UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

CARRERA INGENIERÍA SOFTWARE

TEMAS:

INTRODUCCIÓN AL DISEÑO DE SOFTWARE
Y
PRINCIPIOS DE DISEÑO DE SOFTWARE

PRESENTADO POR:
BRYAN JAVIER FIGUEROA MORALES

FECHA:

SÁBADO, 22 DE NOVIEMBRE DEL 2025
(*Semana 1-2*)

INTRODUCCION

La ingeniería del software es una disciplina de la ciencia de la información que recurre a metodologías de diseño variadas cuyas características se adaptan a las necesidades del usuario o cliente [1]. El diseño de software se define como el conjunto de principios, conceptos y prácticas que conducen al desarrollo de un sistema o producto software. Esta fase une la creatividad y la concepción del software, donde deben reflejarse todos los requisitos y necesidades del negocio para formular un producto que brinde servicios a los usuarios [2].

El modelo de diseño resultante proporciona detalles específicos sobre la arquitectura del software, la estructura de datos, las interfaces o prototipo y todos los componentes necesarios para la implementación del sistema. La importancia del diseño se resume en la calidad del producto final [2].

Desarrollo

1. Introducción al Diseño de Software

La arquitectura de software, según Miriam Martínez es la clave en esta fase, se refiere a la estructura organizacional y conceptual que se establece en el diseño y desarrollo de sistemas de software. Su objetivo es definir una solución estructural que cumpla con todos los requisitos, tanto funcionales como no funcionales (como la escalabilidad, el rendimiento y la seguridad) [3]. La arquitectura se considera como el plano principal de una obra arquitectónica y determina la robustez, flexibilidad y eficiencia del sistema, aunque no siempre sea visible para los usuarios finales.

1.1. Relación con el ciclo de vida del software (SDLC)

El Ciclo de Vida del Desarrollo de Software (SDLC) es un proceso organizado que abarca todas las fases del desarrollo, desde la planificación inicial hasta el mantenimiento y la eventual descontinuación del software. Este ciclo es un elemento clave dentro de la ingeniería de software, ya que establece los procedimientos, recursos y productos que se emplean y generan durante el desarrollo [4].

El Diseño de Software constituye una etapa esencial dentro del SDLC. Su propósito es analizar los requisitos para definir la estructura interna del sistema, la cual servirá como base para su implementación. En el flujo del desarrollo, esta fase se lleva a cabo después de la recopilación y el análisis de requisitos. Como resultado del diseño, se obtiene un conjunto de modelos y artefactos que documentan las decisiones más importantes tomadas durante esta etapa.

1.2. Diseño vs. implementación

La distinción entre diseño e implementación es la siguiente:

- Diseño: Es la fase estratégica y crucial, donde se planea la estructura general y se definen aspectos importantes como la arquitectura, los componentes y las interfaces del sistema. Es en este momento donde realmente se hace ingeniería, ya que se concibe la solución del sistema. Además, es una fase importante ya que la mayoría de los errores graves se introducen en esta etapa y a menudo solo se descubre cuando el software ya está en producción. Estos errores de diseño son más difíciles de detectar y corregir en fases posteriores [2].

- Implementación (o Codificación): Es la fase operativa que traduce el diseño definido en código ejecutable.

2. Principios de Diseño de Software

Los principios de la ingeniería de software son pautas generales que describen las propiedades deseables tanto de los procesos de software como del producto final. Estos principios sirven como la base de todos los métodos, técnicas, metodologías y herramientas[5].

Se destacan los siguientes principios fundamentales:

2.1. Abstracción

La Abstracción es el proceso de identificar los aspectos relevantes de un problema, ignorando los detalles de menor importancia. Es de hecho, un caso especial del principio de Separación de intereses [6].

- Lo que se abstrae depende del propósito de la abstracción, permitiendo diferentes visiones de una misma realidad para fines específicos.

- En el contexto de la Programación Orientada a Objetos (POO), la abstracción se relaciona con la comprensión de un problema antes del diseño OO.

- Los lenguajes de programación son una forma de abstracción construida sobre el hardware, proveyendo constructores que permiten al desarrollador concentrarse en el problema a resolver, ignorando detalles de bajo nivel.

2.2. Encapsulamiento

El Encapsulamiento es el proceso de agrupar los datos y procedimientos (métodos) dentro de un objeto. Es uno de los cuatro aspectos básicos de la POO, junto con la Abstracción, el Polimorfismo y la Herencia [6].

- Los patrones de diseño hacen un fuerte énfasis en el encapsulamiento de acciones o datos.

