Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web

Resumen

Este articulo analiza cómo se usan los *Patrones de Diseño* del famoso grupo de programadores "The Gang of Four" en proyectos de desarrollo web. Primero, se establecen unos criterios para evaluar procesos de desarrollo grandes y formales. Después, se toma una muestra específica y se revisa el código fuente para ver en qué casos se usan estos patrones. La idea es ver qué patrones usan los expertos en ingeniería de software. Al final, se concluye que los patrones sí se aplican en el mundo laboral, pero no mucho, ya sea porque la gente no sabe de su existencia o porque no tienen la experiencia para usarlos bien.

Reflexión

Este artículo nos lleva a reflexionar, que en el desarrollo de software tenemos herramientas valiosas, como los patrones de diseño de The Gang of Four, que muchos desarrolladores no aprovechan por desconocimiento o falta de práctica. Estos patrones no solo organizan el código, sino que facilitan el desarrollo y mantenimiento. La lección es clara: entender y aplicar estas herramientas nos ayuda a trabajar con mayor calidad y profesionalismo, elevando nuestros proyectos y nuestras habilidades.

Bibliografía

Carlos A. Guerrero, Johanna M. Suárez\* y Luz E. Gutiérrez 2018 <https://www.scielo.cl/scielo.php?pid=S071807642013000300012&script=sci_arttext&tlng=en>

Análisis comparativo de patrones de diseño de software para el desarrollo de aplicaciones móviles de calidad: Una revisión sistemática de la literatura

Resumen

Este artículo revisa investigaciones sobre patrones de diseño de software para hacer apps móviles de buena calidad. Primero, buscaron en tres bases de datos (IEEE, Explorer y EBSCO) y encontraron 3072 artículos. Después, seleccionaron 16 que realmente aportaban al tema. Usaron la estrategia PICO para hacer la búsqueda y seleccionar los artículos más relevantes. Al final, descubrieron que hay cinco patrones de diseño principales para apps móviles de calidad y ciertos criterios para identificarlos. Estos criterios pueden usarse como herramienta para comparar patrones y elegir el mejor para cada proyecto.

Reflexión

Este artículo nos hace reflexionar sobre este tipo de estudios nos provee herramientas para elegir los patrones de diseño más convenientes en función de las necesidades de cada proyecto. Con una selección fundamentada, los desarrolladores pueden optimizar la calidad de las aplicaciones móviles, alineando cada diseño con los objetivos específicos del software y, al mismo tiempo, haciendo que el desarrollo sea más sólido y eficiente.

Bibliografía

Abanto Cruz, Jesús Alberto Gonzales Ramírez, Omar Fernando 2019

<http://repositorio.upeu.edu.pe/handle/20.500.12840/2525>

Modelado y verificación de patrones de diseño de arquitectura de software para entornos de computación en la nube

Resumen

Este articulo presenta una herramienta completa para diseñar la arquitectura de aplicaciones web. La herramienta simplifica los problemas más comunes en el diseño, ofreciendo módulos que ayudan a los arquitectos a crear diseños de calidad. Funciona a partir de un modelo que identifica los elementos clave usados en este tipo de arquitecturas. Además, incluye una herramienta gráfica para personalizar diseños y verificar que se apliquen correctamente los patrones de diseño, asegurando que todo esté bien hecho desde el principio.

Reflexión

Este articulo nos hace reflexionar sobre como contamos con un entorno de diseño que simplifique y ordene el trabajo no solo ayuda a crear aplicaciones web más robustas, sino que también asegura que el diseño esté bien hecho desde el principio. Esto ahorra tiempo, esfuerzo y reduce errores, aportando a una mejor calidad en el desarrollo de software.

Bibliografia

Blas, María JuliaIcon ; Leone, Horacio PascualIcon ; Gonnet, Silvio MiguelIcon 2019

<https://ri.conicet.gov.ar/handle/11336/125130>

Análisis de secuencias discretas para la detección de patrones de diseño de software

Resumen

Los patrones de diseño de software ayudan a reutilizar lo que ya se ha aprendido en proyectos anteriores. Pero los diseñadores que no tienen mucha experiencia tienen que pasar mucho tiempo leyendo listas de patrones para aprender, lo que hace que no practiquen tanto. Para solucionar esto, el artículo propone usar Modelos de Markov de Orden Variable. Básicamente, estos modelos toman un conjunto limitado de actividades que hace un diseñador y una base de conocimientos previa para que una herramienta de interfaz pueda darle consejos útiles al usuario mientras está diseñando software. Así, los diseñadores pueden aprender y mejorar mientras trabajan, sin tener que depender tanto de leer manuales.

Reflexión

Esta herramienta tiene el potencial de mejorar la curva de aprendizaje para los diseñadores novatos. Al ofrecerles una guía basada en patrones y experiencia previa, pueden avanzar más rápido y de manera más efectiva, aprendiendo a diseñar con calidad sin depender tanto de la teoría y acercándose más a la práctica.

Bibliografía

Silva Logroño, Juan Francisco | Berdun, Luis S. | Armentano, Marcelo G. | Amandi, Analía

2010

<http://sedici.unlp.edu.ar/handle/10915/152663>

Arquitectura de software académica para la comprensión del desarrollo de software en capas

Resumen

El desarrollo de software requiere tener en cuenta muchos aspectos, como el acceso a datos, la seguridad, las interfaces y el control de transacciones. Para lograr un diseño que cumpla con todos estos requisitos, sea flexible y fácil de mantener, es importante pensar en una arquitectura bien estructurada. Un enfoque popular es el diseño en capas, donde el software se divide en grupos que representan partes físicas del sistema. Cada capa es independiente, pero se conecta bien con las demás. Esto permite que, si se hace un cambio en una parte, el impacto sea mínimo en el resto del sistema. Este trabajo presenta un diseño en capas que es fácil de entender y sirve como base para construir arquitecturas más complejas en el futuro.

Reflexión

Una arquitectura en capas no solo facilita el mantenimiento y la flexibilidad del sistema, sino que también establece una base sólida para crear diseños más complejos. Este enfoque ayuda a los desarrolladores a comprender mejor el sistema y a crear software que sea estable, adaptable y fácil de evolucionar según las necesidades del proyecto.

