Ingeniería del Software I

2º

TEMA 5.1
Diseño arquitectónico

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Diseño arquitectónico

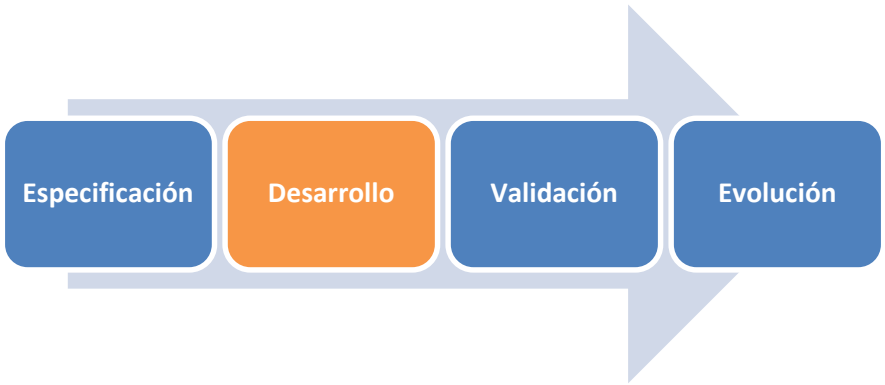
Universidad Francisco de Vitoria UFV Madrid

Objetivos

- Por qué es importante el diseño arquitectónico del software.
- Decisiones que deben tomarse sobre la arquitectura de software.
- Entender patrones arquitectónicos.
- Identificará los patrones arquitectónicos.

Índice

1. Identificará los patrones arquitectónicos
2. Vistas arquitectónicas
3. Patrones arquitectónicos
4. Arquitecturas de aplicación



Diseño arquitectónico

Universidad Francisco de Vitoria UFV Madrid

- El **diseño arquitectónico** es **la primera etapa** en el proceso de diseño del software. Es el enlace crucial entre el diseño y la ingeniería de requerimientos, ya que identifica los principales componentes estructurales en un sistema y la relación entre ellos.
- La **salida del proceso** de diseño arquitectónico consiste en un modelo arquitectónico que describe la forma en que se organiza el sistema como un conjunto de componentes en comunicación.
- El modelo arquitectónico presenta componentes y los vínculos entre ellos.

Diseño arquitectónico

Universidad Francisco de Vitoria UFV Madrid

- Los **componentes individuales** implementan los **requerimientos funcionales del sistema**.
- Los **requerimientos no funcionales** dependen de la **arquitectura del sistema**, es decir, la forma en que dichos componentes se organizan y comunican.
- **Tres ventajas de diseñar y documentar de manera explícita la arquitectura de software**:
  - **Comunicación con los participantes**: La arquitectura es una presentación de alto nivel del sistema, que puede usarse como un enfoque para la discusión de un amplio número de participantes.
  - **Análisis del sistema**: En una etapa temprana en el desarrollo del sistema, aclarar la arquitectura del sistema requiere cierto análisis. Las decisiones de diseño arquitectónico tienen un efecto profundo sobre si el sistema puede o no cubrir requerimientos críticos como rendimiento, fiabilidad y mantenibilidad.
  - **Reutilización a gran escala**: Un modelo de una arquitectura de sistema es una descripción corta y manejable de cómo se organiza un sistema y cómo interoperan sus componentes.

Diseño arquitectónico

Universidad Francisco de Vitoria UFV Madrid

- Las aparentes **contradicciones entre práctica y teoría arquitectónica** surgen porque hay **dos formas en que se utiliza un modelo arquitectónico** de un programa:
  - ***Como una forma de facilitar la discusión acerca del diseño del sistema***
    - Una visión arquitectónica de alto nivel de un sistema es útil para la comunicación con los participantes de un sistema y la planificación del proyecto, ya que **no se satura con detalles**
    - El modelo arquitectónico **identifica los componentes clave** que se desarrollarán, de modo que los administradores pueden asignar a individuos para planificar el desarrollo de dichos sistemas.
  - ***Como una forma de documentar una arquitectura que se haya diseñado***
    - La meta aquí es producir un **modelo de sistema completo** que muestre los diferentes componentes en un sistema, sus interfaces y conexiones. tal descripción arquitectónica detallada facilita la comprensión y la evolución del sistema
- Los diagramas de bloque son una forma adecuada para describir la arquitectura del sistema durante el proceso de diseño, pues son una buena manera de soportar las comunicaciones entre las personas involucradas en el proceso

5.1 Decisiones en el diseño arquitectónico

Universidad Francisco de Vitoria UFV Madrid

- **Proceso creativo.**
  - El diseño arquitectónico es un proceso creativo en el cual se diseña una organización del sistema que cubrirá los requerimientos funcionales y no funcionales de éste.
- Para **sistemas embebidos y sistemas diseñados** para computadoras personales, por lo general, hay **un solo procesador** y **no tendrá que diseñar una arquitectura distribuida** para el sistema.
- Un **patrón arquitectónico** es una descripción de una organización del sistema (Garlan y Shaw, 1993), tal como una organización cliente-servidor o una arquitectura por capas.
- Existe una **estrecha relación entre los requerimientos no funcionales y la arquitectura de software**.

5.1 Decisiones en el diseño arquitectónico

Universidad Francisco de Vitoria UFV Madrid

- **Debido a esa estrecha relación** entre los requerimientos no funcionales y la arquitectura de software, **estilo y estructura arquitectónicos particulares** que se elijan para un sistema dependerán de los **requerimientos de sistema no funcionales**:
  - ***Rendimiento*** Si el rendimiento es un requerimiento crítico, la **arquitectura** debe diseñarse para localizar operaciones críticas dentro de un **pequeño número de componentes**, con todos estos componentes desplegados en la misma computadora en vez de distribuirlos por la red. Significaría utilizar componentes grandes, en vez de pequeños, lo que reduciría número de comunicaciones entre componentes.
  - ***Seguridad*** Si **la seguridad es un requerimiento crítico**, será necesario usar una estructura en capas para la arquitectura, con los activos más críticos protegidos en las capas más internas, y con un alto nivel de validación de seguridad aplicado a dichas capas.

5.1 Decisiones en el diseño arquitectónico

Universidad Francisco de Vitoria UFV Madrid

- **Protección**. Componentes pequeños.
  - Si la protección es un requerimiento crítico, la arquitectura debe diseñarse de modo que las operaciones relacionadas con la protección se ubiquen en algún componente individual o en un pequeño número de componentes.
  - Esto reduce los costos y problemas de validación de la protección, y hace posible ofrecer sistemas de protección relacionados que, en caso de falla, desactiven con seguridad el sistema
- **Disponibilidad**. Redundancia.
  - Si la disponibilidad es un requerimiento crítico, la arquitectura tiene que diseñarse para incluir componentes redundantes de manera que sea posible sustituir y actualizar componentes sin detener el sistema.
- **Mantenibilidad**. Componentes pequeños.
  - Si la mantenibilidad es un requerimiento crítico, la arquitectura del sistema debe diseñarse usando **componentes autocontenidos de grano fino** que puedan cambiarse con facilidad. Los productores de datos tienen que separarse de los consumidores y hay que evitar compartir las estructuras de datos.

5.1 Decisiones en el diseño arquitectónico

Universidad Francisco de Vitoria UFV Madrid

- Surge un **conflicto** potencial **entre algunas de estas arquitecturas**. Por ejemplo, usar componentes grandes mejora el rendimiento, y utilizar componentes pequeños aumenta la mantenibilidad.
- Si tanto el rendimiento como la mantenibilidad son requerimientos importantes del sistema, **entonces debe encontrarse algún compromiso**.
- Esto en ocasiones se logra **usando diferentes patrones o estilos arquitectónicos** para distintas partes del sistema.

