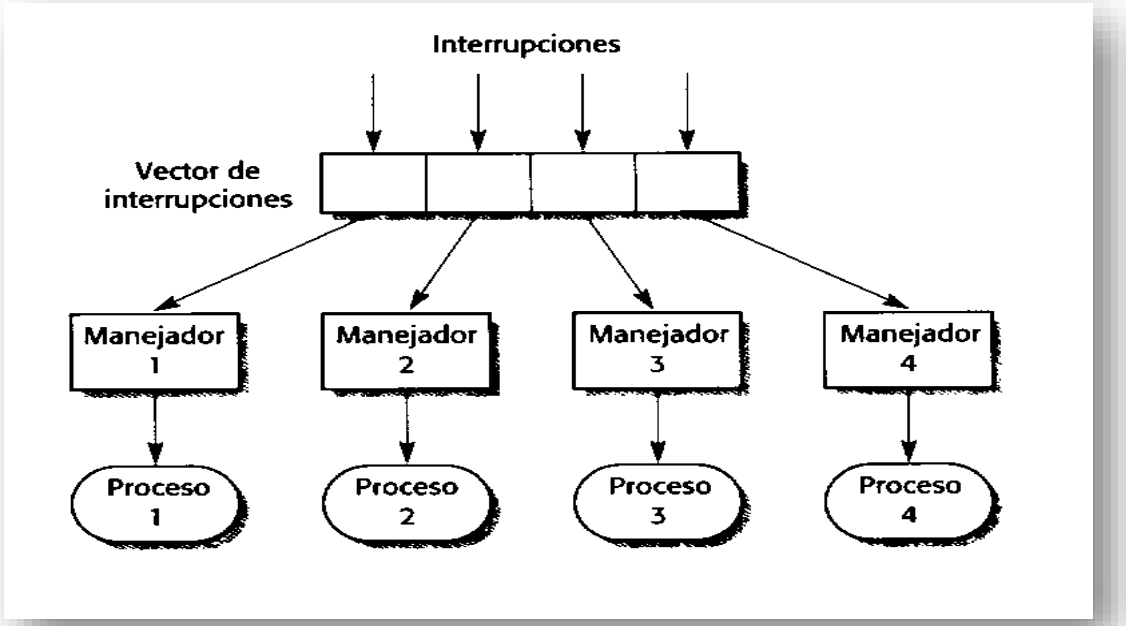


Fuente: Sommerville Cap 11

MODELOS DE CONTROL

»Sistema Dirigido Por Eventos

- Modelo dirigido por interrupciones
  - Se utilizan en sistemas de tiempo real donde las interrupciones externas son detectadas por un manejador de interrupciones y se envía a algún componente para su procesamiento



Fuente: Sommerville Cap 11

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Un sistema distribuido es un sistema en el que el procesamiento de información se distribuye sobre varias computadoras.

»Tipos genéricos de sistemas distribuidos

- Cliente Servidor
- Objetos distribuidos

»Características de los sistemas distribuidos

- Compartir recursos
- Apertura
- Concurrencia
- Escalabilidad
- Tolerancia a fallos

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Características de los sistemas distribuidos

- Compartir recursos
  - Un sistema distribuido permite compartir recursos
- Apertura
  - Son sistemas abiertos y se diseñan con protocolos estándar para simplificar la combinación de los recursos
- Concurrencia
  - Varios procesos puede operar al mismo tiempo sobre diferentes computadoras
- Escalabilidad
  - La capacidad puede incrementarse añadiendo nuevos recursos para cubrir nuevas demandas
- Tolerancia a fallos
  - La disponibilidad de varias computadoras y el potencial para reproducir información hace que los sistemas distribuidos sean mas tolerantes a fallos de funcionamiento de hardware y software

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Desventajas de los sistemas distribuidos

- Complejidad
  - Son mas complejos que los centralizados, además del procesamiento hay que tener en cuenta los problemas de la comunicación y sincronización entre los equipos
- Seguridad
  - Se accede al sistema desde varias computadoras generando tráfico en la red que puede ser intervenido
- Manejabilidad
  - Las computadoras del sistema pueden ser de diferentes tipos y diferentes S.O. lo que genera más dificultades para gestionar y mantener el sistema
- Impredecibilidad
  - La respuesta depende de la carga del sistema y del estado de la red, lo que hace que el tiempo de respuesta varíe entre una petición y otra