Bibliografía

Cardacci, Darío G 2015

https://hdl.handle.net/10419/130825

Modelo Teórico para la Identificacion del Antipatron  
"Stovepipe System " en la Etapa de la Implementacion de una Arquitectura de Software

Resumen

Los anti patrones son soluciones que, en lugar de resolver problemas, terminan generando más complicaciones. Aunque una arquitectura de software debería facilitar el trabajo en equipo y guiar el desarrollo, a veces se elige una solución que no da los resultados esperados, lo que puede retrasar o hacer fracasar un proyecto. Esto es aún peor en instituciones grandes, como las del Estado en Bolivia, donde los sistemas se extienden por diferentes regiones y es crucial que todo funcione en conjunto.

Un problema común en este tipo de instituciones es el anti patrón llamado STOVEPIPE SYSTEM, que aparece cuando los componentes de un sistema son tan complejos y desorganizados que no se pueden comunicar entre sí. Para resolver este problema, se propone crear un modelo teórico durante la implementación que ayude a identificar el STOVEPIPE SYSTEM y a aplicar soluciones prácticas usando redes neuronales para tomar decisiones.

Reflexión

La problemática del anti patrón *STOVEPIPE SYSTEM* refleja la importancia de construir arquitecturas de software bien integradas, especialmente en instituciones grandes y dispersas. Cuando los sistemas no logran comunicarse entre sí, la falta de coordinación afecta la eficiencia y puede llevar a que proyectos importantes fracasen. La propuesta de usar redes neuronales para identificar este problema invita a los arquitectos de software a pensar en soluciones sostenibles, enfocadas en la integración y adaptabilidad a largo plazo, más allá de soluciones rápidas que puedan traer complicaciones futuras.

Bibliografía

Jakeline Cristina Candia Peñaloza 2014

http://www.revistasbolivianas.ciencia.bo/scielo.php?pid=S333377772014000100023&script=sci\_arttext&tlng=es

Marco de trabajo para la selección de la arquitectura de un proyecto de software mediante la aplicación de patrones arquitectónicos

Resumen

Analiza problemas comunes en el diseño de software que afectan la calidad del producto final, como la falta de integración entre la lógica de negocio, la interfaz, la navegación y la estructura de la información. Para abordar esto, se proponen varios patrones de diseño que ofrecen soluciones prácticas en diferentes áreas, desde la estructura de información hasta la navegación y visualización.

A través de una revisión de literatura y encuestas a desarrolladores, se identificaron los patrones más populares hoy en día. De los encuestados, el 40% recomendó la arquitectura en la nube como la mejor para proyectos de software, seguida del patrón MVC (26,7%) y los microservicios (13,3%). También se destacó que patrones como MVP y MVC son ideales para dispositivos móviles y sitios web por su flexibilidad y facilidad de uso, mientras que la arquitectura en la nube y los microservicios son más útiles en aplicaciones y servicios web, simplificando el diseño y reduciendo los riesgos de fallos en el proyecto.

Reflexión

A la hora de desarrollar software, elegir el patrón de diseño adecuado es clave para evitar problemas y asegurar que el producto final sea de calidad. Al integrar bien la lógica, la interfaz y la navegación, se logra un sistema más eficiente y fácil de mantener. Los resultados de la encuesta muestran que, hoy en día, las arquitecturas como la nube, MVC y microservicios son las más recomendadas, dependiendo del tipo de proyecto. Esto nos enseña que no hay una solución única, sino que cada proyecto requiere su propio enfoque, y saber elegir la arquitectura adecuada puede reducir riesgos y facilitar el trabajo a largo plazo.

Bibliografía

[Garzón Gil, Kelly Johana](https://repositorio.tdea.edu.co/browse?type=author&value=Garz%C3%B3n%20Gil,%20Kelly%20Johana) 2020

<https://dspace.tdea.edu.co/handle/tdea/923>

Un Marco de Trabajo para la Integración de Arquitecturas de Software con Metodologías Ágiles de Desarrollo

Resumen

Este articulo propone un marco de trabajo que combina una arquitectura en capas con una técnica de programación enfocada en las pruebas de unidad, dentro de un enfoque ágil de desarrollo de software, como *eXtreme Programming*. Esta propuesta busca asegurar un código de calidad desde el inicio, utilizando procesos sencillos y livianos, pero efectivos. Se integran metodologías ágiles con técnicas probadas, como las pruebas de unidad, que ayudan a mejorar la calidad del software de manera continua. El marco propuesto se valida a través de un caso de estudio, mostrando su efectividad en la práctica.

Reflexión

El uso de metodologías ágiles, como *eXtreme Programming*, permite que el desarrollo de software sea más flexible y rápido, asegurando al mismo tiempo que el código sea de calidad desde el principio. La integración de pruebas de unidad desde las primeras etapas ayuda a detectar problemas antes, lo que ahorra tiempo y esfuerzo a largo plazo. La propuesta de usar una arquitectura en capas le da estructura al proyecto, lo que facilita el manejo de las diferentes partes del software. Al final, este enfoque demuestra que combinar lo mejor de las metodologías ágiles con técnicas probadas mejora la eficiencia y la calidad del desarrollo.

Bibliografía

[Vivas, Héctor Luis](https://rid.unrn.edu.ar/browse?type=author&value=Vivas%2C+H%C3%A9ctor+Luis) [Cambarieri, Mauro Germán](https://rid.unrn.edu.ar/browse?type=author&value=Cambarieri%2C+Mauro+Germ%C3%A1n) [García Martínez, Nicolás](https://rid.unrn.edu.ar/browse?type=author&value=Garc%C3%ADa+Mart%C3%ADnez%2C+Nicol%C3%A1s) [Muñoz Abbate, Horacio](https://rid.unrn.edu.ar/browse?type=author&value=Mu%C3%B1oz+Abbate%2C+Horacio) [Petroff, Marcelo](https://rid.unrn.edu.ar/browse?type=author&value=Petroff%2C+Marcelo) 2013

<http://hdl.handle.net/20.500.12049/146>

Patrones de Diseño, Refactorización y Antipatrones. Ventajas y Desventajas de su Utilización en el Software Orientado a Objetos

Resumen

Este artículo tiene como objetivo explicar los conceptos de Patrón de Diseño, Refactorización y Antipatrón, y cómo afectan el desarrollo de software orientado a objetos. No se enfoca en detalles técnicos de la programación orientada a objetos como herencia o polimorfismo, sino en cómo usar estos conceptos para mejorar la calidad del código y evitar soluciones que puedan generar más problemas de los que resuelven.