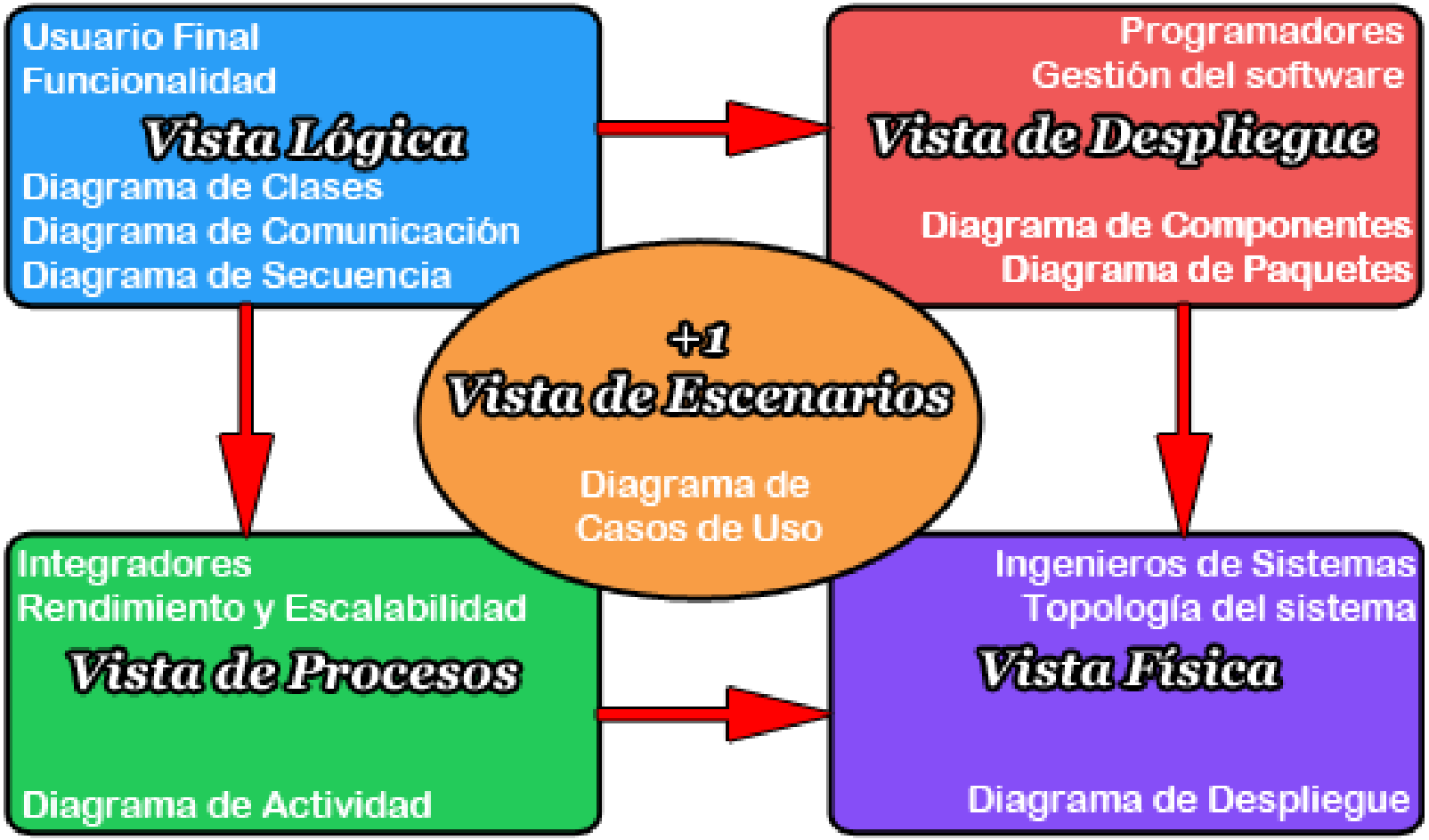
5.2 Vistas arquitectónicas

Universidad Francisco de Vitoria UFV Madrid

- Krutchen (1995), en su bien conocido **modelo de vista 4+1 de la arquitectura de software**, sugiere que deben existir **cuatro vistas arquitectónicas fundamentales**, que se relacionan usando casos de uso o escenarios. Las vistas son:
  - **Una vista lógica**, que indique las abstracciones clave en el sistema como objetos o clases de objeto. En este tipo de vista se tienen que relacionar los requerimientos del sistema con entidades. **Modelo de clases y diagramas de Entidad Relación**.
  - **Una vista de proceso**, que muestre cómo, en el tiempo de operación, el sistema está compuesto de procesos en interacción. Esta vista es útil para hacer juicios acerca de las características no funcionales del sistema, como el rendimiento y la disponibilidad. **DFDs (Data Flow Diagram)**
  - **Una vista de desarrollo (despliegue)**, que muestre cómo el software está descompuesto para su desarrollo, esto es, indica la descomposición del software en elementos que se implementen mediante un solo desarrollador o equipo de desarrollo. Esta vista es útil para administradores y programadores de software. **Descomposición modular**
  - **Una vista física**, que exponga el hardware del sistema y cómo los componentes de software se distribuyen a través de los procesadores en el sistema. Esta vista es útil para los ingenieros de sistemas que planean una implementación de sistema. **Diagramas de despliegue**

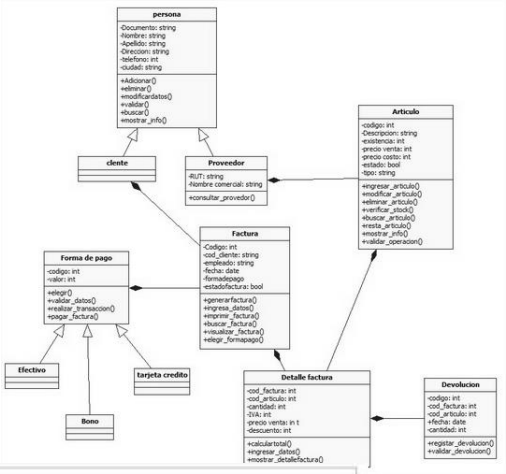
5.2 Vistas arquitectónicas

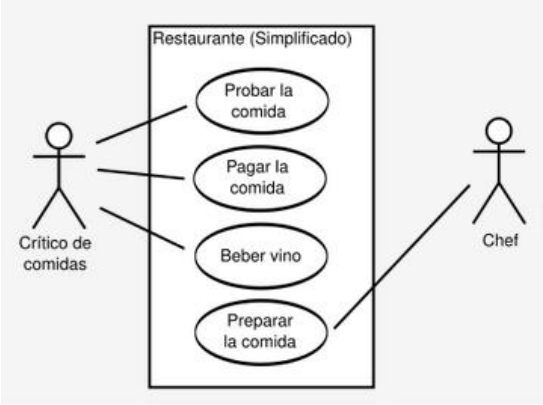
Universidad Francisco de Vitoria UFV Madrid