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

- Arquitectura
  - Multiprocesador
  - Cliente Servidor
    - Dos Capas
    - Tres o mas Capas
  - Objetos Distribuidos
  - Computación distribuida inter-organizacional
    - Peer to peer
    - Orientada a servicios

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Multiprocesador

- El sistema de software está formado por varios procesos que pueden o no ejecutarse en procesadores diferentes
- La asignación de los procesos a los procesadores puede ser predeterminada o mediante un dispatcher
- Es común en sistemas grandes de tiempo real que recolectan información, toman decisiones y envían señales para modificar el entorno

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Multiprocesador

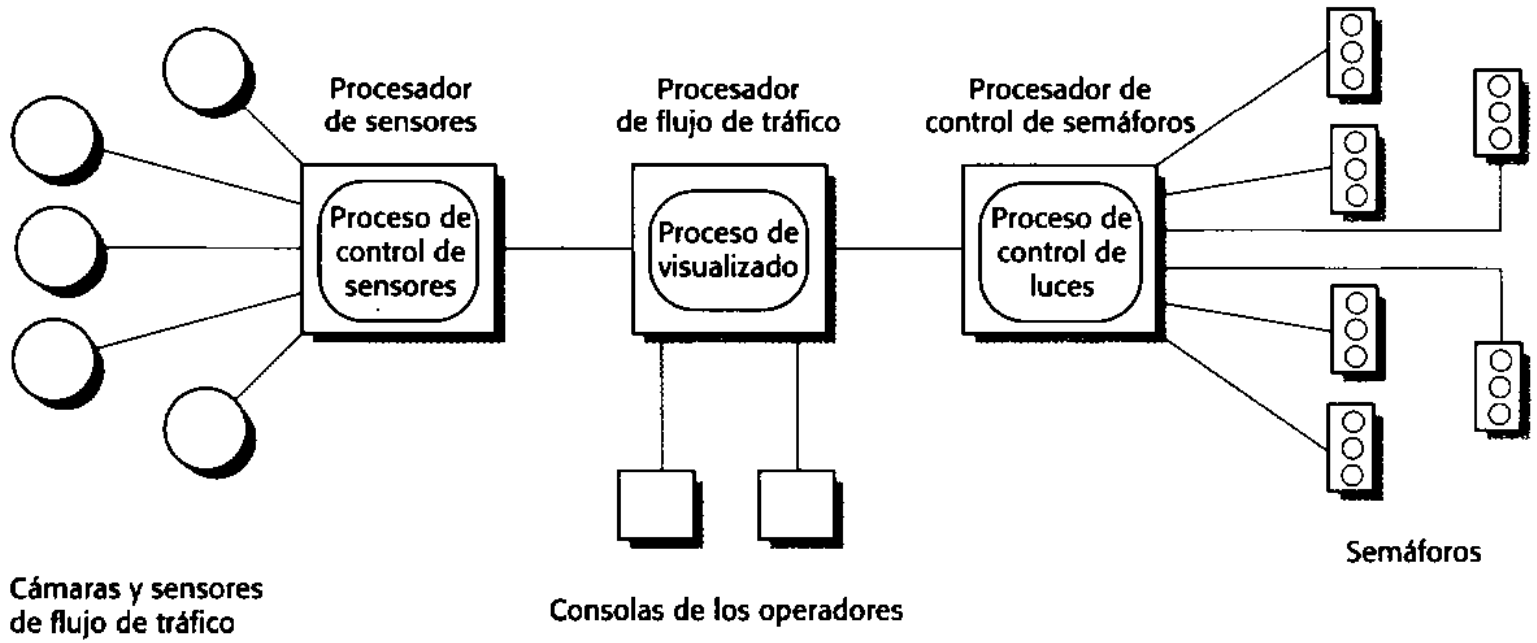


Figura 12.1 Un sistema multiprocesador de control de tráfico.