Reflexión

Entender los patrones de diseño y los antipatrón es clave para mejorar cómo desarrollamos software. Los patrones de diseño nos ayudan a aplicar soluciones probadas, mientras que la refactorización permite mejorar el código sin cambiar su funcionalidad. Por otro lado, los antipatrón nos advierten sobre prácticas que, aunque parecen buenas al principio, terminan complicando el proyecto. Usarlos correctamente puede hacer una gran diferencia en la calidad del software y en la facilidad para mantenerlo a largo plazo.

Bibliografía

Gustavo Damián Campo 2019

https://revistas.ucasal.edu.ar/index.php/CI/article/view/178

Aplicación de patrones de diseño para garantizar alta flexibilidad en el software

Resumen

Este artículo explica cómo las aplicaciones empresariales, por su complejidad y la constante variabilidad de sus reglas de negocio, pueden beneficiarse del uso de patrones de diseño. A través de un ejemplo práctico, se muestra cómo los patrones como el patrón estrategia, compuesto y fábrica ayudan a crear aplicaciones más flexibles frente a cambios. Además, se recomienda usar un framework que facilite la creación de objetos y la inyección de dependencias, mejorando la flexibilidad y la mantenibilidad del software.

Reflexión

Las aplicaciones empresariales suelen enfrentarse a constantes cambios en las reglas de negocio, lo que puede hacerlas difíciles de manejar. Usar patrones de diseño como estrategia, compuesto y fábrica ayuda a que estas aplicaciones sean más flexibles y fáciles de modificar sin romper su estructura. Además, recurrir a un framework para manejar la creación de objetos y dependencias facilita aún más el proceso, permitiendo que el software se adapte rápidamente a nuevas necesidades sin complicarse demasiado.

Bibliografía

Lain Cárdenas Escalante 2014

<https://doi.org/10.18050/td.v12i1.696>

Una metodología para el desarrollo de patrones de diseño para objetos de aprendizaje constructivista colaborativo

Resumen

El articulo propone una manera práctica de crear patrones para diseñar recursos educativos digitales, basándose en el trabajo en equipo. Todo gira en torno a que los estudiantes sean el centro del aprendizaje, usando ideas del constructivismo, que es básicamente aprender haciendo y entendiendo por cuenta propia. Además, sugiere que el equipo que haga estos recursos tenga guías claras para que el proceso sea más fácil y organizado.

Reflexión

Este enfoque es súper útil porque pone a los estudiantes como los verdaderos protagonistas, lo que los motiva más a aprender. También resalta la importancia del trabajo en equipo, porque crear buenos recursos educativos no es cosa de una sola persona, sino de muchas ideas y habilidades juntas. Es como un recordatorio de que la educación funciona mejor cuando es colaborativa y bien planeada. En el fondo, esto nos muestra que aprender no es solo recibir información, sino participar activamente y construir conocimiento.

Bibliografía

[Calvo, Patricia](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/67118) 2015

<http://sedici.unlp.edu.ar/handle/10915/47765>

Impacto del uso de patrones de diseño en la industria del software en Costa Rica

Resumen

Este articulo habla sobre la importancia de los estándares y patrones de diseño en el desarrollo de software. Aunque las empresas usan estas herramientas para garantizar calidad, también es clave saber si están actualizadas con las tecnologías más nuevas. Esto no solo afecta a los productos que hacen, sino también a los egresados universitarios, que necesitan estar preparados para enfrentar el mercado laboral. Usar patrones de diseño es una señal de un buen ingeniero, porque implica trabajar con eficiencia, calidad y buenas prácticas. El artículo se centra en analizar cómo se usan estos patrones en la industria de software en Costa Rica: qué tan comunes son, qué tipos se prefieren y qué tan importantes resultan.

Reflexión

Es muy importante que las empresas y las universidades vayan al ritmo de las tecnologías actuales. Si los ingenieros recién graduados no saben usar las herramientas y metodologías que las empresas necesitan, será difícil que se adapten rápidamente al mercado laboral. Por otro lado, usar patrones de diseño es como tener un sello de calidad: no solo ayuda a hacer software más organizado y eficiente, sino que también mejora la reputación del ingeniero y de la empresa. Este tema nos recuerda que la tecnología no es estática, y tanto la educación como la industria deben trabajar juntas para mantenerse actualizadas y competitivas.

Bibliografía

Francisco Picado Corao, Mariana Pérez Vanegas 2019

https://revistas.ulatina.ac.cr/index.php/tecnologiavital/article/view/249

Patrón de diseño Strategy

Resumen

El patrón **Strategy** es como una caja de herramientas con varios algoritmos que funcionan bajo la misma estructura. Lo genial es que puedes cambiar de herramienta (algoritmo) cuando quieras, incluso mientras el programa está funcionando. Además, este patrón organiza todo de manera que cualquier parte del programa pueda usar esos algoritmos fácilmente, sin depender de detalles específicos.

Reflexión

El patrón Strategy es una solución súper útil para hacer programas más flexibles y fáciles de manejar. En lugar de amarrar todo a un solo método, te da la libertad de probar diferentes enfoques sin complicar el código. Es como tener un plan B o C o D, siempre listo para que todo fluya mejor. Al final, esto se traduce en un software más ordenado, eficiente y adaptable, que es justo lo que cualquier proyecto necesita para evolucionar con el tiempo.

Bibliografía

[Rodríguez Portela, Arles Ernesto](https://repositorio.konradlorenz.edu.co/browse/author?value=Rodr%C3%ADguez%20Portela,%20Arles%20Ernesto) 2019

<https://repositorio.konradlorenz.edu.co/handle/001/1535>

Construyendo software innovador: Implementación de patrones de diseño creacionales

Resumen

La Programación Orientada a Objetos (POO) utiliza objetos para modelar el mundo real y crear software fácil de mantener. Los patrones de diseño, surgidos en los años 90, son soluciones probadas para problemas comunes en el desarrollo de software, ayudando a organizar programas complejos. Este libro se centra en patrones creacionales, ofreciendo ejemplos prácticos para aplicarlos y mejorar la calidad del código.

Reflexión

La POO y los patrones de diseño son herramientas clave para escribir código eficiente y mantener proyectos organizados. Aplicarlos no solo mejora los programas, sino también las habilidades del desarrollador.

Bibliografía

Patricio Celi Vivanco 2023

https://itq.edu.ec/wp-content/uploads/2023/12/2023-12-06\_construyendo\_software\_innovador.pdf

Incidencia de los patrones de diseño de software en la seguridad de aplicaciones web

Resumen

Los patrones de diseño son como trucos ya probados para organizar y estructurar el código de manera eficiente. Esta investigación analiza cómo estos patrones pueden ayudar a hacer que las aplicaciones web sean más seguras frente a amenazas comunes. Usar patrones bien pensados facilita tareas clave como validar datos, manejar usuarios y evitar errores graves. Al final, esto reduce los riesgos de ataques y los costos de mantenimiento del software, porque desde el inicio se trabaja con estructuras claras y seguras.