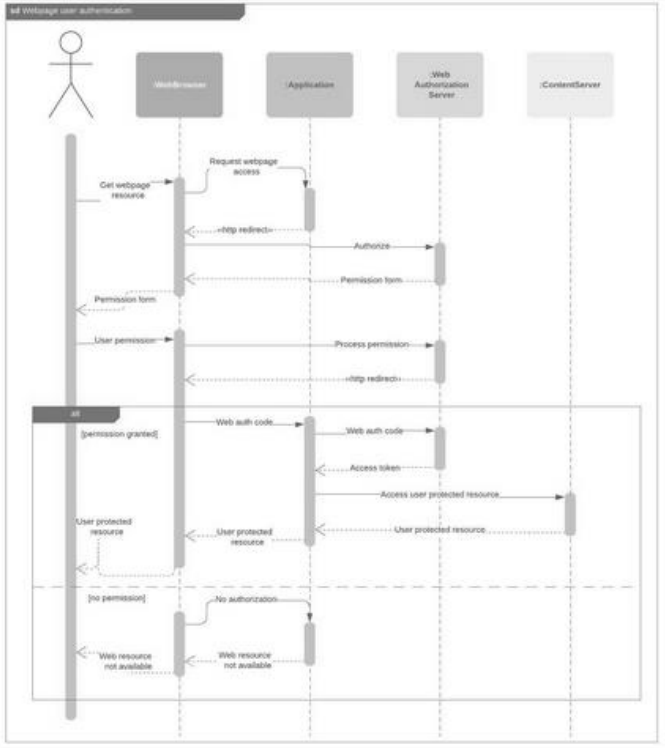


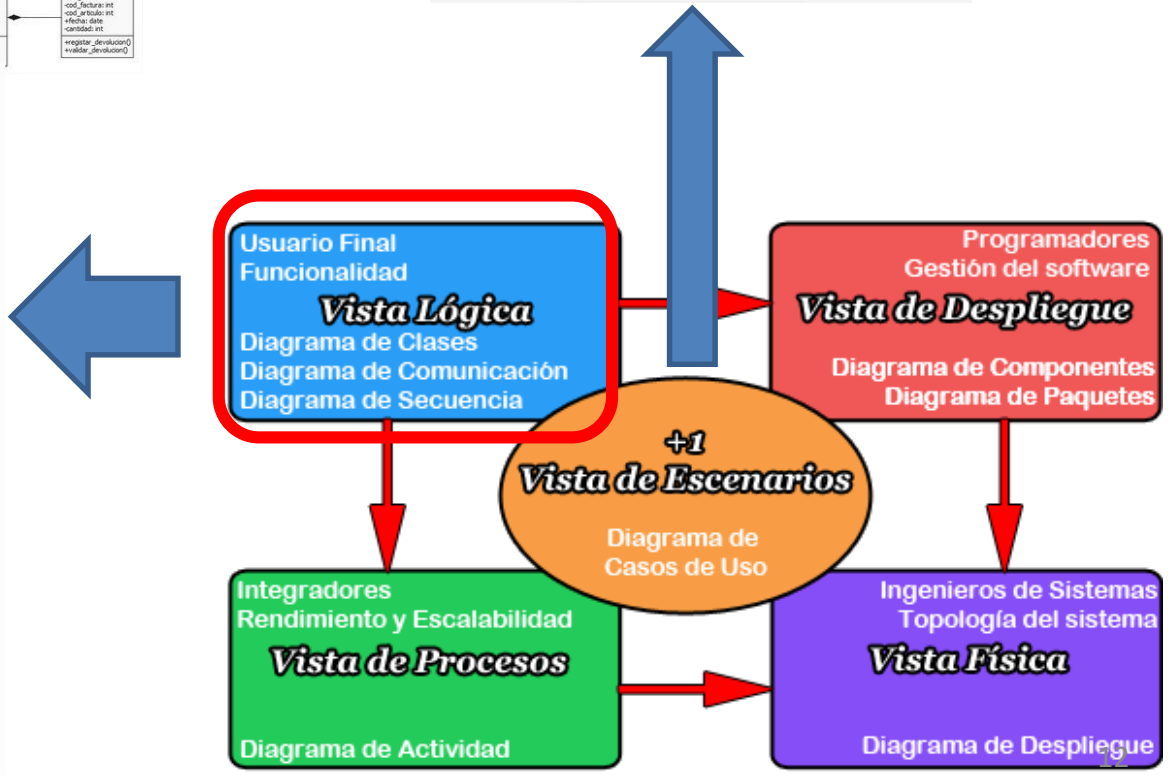
5.2 Vistas arquitectónicas

Universidad Francisco de Vitoria UFV Madrid

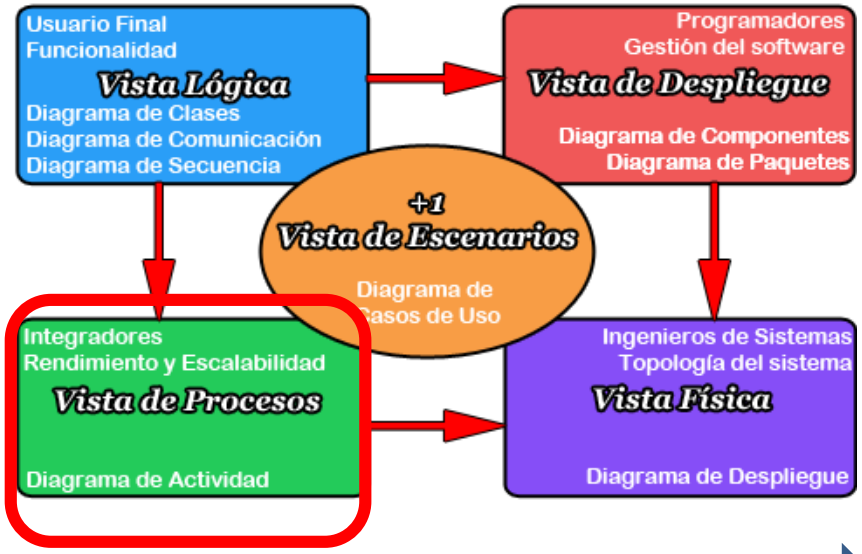


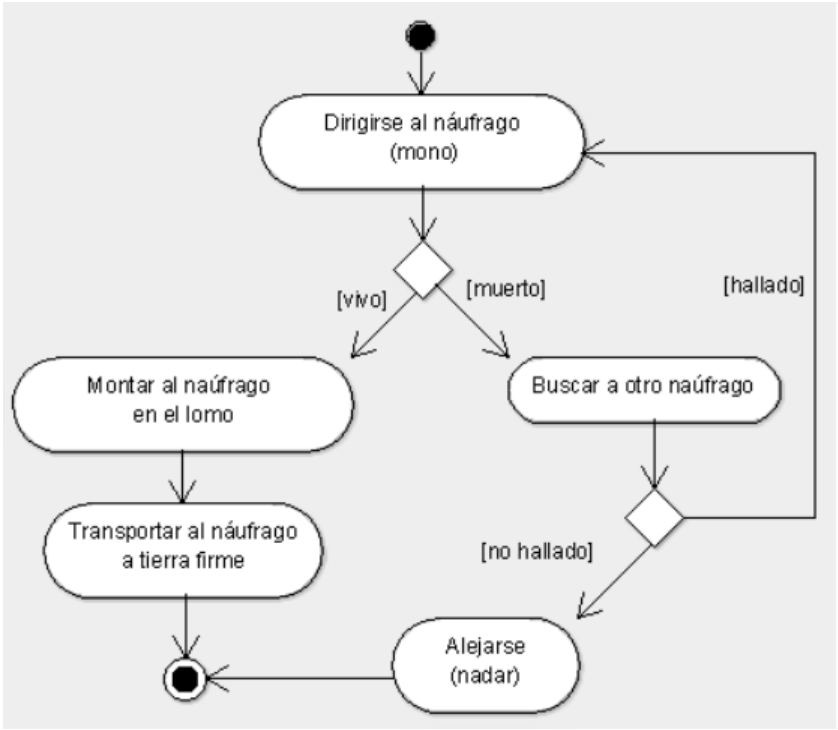




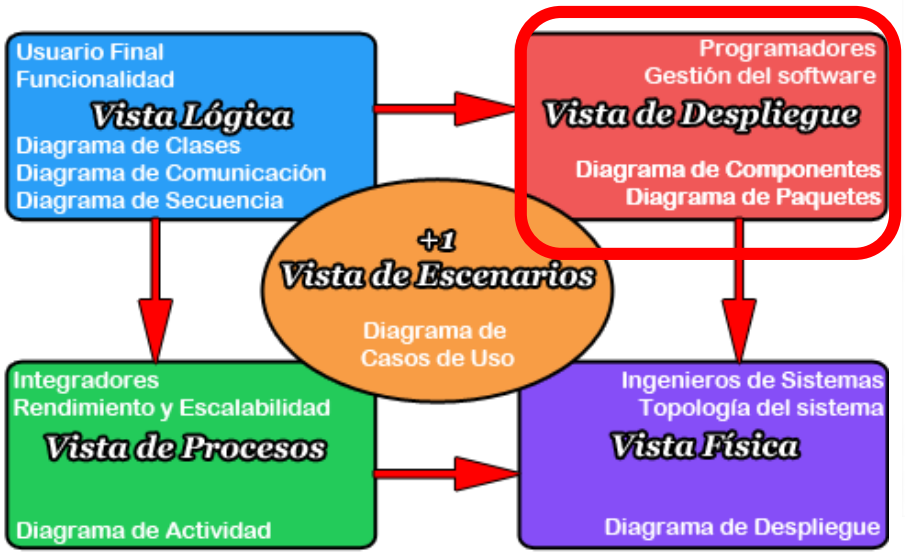


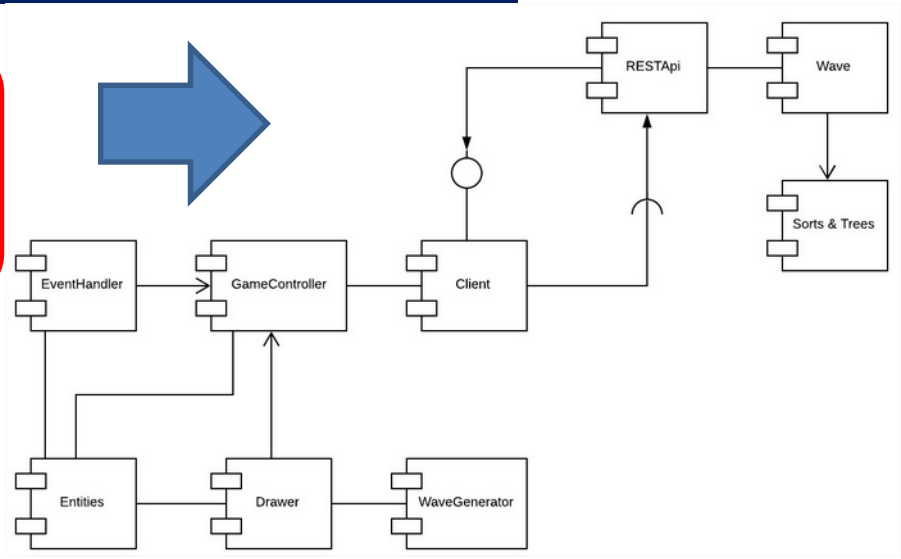
5.2 Vistas arquitectónicas

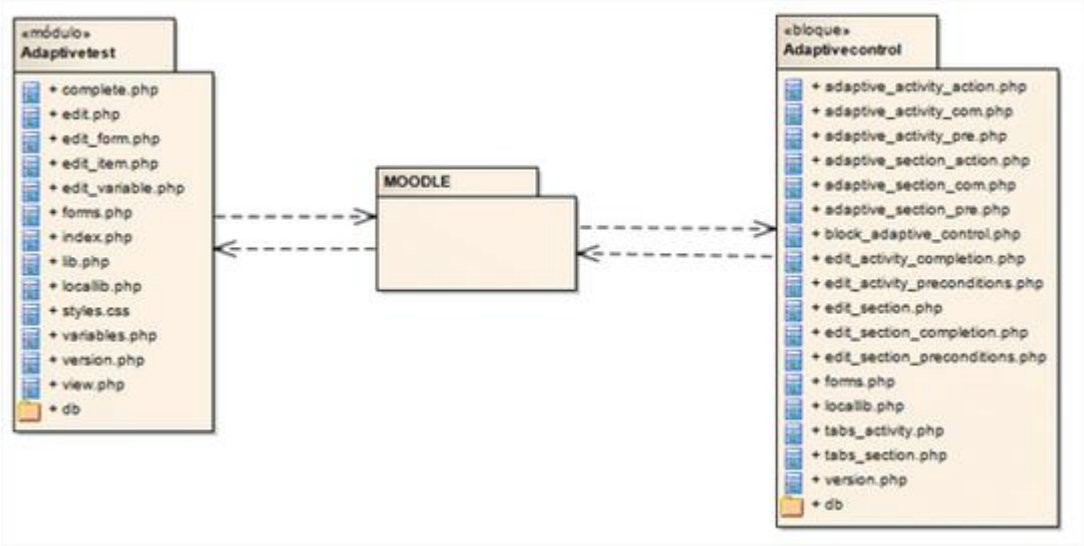




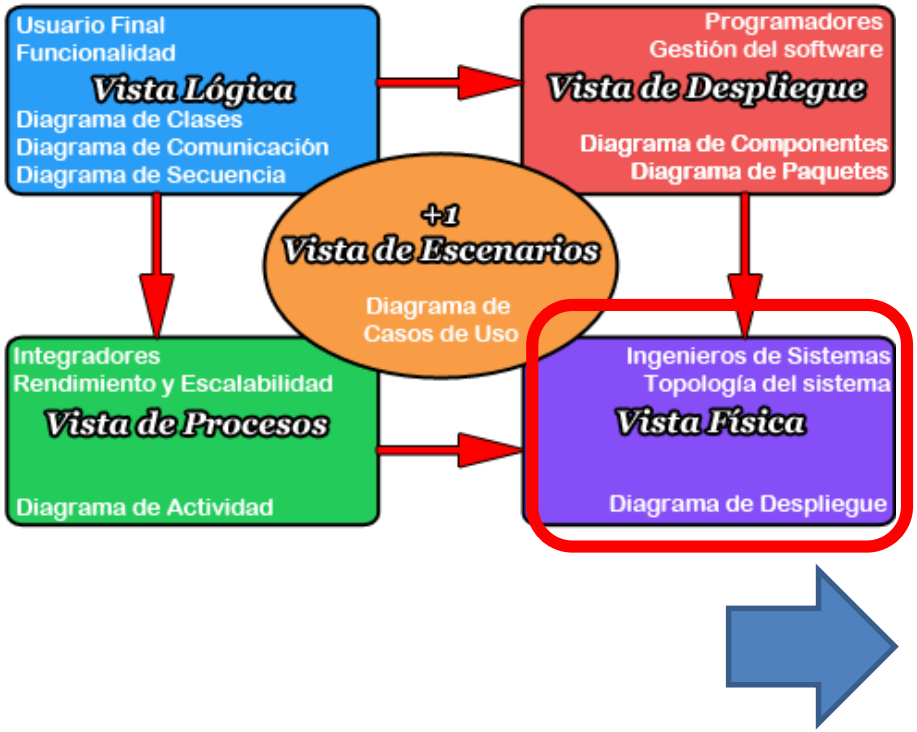
5.2 Vistas arquitectónicas

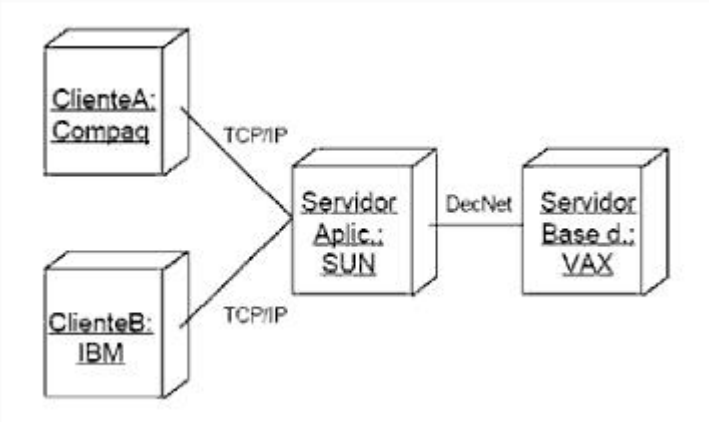






5.2 Vistas arquitectónicas





5.2 Vistas arquitectónicas

Universidad Francisco de Vitoria UFV Madrid

- Hofmeister y sus colaboradores (2000) sugieren el uso de vistas similares, pero a éstas agregan la noción de vista conceptual.
- Esta última es una **vista abstracta del sistema** que puede ser la base para descomponer los requerimientos de alto nivel en especificaciones más detalladas, ayudar a los ingenieros a tomar decisiones sobre componentes que puedan reutilizarse, y representar una línea de producto en vez de un solo producto.
- **UML no es lo mejor. Es una notación informal.**
  - El UML no es útil durante el proceso de diseño en sí y prefiere notaciones informales que sean más rápidas de escribir y puedan dibujarse fácilmente.
- Los **usuarios de métodos ágiles** afirman que, por lo general, **no se utiliza la documentación detallada del diseño**. Por lo tanto, desarrollarla es un desperdicio de tiempo y dinero.
- En general, para la mayoría de los sistemas, **no vale la pena desarrollar una descripción arquitectónica detallada** desde estas cuatro perspectivas.
  - **No siempre se aplica el modelo 4+1**

