Ingeniería de software II

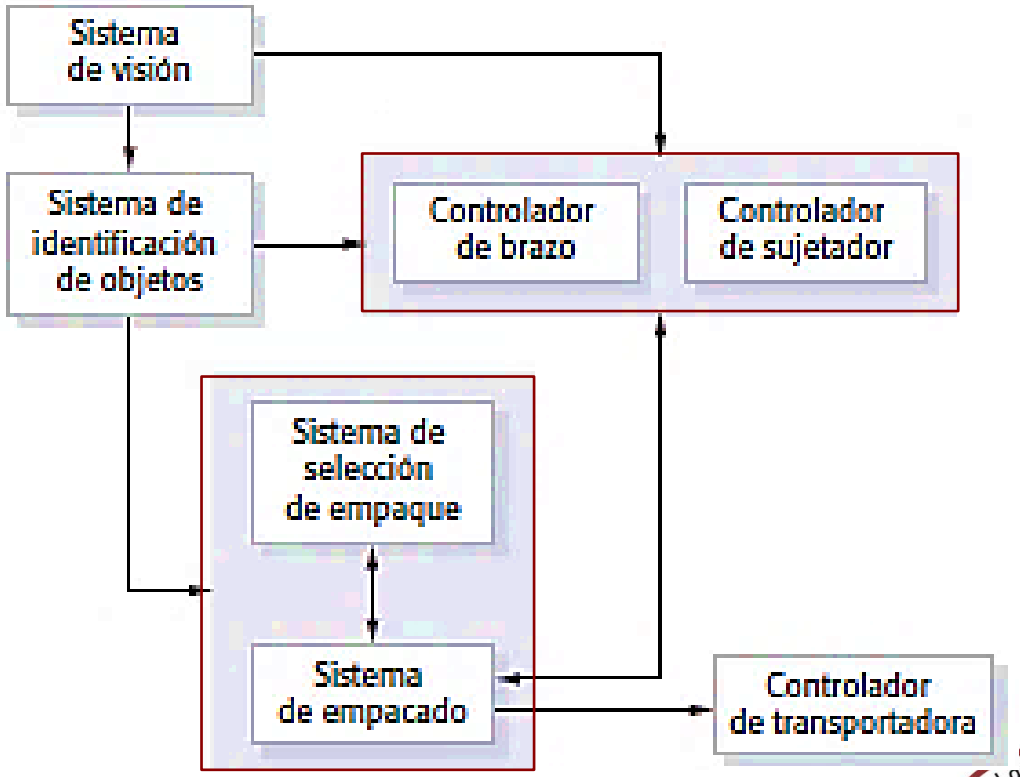
Diseño de Software – Arquitecturas

Diseño Arquitectónico

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

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



Fuente: »Sommerville 9ª Edición Cap 6

Diseño Arquitectónico

»La arquitectura afecta directamente a los requerimientos no funcionales

Los más CRÍTICOS

Rendimiento, Protección, Seguridad, Disponibilidad, Mantenibilidad

Rendimiento

Se deben agrupar las operaciones críticas en un grupo reducido de sub-sistemas (componentes de grano grueso, baja comunicación).

Seguridad

Se debe utilizar una arquitectura en capas, protegiendo los recursos más críticos en las capas más internas.

Fuente: »Sommerville 9ª Edición Cap 6

Diseño Arquitectónico

»Arquitectura y requisitos no funcionales

Protección

La arquitectura deberá diseñarse para que las operaciones relacionadas con la protección se localicen en un único sub-sistema (o grupo pequeño), para reducir los costos y problemas de validación de la protección.

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 autocontenidos de grano fino que puedan modificarse con facilidad.

Fuente: »Sommerville 9ª Edición Cap 6

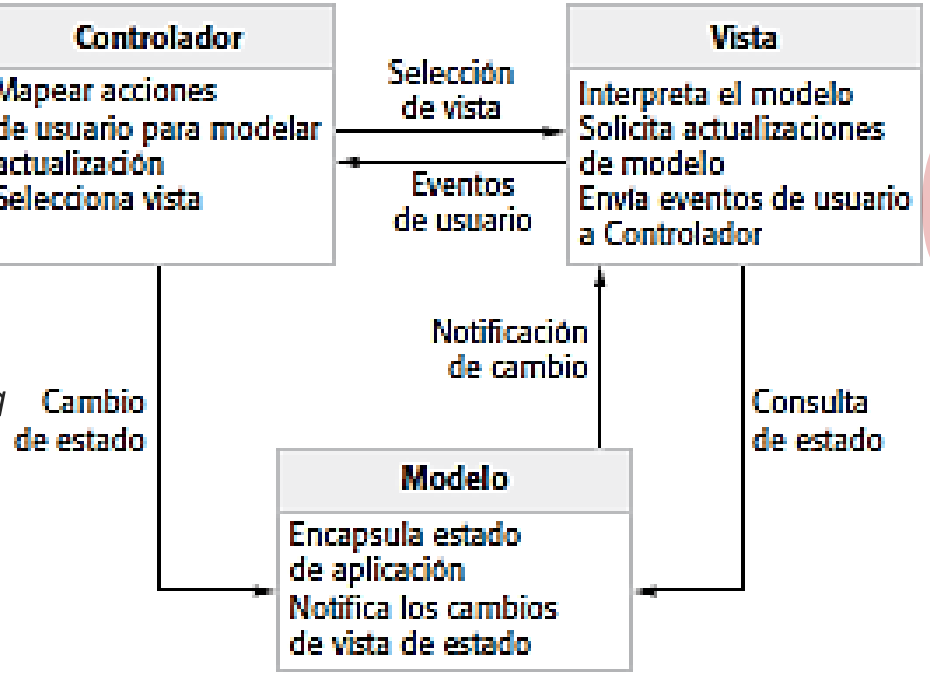
Diseño Arquitectónico

1. Organización del sistema
2. Descomposición modular
3. Modelos de control
4. Arquitectura de los Sistemas Distribuidos

Fuente: »Sommerville 9ª Edición Cap 6

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 (***Patrones*** arquitectónicos)
    - *Repositorio, cliente-servidor, capas o combinaciones entre ellos, entre otros*



Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

Patrón 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 subsistemas

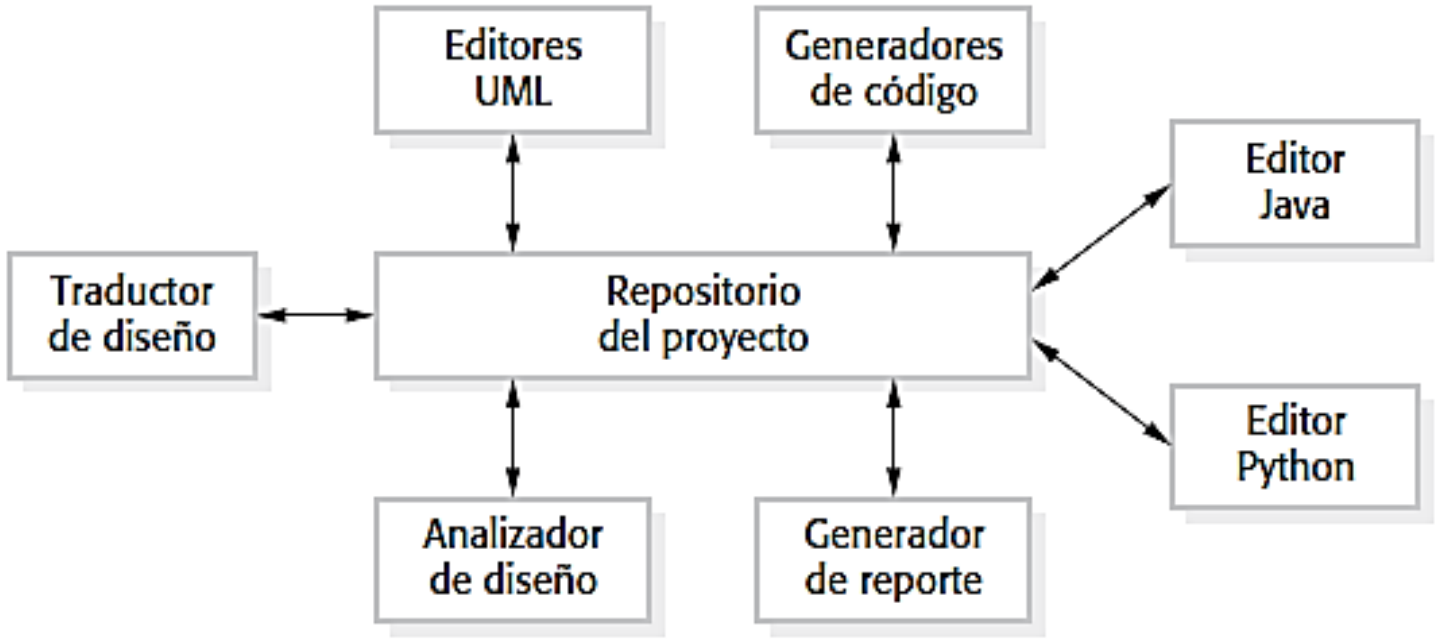
Ejemplo

Sistemas de gestión, Sistemas CAD, Herramientas Case, etc.

Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

Patrón de repositorio



Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

Patrón 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 9ª Edición Cap 6

Organización del Sistema

Patrón 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 algunos casos imposible.

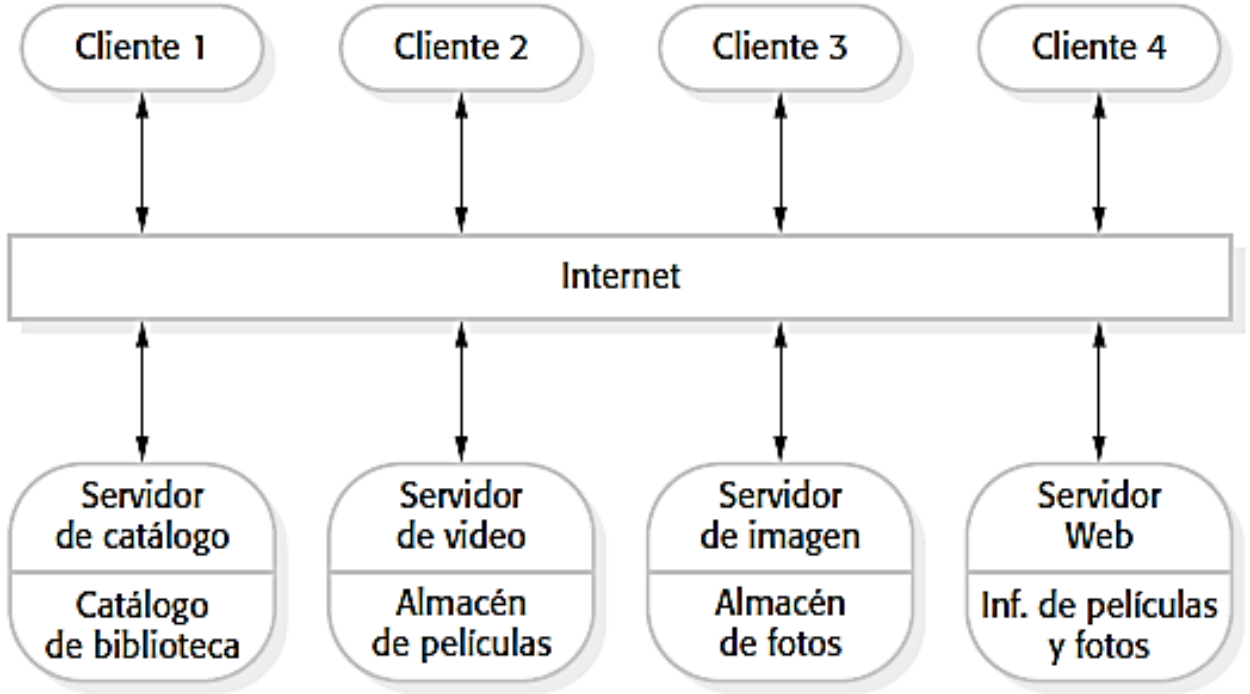
Diferentes subsistemas pueden tener distintos requerimientos de protección o políticas de seguridad y el modelo de repositorio impone las mismas para todos.

Es difícil distribuir el repositorio en varias máquinas, existen repositorios centralizados lógicamente pero pueden ocasionar problemas de redundancia e inconsistencias.

Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

Patrón cliente-servidor



Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

Patrón cliente-servidor

Es un modelo donde el sistema se organiza como un conjunto de servicios y servidores asociados, más unos clientes que utilizan los servicios

Componentes

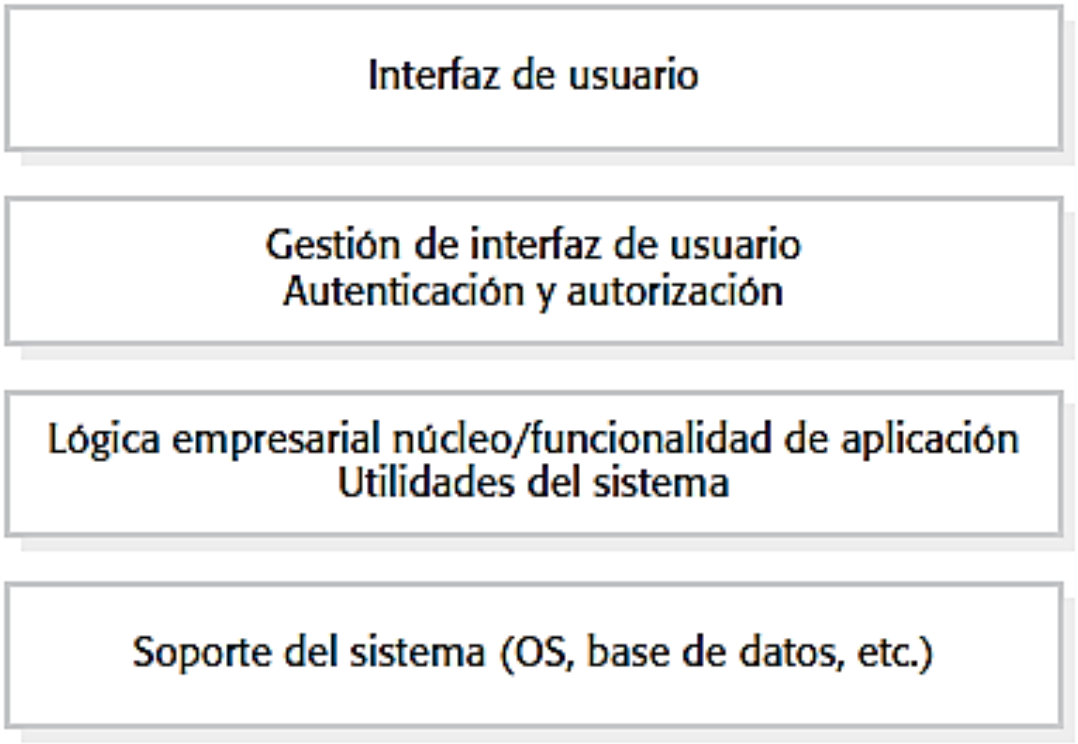
- Un conjunto de servidores que ofrecen servicios, otros sistemas
- Un conjunto de clientes que llaman a los servicios
- Una red que permite a los clientes acceder a los servicios
  - Caso particular cuando los servicios y el cliente corren en la misma máquina

Los clientes conocen el nombre del servidor y el servicio que brinda, pero el servidor no necesita conocer al cliente

Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

»*Patrón* de arquitectura en capas



Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

»***Patrón*** de arquitectura en 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 más internas son dependientes de la plataforma (se genera una capa interna para cada plataforma)

Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

»***Patrón*** de arquitectura en capas

Desventajas

Difícil de estructurar

Las capas internas proporcionas servicios que son requeridos por todos los niveles

