UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

DEPARTAMENTO DE CIENCIAS DE LA COMUNICACIÓN

SOFTWARE

MATERIA:

ANALISIS Y DISEÑO DE SOFTWARE

Nombre:
Alisson Nicole Clavijo Gutiérrez

NRC:
9864

Docente:

Ing. Alexis Darío Estévez Salazar

1. Tema

Cohesión y Tipos de Acoplamiento.

2. Introducción:

La arquitectura de software es un elemento crucial en el desarrollo de sistemas y aplicaciones, ya que define la estructura y organización de un sistema, estableciendo las bases para su diseño, implementación y mantenimiento. Los estilos arquitectónicos son patrones generales que guían la organización y estructura de los componentes de software en un sistema. Este informe explora la importancia de la arquitectura de software y presenta algunos ejemplos de estilos arquitectónicos.

En la ingeniería de software, la descomposición es un proceso clave que implica dividir un sistema complejo en componentes más pequeños y manejables. Además, la cohesión y el acoplamiento son conceptos esenciales para lograr una arquitectura de software eficiente y mantenible. Este informe explora la descomposición, los tipos de cohesión y los niveles de acoplamiento, destacando su importancia en el diseño de sistemas de software.

3. Objetivos:

- Comprender la importancia de la arquitectura de software en el desarrollo de sistemas.
- Identificar y describir diferentes estilos arquitectónicos.
- Analizar ejemplos de sistemas que utilizan diferentes estilos arquitectónicos.
- Extraer conclusiones y recomendaciones para el diseño arquitectónico de sistemas.

4. Marco Teórico:

La arquitectura de software se refiere a la estructura y organización de un sistema de software. Define cómo los componentes interactúan entre sí y con el entorno. Los estilos arquitectónicos son patrones que establecen directrices para la organización y comunicación de estos componentes. Algunos estilos arquitectónicos comunes incluyen:

- Arquitectura Monolítica: Un sistema único y compacto en el que todos los componentes están integrados en una sola aplicación. Ejemplo: Aplicaciones de procesamiento de textos como Microsoft Word.
- Cliente-Servidor: Divide el sistema en dos partes: el cliente que solicita servicios y el servidor que los proporciona. Ejemplo: Aplicaciones web, donde el navegador es el cliente y el servidor web maneja la lógica de negocio y los datos.
- Arquitectura en Capas: Divide el sistema en capas jerárquicas, donde cada capa se encarga de una funcionalidad específica. Ejemplo: Aplicaciones empresariales, donde la capa de presentación, lógica de negocio y acceso a datos están separadas.
- Arquitectura Orientada a Microservicios: El sistema se compone de múltiples servicios pequeños e independientes, cada uno ejecutando una función específica.

Ejemplo: Aplicaciones en la nube, donde cada microservicio puede escalarse y actualizarse de manera independiente.

- Arquitectura Basada en Eventos: Los componentes del sistema interactúan mediante la emisión y recepción de eventos. Ejemplo: Sistemas de mensajería en tiempo real.

5. Descomposición:

La descomposición es el proceso de dividir un sistema en partes más pequeñas y manejables, lo que facilita el diseño, la implementación y el mantenimiento. Esto permite que el sistema sea más fácilmente comprensible y modificable. La descomposición puede realizarse de manera jerárquica, funcional o modular, según los objetivos y la estructura del sistema.

6. Tipos de Cohesión:

La cohesión se refiere al grado en que los componentes de un módulo están relacionados y se centran en un objetivo común. Los siete tipos de cohesión, en orden ascendente de calidad, son:

- Cohesión Coincidental: Los componentes no tienen relación lógica y están agrupados por conveniencia.
- Cohesión Lógica: Los componentes están agrupados según su funcionalidad lógica.
- Cohesión Temporal: Los componentes se agrupan porque se ejecutan en el mismo período de tiempo.
- Cohesión de Comunicación: Los componentes se agrupan debido a que interactúan entre sí.
- Cohesión de Secuencia: Los componentes se agrupan porque uno depende del resultado del otro.
- Cohesión Funcional: Los componentes están agrupados porque realizan funciones similares y relacionadas.
- Cohesión de Información: Los componentes se agrupan debido a que comparten y operan sobre el mismo conjunto de datos.

7. Niveles de Acoplamiento:

El acoplamiento se refiere a la interdependencia entre los componentes de un sistema. Los cinco niveles de acoplamiento, en orden descendente de deseabilidad, son:

- Acoplamiento de Contenido: Los componentes acceden a detalles internos de otros componentes.
- Acoplamiento Común: Los componentes comparten datos globales.
- Acoplamiento de Control: Un componente controla el flujo de otro componente.

- Acoplamiento de Datos: Los componentes comparten datos a través de parámetros.
- Acoplamiento Sin Datos (Acoplamiento Bajo): Los componentes no comparten información, lo que resulta en una independencia más alta.

8. **Ejemplos**:

- Arquitectura Monolítica: Microsoft Office Suite.
- Cliente-Servidor: Aplicación de correo electrónico como Microsoft Outlook.
- Arquitectura en Capas: Sistema bancario con capas de presentación, lógica de negocio y acceso a la base de datos.
- Arquitectura de Microservicios: Netflix, donde diferentes microservicios manejan la reproducción, recomendaciones, autenticación, etc.
- Arquitectura Basada en Eventos: Aplicaciones de redes sociales como Twitter, donde los eventos incluyen publicaciones, retweets, etc.

9. Conclusiones y Recomendaciones:

- La arquitectura de software es esencial para crear sistemas eficientes, escalables y mantenibles. La elección del estilo arquitectónico debe basarse en las necesidades y requisitos del sistema. La modularidad, la separación de preocupaciones y la flexibilidad son factores clave a considerar al diseñar la arquitectura. Es importante evaluar los pros y contras de cada estilo arquitectónico y adaptarlos a las circunstancias específicas del proyecto.
- La descomposición, los tipos de cohesión y los niveles de acoplamiento son fundamentales para el diseño y la construcción de sistemas de software eficientes y mantenibles. Un sistema bien descompuesto, con una alta cohesión y un bajo acoplamiento, es más fácil de entender, modificar y mantener a lo largo del tiempo.

10. Bibliografía:

- Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Addison-Wesley Professional.
- Fowler, M. (2002). Patterns of Enterprise Application Architecture. Addison-Wesley Professional.
- Newman, S. (2015). Building Microservices. O'Reilly Media.
- Shaw, M., & Garlan, D. (1996). Software Architecture: Perspectives on an Emerging Discipline
- Sommerville, I. (2011). Software Engineering (9th ed.). Addison-Wesley.
- Pressman, R. S. (2014). Software Engineering: A Practitioner's Approach (8th ed.). McGraw-Hill.
- Bass, L., Clements, P., & Kazman, R. (2012). Software Architecture in Practice. Addison-Wesley Professional.
- Yourdon, E. (1979). Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design. Prentice-Hall.