5.3 Patrones arquitectónicos

Universidad Francisco de Vitoria UFV Madrid

- Solución común a un problema común en un contexto dado.
- **Origen: patrones de diseño orientados a objetos** que incitó el desarrollo de otros tipos de patrón, como los patrones para el diseño organizacional, patrones de usabilidad, interacción, administración de la configuración.
- **Los patrones arquitectónicos se propusieron en la década de 1990, con el nombre de «estilos arquitectónicos»**
- Un patrón arquitectónico se puede considerar como una descripción abstracta estilizada de buena práctica, que se ensayó y puso a prueba en diferentes sistemas y entornos.
- La idea de los patrones como una forma de **presentar, compartir y reutilizar el conocimiento** sobre los sistemas de software se usa ahora ampliamente.
- Los **estilos arquitecturales** pueden ser vistos como patrones para la organización de alto nivel del software (patrones macro-arquitecturales).
- **Tipos patrones arquitectónicos**:
    1. Arquitectura por capas.
    2. Arquitectura de repositorio.
    3. Arquitectura cliente – servidor.
    4. Arquitectura pipe and filter.

5.3 Patrones arquitectónicos

Universidad Francisco de Vitoria UFV Madrid

1. Arquitectura por capas

- El patrón de arquitectura en capas es otra forma de lograr **separación e independencia**.

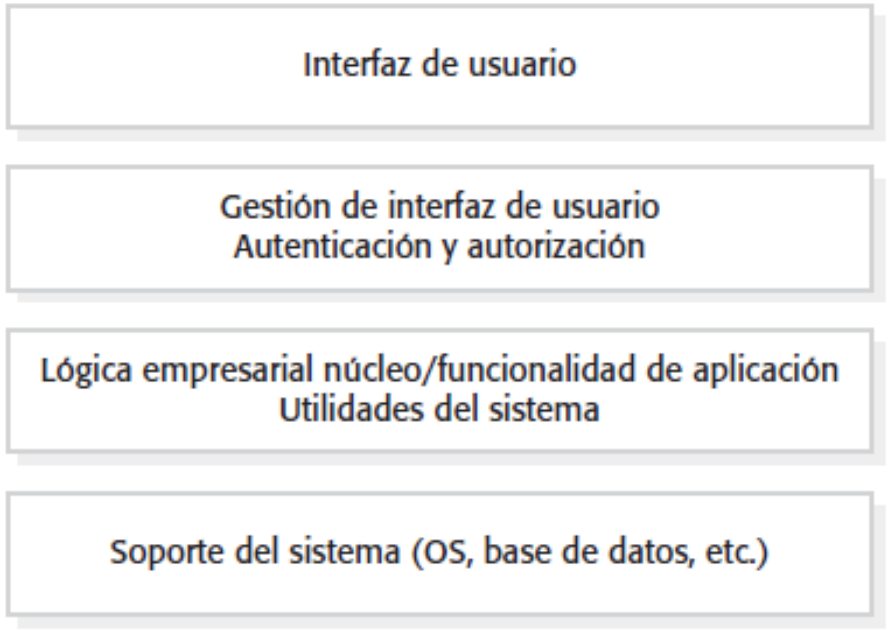


Figura 6.6 Arquitectura genérica en capas

5.3 Patrones arquitectónicos

Universidad Francisco de Vitoria UFV Madrid

2. **Arquitectura de repositorio**

- El siguiente ejemplo, el patrón de repositorio describe cómo comparte datos un conjunto de componentes en interacción.
- Para aplicaciones en las que un componente genere datos y otro los use.

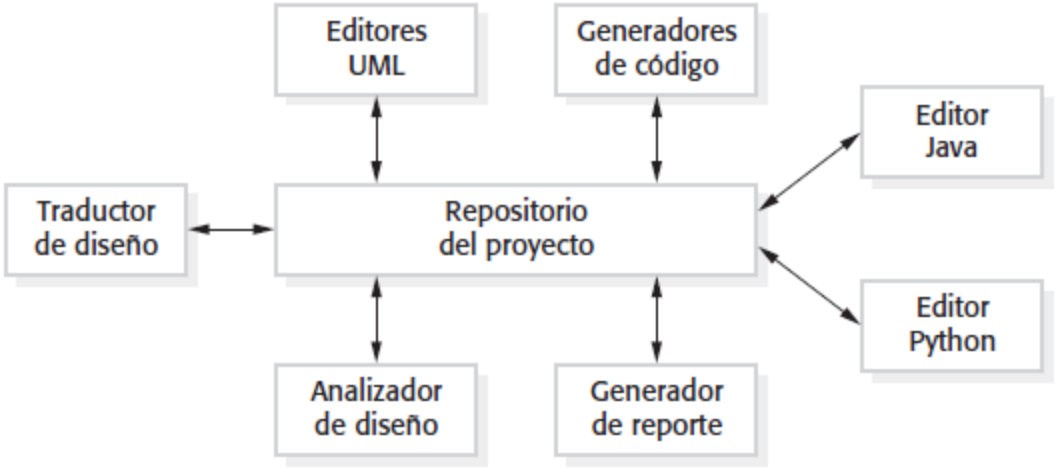


Figura 6.9 Arquitectura de repositorio para un IDE

5.3 Patrones arquitectónicos

Universidad Francisco de Vitoria UFV Madrid

3. Arquitectura cliente - servidor

- Se usa cuando se tiene que ingresar los datos en una base de datos compartida desde varias ubicaciones. Como los servidores se pueden replicar, también se usan cuando la carga de un sistema es variable.
- Se organiza como un conjunto de servicios y servidores asociados, y de clientes que acceden y usan los servicios. Los principales componentes de este modelo son:
  - **Un conjunto de servidores que ofrecen servicios a otros componentes.** Servidores de impresión; servidores de archivo que brindan servicios de administración de archivos.
  - **Un conjunto de clientes que solicitan los servicios que ofrecen los servidores**. Habrá usualmente varias instancias de un programa cliente que se ejecuten de manera concurrente en diferentes computadoras.
  - **Una red que permite a los clientes acceder a dichos servicios**. La mayoría de los sistemas cliente-servidor se implementan como sistemas distribuidos, conectados mediante protocolos de Internet.
- Los clientes acceden a los servicios que proporciona un servidor través de llamadas a procedimiento remoto usando un protocolo solicitud-respuesta, como el protocolo http utilizado en Internet.