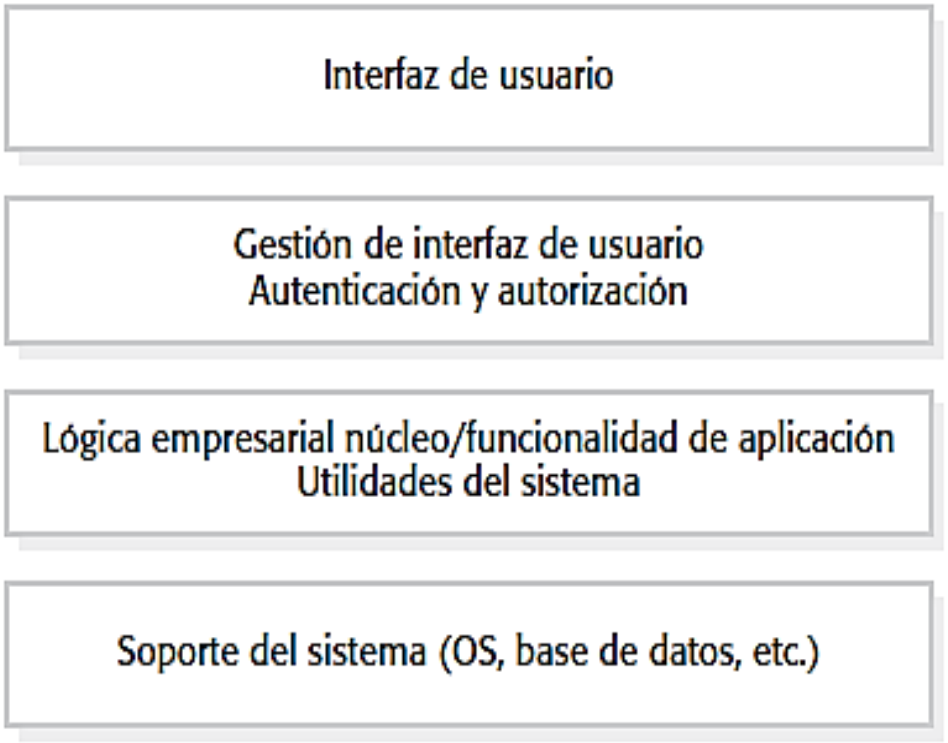
Los servicios requeridos por el usuario pueden estar brindados por las capas internas teniendo que atravesar varias capas adyacentes

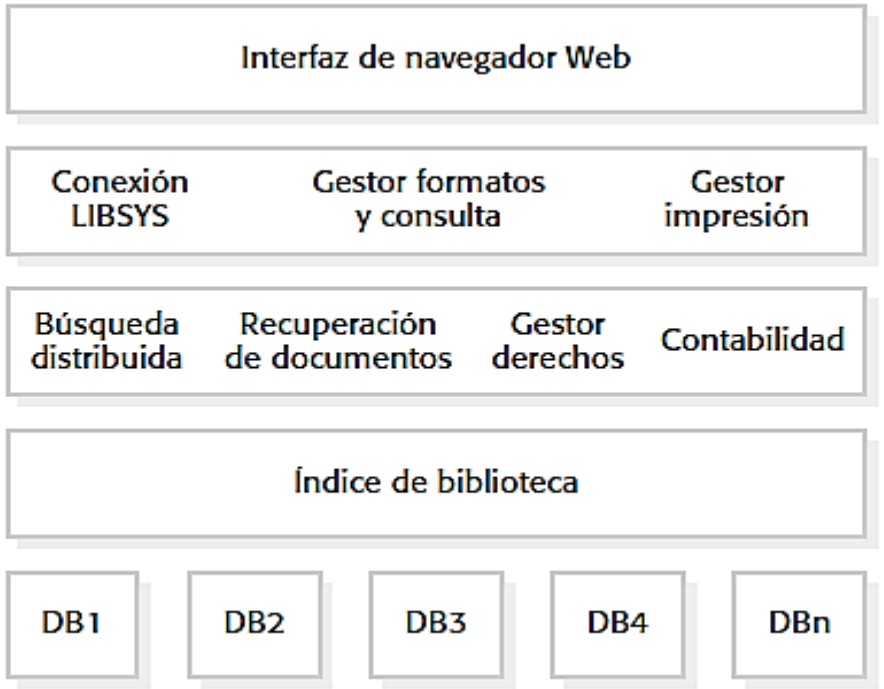
Si hay muchas capas, un servicio solicitado de la capa superior puede tener que ser interpretado varias veces en diferentes capas

Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

»Ejemplo de ***Patrón*** de arquitectura en capas:

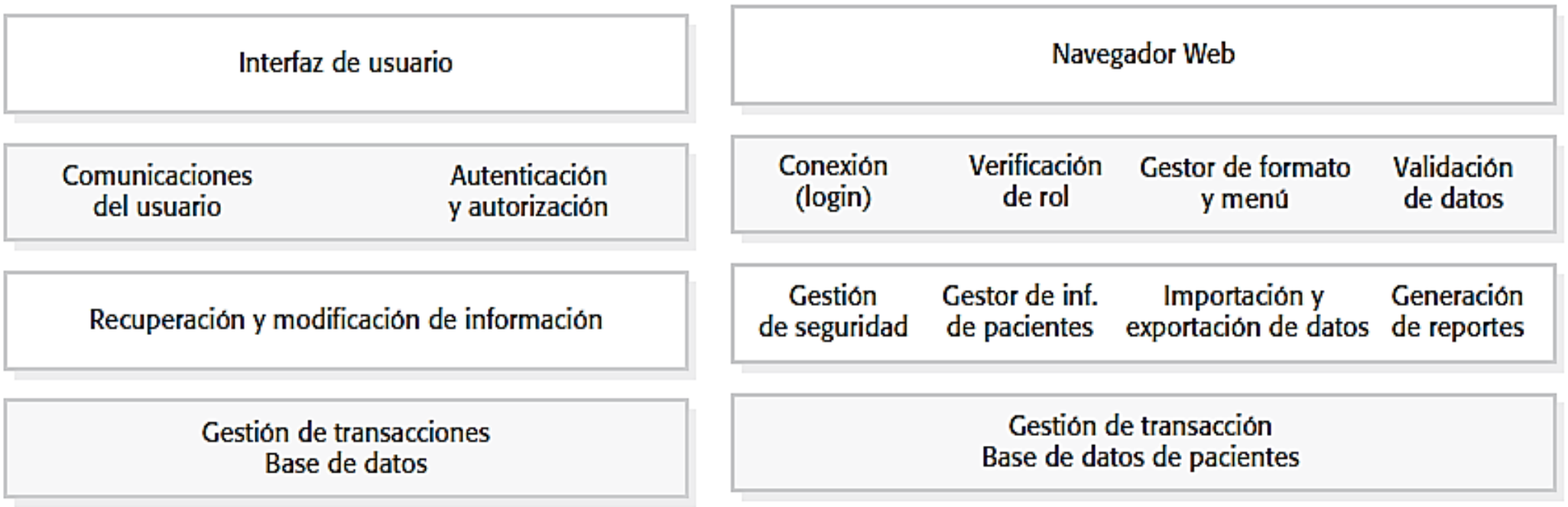




Fuente: »Sommerville 9ª Edición Cap 6

Organización del Sistema

»Ejemplo de ***Patrón*** de arquitectura en capas:



Fuente: »Sommerville 9ª Edición Cap 6

Diseño Arquitectónico

1. Organización del sistema
2. Descomposición modular
3. Modelos de control
4. Arquitectura de los Sistemas Distribuidos

Fuente: »Sommerville 9ª Edición Cap 6

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 más 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 9ª Edición Cap 6

Descomposición Modular

»Definiciones

Subsistema

Es un sistema en sí 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 otro subsistemas.