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Cliente Servidor

- Una aplicación se modela como un conjunto de servicios proporcionado por los servidores y un conjunto de clientes que usan estos servicios
- Los clientes y servidores son procesos diferentes
- Los servidores pueden atender varios clientes
- Un servidor puede brindar varios servicios
- Los clientes no se conocen entre si

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Cliente Servidor

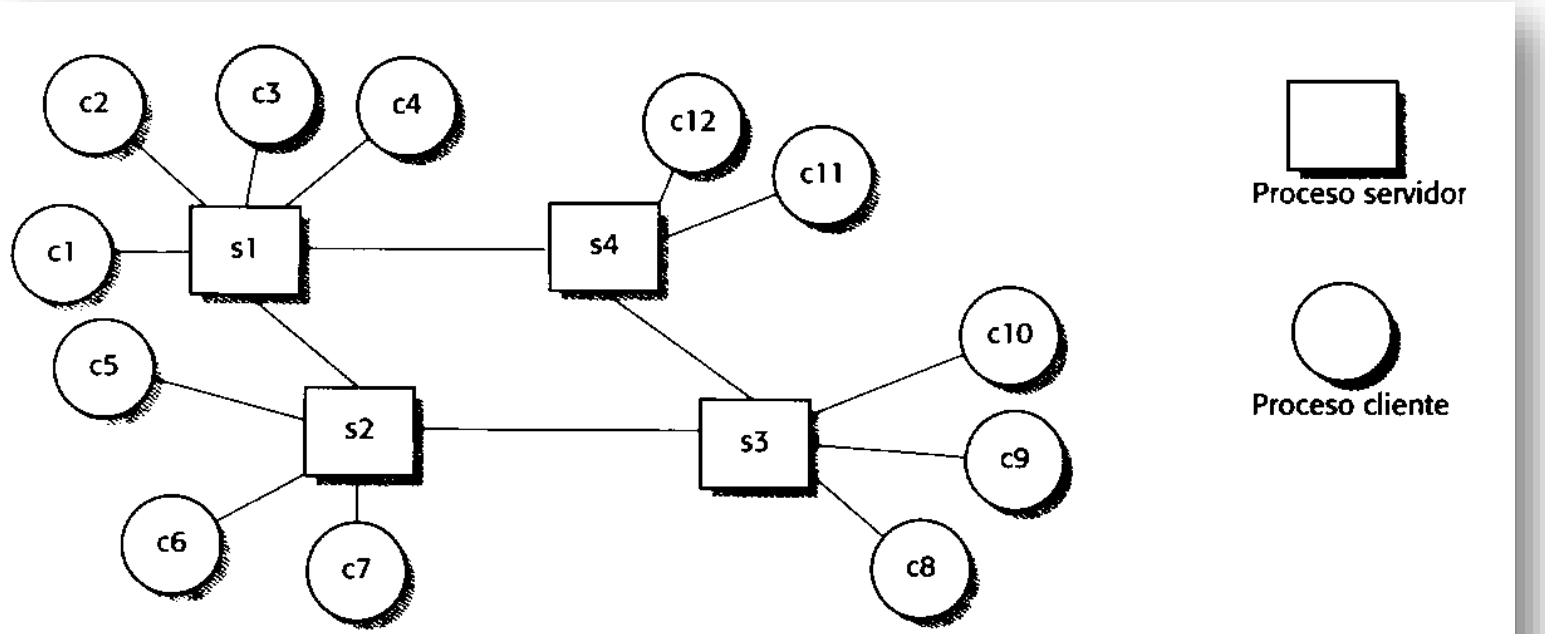
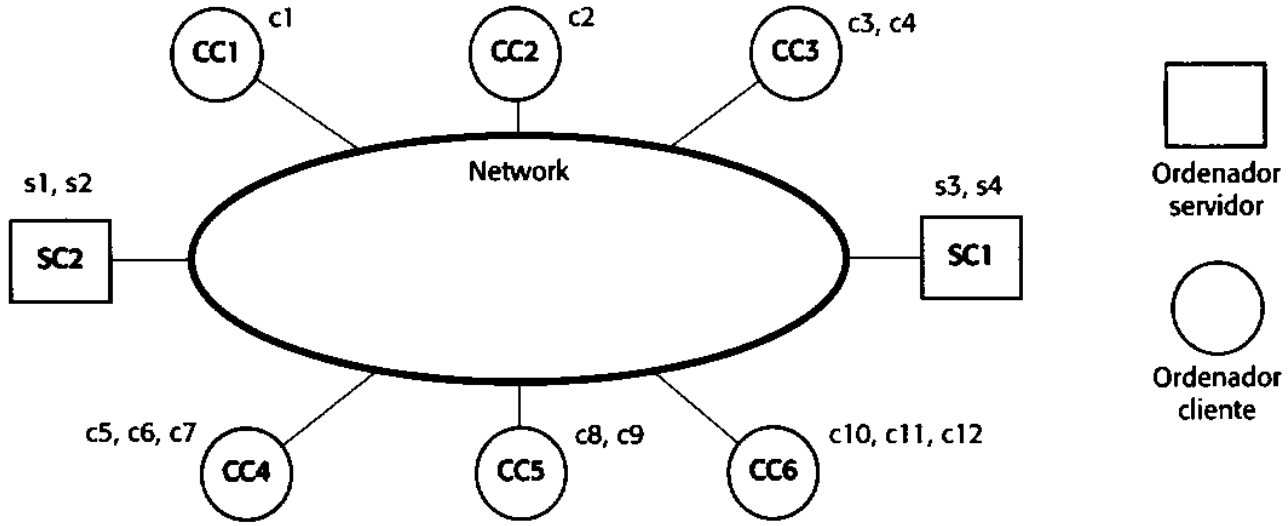