5.3 Patrones arquitectónicos

Universidad Francisco de Vitoria UFV Madrid

4. Arquitectura de tubería y filtro (pipe and filter)

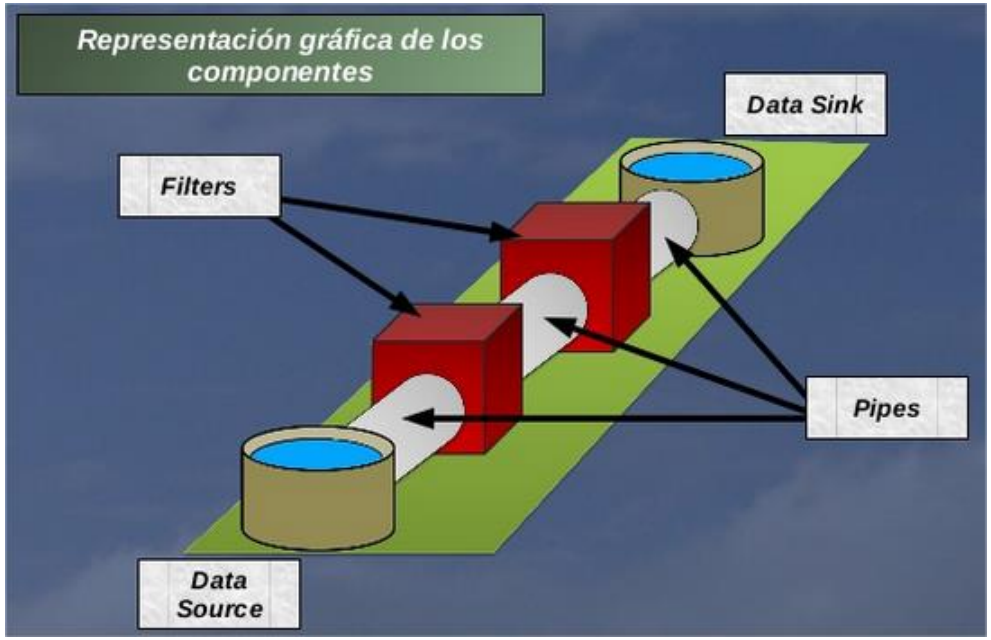
- Modelo de la organización en tiempo de operación de un sistema, donde las transformaciones funcionales procesan sus entradas y producen salidas. Los datos fluyen y se transforman conforme se desplazan a través de la secuencia. Cada **paso** de la secuencia es un **transformador**.
- Cuando las **transformaciones son secuenciales**, con datos procesados en lotes, este modelo arquitectónico de tubería y filtro se convierte en un **modelo secuencial en lote**, una arquitectura común para sistemas de procesamiento de datos (por ejemplo, un sistema de facturación).
- La **arquitectura de un sistema embebido** puede organizarse también como un proceso por entubamiento, donde cada proceso se ejecuta de manera concurrente.
- «*El patrón de arquitectura Pipe and Filter provee una estructura para los sistemas que procesan un flujo de datos. Cada paso de procesamiento se encapsula en un componente de filtro. Los datos pasan a través de tuberías entre filtros adyacentes. Combinar filtros permite construir familias de filtros relacionados*» Buschmann

5.3 Patrones arquitectónicos

Universidad Francisco de Vitoria UFV Madrid

4. Arquitectura de tubería y filtro (pipe and filter). Características

- Dividir la tarea en una secuencia de etapas de procesamiento.
- Cada etapa será implementada por un programa filtro que consume por su entrada y produce datos en su salida de manera incremental.
- Conectar la saluda de una etapa (filter) como la entrada de su etapa sucesora por medio de una tubería (pipe).
- Permitir que los filtros se ejecuten de manera concurrente.
- Conectar la entrada de la secuencia a alguna fuente de datos (data source), como un archivo.
- Conectar la salida de la secuencia a algún sumidero de datos (data sink) como un archivo o una plantilla.



5.3 Patrones arquitectónicos

Universidad Francisco de Vitoria UFV Madrid

Filter	
Unidades de procesamiento	
Responsabilidades	**Características**
• Recibir datos de entrada • Ejecutar una función sobre dichos datos • Proporcionar datos de salida	• Enriquecen, refinan o transforman los datos • Pueden ser pasivos o activos
	Colaboradores
	• Pipe

Pipe	
Conexiones entre dos componentes	
Responsabilidades	**Colaboradores**
• Transferir datos • Almacena datos temporalmente • Sincroniza los vecinos activos	• Filter • Data Source • Data Sink

Data Source	
Responsabilidades	**Colaboradores**
• Proporciona los datos de entrada a la linea de procesamiento	• Pipe
Variantes	**Ejemplos**
-Pasivas (proveen datos al ser solicitados) -Activas (constantemente entregan información)	• Archivos de texto plano • Sensores externos • Entrada estándar (stdin)

Data Sink	
Responsabilidades	**Colaboradores**
• Almacenar los datos de salida de la linea de procesamiento	• Pipe
	Variantes
	Pasivas y activas

5.4 Arquitecturas de aplicación

Universidad Francisco de Vitoria UFV Madrid

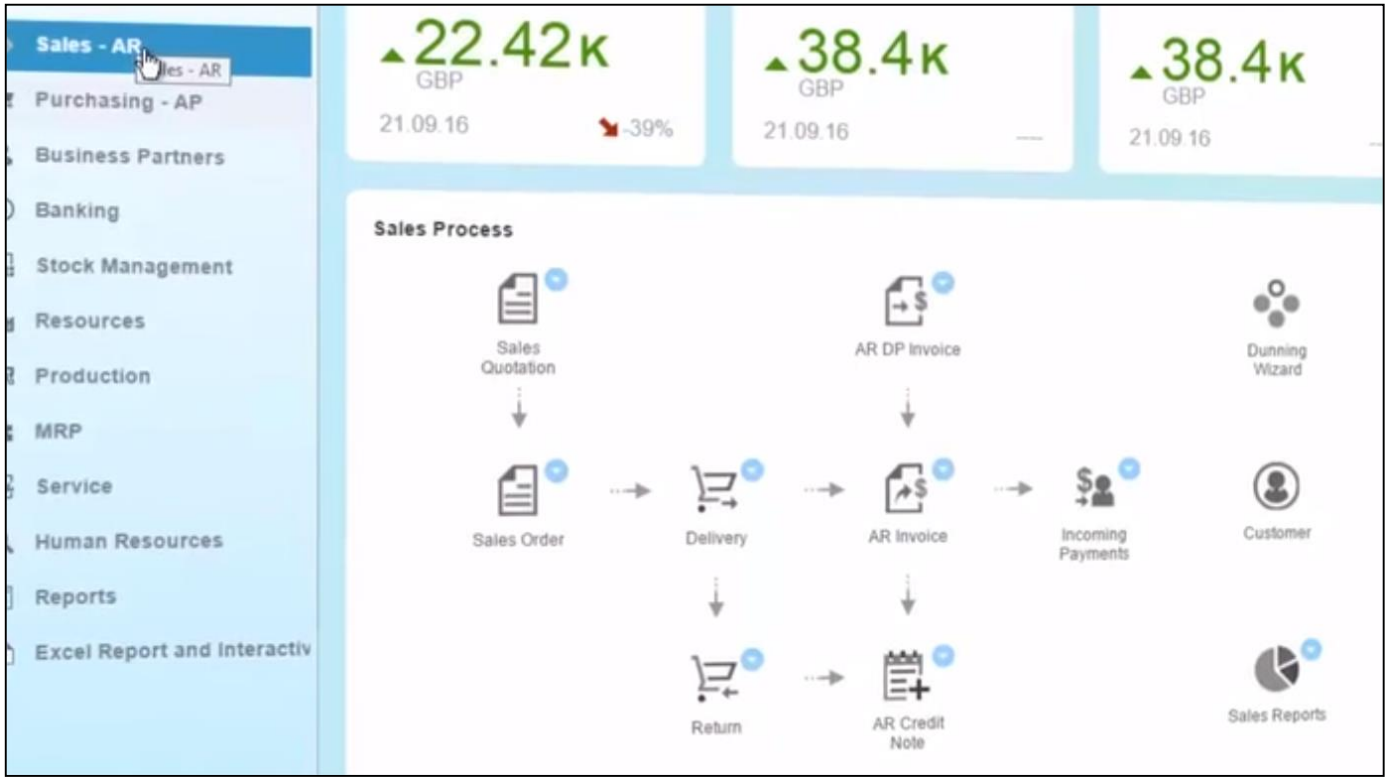
- Los sistemas de aplicación tienen la intención de cubrir las necesidades de una empresa u organización.
- Todas las empresas tienen mucho en común: necesitan contratar personal, emitir facturas, llevar la contabilidad.
- Estos factores en común condujeron al desarrollo de arquitecturas de software que describen la estructura y la organización de tipos particulares de sistemas de software.
- Las **arquitecturas de aplicación** encapsulan las principales características de una clase de sistemas.
- La **arquitectura de aplicación** puede reimplantarse cuando se desarrollen nuevos sistemas, pero, **para diversos sistemas empresariales, la reutilización de aplicaciones es posible sin reimplementación**.
  - Esto se observa en el crecimiento de los sistemas de planificación de recursos empresariales .
  - (ERP, por las siglas de Enterprise Resource Planning) de compañías como SAP y Oracle, y **paquetes de software vertical** (**COTS Commercial off the shelf**) para aplicaciones especializadas en diferentes **áreas de negocios**.