Módulo

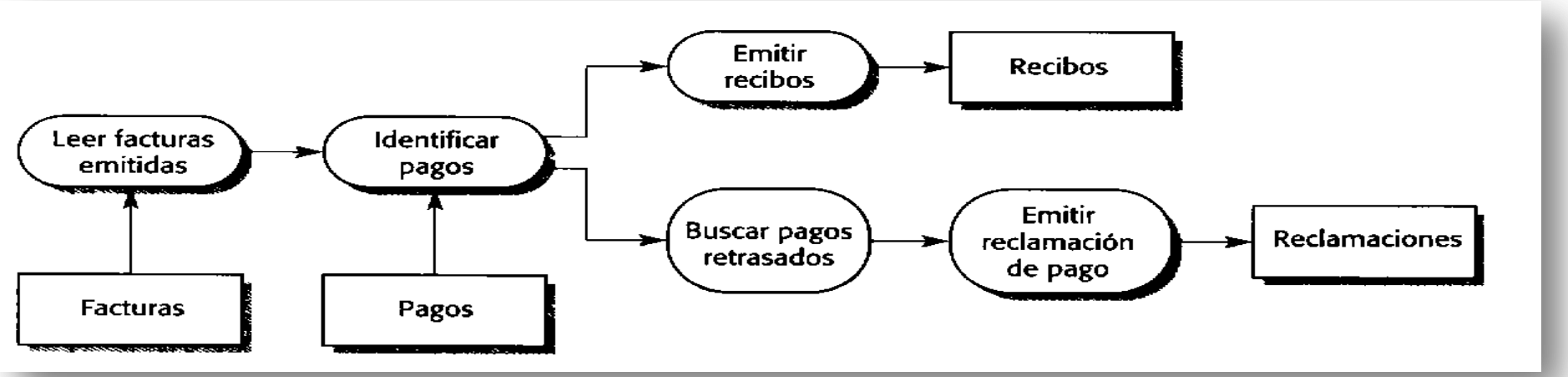
Es un componente de un subsistema que proporciona uno o más servicios a otros módulos. A su vez utiliza servicios proporcionados por otros módulos. Por lo general no se los considera un sistema independiente.

Fuente: »Sommerville 9ª Edición Cap 6

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.

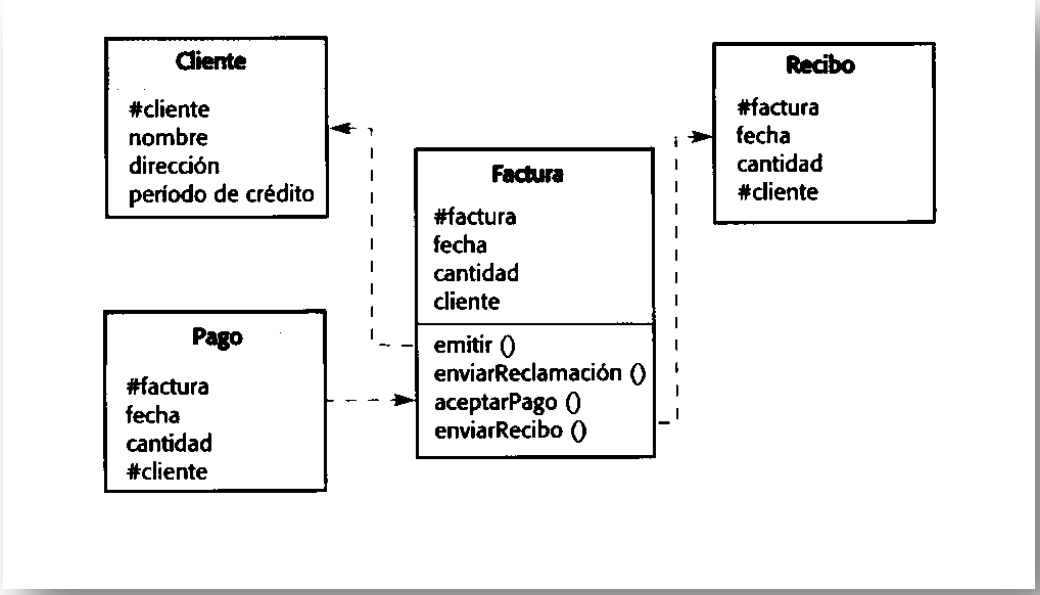


Fuente: »Sommerville 9ª Edición Cap 6

Descomposición Modular

»Descomposición orientada a objetos

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



Fuente: »Sommerville 9ª Edición Cap 6

Diseño Arquitectónico

1. Organización del sistema
2. Descomposición modular
3. Modelos de control
4. Arquitectura de los Sistemas Distribuidos

Fuente: »Sommerville 9ª Edición Cap 6

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 Basado en Eventos

Cada subsistema responde a eventos externos al subsistema.

Fuente: »Sommerville 9ª Edición Cap 6

Modelos de Control

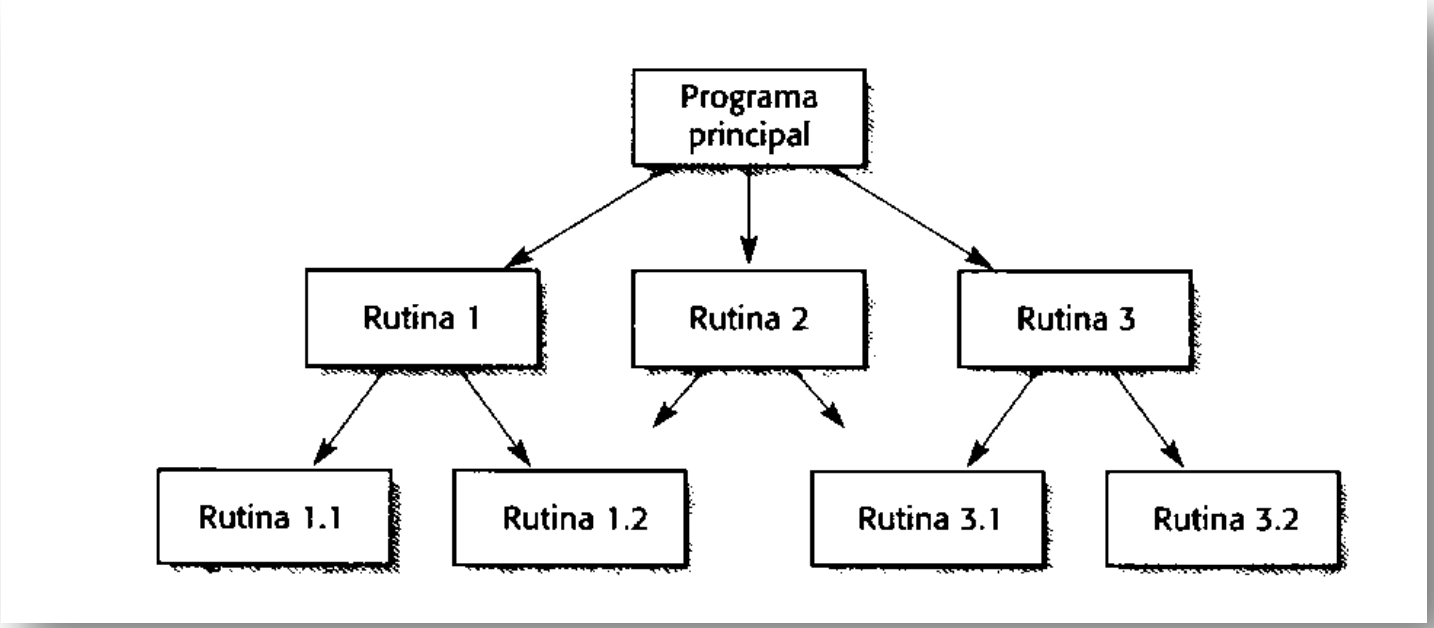
- »***Control*** Centralizado
  - Un subsistema se diseña como controlador y tiene la responsabilidad de gestionar la ejecución de otros subsistemas, la ejecución pude ser secuencial o en paralelo
    - *Modelo de llamada y retorno*
      - Modelo de subrutinas descendentes
      - Aplicable a modelos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

