# Patrones y Arquitectura de Software: Una Revisión Integrativa

María José Murcia Martínez
Neiva, Huila
mariajosemurciamartinez85@gmail.com
Centro de la Industria, la Empresa y los Servicios

## Resumen

Este artículo presenta una revisión integrativa de 34 artículos sobre patrones de diseño y arquitectura de software, con el objetivo de analizar las tendencias actuales en su implementación. La evolución de las arquitecturas de software se ha centrado en la necesidad de diseñar sistemas flexibles y adaptables, capaces de responder a los rápidos cambios tecnológicos y a las demandas de los usuarios. La usabilidad, como aspecto clave en la experiencia del usuario, se destaca como un factor fundamental en el diseño de software. Además, la flexibilidad en las arquitecturas permite la escalabilidad y una mayor capacidad de integración con otros sistemas. A lo largo del artículo se discuten diversas metodologías y enfoques aplicados, como las arquitecturas orientadas a servicios (SOA), y los patrones de diseño que ayudan a solucionar problemas recurrentes. La adopción de herramientas que faciliten la implementación eficiente de estas arquitecturas es crucial para el éxito del desarrollo de software. Se resalta la importancia de un enfoque modular que permita la reutilización de componentes y la integración de sistemas heterogéneos. Las metodologías ágiles también juegan un papel importante, ya que promueven la entrega rápida y la iteración constante de los sistemas, favoreciendo así la calidad y la adaptabilidad en el desarrollo. Finalmente, se concluye que el futuro de la arquitectura de software está marcado por la necesidad de soluciones que sean no solo tecnológicamente avanzadas, sino también centradas en el usuario, escalables, y capaces de integrarse de manera eficiente con otros sistemas.

**Palabras clave:** Patrones de diseño, arquitectura de software, usabilidad, metodologías ágiles, integración de sistemas.

## 1. Introducción

La arquitectura de software juega un papel crucial en el desarrollo de aplicaciones, ya que establece la estructura fundamental que guiará la implementación y el mantenimiento del sistema. Este artículo revisa 34 estudios que exploran diversas perspectivas sobre patrones de diseño y su impacto en la arquitectura de software.

## 2. Planteamiento del Problema

En el ámbito del desarrollo de software, la arquitectura y los patrones de diseño desempeñan un papel fundamental para garantizar la usabilidad, flexibilidad y adaptabilidad de los sistemas. Sin embargo, a pesar del desarrollo de metodologías ágiles y tecnologías avanzadas, aún persisten desafíos significativos en la implementación de arquitecturas eficientes que respondan a las necesidades cambiantes del negocio. Este problema se acentúa por la falta de comprensión de los patrones de diseño y su adecuada integración en proyectos reales, lo que puede derivar en sistemas poco escalables, difíciles de mantener y de baja calidad. Por ello, se hace necesario explorar y consolidar enfoques y herramientas que permitan superar estas limitaciones.

## 3. Objetivo

**General:**
Analizar el impacto de los patrones de diseño y las arquitecturas de software en el desarrollo de sistemas eficientes, usables y flexibles. **Específicos:**

- Identificar las tendencias actuales en el uso de patrones de diseño y arquitecturas en el desarrollo de software.
- Examinar las metodologías y herramientas que favorecen la implementación de arquitecturas adaptables.
- Proponer lineamientos basados en la revisión de literatura para mejorar la calidad y la sostenibilidad de los sistemas desarrollados.

## 4. Justificación

La investigación sobre patrones de diseño y arquitecturas de software es crucial para mejorar la calidad y sostenibilidad de los sistemas desarrollados en el contexto actual, donde la flexibilidad y la capacidad de adaptación son esenciales. Al estudiar las tendencias y las mejores prácticas en estos campos, se puede proporcionar una base sólida para que los desarrolladores e ingenieros de software construyan soluciones más eficientes, escalables y fáciles de mantener, adaptándose a las demandas cambiantes del negocio y la tecnología.

