

Asignatura

Arquitectura del Software



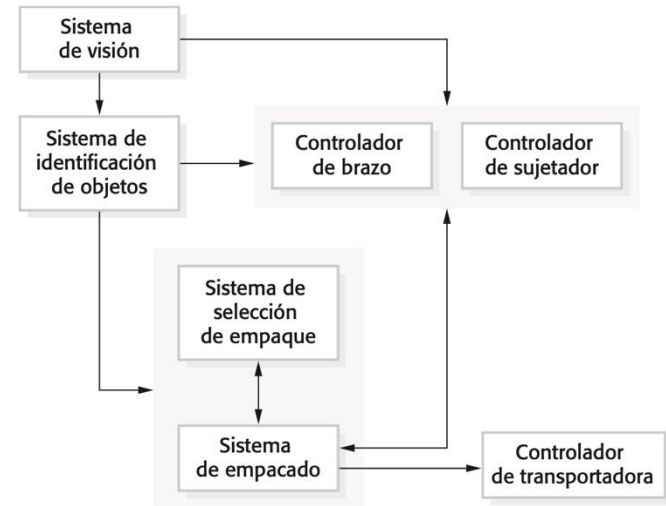
Profesor

Yago Fontenla Seco

{yago.fontenla1@uie.edu}

Arquitectura del Software - Puntos clave

- La arquitectura de software define la **estructura** fundamental de un sistema. Pero incluye tanto decisiones estructurales como no estructurales.
- Se centra en la **organización** de componentes del software y cómo **interactúan**.
- Es esencial para la calidad, el rendimiento y la escalabilidad del sistema.



Arquitectura de un sistema de control para un robot empacador

Decisiones Arquitectónicas

Las decisiones arquitectónicas determinan cómo se **organiza** un sistema de software. Consideran tanto **restricciones funcionales** como **no funcionales** (rendimiento, seguridad, etc.). Algunas decisiones clave incluyen:

- Selección de **estilos** y **patrones** arquitectónicos.
- Definición de **vistas** y **diagramas** que representen la arquitectura.
- Identificación de **componentes** principales y sus relaciones.

Los **componentes** del sistema implementan los requisitos **funcionales**. La forma en que los componentes se **organizan y comunican**, representa los requisitos **no funcionales**.

Decisiones Arquitectónicas


Preguntas clave:

1. ¿Existe alguna **arquitectura de aplicación genérica** que actúe como plantilla para el sistema que se está diseñando?
2. ¿Cómo se distribuirá el sistema a través de algunos núcleos o procesadores?
3. ¿Qué **patrones o estilos arquitectónicos** pueden usarse?
4. ¿Cuál será el enfoque fundamental usado para estructurar el sistema?
5. ¿Cómo los componentes estructurales en el sistema se separarán en **subcomponentes**?
6. ¿Qué estrategia se usará para controlar la operación de los componentes en el sistema?
7. ¿Cuál **organización arquitectónica** es mejor para entregar los **requerimientos no funcionales del sistema**?
8. ¿Cómo se evaluará el diseño arquitectónico?
9. ¿Cómo se documentará la arquitectura del sistema?

Requisitos no funcionales y arquitectura

- **Rendimiento:** Cuando el rendimiento es esencial, la arquitectura debe **agrupar las operaciones críticas** en pocos componentes grandes, reduciendo la necesidad de comunicaciones frecuentes entre ellos.
- **Seguridad:** Para sistemas con altos requisitos de seguridad, una **arquitectura en capas** es ideal. Los activos más sensibles se protegen en las capas internas, y se aplican rigurosas validaciones de seguridad a esas capas, minimizando los riesgos de ataques o accesos no autorizados.
- **Protección:** En sistemas donde la protección es prioritaria, las operaciones relacionadas con la protección deben concentrarse en un **pequeño número de componentes**. Esto facilita la validación y el control de la protección, permitiendo un apagado seguro del en caso de fallos.
- **Disponibilidad:** Para garantizar la alta disponibilidad, es necesario incluir **componentes redundantes** en la arquitectura. Esto permite que se realicen actualizaciones o reparaciones sin interrumpir el servicio del sistema.
- **Mantenibilidad:** Una arquitectura diseñada para ser mantenible debe componerse de pequeños **componentes independientes**. Estos componentes deben ser intercambiables, con una clara separación, evitando la interdependencia de estructuras de datos.

Patrones Arquitectónicos

 Un **patrón arquitectónico** define una solución reutilizable a problemas recurrentes en el diseño de software a nivel estructural. Estos patrones son plantillas que ayudan a diseñar sistemas escalables, mantenibles y eficientes.