Fuente: »Sommerville 9ª Edición Cap 6

Modelos de Control

»*Control* Centralizado

Modelo de llamada y retorno

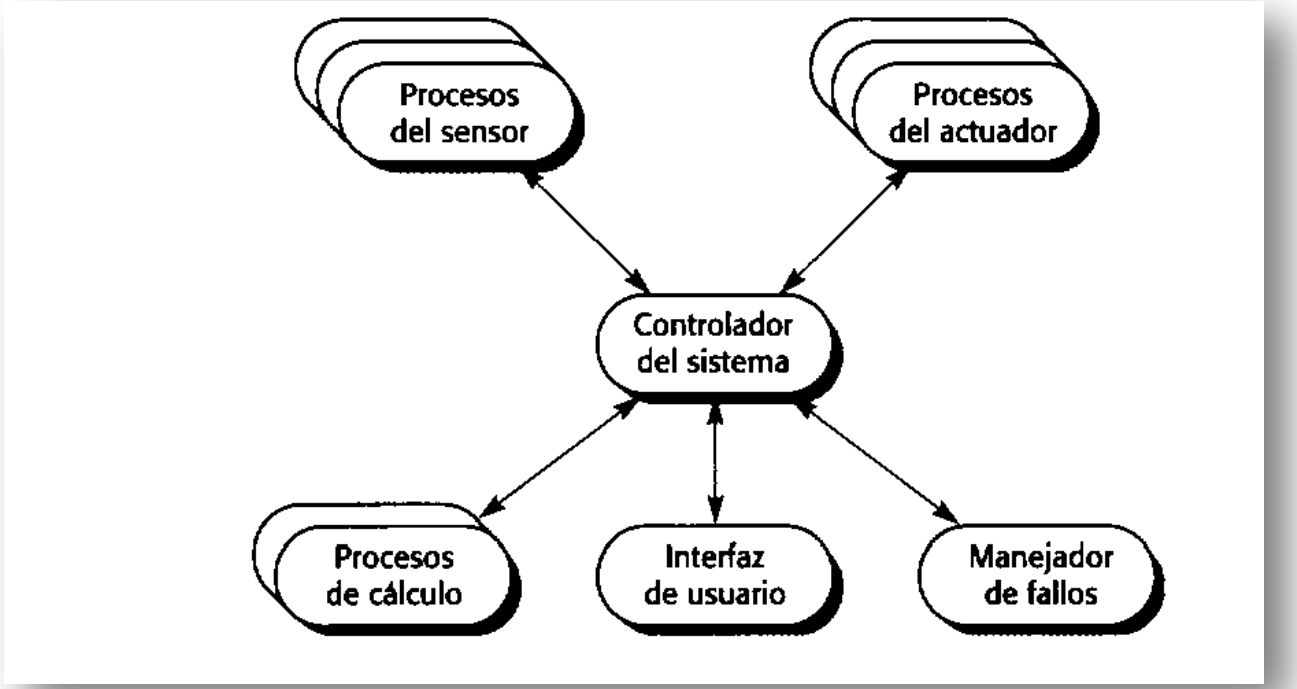


Fuente: »Sommerville 9ª Edición Cap 6

Modelos de Control

»*Control* Centralizado

Modelo de gestor



Fuente: »Sommerville 9ª Edición Cap 6

Modelos de Control

»Sistemas Dirigidos Por Eventos

Se rigen por eventos generados externamente al proceso

Eventos

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

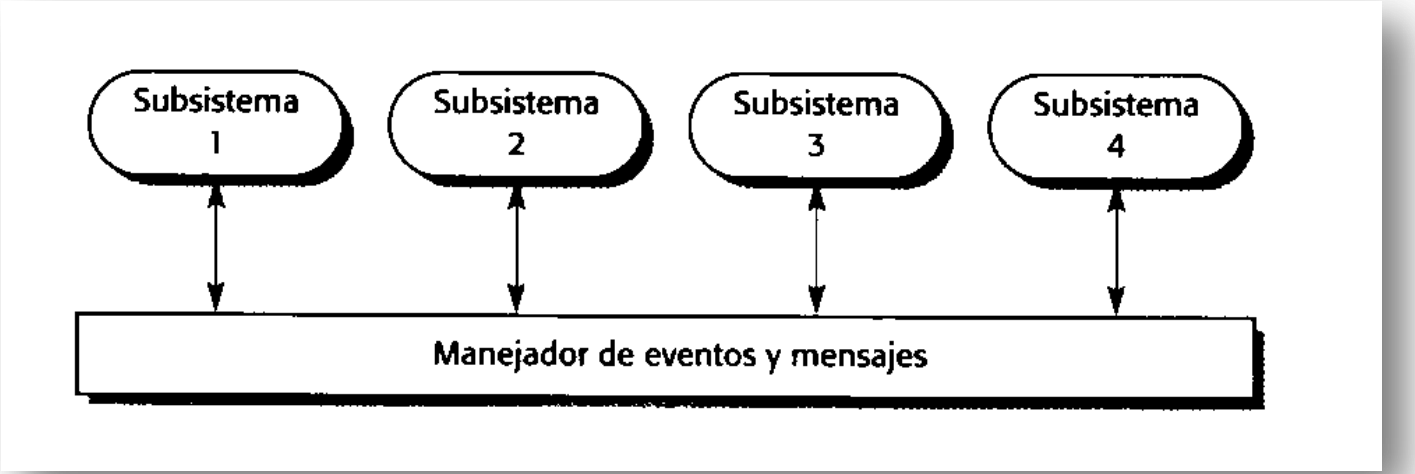
Fuente: »Sommerville 9ª Edición Cap 6

Modelos de Control

»Sistema ***Dirigido*** Por Eventos

Modelos de transmisión (Broadcast)

Un evento se trasmite a todos los subsistemas, cualquier subsistema programado para manejar ese evento lo atenderá



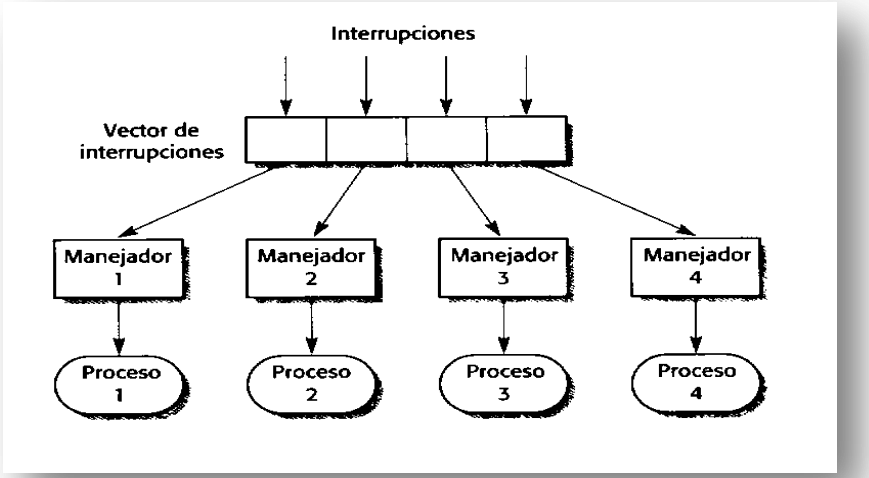
Fuente: »Sommerville 9ª Edición Cap 6

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 9ª Edición Cap 6

Diseño Arquitectónico

1. Organización del sistema
2. Descomposición modular
3. Modelos de control
4. Arquitectura de los Sistemas Distribuidos

Fuente: »Sommerville 9ª Edición Cap 6

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
  - Componentes distribuidos
- Características de los sistemas distribuidos
  - Compartir recursos
  - Apertura
  - Concurrencia
  - Escalabilidad
  - Tolerancia a fallos

Fuente: Sommerville 9ª Edición Cap 18

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 pueden operar al mismo tiempo sobre diferente 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 9ª Edición Cap 18

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 trafico en la red que puede ser intervenido*
  - Manejabilidad
    - *Las computadoras del sistema pueden ser de diferentes tipos y diferentes S.O. lo que genera mas 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 varié entre una petición y otra*

Fuente: Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

- »Arquitectura
  - Multiprocesador
  - Cliente-Servidor
    - *Dos niveles*
    - *Multinivel*
  - Objetos Distribuidos
  - Computación distribuida inter-organizacional
    - *Peer to peer*
    - *Orientada a servicios*

Fuente: »Sommerville 9ª Edición Cap 18

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 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Arquitectura Multiprocesador