5.4 Arquitecturas de aplicación

Universidad Francisco de Vitoria UFV Madrid

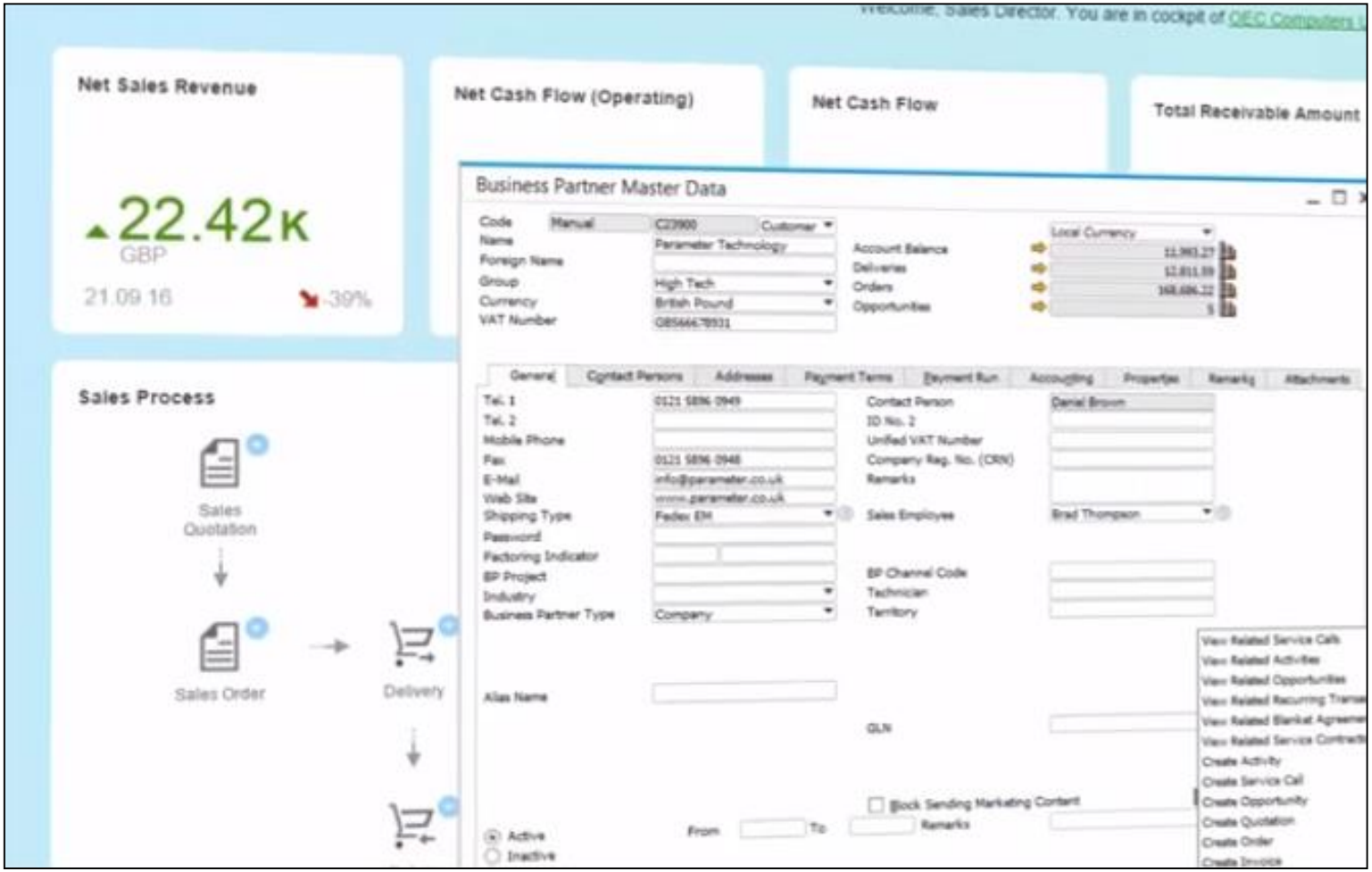
SAP ERP (Enterprise Resource Planning):

- Da soporte a las funciones esenciales de los procesos y operaciones de la empresa. A su vez, se subdivide en: SAP ERP Finanzas, SAP ERP Gestión del capital financiero, SAP ERP Operaciones.
- Crear y suministrar productos innovadores.
- Aumentar los ingresos por la venta de estos.
- Mejora el servicio a los clientes.