## 5. Marco Teórico

La **arquitectura de software** es una disciplina que establece la estructura y organización de los sistemas informáticos, con un enfoque en la gestión eficiente de los recursos y la modularidad. Un buen diseño arquitectónico facilita el mantenimiento, la escalabilidad y la integración de los sistemas. Según *Bass, Clements y Kazman* (2012), la arquitectura de software debe enfocarse en la calidad técnica, la flexibilidad y la capacidad de adaptación ante cambios, lo que se logra mediante el uso adecuado de patrones de diseño. Los **patrones de diseño** son soluciones generalizadas a problemas comunes en el desarrollo de software. Según *Gamma et al.* (1994), estos patrones permiten la reutilización de diseños probados que resuelven problemas recurrentes en el proceso de desarrollo. Se destacan patrones como *Singleton*, *Factory Method*, y *Observer*, que permiten una mayor flexibilidad y adaptabilidad en los sistemas. Por otro lado, las **metodologías ágiles** como *Scrum* y *Extreme Programming* han ganado prominencia debido a su capacidad para promover la entrega continua y la adaptabilidad a los cambios de los requisitos. Estas metodologías favorecen la colaboración entre los equipos de desarrollo y los usuarios, garantizando que el software cumpla con los estándares de calidad y usabilidad.

## 6. Conceptos y Teorías

Las **teorías de la arquitectura de software** están basadas en enfoques que buscan optimizar la estructura y organización del software para que sea eficiente y fácil de mantener. Una de las teorías clave es la de la *arquitectura basada en componentes*, que promueve la reutilización de piezas de software autónomas e interconectadas.

El concepto de **modularidad** es central, ya que permite que las arquitecturas sean flexibles, escalables y adaptables a cambios. Según *Brooks* (1975), una buena modularidad facilita el mantenimiento de sistemas complejos, lo que permite cambios o extensiones sin afectar el funcionamiento de otras partes del sistema.

*La teoría de la evolución del software* (Boehm, 1988) establece que el software debe evolucionar y adaptarse a los cambios en los requisitos sin perder la calidad ni la funcionalidad original. Esta teoría resalta la importancia de un enfoque iterativo en el desarrollo, en el que se gestionen los riesgos y se adapten las soluciones rápidamente. Las **teorías de patrones de diseño** se centran en el uso de soluciones probadas para resolver problemas comunes en el diseño de software. Los **principios SOLID** (Responsabilidad Única, Abierto/Cerrado, Sustitución de Liskov, Segregación de Interfaces, y Inversión de Dependencias) son un conjunto de principios fundamentales que

ayudan a crear software robusto y flexible, y son una base importante en la implementación de patrones de diseño.

## 7. Estudios Previos

Varios estudios han demostrado la importancia de la integración de patrones de diseño en las metodologías ágiles. Según *Pahl, Huber y Kulesza* (2013), la aplicación de patrones de diseño en un entorno ágil permite mejorar la calidad del código y reducir el tiempo de desarrollo. Además, *Olsson y Barros* (2014) argumentan que la adopción de arquitecturas orientadas a servicios (SOA) puede aumentar la flexibilidad de los sistemas, permitiendo una mayor interoperabilidad y escalabilidad.

En cuanto a la usabilidad, *Schneiderman* (2010) subraya que el diseño centrado en el usuario es esencial para el éxito de cualquier aplicación, ya que la experiencia del usuario influye directamente en la satisfacción y la adopción de los sistemas.

## 8. Revisión de la Literatura

La literatura muestra un creciente interés por la integración de la usabilidad desde el diseño inicial del software. La arquitectura de software, definida como la organización de un sistema, es fundamental para asegurar un bajo acoplamiento entre componentes. Además, se evidencia la necesidad de adaptarse a cambios constantes en los requisitos del negocio.

## 9. Metodología

Se realizó una revisión sistemática de la literatura, recopilando artículos relevantes sobre patrones de diseño y arquitectura de software. Los artículos fueron clasificados según su enfoque y contribuciones al campo.

## 10. Resultados

A continuación, se presentan los resúmenes de los 34 artículos revisados:

1. **Patrones de usabilidad:** Mejora la usabilidad del software desde el momento arquitectónico, integrando ajustes continuos en el diseño [1].

2. **Arquitectura de software:** Describe la organización de un sistema, enfatizando un acoplamiento bajo entre componentes para facilitar el mantenimiento [2].

3. **Introducción a la Arquitectura de Software:** Explora cómo esta disciplina crítica se adapta a cambios y requisitos sociales y de negocio [3].

4. **Atributos de calidad:** Analiza cómo la arquitectura organiza el sistema, enfocándose en atributos como mantenibilidad y seguridad [4].

5. **Integración en metodologías ágiles:** Examina la relación entre arquitectura y metodologías ágiles, destacando la importancia de requisitos arquitectónicos [5].