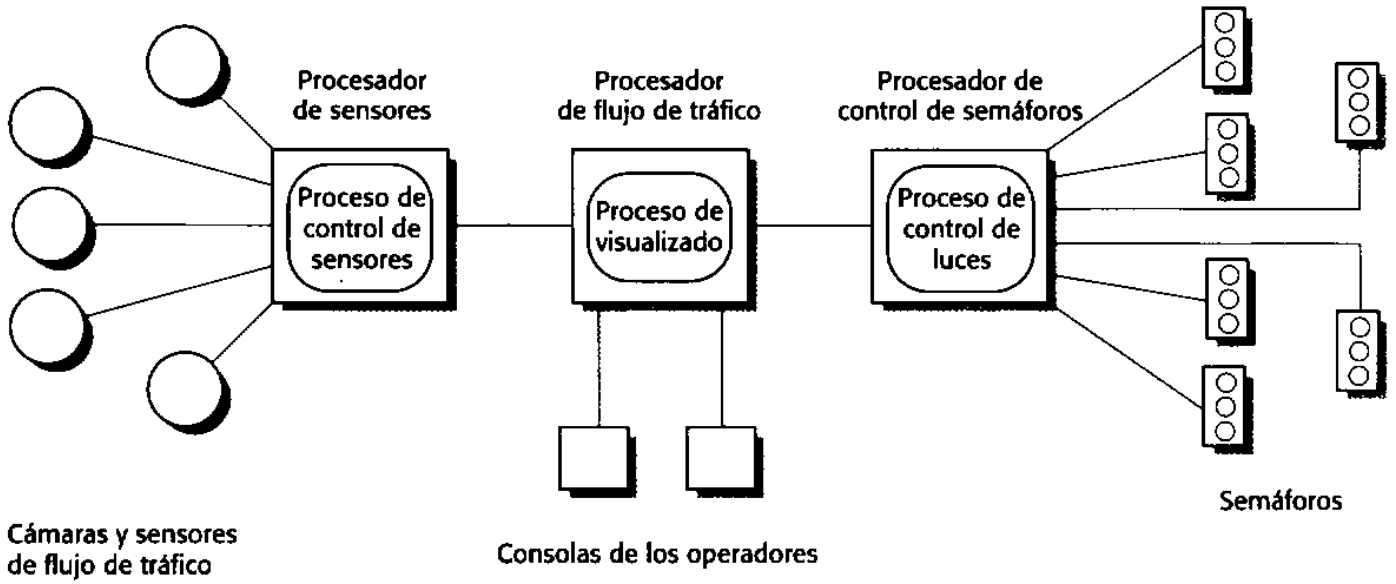


Figura 12.1 Un sistema multiprocesador de control de tráfico.

Fuente: »Sommerville 9ª Edición Cap 18

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 sí

Fuente: Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Arquitectura Cliente-Servidor

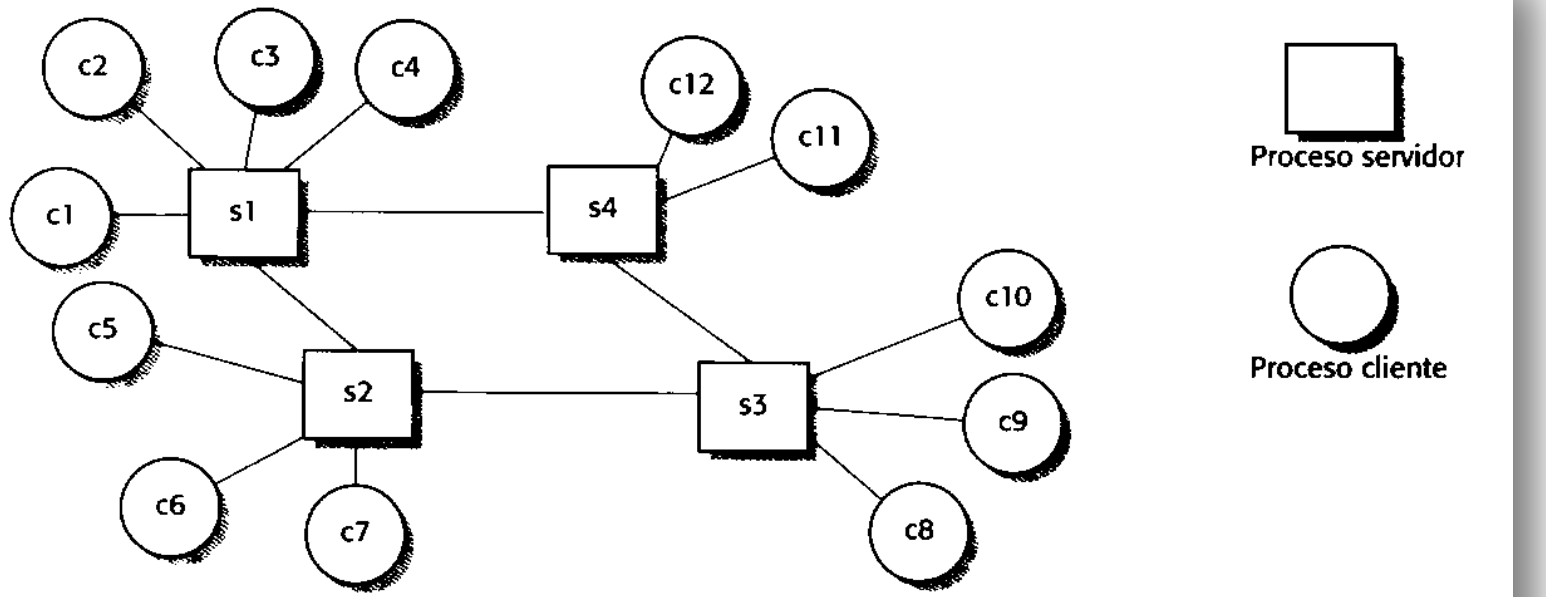
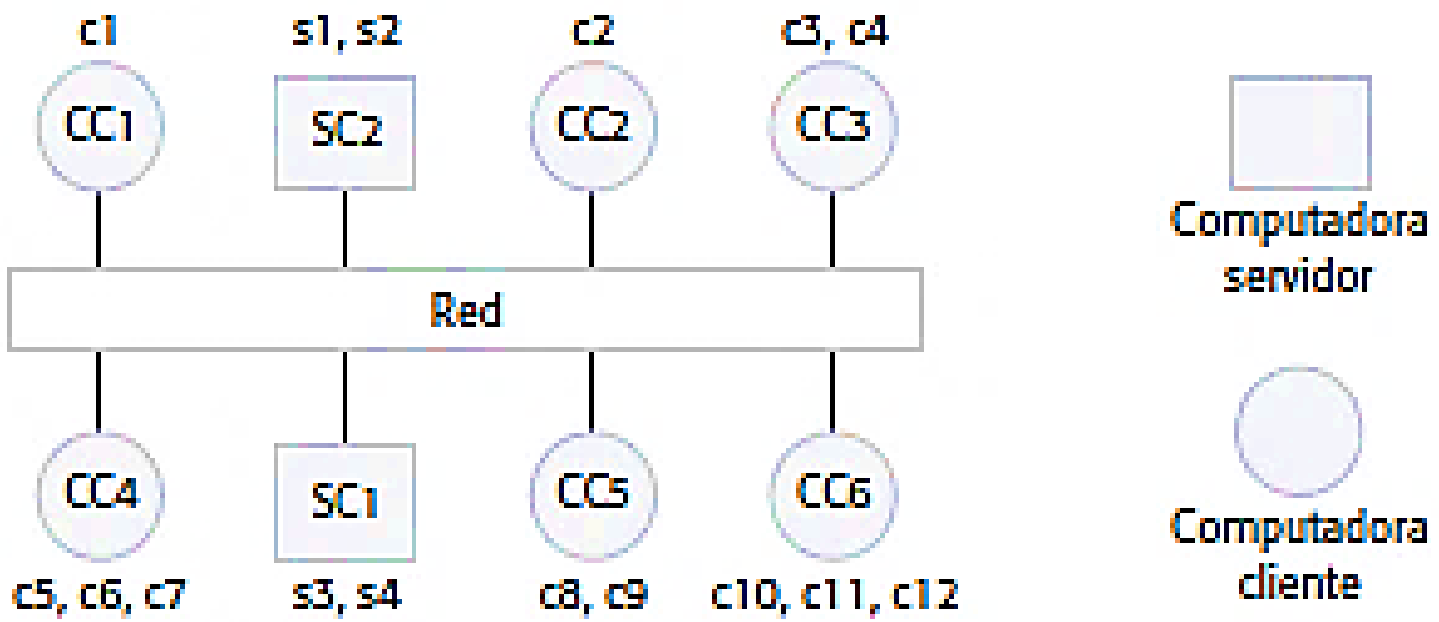


Figura 12.2 Un sistema cliente-servidor.