Importancia en el Diseño de Software

- **Modularidad:** Facilitan la división del sistema en componentes manejables.
- **Escalabilidad:** Permiten construir sistemas capaces de crecer en tamaño y complejidad.
- **Mantenibilidad:** Fomentan una estructura que facilita actualizaciones y correcciones.
- **Reutilización de Código:** Posibilitan la reutilización de soluciones probadas, reduciendo tiempos y costos.

Cada patrón afecta distintos atributos de calidad como rendimiento, seguridad, y facilidad de mantenimiento. Elegir el patrón adecuado es esencial para cumplir con los requisitos del sistema.

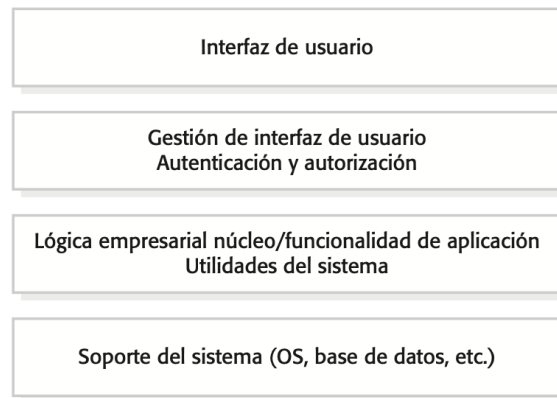
Arquitectura de Capas y Evoluciones

Organiza el sistema en capas jerárquicas, cada una de las cuales provee servicios a la capa superior y utiliza los servicios de la capa inferior. Este modelo es común en sistemas donde la separación de responsabilidades es clave.

Aspectos Clave: Cada capa tiene una función específica, como presentación, lógica de negocio, o gestión de datos. La comunicación es unidireccional: de capa superior a inferior.

Evoluciones: Una ventaja de este modelo es su facilidad para evolucionar. Las capas superiores pueden cambiar sin afectar a las inferiores.

Beneficios: Facilita la modularidad, el mantenimiento y la escalabilidad. Además, permite controlar y limitar el acceso a funcionalidades específicas mediante capas.



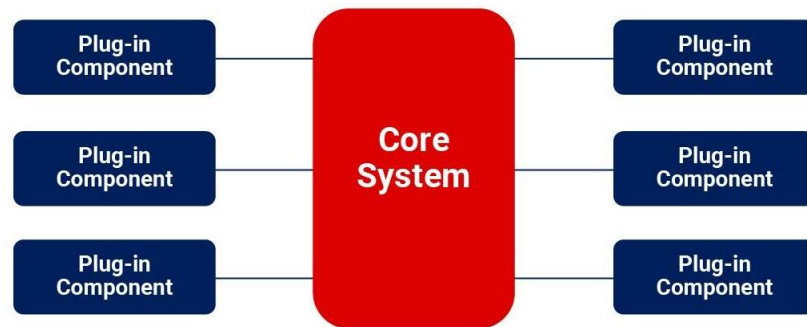
Arquitectura Microkernel

En una arquitectura microkernel, el sistema se construye alrededor de un núcleo (core) mínimo y esencial que ofrece las funciones básicas, mientras que las funcionalidades adicionales se implementan mediante complementos (plug-ins) que interactúan con el núcleo a través de APIs.

Aspectos Clave: El núcleo es responsable de la gestión principal, como servicios de seguridad y comunicación. Los plug-ins añaden funcionalidades accediendo al core a través de APIs predefinidas.

Características: Modularidad elevada, el núcleo permanece inalterado y pueden añadirse nuevas funcionalidad.

Beneficios: Permite flexibilidad y personalización mediante plug-ins, facilitando la evolución sin afectar la estabilidad.

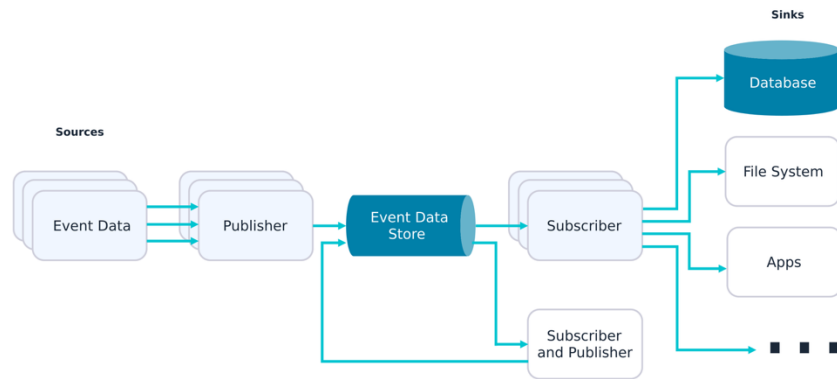


Arquitectura Manejada por Eventos

En la arquitectura manejada por eventos, los componentes del sistema se comunican a través de eventos. Los componentes pueden actuar como productores de eventos, consumidores o ambos, lo que facilita un alto grado de flexibilidad en la interacción.