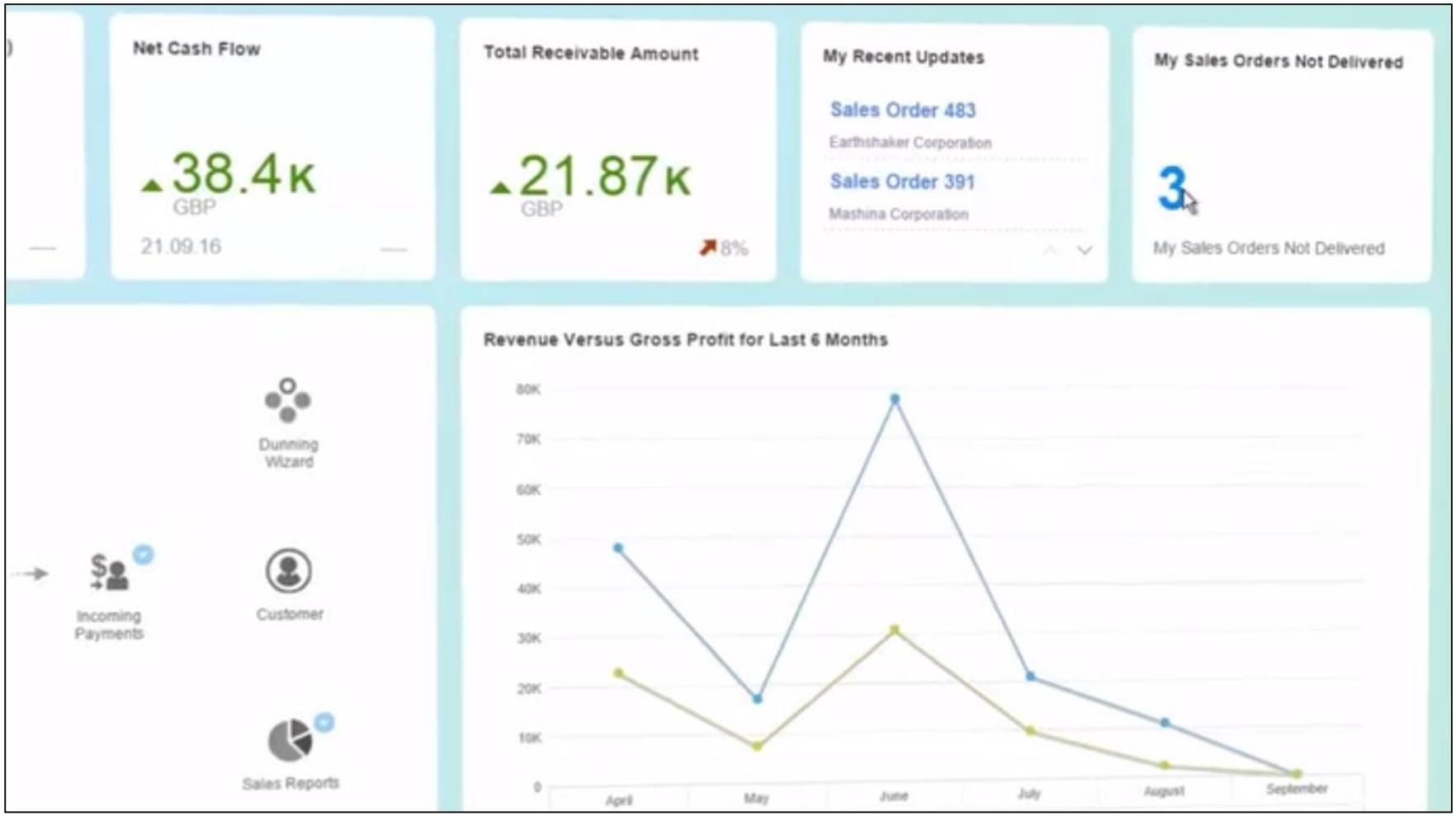
5.4 Arquitecturas de aplicación

Universidad Francisco de Vitoria UFV Madrid



5.4 Arquitecturas de aplicación

Universidad Francisco de Vitoria UFV Madrid



5.4 Arquitecturas de aplicación

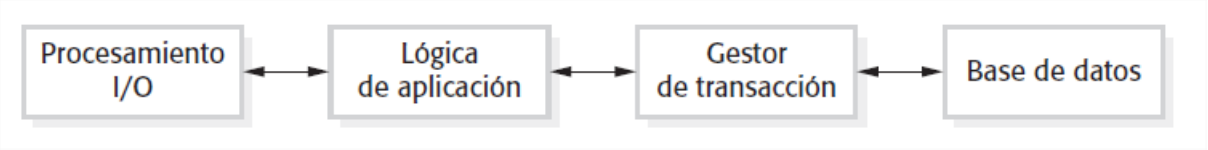
Universidad Francisco de Vitoria UFV Madrid

- Como diseñador de software, usted puede usar modelos de arquitecturas de aplicación en varias formas:
  1. ***Como punto de partida para el proceso de diseño arquitectónico***. Si no está familiarizado con el tipo de aplicación que desarrolla, podría basar su **diseño inicial** en una **arquitectura de aplicación genérica**. *Tiene que ser especializada… pero es un buen comienzo para hacer el diseño.*
  2. ***Como lista de verificación del diseño***. Si se desarrolló un diseño arquitectónico para un sistema de **aplicación**, puede comparar éste con la arquitectura de **aplicación genérica** y luego, **verificar que su diseño sea consistente** con la arquitectura genérica.
  3. ***Como una forma de organizar el trabajo del equipo de desarrollo***. Las **arquitecturas de aplicación** identifican características estructurales estables de las **arquitecturas del sistema** y, en muchos casos, es posible desarrollar éstas en **paralelo**.
  4. ***Como un medio para valorar los componentes a reutilizar***. Comparar éstos **componentes** con las estructuras **genéricas** para saber **si existen componentes similares** en la arquitectura de aplicación (reutilizables).
  5. ***Como un vocabulario para hablar acerca de los tipos de aplicaciones***. Puede usar los conceptos identificados en la arquitectura genérica para hablar sobre las aplicaciones

5.4 Arquitecturas de aplicación

Universidad Francisco de Vitoria UFV Madrid

- Hay muchos tipos de sistema de aplicación y, en algunos casos, parecerían muy diferentes.
- Sin embargo, muchas de estas aplicaciones distintas que en realidad **tienen mucho en común** y que suelen representarse mediante **una sola arquitectura de aplicación abstracta**.
- Se ilustra en las arquitecturas de **dos tipos de aplicación**; (como los tipos que estudiamos al comienzo)
  - **Aplicaciones de procesamiento de transacción** Este tipo de aplicaciones son aplicaciones centradas en **bases de datos**, que procesan los requerimientos del usuario mediante la información y actualizan ésta en una base de datos. Se trata del tipo más común de sistemas empresariales interactivos.



  - **Sistemas de procesamiento de lenguaje** Son sistemas en los que las intenciones del usuario se expresan en **un lenguaje formal** (como Java). El sistema de procesamiento de lenguaje elabora este lenguaje en un formato interno y después interpreta dicha representación interna. Los sistemas de procesamiento de lenguaje mejor conocidos son los compiladores, que traducen los programas en lenguaje de alto nivel dentro de un código de máquina.

5.4 Arquitecturas de aplicación

Universidad Francisco de Vitoria UFV Madrid

5.4.1 Sistemas de procesamiento de transacciones

- Técnicamente, una **transacción de base de datos** es una secuencia de operaciones que se trata como una **sola unidad** (una unidad atómica).
- Todas las operaciones en una transacción tienen que **completarse antes de que sean permanentes los cambios** en la base de datos.
- Desde una perspectiva de usuario, una transacción es cualquier **secuencia coherente** de operaciones que satisface un objetivo.
- Por lo general, los sistemas de procesamiento de transacción son sistemas interactivos donde los usuarios hacen **peticiones asíncronas de servicios**.

5.4.2 Sistemas de Información

- Todos los sistemas que incluyen interacción con una base de datos compartida se consideran sistemas de información basados en transacciones.
- Considerar el problema de la clave única
- Dichos sistemas se suelen implementar como **arquitecturas cliente-servidor de multinivel**:
  - El servidor Web responsable de todas las comunicaciones del usuario, y la interfaz de usuario.
  - El servidor de aplicación responsable de implementar la lógica específica de la aplicación
  - El servidor de la base de datos El servidor de la base de datos mueve la información hacia y desde la base de datos. Gestión de transacciones.

5.4 Arquitecturas de aplicación

Universidad Francisco de Vitoria UFV Madrid

5.4.3 Sistemas de procesamiento de lenguaje

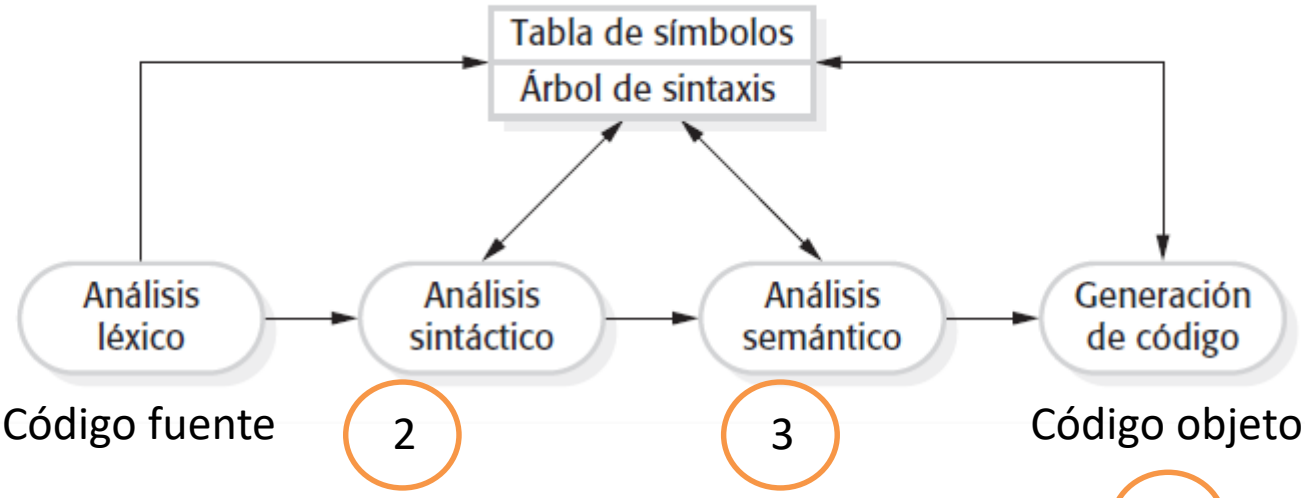


Figura 6.19 Una arquitectura de compilador de tubería y filtro

- Los sucesivos filtros comparten un cierto estado: la tabla de símbolos.
- No es una arquitectura pipe and filter estricta.
- La implementación es un único programa.