Fuente: »Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Arquitectura Cliente-Servidor



Fuente: »Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

- Arquitectura Cliente-Servidor
  - Clasifican en niveles
    - *Dos Niveles*
      - 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
    - *Multinivel*
      - La presentación, el procesamiento y la gestión de los datos son procesos lógicamente separados y se pueden ejecutar en procesadores diferentes

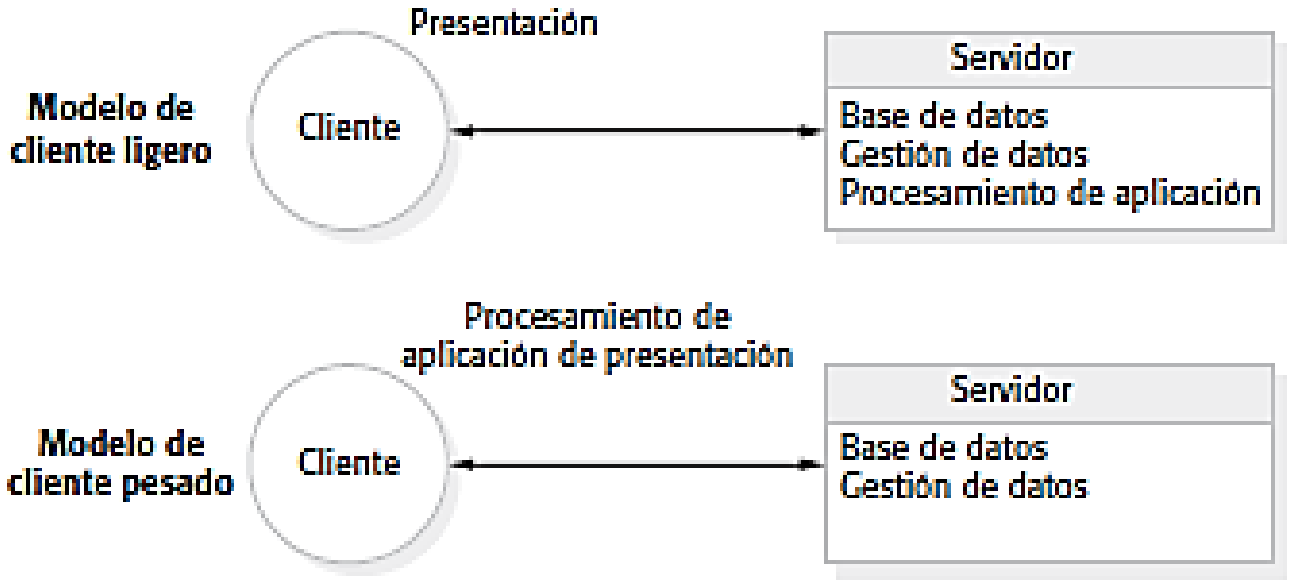
Fuente: Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Arquitectura Cliente-Servidor

Clasifican en niveles

Dos Niveles



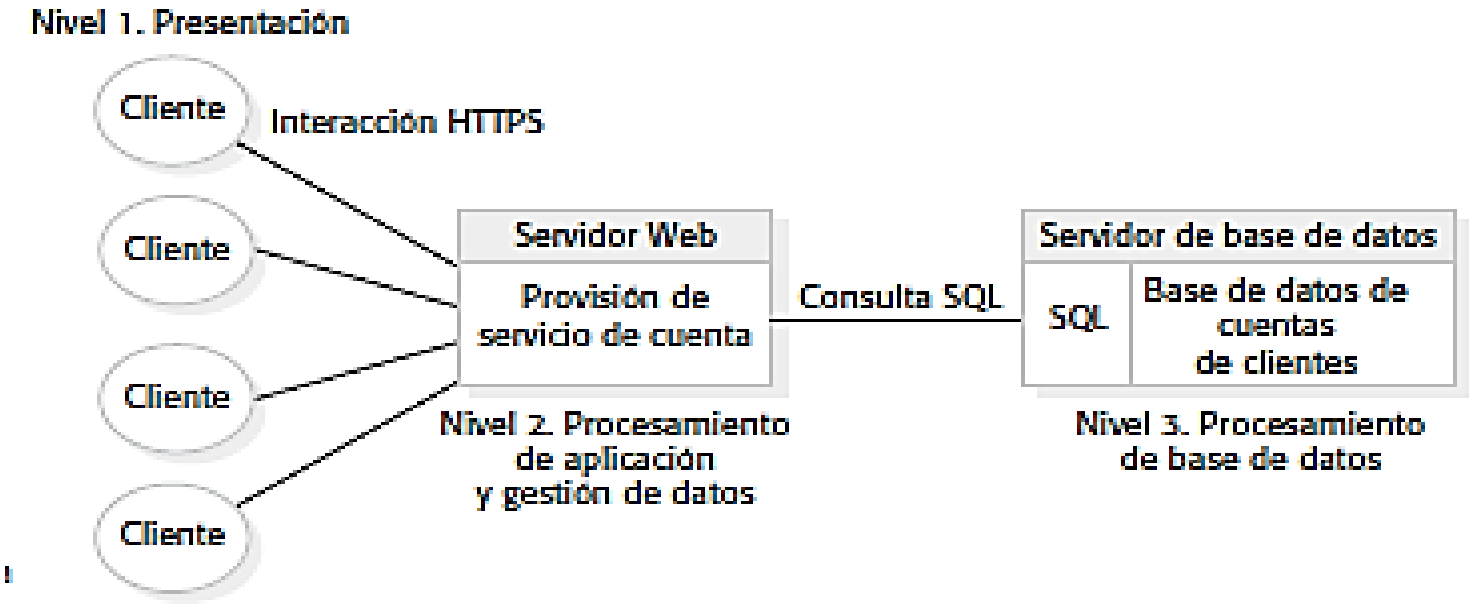
Fuente: »Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Arquitectura Cliente-Servidor

Clasifican en niveles

Multinivel



Fuente: »Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Arquitectura de Componentes Distribuidos

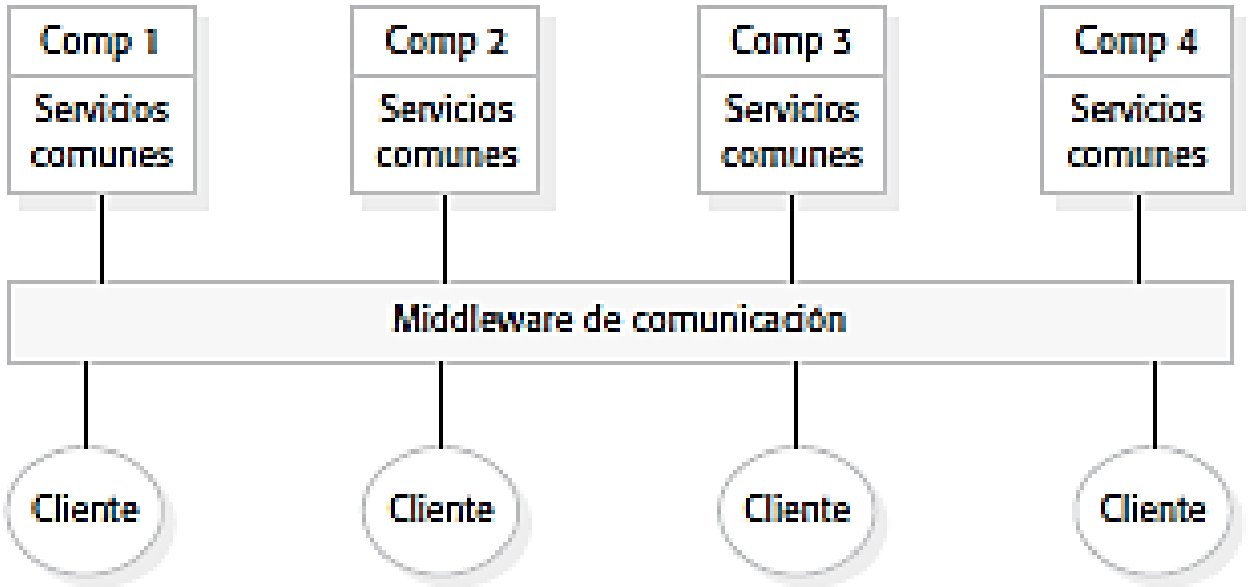
Diseña al sistema como un conjunto de componentes u objetos que brindan una interfaz de un conjunto de servicios que ellos suministran. Otros componentes u objetos solicitan estos servicios. No hay distinción tajante entre clientes y servidores.