Figura 12.2 Un sistema cliente-servidor.

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Cliente Servidor



Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Cliente Servidor

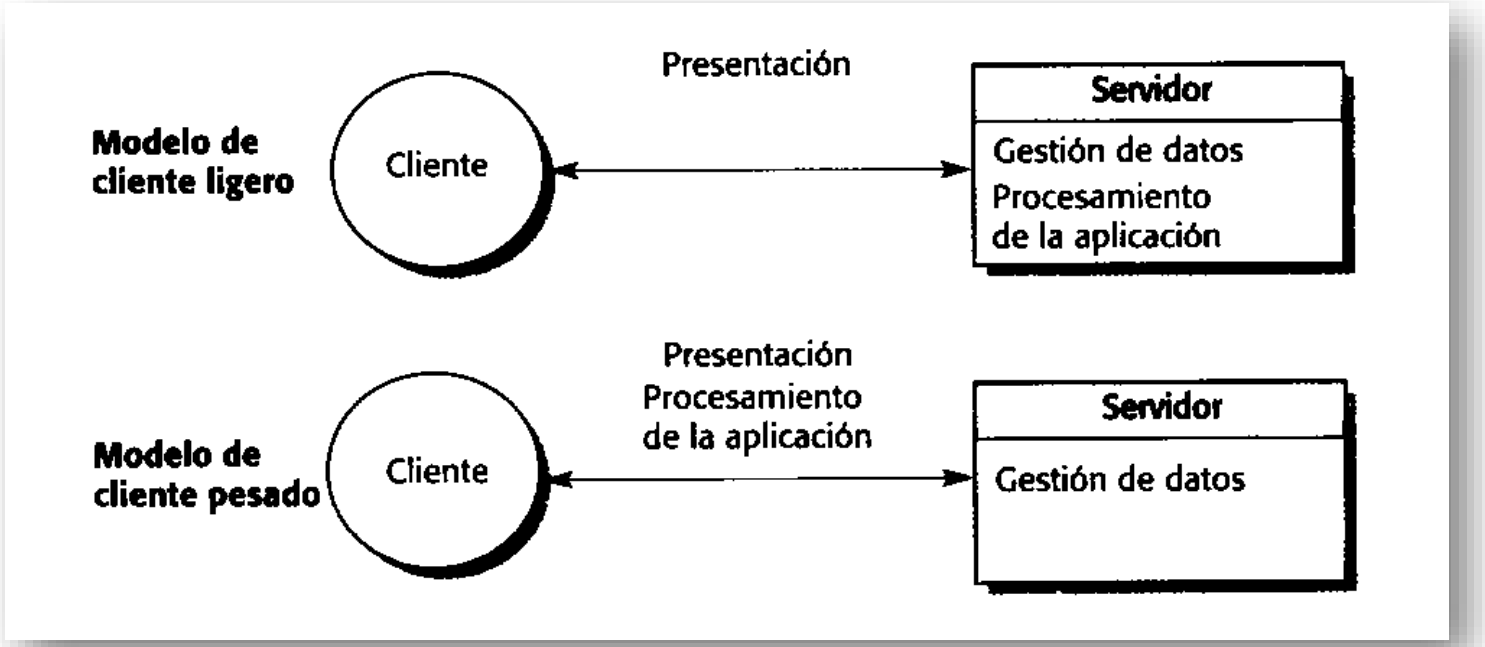
- Clasifican en Capas
  - Dos Capas
    - Cliente ligero
      - El procesamiento y gestión de datos se lleva a cabo en el servidor
    - Cliente pesado
      - El cliente implementa la lógica de la aplicación y el servidor solo gestiona los datos
  - Tres o más capas
    - La presentación, el procesamiento y la gestión de los datos son procesos lógicamente separados y se pueden ejecutar en procesadores diferentes