Aspectos Clave: Escuchar, producir y consumir eventos. Los eventos son emitidos por productores y escuchados por consumidores que responden de acuerdo con reglas predefinidas.

Características: Desacoplamiento entre componentes, permitiendo que los productores y consumidores de eventos se modifiquen independientemente. Facilita la respuesta rápida a cambios.

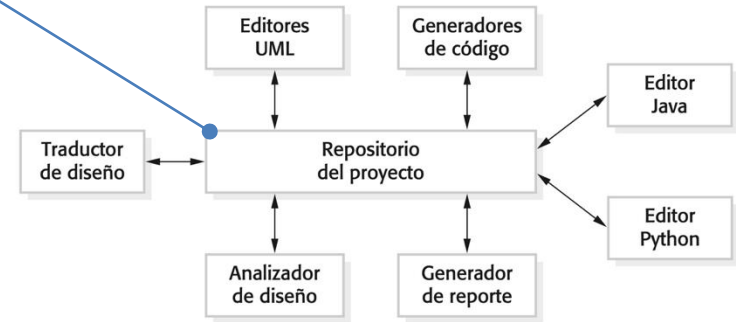


Arquitectura Centrada en Datos

La arquitectura centrada en datos, o basada en “pizarra” (blackboard), organiza el sistema en torno a un repositorio central de datos que es accedido y modificado por distintos componentes del sistema. Los componentes se comunican indirectamente a través de actualizaciones en el repositorio.

Aspectos Clave: “Pizarra” (blackboard), que actúa como área de almacenamiento compartido. Los componentes observan o contribuyen a este espacio de datos central, accediendo y modificando el estado del sistema.

Características: Alta flexibilidad en el acceso a los datos compartidos y una clara separación entre procesamiento de datos y almacenamiento. Cada componente puede reaccionar a los cambios en la pizarra según las reglas definidas.



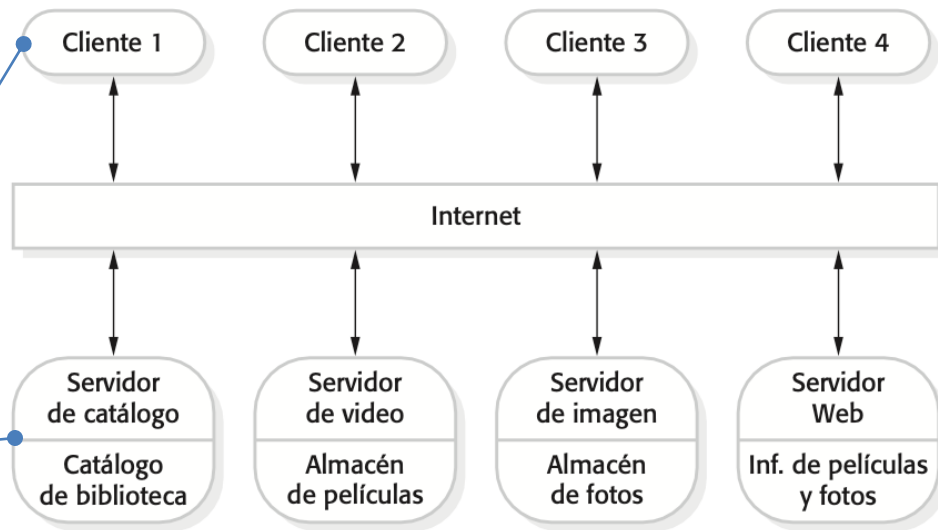
Sistemas distribuidos: Cliente-Servidor

Un sistema distribuido es aquel que implica numerosas computadoras, en contraste con los sistemas centralizados en que todos los componentes de sistema se ejecutan en una sola computadora.

En una arquitectura **cliente-servidor**, la funcionalidad del sistema se organiza en servicios, y cada servicio lo entrega un servidor independiente.

Roles:

- **Cliente** es quien solicita recursos o servicios.
- **Servidor** es quien responde a estas solicitudes y proporciona los recursos.



Asignatura

Arquitectura del Software



11

Profesor

Yago Fontenla Seco

{yago.fontenla1@uie.edu}