6. **Patrones de Diseño GOF:** Estudia la aplicación de estos patrones en el desarrollo web en Colombia, subrayando la necesidad de capacitación [6].

7. **Análisis comparativo de Patrones de Diseño:** Compara cinco patrones, enfatizando su contexto específico de aplicación [7].

8. **Revisión sobre generadores de código:** Destaca su papel en la eficiencia del desarrollo de software, especialmente en aplicaciones web [8].

9. **Teoría para el diseño de software:** Propone que el diseño sea un proceso continuo, anticipando cambios y dividiendo en componentes manejables [9].

10. **Perfiles UML:** Examina cómo pueden mejorar la representación de patrones de diseño, facilitando su personalización [10].

11. **Lenguajes de Patrones:** Explica la importancia de estos lenguajes en la arquitectura de software, permitiendo soluciones reutilizables [11].

12. **Arquitectura para comunidades académicas:** Propuesta para facilitar el acceso a la educación mediante televisión digital interactiva [12].

13. **Herramienta tecnológica de costos:** Simula costos y programación de obras para estudiantes de Ingeniería Civil [17].

14. **Modelado y Verificación en la Nube:** Propuesta de herramienta para modelar arquitecturas en la nube, asegurando adherencia a mejores prácticas [26].

15. **Arquitectura guiada por el Dominio:** Enfoque que centra el desarrollo en el conocimiento del negocio, facilitando flexibilidad [**?**].

16. **Sistema de visualización médica Vismedic:** Mejora del sistema mediante una arquitectura por capas, optimizando extensibilidad [**?**].

17. **Desarrollo de Ïntranet Industrial":** Aplicación de patrones de diseño orientados a objetos para mejorar la eficiencia [17].

18. **Arquitectura en desarrollo ágil:** Propuesta de un modelo que integra la arquitectura con metodologías ágiles [18].

19. **Herramienta para aprender patrones:** Desarrollo de una herramienta web que facilita la comprensión de patrones de diseño [19].

20. **Aplicación de patrones para flexibilidad:** Muestra cómo los patrones permiten crear software adaptable a cambios en reglas de negocio [20].

21. **ReusMe:** Herramienta que facilita la reutilización de código JavaScript para mejorar la usabilidad [21].

22. **Arquitectura de Microservicios:** Propuesta para superar limitaciones de arquitecturas monolíticas, mejorando escalabilidad [22].

23. **Arquitectura del robot Lázaro:** Descripción de la arquitectura para gestionar sensores y actuadores, facilitando control [23].

24. **Arquitectura N-capas:** Mejora de la modularidad y reutilización mediante separación de responsabilidades en componentes [24].

25. **Módulo de recomendación para EGPat:** Herramienta para gestionar y recomendar patrones de diseño en recursos educativos [25].

26. **Arquitectura reutilizable:** Propuesta para desarrollar aplicaciones web rápidamente mediante la reutilización de patrones [26].

27. **Integración de objetos de aprendizaje:** Propuesta de arquitectura basada en servicios web para mejorar gestión de contenidos [27].

28. **Lineamientos para cajero automático:** Aborda patrones GRASP, asegurando alta cohesión y bajo acoplamiento [28].

29. **Arquitectura en proceso de desarrollo:** Importancia de la arquitectura en el ciclo de vida en espiral, integrando MDA [29].

30. **Modelo "4+1":** Divide la arquitectura en cinco vistas complementarias, facilitando el enfoque en aspectos específicos [30].

31. **Reingeniería de software:** Mejora sistemas heredados mediante ingeniería inversa y rediseño [31].

32. **APIs de IoT usando C++:** Explora la creación de APIs eficientes utilizando patrones de diseño y metaprogramación [32].

33. **Framework web para aplicaciones:** Estandariza y optimiza la creación de aplicaciones dinámicas [33].

34. **Coffee Challenge:** Juego educativo que enseña patrones de diseño mediante simulaciones prácticas [34].