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Cliente Servidor

- Clasifican en Capas
  - Dos Capas

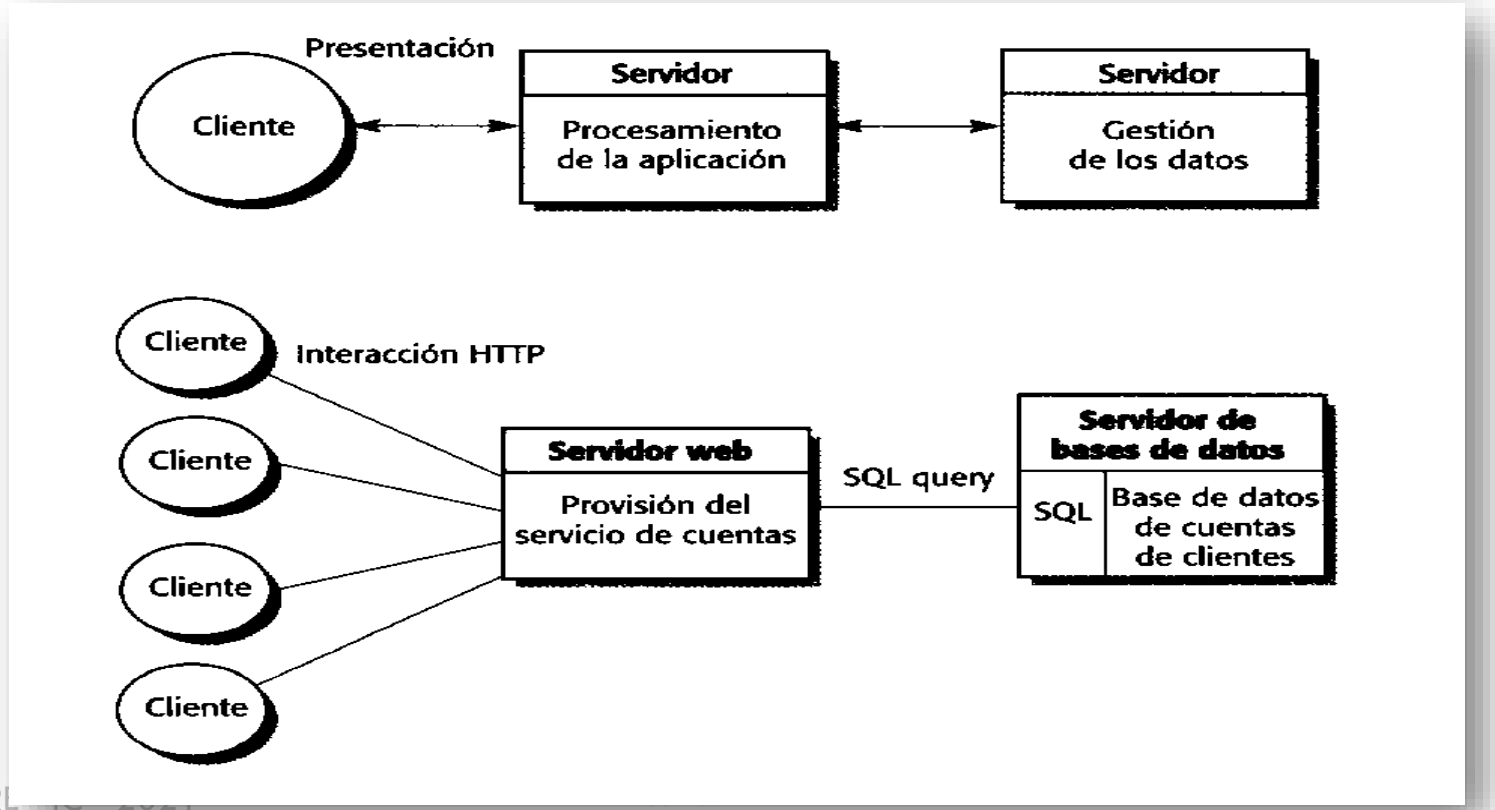


Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura Cliente Servidor

- Clasifican en Capas
  - Tres o Más Capas



ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

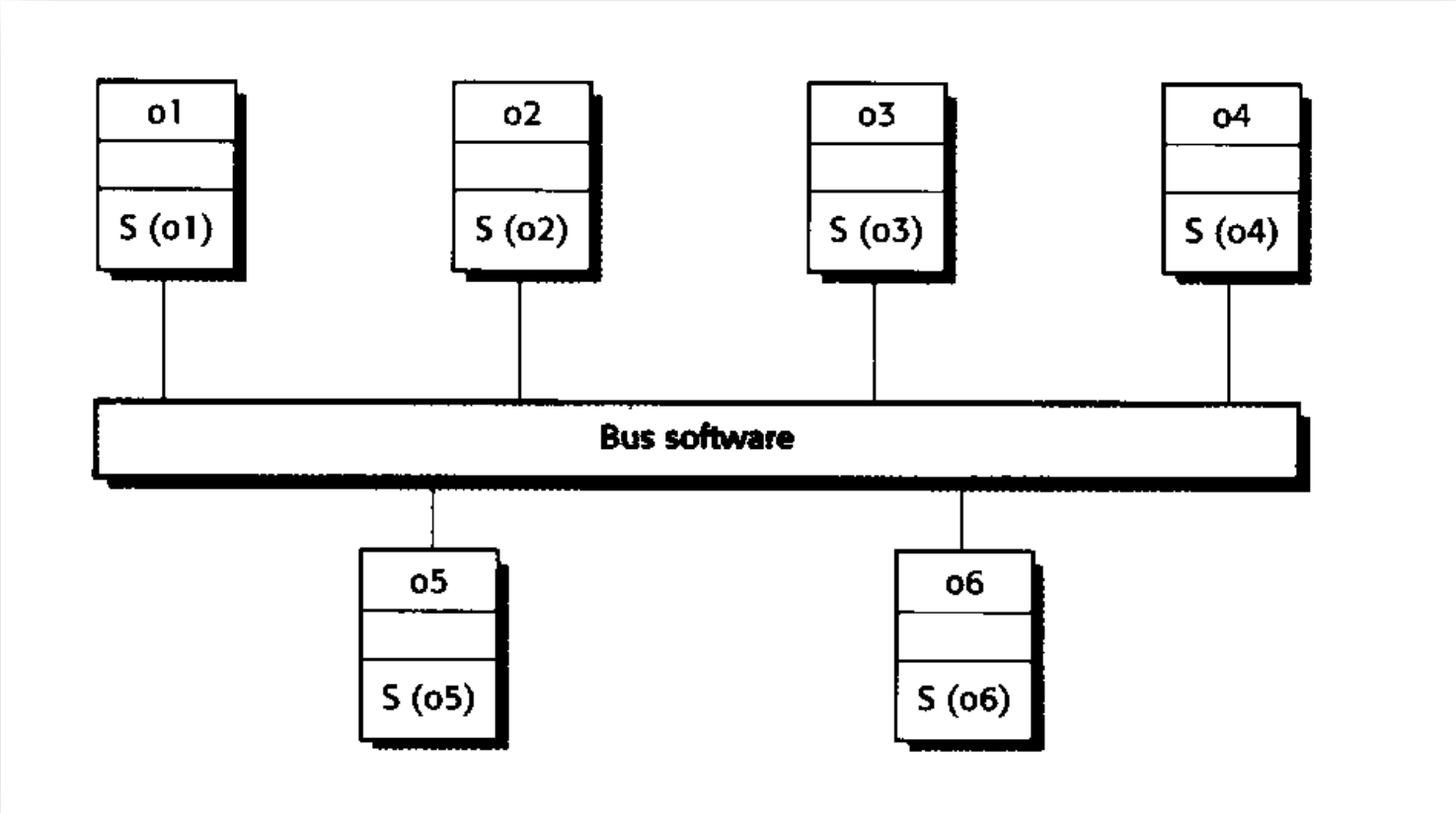
»Arquitectura de Objetos Distribuidos

- Los componentes fundamentales son objetos que brindan una interfaz de un conjunto de servicios que ellos suministran. Otros objetos realizan llamadas a estos servicios, sin hacer distinción entre cliente y servidores