Reflexión

Los patrones de diseño no solo sirven para que el código se vea más ordenado, sino también para blindar las aplicaciones contra problemas de seguridad. Para los desarrolladores, usar estas herramientas no es un lujo, es una necesidad. Si construyes bien desde el principio, no solo ahorras tiempo y dinero, también proteges la confianza de los usuarios, que confían su información a estas aplicaciones.

Bibliografía

Mesías-Valencia, J. J., & Cevallos-Muñoz, F. D. (2024). Incidencia de los patrones de diseño de software en la seguridad de aplicaciones web. *MQRInvestigar*, *8*(1), 236–259. https://doi.org/10.56048/MQR20225.8.1.2024.236-259

Análisis del impacto del uso de patrones de diseño en la fase de mantenimiento

Resumen

Este proyecto busca medir cómo el uso de patrones de diseño afecta la fase de mantenimiento en proyectos de software. A través de encuestas, se evaluó qué tan eficiente y cuánto esfuerzo requiere el mantenimiento cuando se aplican estas técnicas. Además, se plantea crear un software que evalúe buenas prácticas en el desarrollo, usando un modelo arquitectónico basado en **Archimate**.

Reflexión

Aplicar patrones de diseño no solo organiza el código, sino que también reduce el esfuerzo al hacer cambios o correcciones en el futuro. Este estudio resalta cómo las buenas prácticas no son solo teoría, sino herramientas reales que facilitan el trabajo. La idea de un software para evaluar estas prácticas es un paso clave para garantizar que los proyectos sigan estándares de calidad desde el principio.

Bibliografía

Martha Janeth de León Correa , Ivon Rosmery Hoyos Beltrán, Fabio Enrique Quintero Díaz-Granados 2017

https://revistas.udistrital.edu.co/index.php/tia/article/view/8811

Implementación de un modelo alternativo de arquitectura de software para proyectar y construir sistemas lógicos de computadoras.

Resumen

Este articulo implementó un modelo alternativo de arquitectura de software en la fundación "San Luis" de Pronaca, en Bucay, Guayas. La investigación abarcó cuatro etapas clave: recopilar información de usuarios y desarrolladores, analizar arquitecturas de referencia, implementar una nueva arquitectura con herramientas .NET y evaluarla con un software administrativo automatizado. Los resultados mostraron mejoras significativas en funcionalidad (88%) y mantenibilidad (75%), logrando un aumento general de eficiencia del 82%. Se concluye que una arquitectura bien diseñada mejora la calidad, escalabilidad y competitividad de los sistemas informáticos.

Reflexión

Una buena arquitectura de software no solo hace que los sistemas sean más fáciles de mantener y escalar, sino que también impulsa la eficiencia y la capacidad de adaptarse a cambios futuros. Este estudio demuestra que invertir en estructuras sólidas desde el inicio beneficia tanto a los desarrolladores como a las empresas, ofreciendo soluciones más estratégicas y productos de calidad.

Bibliografía

Avalos Pérez, Miguel Angel. (2012). Implementación de un modelo alternativo de arquitectura de software para proyectar y construir sistemas lógicos de computadoras. Escuela Superior Politécnica de Chimborazo. Riobamba.

Diseño de un prototipo de arquitectura de software integral que permita gestionar y controlar eficazmente las etapas del ciclo de desarrollo de software en un entorno empresarial

Resumen

En el mundo empresarial actual, el desarrollo de software es clave para competir, pero gestionar todo el ciclo de desarrollo puede ser un gran desafío. Muchas empresas carecen de herramientas que permitan un seguimiento claro y preciso del progreso de los proyectos, lo que puede generar incertidumbre, malas decisiones y clientes insatisfechos. Este problema ha llevado a la propuesta de un prototipo de arquitectura de software integral que facilite un control efectivo y transparente en cada etapa del desarrollo, abordando así estas dificultades.

Reflexión

Controlar el avance de los proyectos de software no es solo una cuestión técnica, sino estratégica. Una arquitectura de software bien diseñada puede dar claridad al proceso, mejorar la toma de decisiones y aumentar la confianza de los clientes. En un mercado tan dinámico, tener herramientas que mantengan todo bajo control no es solo útil, es esencial para garantizar calidad, eficiencia y éxito a largo plazo.

Bibliografía

[Torres Reyes, Alvaro Javier](https://repository.udistrital.edu.co/browse/author?startsWith=Torres%20Reyes,%20Alvaro%20Javier), [Vargas Velásquez, Oscar Javier](https://repository.udistrital.edu.co/browse/author?startsWith=Vargas%20Vel%C3%A1squez,%20Oscar%20Javier) 2024

<http://hdl.handle.net/11349/41990>

Modelo de arquitectura de software para el procesamiento de datos en arquitecturas actuales

Resumen

Una arquitectura de software es clave para el éxito de cualquier aplicación, ya que una mala elección puede generar problemas de efectividad, eficiencia y medición. Al seleccionar la arquitectura, no solo se deben considerar las habilidades del equipo, sino también un análisis de las necesidades específicas del proyecto. El manejo de datos masivos es una de las áreas que pone a prueba muchas arquitecturas, especialmente en cuanto a rendimiento y control de la información. Este trabajo propone un prototipo arquitectónico basado en la lógica de negocio y las herramientas existentes, para abordar problemas de procesamiento de datos masivos y ayudar a las empresas a tomar mejores decisiones.

Reflexión

Elegir la arquitectura adecuada no es una decisión trivial; debe ser un proceso bien pensado y adaptado a las necesidades del proyecto, no solo a lo que el equipo pueda manejar fácilmente. Cuando se trata de datos masivos, la arquitectura tiene que ser capaz de manejar grandes volúmenes de información sin afectar el rendimiento. Este enfoque ofrece una guía útil para que las empresas tomen decisiones más informadas y encuentren la mejor solución para sus necesidades específicas.