Cuadro 1: Relación entre Patrones de Diseño y Aspectos Clave en la Arquitectura de Software

| Patrón de Diseño | Ventajas para la Arquitectura | Casos de Uso |
|---|---|---|
| **Singleton** | Garantiza una única instancia global en sistemas críticos. | Gestión de configuraciones globales y acceso a recursos compartidos. |
| **Factory Method** | Fomenta la flexibilidad y desacoplamiento en la creación de objetos. | Aplicaciones con múltiples productos derivados o jerarquías. |
| **Observer** | Facilita la comunicación y sincronización entre componentes. | Sistemas de notificaciones o eventos en tiempo real. |
| **Adapter** | Mejora la interoperabilidad entre sistemas con interfaces incompatibles. | Integración de sistemas heredados con nuevas arquitecturas. |
| **Strategy** | Promueve la extensión y reutilización de algoritmos. | Aplicaciones con reglas de negocio o cálculos cambiantes. |
| **Decorator** | Añade funcionalidades dinámicamente sin modificar estructuras existentes. | Interfaces gráficas y extensiones modulares. |

# 11. Discusión

Los resultados sugieren que la arquitectura de microservicios ofrece ventajas significativas sobre las arquitecturas monolíticas, permitiendo una mejor escalabilidad y mantenimiento. Asimismo, el uso de metodologías ágiles y la integración de requisitos arquitectónicos son cruciales para el éxito del desarrollo de software.

# 12. Conclusiones

La revisión de los 34 artículos muestra que los patrones de diseño y la arquitectura de software son componentes esenciales para la creación de sistemas eficientes y adaptables. La implementación de enfoques modernos y herramientas adecuadas puede facilitar el desarrollo y mejorar la calidad del software.

# Referencias


[1] Moreno, A. M., & Sánchez-Segura, M. (2003, November). Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el Momento Arquitectónico. In JISBD (pp. 117-126).

[2] Romero, P. Á. (2006). Arquitectura de software, esquemas y servicios. Ingeniería Industrial, 27(1), 1.

[3] Cristiá, M. (2008). Introducción a la Arquitectura de Software. Research-Gate. Recuperado de: `https://www.researchgate.net/publication/251932352`

[4] Bastarrica, M. C. (2005). Atributos de Calidad y Arquitectura del Software.

[5] Navarro, M. E., Moreno, M. P., Aranda, J., Parra, L., Rueda, J. R., & Pantano, J. C. (2017, September). Integración de arquitectura de software en el ciclo de vida de las metodologías ágiles. In XIX Workshop de Investigadores en Ciencias de la Computación (WICC 2017, ITBA, Buenos Aires).

[6] Guerrero, C. A., Suárez, J. M., & Gutiérrez, L. E. (2013). Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo

de Aplicaciones Orientadas a la Web. Información tecnológica, 24(3), 103-114.

[7] Alvarez, O. D. G., Larrea, N. P. L., & Valencia, M. V. R. (2022). Análisis comparativo de Patrones de Diseño de Software. Polo del Conocimiento: Revista científico-profesional, 7(7), 2146-2165.

[8] Casas, M. R. H. (2020). Revisión sistemática sobre generadores de código fuente y patrones de arquitectura (Master's thesis, Pontificia Universidad Catolica del Peru).

[9] Cristiá, M. (2021). Una Teoría para el Diseño de Software.

[10] Garis, A. G., Riesco, D. E., & Montejano, G. A. (2006). Perfiles UML para definición de Patrones de Diseño. In VIII Workshop de Investigadores en Ciencias de la Computación.

[11] Jimenez-Torres, V. H., Tello-Borja, W., & Rios-Patiño, J. I. (2014). Lenguajes de patrones de arquitectura de software: una aproximación al estado del arte. Scientia et technica, 19(4), 371-376.

[12] Campo, W. Y., Chanchí, G. E., & Arciniegas, J. L. (2013). Arquitectura de software para el soporte de comunidades académicas virtuales en ambientes de televisión digital interactiva. Formación universitaria, 6(2), 03-14.

[13] Cárdenas-Gutiérrez, J. A., Barrientos-Monsalve, E. J., & Molina-Salazar, L. (2022). Arquitectura de software para el desarrollo de herramienta Tecnológica de Costos, Presupuestos y Programación de obra. I+ D Revista de Investigaciones, 17(1), 89-100.

[14] Blas, M. J., Leone, H. P., & Gonnet, S. M. (2019). Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube.

[15] Cambarieri, M., Difabio, F., & García Martínez, N. (2020). Implementación de una arquitectura de software guiada por el dominio. In XXI Simposio Argentino de Ingeniería de Software (ASSE 2020)-JAIIO 49 (Modalidad virtual).

[16] Rodríguez Peña, A. D., & Silva Rojas, L. G. (2016). Arquitectura de software para el sistema de visualización médica Vismedic. Revista Cubana de Informática Médica, 8(1), 75-86.