- Los objetos pueden distribuirse en varias maquinas a través de la red utilizando un middleware como intermediario de peticiones

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Arquitectura de Objetos Distribuidos



Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Computación Distribuida interorganizacional

- Una organización tiene varios servidores y reparte su carga computacional entre ellos.
- Extender este concepto a varias organizaciones.
- Arquitecturas
  - Peer-to-Peer
  - Orientados a servicios

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Computación Distribuida interorganizacional

- Arquitecturas Peer-to-Peer (P2P)
- Sistemas descentralizados en los que el cálculo puede llevarse a cabo en cualquier nodo de la red
- Se diseñan para aprovechar la ventaja de la potencia computacional y el almacenamiento a través de una red
- Puedes utilizar una arquitectura
  - Descentralizada
    - donde cada nodo rutea los paquetes a sus vecinos hasta encontrar el destino
  - Semi-centralizada
    - donde un servidor ayuda a conectarse a los nodos o coordinar resultados
- Ejemplos: Kazza, Torrents, Emule, SETI@Home

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Computación Distribuida interorganizacional

- Arquitecturas Peer-to-Peer (P2P)

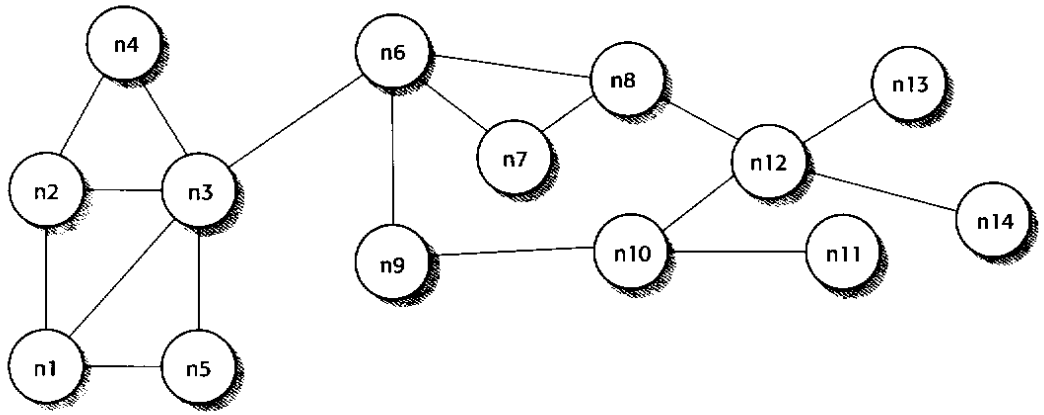
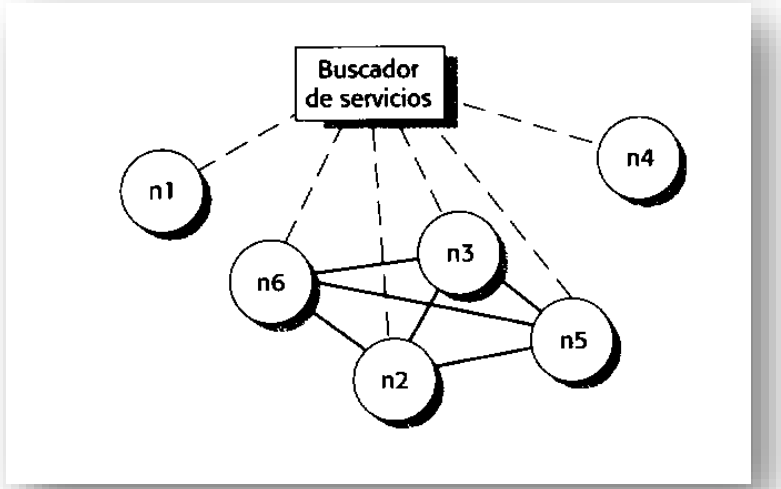


Figura 12.15 Arquitectura p2p descentralizada.



Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Computación Distribuida inter-organizacional

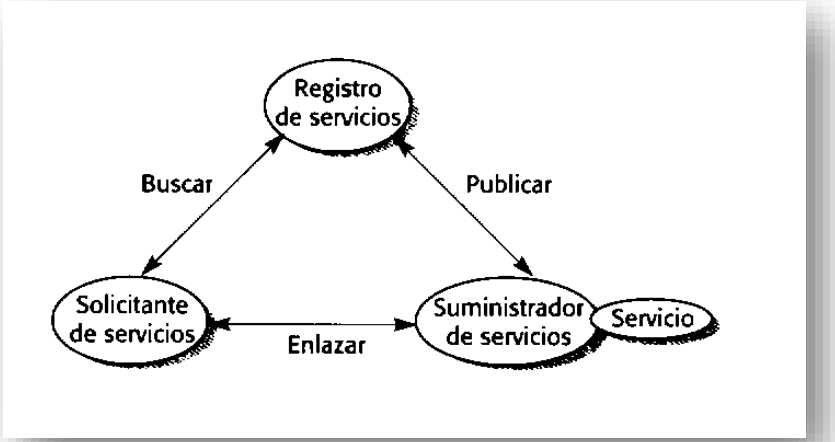
- Arquitectura de sistemas orientadas a servicios
  - Servicio
    - Representación de un recurso computacional o de información que puede ser utilizado por otros programas.
    - Un servicio es independiente de la aplicación que lo utiliza
    - Un servicio puede ser utilizado por varias organizaciones
    - Una aplicación puede construirse enlazando servicios
    - Las arquitecturas de las aplicaciones de servicios web son arquitecturas débilmente acopladas

Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Computación Distribuida inter-organizacional

- Arquitectura de sistemas orientadas a servicios
  - Funcionamiento
    - Un proveedor de servicios oferta servicios definiendo su interfaz y su funcionalidad
    - Para que el servicio sea externo, el proveedor publica el servicio en un “servicio de registro” con información del mismo
    - Un solicitante enlaza este
    - servicio a su aplicación,
    - es decir que el solicitante
    - incluye el código para
    - invocarlo y procesa el
    - resultado del mismo



Fuente: Sommerville Cap 12

ARQUITECTURA DE LOS SISTEMAS DISTRIBUIDOS

»Computación Distribuida inter-organizacional

- Arquitectura de sistemas orientadas a servicios
  - Los estándares fundamentales que permiten la comunicación entre servicios
  - SOAP (Simple Object Access Protocol)
    - Define una organización para intercambio de datos estructurados entre servicios web
  - WSDL (Web Service Description Language)
    - Define como pueden representarse las interfaces web
  - UDDI (Universal Description Discovery and Integration)
    - Estándar de búsqueda que define como puede organizarse la información de descripción de servicios

Fuente: Sommerville Cap 12

RESUMEN

Diseño de Software

- Definiciones generales

- Diseño de la interfaz
  - Dar control al usuario
  - Reducir la carga de memoria del usuario
  - Lograr una Interfaz consistente
  - Factores Humanos

- Conceptos de Diseño
  - Abstracción Refinamiento Modularidad Arquitectura Jerarquía de control División estructural Estructuras de datos Procedimiento de software Ocultamiento de información

- Diseño Arquitectónico
  - Organización del sistema
    - Repositorio, Cliente Servidor, Capas o combinaciones entre ellos
  - Descomposición modular
    - Orientada a flujo de funciones o a objetos
  - Modelos de control
    - Control Centralizado
      - Modelo de llamada y retorno /Modelo de gestor
    - Control Basadas en Eventos
  - Arquitectura de los Sistemas Distribuidos
    - Multiprocesador
    - Cliente Servidor
      - Dos Capas
      - Tres o mas Capas
    - Objetos Distribuidos
    - Computación distribuida inter-organizacional
      - Peer to peer
    - Orientada a servicios