Bibliografía

[Quilindo Celis, Luz Amanda](https://repository.udistrital.edu.co/browse/author?startsWith=Quilindo%20Celis,%20Luz%20Amanda), [Vega Guarín, Juan Sebastián](https://repository.udistrital.edu.co/browse/author?startsWith=Vega%20Guar%C3%ADn,%20Juan%20Sebasti%C3%A1n) 2021

<http://hdl.handle.net/11349/29557>

Patrones de adaptación para arquitecturas de software basadas en tecnologías del acuerdo

Resumen

La creciente complejidad de los sistemas de software requiere nuevas estrategias de desarrollo, y la auto-adaptación se vuelve crucial en la arquitectura. Los sistemas multiagentes son útiles para problemas complejos, pero necesitan mecanismos para adaptarse dinámicamente. Este artículo propone una arquitectura evolutiva que permita a los sistemas adaptarse mediante una estructura de controles y protocolos definidos, creando organizaciones adaptativas que evolucionan hacia una estructura estable.

Reflexión

La auto-adaptación es clave para que los sistemas software se mantengan eficaces frente a cambios. Este enfoque basado en agentes ofrece flexibilidad y evolución, lo que podría ser fundamental para manejar la complejidad de los sistemas en el futuro.

Bibliografía

[J. Santiago Pérez Sotelo](https://dialnet.unirioja.es/servlet/autor?codigo=3030979), [Carlos Enrique Cuesta Quintero](https://dialnet.unirioja.es/servlet/autor?codigo=3137966), [Sascha Ossowski](https://dialnet.unirioja.es/servlet/autor?codigo=1101178) 2011

<https://dialnet.unirioja.es/servlet/articulo?codigo=3856682>

Una metodología basada en patrones de diseño de software para una óptima implementación de mecánicas en videojuegos

Resumen

En los videojuegos, las mecánicas son como el "manual de instrucciones" del juego: las reglas que detallan qué puede hacer el jugador, cómo interactúa con el mundo del juego, cómo funcionan los enemigos, la economía, y todo lo demás. Para que todo esto funcione, el jugador necesita saber cómo usar los controles y qué pasará en el juego cuando haga algo. Los diseñadores y programadores trabajan juntos para asegurarse de que todo esté en orden. Tener unas mecánicas bien diseñadas es clave para que el juego sea divertido, pero no es lo único importante. También se usan patrones de diseño, que son herramientas súper útiles para ahorrar tiempo y recursos al desarrollar y ejecutar el juego. Este enfoque busca ayudar a los equipos de desarrollo a planear y crear mecánicas de forma eficiente.

Reflexión

Un juego divertido no solo depende de ideas creativas, sino también de cómo se planifica y ejecuta su diseño. Las mecánicas claras y bien pensadas son la base para que los jugadores se sientan conectados al juego, pero los patrones de diseño son el secreto para que el desarrollo sea más rápido y efectivo. Este equilibrio entre creatividad y organización técnica puede marcar la diferencia entre un juego promedio y uno excepcional.

Bibliografía

[Ruiz Ponce, Erick Blas](https://repositorioinstitucional.buap.mx/browse/author?startsWith=Ruiz%20Ponce,%20Erick%20Blas) 2023

<https://hdl.handle.net/20.500.12371/19755>

Modelo de toma de decisiones para facilitar la elección de patrones de diseño en una arquitectura orientada a servicios

Resumen

Hoy en día, hay muchos patrones de diseño para SOA que solucionan problemas comunes en proyectos de software, pero puede ser difícil evaluarlos todos. Este estudio identificó once problemas frecuentes en el diseño de SOA y analizó patrones para resolverlos. Como resultado, se creó una wiki con un modelo que facilita la toma de decisiones y organiza esta información para consulta fácil. La herramienta fue validada con profesionales y recibió comentarios positivos.

Reflexión

Elegir el patrón de diseño correcto no solo mejora la calidad del software, sino que también ahorra tiempo y recursos. Esta investigación es un gran aporte porque no solo organiza soluciones, sino que también guía a los profesionales para elegirlas de manera más racional y estratégica. La creación de una herramienta accesible como una wiki demuestra que, con la tecnología adecuada, se pueden simplificar procesos complejos y aumentar la eficiencia en proyectos de software.

Bibliografía

Cuervo Roa, María Alejandra, Chacón Torres, Leidy Jhoana 2015

<http://repository.unipiloto.edu.co/handle/20.500.12277/1135>

Utilización de patrones de diseño en la asignatura de Procesadores de Lenguaje

Resumen

La materia de Procesadores de Lenguaje trata sobre entender la teoría detrás de los lenguajes de programación y aprender a crear herramientas para procesarlos, como los compiladores. Una parte clave es el análisis semántico, que consiste en verificar que las reglas del lenguaje se cumplan. Para enseñar esto, se usa algo llamado gramáticas atribuidas y patrones de diseño, lo que hace que los estudiantes puedan conectar la teoría con la práctica. Además, usan diagramas UML, que ya conocen, para explicar conceptos complejos, y luego aplican todo esto desarrollando un procesador de lenguaje, lo que refuerza su aprendizaje práctico y teórico.

Reflexión

El enfoque que combina teoría y práctica es muy valioso porque permite que los estudiantes entiendan mejor conceptos complicados al verlos aplicados en algo real. Además, usar herramientas conocidas como UML ayuda a reducir la barrera de aprendizaje, haciendo que sea más fácil conectar los puntos. Esto no solo mejora el conocimiento técnico, sino que también desarrolla habilidades importantes para trabajar en proyectos de software en el mundo real.

Bibliografía

[Ortín Soler, Francisco](https://upcommons.upc.edu/browse?value=Ort%C3%ADn%20Soler,%20Francisco&type=author), [Zapico Rodríguez, Daniel](https://upcommons.upc.edu/browse?value=Zapico%20Rodr%C3%ADguez,%20Daniel&type=author), [Redondo López, José Manuel](https://upcommons.upc.edu/browse?value=Redondo%20L%C3%B3pez,%20Jos%C3%A9%20Manuel&type=author) 2009

<http://hdl.handle.net/2099/7895>

Pruebas de Regresión Funcional Mediante el Uso de Patrones de Diseño

Resumen

En el desarrollo de software, lo más importante es que el sistema cumpla con lo que se necesita. Si no lo hace o las necesidades cambian (por nuevas tecnologías o cambios en los negocios), se pierden tiempo y dinero. Por eso, las pruebas son clave: aseguran que todo funcione bien. Un método llamado "regresión funcional" evalúa si los requerimientos se cumplen en la arquitectura del software, pero es caro y difícil de automatizar. Esta propuesta busca automatizar ese proceso, usando una matriz de trazabilidad junto con patrones de diseño para generar casos de prueba de manera más rápida y confiable.