2.3. Modularidad

La Modularidad implica dividir un sistema complejo en partes más simples llamadas módulos. Un sistema compuesto de módulos facilita la aplicación del principio de Separación de intereses en dos fases: el manejo de los detalles de cada módulo individual y la integración de las características globales y sus relaciones para formar un sistema coherente [3].

Los objetivos principales de la modularidad son tres:

- La capacidad de descomponer un sistema complejo, aplicando el principio de Divide y Vencerás.

- La capacidad de componerlo a partir de módulos existentes.

- La comprensión del sistema en piezas separadas, lo que facilita la modificabilidad y ayuda a restringir la búsqueda de errores.

Para alcanzar estos objetivos, los módulos deben tener alta cohesión (todos los elementos están fuertemente relacionados y cooperan para el objetivo común del módulo) y bajo acoplamiento (la interdependencia entre módulos es baja), lo que permite que los módulos sean más fáciles de analizar, comprender, modificar, probar o reutilizar en forma separada.

2.4. Separación de intereses

Este principio permite enfrentar los distintos aspectos individuales de un problema concentrándose en cada uno por separado. Es esencial para abordar la complejidad inherente a un proyecto de software [5].

La Separación de intereses se puede aplicar de varias maneras:

- Según el tiempo: Esto motiva el ciclo de vida del software, al planificar las distintas actividades de forma secuencial.

- Por cualidades: Tratando cualidades de software separadamente, por ejemplo, primero la correctitud y luego la eficiencia.

- Por visiones: Analizando distintas perspectivas del software por separado, como el flujo de datos versus el flujo de control.

- Por tamaño: Enfrentando partes del sistema separadamente, lo cual es el concepto fundamental de la Modularidad.

En última instancia, este principio sienta las bases para dividir el trabajo en un problema complejo, asignando responsabilidades específicas a personas con distintas habilidades.

Conclusión

El diseño de software es la piedra angular del desarrollo de aplicaciones, actuando como el plano maestro que define la arquitectura y la estructura interna del sistema. Como actividad esencial del Ciclo de Vida del Desarrollo de Software (SDLC), el diseño es una fase estratégica y de ingeniería que precede a la implementación operativa (codificación). La calidad del software está estrechamente ligada al diseño, y los errores conceptuales introducidos en esta etapa son costosos y difíciles de corregir una vez que el producto está avanzado o en producción.

Para gestionar la complejidad y asegurar la calidad, se aplican principios fundamentales:

• La Separación de intereses es la estrategia macro para dividir un problema complejo y asignar responsabilidades.

• La Modularidad aplica esta separación dividiendo el sistema en módulos con alta cohesión y bajo acoplamiento, lo que facilita la descomposición, la composición y la comprensión.

• La Abstracción permite concentrarse en los aspectos relevantes de un problema, ignorando los detalles para un propósito específico.

• El Encapsulamiento agrupa lógicamente los datos y las operaciones de un objeto, siendo fundamental en el diseño orientado a objetos.

La aplicación rigurosa de estos principios garantiza que el software no solo satisfaga sus requisitos funcionales, sino que también muestre capacidades cruciales para el éxito a largo plazo, como la escalabilidad, la capacidad de evolución y la agilidad del mismo.

4.- REFERENCIAS BIBLIOGRÁFICAS.

[1] R. D. Gordón Graell, “INGENIERÍA DE SOFTWARE: : APORTES, DESAFÍOS DE LA METODOLOGÍA SCRUM PARA SISTEMAS DE INFORMACIÓN EN PANAMÁ,” *Revista FAECO Sapiens*, vol. 6, no. 2, pp. 103–121, Jul. 2023, doi: 10.48204/j.faeco.v6n2.a4014.

[2] Practitioner Lab, “¿Qué es el diseño de software?” Accessed: Nov. 21, 2025. [Online]. Available: https://www.youtube.com/watch?v=56kAfd1DQX8

[3] Miriam Martínez Canelo, “Arquitectura de Software: Qué es y fundamentos clave,” Profile Software Services, S.L. 2025. Accessed: Nov. 21, 2025. [Online]. Available: https://profile.es/blog/que-es-la-arquitectura-de-software/

[4] M. O. Sydorov, “Toward software engineering ecosystems definition,” *PROBLEMS IN PROGRAMMING*, no. 3–4, 2022, doi: 10.15407/pp2022.03-04.092.

[5] Carlo. Ghezzi, Mehdi. Jazayeri, and Dino. Mandrioli, *Fundamentals of software engineering*. Prentice Hall, 1991.

[6] Josué Figueroa González, “Introducción a los Patrones de Diseño de Software.” Accessed: Nov. 21, 2025. [Online]. Available: https://academicos.azc.uam.mx/jfg/diapositivas/patrones/Unidad_1.pdf