Los componentes pueden distribuirse en varias máquinas a través de la red utilizando un middleware como intermediario de peticiones

Fuente: »Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Arquitectura de Objetos Distribuidos



Fuente: »Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

- Computación Distribuida inter-organizacional
  - 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 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

- Computación Distribuida inter-organizacional
  - 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
  - Pueden 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: Torrents, Skype, ICQ, SETI@Home

Fuente: »Sommerville 9ª Edición Cap 18

Arquitectura de los Sistemas Distribuidos

»Computación Distribuida inter-organizacional

Arquitecturas Peer-to-Peer (P2P)

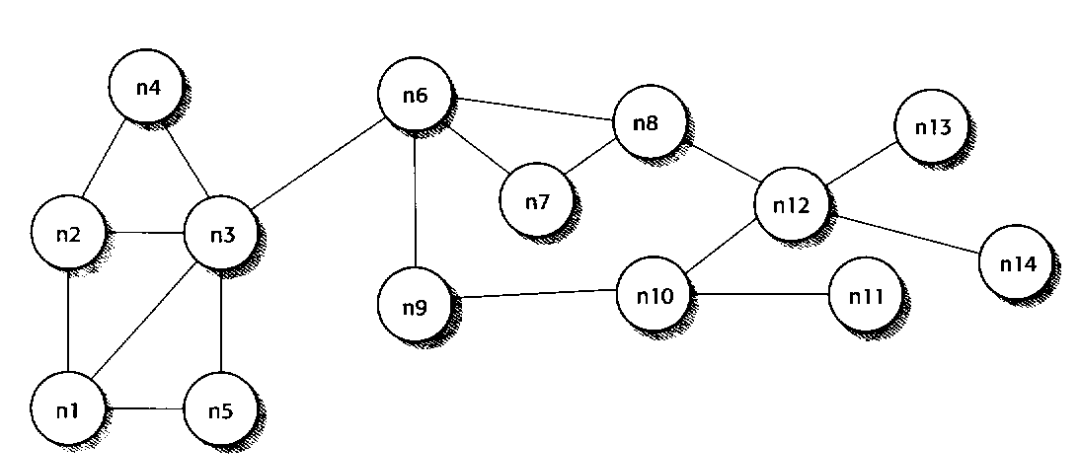
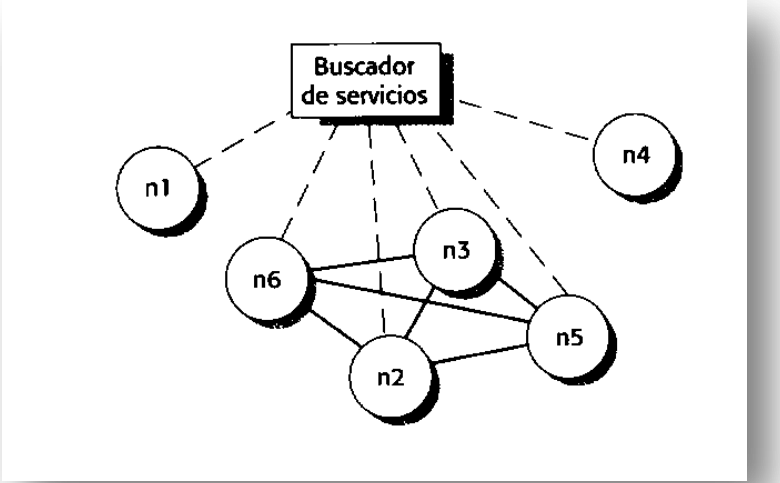


Figura 12.15 Arquitectura p2p descentralizada.



Fuente: »Sommerville 9ª Edición Cap 18

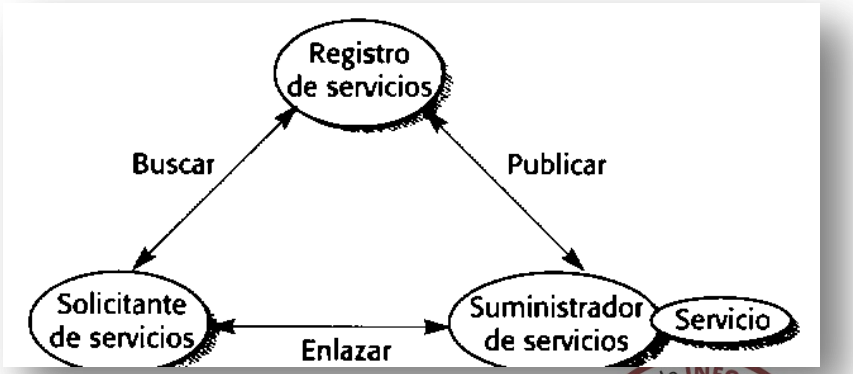
Arquitectura de los Sistemas Distribuidos

- 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 9ª Edición Cap 18

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 “registro de servicio” 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 9ª Edición Cap 18

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 Lenguage)*
      - Define como puede representarse las interfaces web
    - *UDDI (Universal Desscription Discovery and Integration)*
      - Estándar de búsqueda que define como puede organizarse la información de descripción de servicios

Fuente: »Sommerville 9ª Edición Cap 18

Ingeniería de software II

Codificación

Codificación

- Una vez establecido el diseño, se deben escribir los programas que implementen dicho diseño.
- Esto puede resultar una tarea compleja por distintos motivos:
  - Los diseñadores pueden no haber tenido en cuenta las particularidades de la plataforma y el ambiente de programación.
  - Las estructuras e interrelaciones que son fáciles de describir mediante diagramas, no siempre resultan sencillas de escribir en código.
  - Es indispensable escribir el código de forma que resulte comprensible para otras personas.
  - Se deben sacar beneficios de las características de diseño creando código que sea reutilizable.

Fuente: »Pfleeger Cap. 7

Codificación: Pautas Generales

»Resultan útiles para conservar la calidad del diseño en la codificación:

Localización de entrada y salida: es deseable localizarlas en componentes separados del resto del código ya que generalmente son más difíciles de probar.

Inclusión de pseudocódigo: Es útil avanzar el diseño, realizando un pseudocódigo para adaptar el diseño al lenguaje elegido.

Revisión y reescritura, no a los remiendos: Es recomendable realizar un borrador, revisarlo y reescribirlo tantas veces como sea necesario.

Reutilización: Hay dos clases de reutilización:

Productiva: se crean componentes destinados a ser reutilizados por otra aplicación

Consumidora: Se usan componentes originalmente desarrollados para otros proyectos.

Fuente: »Pfleeger Cap. 7

Codificación: Documentación

»Se considera como Documentación del programa al conjunto de descripciones escritas que explican al lector qué hace el programa y cómo lo hace.

»Se divide en:

Documentación interna: Es concisa, escrita en un nivel apropiado para un programador. Contiene información dirigida a quienes leerán el código fuente. Incluye información de algoritmos, estructuras de control, flujos de control.

Documentación externa: Se prepara para ser leída por quienes, tal vez, nunca verán el código real. Por ejemplo, los diseñadores, cuando evalúan modificaciones o mejoras.

Fuente: »Pfleeger Cap. 7