Reflexión

Automatizar las pruebas de software es como tener un asistente que se encarga del trabajo pesado: ahorra tiempo, dinero y errores. Este enfoque no solo facilita la vida a los desarrolladores, sino que también mejora la calidad del producto final. Al usar herramientas como la matriz de trazabilidad y patrones de diseño, se combina lo mejor de la organización y la creatividad para hacer que las pruebas sean más eficientes y útiles.

Bibliografía

Davila Nicanor, Leticia; Marín Guerrero, Omar; Aguilar Juárez, Irene; Ayala De La Vega, Joel 2015

<http://hdl.handle.net/20.500.11799/41174>

Framework de evaluación para modelos formales de patrones de diseño

Resumen

Los patrones de diseño son una gran herramienta para reutilizar ideas y anticiparse a cambios, pero muchas veces su descripción es tan vaga que no queda claro cómo aplicarlos bien. Usar lenguajes formales ayuda a describirlos de manera más precisa. Este trabajo analiza las limitaciones de los lenguajes informales y propone un Framework para evaluar lenguajes y modelos formales, basado en 14 características para los lenguajes y 10 aspectos para los modelos. El objetivo es entender mejor cómo representar patrones de diseño correctamente, facilitando que los diseñadores usen herramientas automáticas para crear diseños más claros y efectivos.

Reflexión

Tener descripciones claras y precisas de los patrones de diseño es clave para que sean realmente útiles. Este enfoque no solo ayuda a evitar errores o malentendidos, sino que también abre la puerta a herramientas automáticas que ahorran tiempo y mejoran el diseño. Esto demuestra que combinar teoría con práctica puede llevar a soluciones más sólidas y funcionales en el desarrollo de software.

Bibliografía

[Flores, Andrés P.](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/67401) | [Fillottrani, Pablo Rubén](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/52246) 2003

<http://sedici.unlp.edu.ar/handle/10915/22886>

Calidad Ágil: Patrones de Diseño en un contexto de Desarrollo Dirigido por Pruebas

Resumen

En el desarrollo ágil, especialmente con TDD (Desarrollo Dirigido por Pruebas), las pruebas unitarias son una práctica común, y JUnit es una herramienta muy usada para esto. Sin embargo, no todos los patrones y estilos de diseño pueden aplicarse fácilmente en este contexto porque deben cumplir ciertas condiciones para ser probados. Este trabajo analiza un pequeño grupo de patrones y estilos arquitectónicos para ver si pueden probarse en un entorno de pruebas automáticas y TDD. Usando un caso práctico, se evaluó su "testeabilidad" y los beneficios que aportan a la calidad del software.

Reflexión

Evaluar qué patrones y estilos son adecuados para pruebas en TDD es clave para hacer que el desarrollo sea más eficiente y el software, de mejor calidad. Este enfoque destaca la importancia de integrar diseño y pruebas desde el principio, asegurándose de que las herramientas y metodologías utilizadas realmente mejoren el proceso de desarrollo y los resultados finales.

Bibliografía

Manuel I, Capel, Anna C. Grimán, and Eladio Garví

<https://www.researchgate.net/profile/AnnaGriman/publication/308054417_Calidad_Agil_Patrones_de_Diseno_en_un_contexto_de_Desarrollo_Dirigido_por_Pruebas/links/57d83fb808ae601b39afa074/Calidad-Agil-Patrones-de-Diseno-en-uncontexto-de-Desarrollo-Dirigido-por-Pruebas.pdf>

Desarrollo de una herramienta para el aprendizaje de patrones de diseño software

Resumen

Este trabajo de fin de grado (TFG) tiene como objetivo crear una herramienta o aplicación web que ayude a aprender y practicar patrones de diseño orientados a objetos. La herramienta permitirá que los profesores configuren y personalicen contenidos que los estudiantes podrán usar para aprender a identificar y aplicar estos patrones. Será flexible y ampliable, permitiendo agregar nuevos patrones en el futuro. En su primera versión, solo se incluirán algunos patrones representativos del libro de Gamma, pero se podrá agregar más contenido según sea necesario.

Reflexión

Crear una herramienta interactiva para aprender patrones de diseño es una gran manera de facilitar la enseñanza práctica. Al permitir la configuración personalizada y la expansión, se asegura que la herramienta no solo sea útil en el presente, sino que también se adapte a las necesidades futuras de los estudiantes. Este enfoque permite una enseñanza más dinámica y acorde a las necesidades reales del aprendizaje de software.

Bibliografía

Ferrandis Homsi, A. (2021). Desarrollo de una herramienta para el aprendizaje de patrones de diseño software. Universitat Politècnica de València. 2017 <http://hdl.handle.net/10251/174122>

Especificando una arquitectura de software

Resumen

Esta investigación explora los diferentes componentes que forman parte de una arquitectura de software y cómo se relacionan mediante los patrones arquitecturales. A medida que la tecnología avanza, las plataformas móviles y web se expanden, lo que cambia la forma en que se diseñan las infraestructuras necesarias para que todo funcione. El software debe enfrentar tareas más complejas, tiempos de entrega más cortos, y asegurar la protección de los datos, todo mientras mantiene una arquitectura sólida. El artículo ofrece un vistazo a los componentes clave de la arquitectura de software actual, su propósito e importancia.

Reflexión

La arquitectura de software es la columna vertebral de cualquier sistema moderno, y con el crecimiento de la tecnología, se vuelve cada vez más crucial tener patrones arquitecturales bien definidos. Estos patrones no solo ayudan a organizar el trabajo, sino que también aseguran que el software pueda enfrentar desafíos complejos y seguir creciendo de manera eficiente. Es fascinante cómo estos patrones pueden hacer frente a las demandas actuales de rapidez, seguridad y escalabilidad.

Bibliografía

luz amanda quilindo 2024

<https://revistas.udistrital.edu.co/index.php/tia/article/view/18076>

Recuperación de Arquitecturas de Software: Un Mapeo Sistemático de la Literatura

Resumen

Este artículo habla de un estudio donde se revisaron un montón de investigaciones sobre cómo recuperar vistas arquitectónicas de sistemas de software, que básicamente son como mapas de cómo están hechos esos sistemas por dentro. Se enfocaron en ver qué tipos de estudios hay, para qué sirven, qué técnicas usan, qué partes del sistema recuperan y cómo muestran esos mapas. Al final, se dieron cuenta de que hace falta un método más claro y uniforme para presentar esos resultados, porque eso haría que todo fuera más fácil de entender, analizar y usar otra vez.