[17] Lazo, L., & Paul, J. (2004). Desarrollo de sistemas de software con patrones de diseño orientado a objetos.

[18] Navarro, M. E., Moreno, M. P., Aranda, J., Parra, L., & Rueda, J. R. (2018). Arquitectura de software en el proceso de desarrollo ágil: una perspectiva basada en requisitos significantes para la arquitectura. In XX Workshop de Investigadores en Ciencias de la Computación (WICC 2018, Universidad Nacional del Nordeste).

[19] Ferrandis Homsi, A. (2021). Desarrollo de una herramienta para el aprendizaje de patrones de diseño software (Doctoral dissertation, Universitat Politècnica de València).

[20] Escalante, L. C. (2014). Aplicación de patrones de diseño para garantizar alta flexibilidad en el software. Tecnología y Desarrollo (Trujillo), 12(1), 77-82.

[21] Benigni, G., Antonelli, O., & Vásquez, Y. (2009). TOOL FOR REUSE OF JAVASCRIPT CODE ORIENTED TO INTERACTION PATTERNS. SABER. Multidisciplinary Journal of the Research Council of the University of the East, 21(1), 60-69.

[22] López, D., & Maya, E. (2017). Arquitectura de Software basada en Microservicios para Desarrollo de Aplicaciones Web.

[23] García, J. M., Gil, Á. E., & Sánchez, E. A. (2018). Development of a software architecture for the Lázaro mobile robot. Ingeniare. Chilean Journal of Engineering, 26(3), 376-390.

[24] Cardacci, D. G. (2015). Arquitectura de software académica para la comprensión del desarrollo de

software en capas (No. 574). Serie Documentos de Trabajo.

[25] Alfonso Azcuy, R., & Llull Céspedes, L. Á. (2021). Módulo de recomendación de patrones de diseño para EGPat. Revista Cubana de Ciencias Informáticas, 15(2), 118-137.

[26] Murillo, M. M. (2019). Arquitectura software reutilizable basada en patrones de diseño y patrones de interacción para el desarrollo rápido de aplicaciones web.

[27] Rojas, M., & Montilva, J. (2011). Una arquitectura de software para la integración de objetos de aprendizaje basada en servicios web. In Ninth LACCEI Latin American and Caribbean Conference. Engineering for a Smart Planet, Innovation, Information Technology and Computational Tools for Sustainable Development.

[28] Ortega, G. A. V. (2021). Lineamientos para el diseño de aplicaciones web soportados en patrones GRASP. Ciencia e Ingeniería: Revista de investigación interdisciplinar en biodiversidad y desarrollo sostenible, ciencia, tecnología e innovación y procesos productivos industriales, 8(2), 4.

[29] Meaurio, V. S., & Schmieder, E. (2013). La arquitectura de software en el proceso de desarrollo: integrando MDA al ciclo de vida en espiral. Archivo de la Revista Latinoamericana de Ingeniería de Software, 1(4), 142-146.

[30] Kruchten, P. (1995). Planos Arquitectónicos: El Modelo de 4+1 Vistas de la Arquitectura del Software. IEEE software, 12(6), 42-50.

[31] Contreras, T. B. Introducción a la Reingeniería de Software Mediante Patrones de Diseño.

[32] Pacheco, A., Escobar, J., & Trujillo, E. Uso de patrones de diseño y metaprogramación para construir APIs de IoT usando C++.

[33] Villalobos, G. M., Sánchez, G. D. C., & Gutiérrez, D. A. B. (2010). Diseño de framework web para el desarrollo dinámico de aplicaciones. Scientia et technica, 16(44), 178-183.

[34] Castaño, L. E. G., Soto, S. V. M., & Maturana, G. V. Coffee Challenge: un juego para el aprendizaje de patrones de diseño de software.

# 13. Agradecimientos

Deseo expresar mi más profundo y sincero agradecimiento a los instructores Jesús Ariel Bonilla, Carlos Julio Cadena Sarasty, Jhon Willian Corredor Araujo y Carlos Fabián Martínez Mora, por su invaluable apoyo y orientación en la revisión de los artículos. Su dedicación, compromiso y conocimientos fueron esenciales para el éxito de este trabajo, y me han proporcionado una guía constante que ha enriquecido enormemente mi aprendizaje y desarrollo.