Reflexión

Lo que dicen en este artículo es súper importante, porque es como cuando armas un rompecabezas y necesitas que las piezas encajen bien para que todo tenga sentido. En el mundo del software, si no tienes una forma clara y consistente de entender cómo está hecho un sistema, es como intentar armar ese rompecabezas con piezas que no sabes dónde van. Tener un mecanismo unificado sería como tener una guía que te ayude a entender, arreglar o mejorar los sistemas sin perderte en el intento. Es un recordatorio de que, incluso en temas técnicos, la claridad y el orden son claves para avanzar mejor.

Bibliografía

Martín E. Monroy, José L. Arciniegas y Julio C. Rodríguez

<https://webcache.icesi.edu.co/revistas/index.php/sistemas_telematica/article/view/918/943>

Formalización de patrones de diseño y su identificación automática en software PLC para evaluación de arquitectura

Resumen

Con la automatización creciendo tanto, el software que controla estos sistemas se está complicando mucho. Por eso, es clave organizarlo mejor con módulos reutilizables. En otras áreas de programación ya usan patrones de diseño para resolver problemas comunes, pero en estos sistemas no es tan común. Este trabajo propone formas de clasificar esos patrones y probar cómo usarlos para hacer el software más claro, reutilizable y fácil de analizar, incluso con herramientas automáticas en la industria.

Reflexión

Esto muestra que no hay que complicarse más de lo necesario. Si ya existen formas efectivas de organizar el software en otros campos, ¿por qué no aplicarlas aquí? Usar patrones de diseño y automatizar su detección puede hacer que trabajar con estos sistemas sea más fácil, rápido y eficiente. Es como organizar un caos con herramientas inteligentes.

Bibliografía

Eva María Neumann [Birgit](https://www.sciencedirect.com/author/6603480302/birgit-vogel-heuser) Vogel [Heuser](https://www.sciencedirect.com/author/6603480302/birgit-vogel-heuser), Juliane Fischer, Félix Ocker, Sebastián Diehm, Michael Schwarz

<https://www.sciencedirect.com/science/article/pii/S2405896320324952>

Garantizar la mantenibilidad del software a nivel de arquitectura de software utilizando patrones arquitectónicos

Resumen

La arquitectura de software es como el esqueleto que sostiene todo un sistema. Si está bien hecha, mejora cosas importantes como que sea fácil de mantener. Una buena forma de lograrlo es usar patrones arquitectónicos que ya están probados. Pero ojo, usar cualquier patrón no garantiza que el software sea súper mantenible, y si eliges mal, puedes terminar con más problemas. Este artículo estudia cómo esos patrones afectan la capacidad de mantenimiento, propone un modelo para medir esa calidad y analiza dos casos reales para ver qué pasa cuando cambias la arquitectura de un sistema.

Reflexión

Esto deja claro que no basta con seguir recetas al pie de la letra; hay que saber cuál patrón arquitectónico usar según lo que necesites. Una mala decisión puede complicar todo en lugar de ayudar. Tener un modelo para comparar opciones y aprender de casos reales es clave para no cometer los mismos errores. Al final, se trata de construir software que no solo funcione bien hoy, sino que sea fácil de arreglar o mejorar mañana.

Bibliografía

[Zahed Rahmati](https://ajmc.aut.ac.ir/?_action=article&au=32952&_au=Zahed++Rahmati), [Mohamed Tanhaei](https://ajmc.aut.ac.ir/?_action=article&au=39895&_au=Mohammad++Tanhaei)

<https://ajmc.aut.ac.ir/article_4272.html>

Una herramienta para reutilizar estrategias en diseños de arquitecturas de software.

Resumen

Diseñar la arquitectura de software es un proceso súper creativo, pero no hay una forma estándar de hacerlo. Cada arquitecto usa lo que sabe y lo que le funciona según su experiencia. Reutilizar ideas de proyectos anteriores es común, pero no hay muchas herramientas que los ayuden con esto. Este trabajo presenta una herramienta que recopila experiencias pasadas, tanto propias como de otros, para que los arquitectos las puedan usar fácilmente al diseñar nuevos sistemas.

Reflexión

Esto muestra que, aunque el diseño de software sea creativo, no hay que empezar de cero cada vez. Tener una herramienta que te permita aprovechar lo que ya funcionó antes es como tener un manual con trucos probados. No solo ahorra tiempo, sino que también eleva la calidad del diseño al aprender de lo que ya se hizo bien (o mal) antes. Es un paso hacia un diseño más inteligente y colaborativo.

Bibliografía

[Carignano, María Celeste](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/65292) | [Gonnet, Silvio M.](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/59312) | [Leone, Horacio P.](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/53632)

<https://sedici.unlp.edu.ar/handle/10915/57164>

Captura del razonamiento y evolución de arquitecturas de software mediante la aplicación de operaciones arquitectónicas orientadas a objetivos

Resumen

Cuando se diseña la arquitectura de software, se toman muchas decisiones importantes, pero casi nunca se documentan porque es un trabajo extra que nadie quiere hacer. Este artículo propone un modelo que facilita guardar esas decisiones y por qué detrás de ellas, de forma automática y sin tanto esfuerzo. La idea es que el modelo permita registrar qué decisiones se tomaron, qué elementos del software se cambiaron y qué objetivos se querían alcanzar, todo de manera clara y útil.

Reflexión

Esto es como tener un diario de decisiones, pero sin tener que escribirlo a mano. Muchas veces, cuando algo sale mal en el software, nadie sabe por qué se decidió hacerlo así. Este modelo ayuda a mantener ese historial sin complicarle la vida a los diseñadores, lo que al final hace que el trabajo sea más claro, ordenado y fácil de revisar o mejorar. Es una forma de aprender del proceso sin perder tiempo extra.

Bibliografía

[Roldán, María Luciana](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=equals&filter=Rold%C3%A1n,%20Mar%C3%ADa%20Luciana) | [Gonnet, Silvio M.](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/59312) | [Leone, Horacio P.](https://sedici.unlp.edu.ar/discover?filtertype=author&filter_relational_operator=authority&filter=http://voc.sedici.unlp.edu.ar/node/53632)

<https://sedici.unlp.edu.ar/handle/10915/52407>

Captura de conocimientos de arquitectura de software para un diseño basado en patrones

Resumen

La arquitectura de software necesita mucho conocimiento, y una forma de organizarlo es usando patrones arquitectónicos. Pero elegir el patrón correcto es complicado porque la información sobre ellos está regada por todos lados en libros, artículos y demás.

Reflexión

Esto muestra que, aunque los patrones arquitectónicos son súper útiles, encontrar el correcto puede ser como buscar una aguja en un pajar. Tener esa información más accesible y bien organizada facilitaría mucho el trabajo de los arquitectos y mejoraría la calidad del software desde el principio.

Bibliografía

Siamak Farshidi, Honda Jansen, Jan Martijn van der Werf

<https://www.sciencedirect.com/science/article/pii/S0164121220301552>

Patrones de Usabilidad: Mejora de la Usabilidad del Software desde el  
momento de Arquitectónico

Resumen

La usabilidad es súper importante en el software, y este trabajo propone mejorarla desde el principio, no cuando ya está terminado. En lugar de solo medir qué tan fácil de usar es un sistema, identificaron 20 patrones de usabilidad, como poder deshacer acciones o usar varios idiomas, y ofrecen ideas de diseño para incluirlos desde el arranque. Esto se basa en investigaciones y experiencia de expertos, garantizando soluciones prácticas aunque no únicas.

Reflexión

Esto cambia la forma tradicional de hacer las cosas. Pensar en la usabilidad desde el diseño, y no como un parche al final, asegura que el software sea más amigable desde el inicio. Además, tener patrones claros facilita mucho el trabajo y da una base sólida para crear sistemas que realmente funcionen bien para los usuarios. Es una manera proactiva de hacer las cosas mejor.

Bibliografía

Ana M. Moreno, M. Isabel Sánchez

[https://www.researchgate.net/profile/Ana-Moreno 56/publication/221595496\_Patrones\_de\_Usabilidad\_Mejora\_de\_la\_Usabilidad\_del\_Software\_desde\_el\_Momento\_Arquitectonico/links/0deec51d6997037422000000/Patrones-de-Usabilidad-Mejora-de-la-Usabilidad-del-Software-desde-el-Momento-Arquitectonico.pdf](https://www.researchgate.net/profile/Ana-Moreno%2056/publication/221595496_Patrones_de_Usabilidad_Mejora_de_la_Usabilidad_del_Software_desde_el_Momento_Arquitectonico/links/0deec51d6997037422000000/Patrones-de-Usabilidad-Mejora-de-la-Usabilidad-del-Software-desde-el-Momento-Arquitectonico.pdf)

Clasificación de los patrones de diseño idóneos en programación Android

Resumen

En programación, la eficiencia es clave. Un código bien hecho no solo es más rápido, sino que también es más fácil de entender. En Android, hay muchos patrones de diseño para evitar el "código espagueti", que complica la vida tanto a los usuarios como a los desarrolladores. Este trabajo analiza y clasifica patrones de diseño para ayudar a los programadores novatos en Java a escribir código más limpio y eficiente según el tipo de programa que estén haciendo.

Reflexión

Esto es como enseñar a los nuevos programadores a no enredarse al escribir código. Usar patrones de diseño desde el principio no solo les facilita la vida a ellos, sino que también mejora los programas para los usuarios. Es un recordatorio de que aprender buenas prácticas desde temprano hace la diferencia entre un código funcional y uno no funcional.

Bibliografía

José Alejandro Ibarra Astorga, Jorge Luis Osuna Tirado, Ricardo Ulises Reyes Ramírez, José Carlos Lepe Mendoza, Alvaro Peraza Garzón

<https://redtis.org/index.php/Redtis/article/view/33>

Identificación de áreas de aplicación de arquitecturas de software basadas en modelos, técnicas y herramientas de social media

Resumen

Las redes sociales no solo sirven para interactuar, sino que también ofrecen mucha información sobre cómo se comportan los usuarios. Este estudio se centró en cómo se están usando esas herramientas y datos dentro de la arquitectura de software. Revisaron 30 artículos publicados entre 2015 y 2020, y descubrieron que el interés en usar estos datos para diseñar arquitecturas de software ha crecido un montón en los últimos años, con un aumento del 57% en las publicaciones sobre este tema.

Reflexión

Lo que muestra este estudio es que las redes sociales no solo están cambiando cómo nos conectamos, sino también cómo diseñamos software. Al incorporar el análisis de estos datos en las arquitecturas de software, se abren nuevas oportunidades para mejorar productos y servicios. Es un ejemplo de cómo los avances tecnológicos en un área pueden transformar otras.

Bibliografía

[Paola E. Velazquez-Solis](https://dialnet.unirioja.es/servlet/autor?codigo=5533957); [Brenda L. Flores-Rios](https://dialnet.unirioja.es/servlet/autor?codigo=4637874); [Jorge E. Ibarra-Esquer](https://dialnet.unirioja.es/servlet/autor?codigo=4637876); [María Angélica Astorga-Vargas](https://dialnet.unirioja.es/servlet/autor?codigo=5533958); [Raúl A. Aguilar Vera](https://dialnet.unirioja.es/servlet/autor?codigo=3968068); [Elizabeth Ramírez-Barreto](https://dialnet.unirioja.es/servlet/autor?codigo=5533959); [Gabriel López-Morteo](https://dialnet.unirioja.es/servlet/autor?codigo=5533960)

<https://dialnet.unirioja.es/servlet/articulo?codigo=8235940>

Representación de la arquitecturade software usando UML

Resumen

En los últimos años, mucha gente ha investigado cómo representar mejor las arquitecturas de software, buscando algo más claro que solo líneas y cajas, pero que no sea tan complicado. Los lenguajes de descripción de arquitecturas están ayudando a los ingenieros a representar sistemas de manera más adecuada, pero muchos de estos lenguajes son difíciles de usar o solo sirven para ciertos tipos de sistemas. Este artículo presenta una forma de representar arquitecturas de software usando UML, que es un lenguaje de modelado más fácil de adaptar a diferentes tipos de sistemas.

Reflexión

Este enfoque es como encontrar una forma de dibujar un mapa de algo complicado sin que se convierta en un rompecabezas difícil de entender. Usar UML para representar arquitecturas de software hace que sea más sencillo para todos los involucrados entender cómo está construido el sistema, sin perder la precisión. Al final, es un buen paso hacia hacer que la representación del software sea más accesible y útil para todos, sin complicar demasiado las cosas.

Bibliografía

Sandra Victoria Hurtado Gi

<https://webcache.icesi.edu.co/revistas/index.php/sistemas_telematica/article/view/918/943>