implementar esta técnica. El modelado de amenazas se presenta como una práctica fundamental para integrar la seguridad en el ciclo de desarrollo del software (SDLC), contribuyendo a prevenir pérdidas económicas y a mantener la confianza de los usuarios. Aunque su implementación puede parecer costosa a corto plazo, sus beneficios a largo plazo justifican el esfuerzo, especialmente en sistemas críticos y aplicaciones web.

34. Revisión sistemática sobre generadores de código fuente y patrones de arquitectura.

Los proyectos de desarrollo de software pueden enfrentar retrasos y baja calidad debido a deficiencias en la organización del código y la integración de componentes. Para mejorar la productividad y reducir estos riesgos, se pueden utilizar generadores de código en tareas repetitivas, como interfaces de usuario, conexiones a bases de datos y reportes. Además, es fundamental planificar y diseñar previamente la arquitectura del software para asegurar una estructura organizada. A través de una revisión sistemática de la literatura, se identificaron patrones de arquitectura y herramientas utilizadas en la generación de código para aplicaciones web. Se concluyó que los patrones de arquitectura son clave para estandarizar y organizar el código en capas, facilitando el desarrollo y mejorando la calidad del software.

Desarrollo de Framework Web para el desarrollo ...ítico de aplicaciones.

El documento concluye subrayando la importancia de reconfigurar el área de juego y presentar opciones de manera accesible para el usuario.

31. Metodología para la construcción de Interfaces Gráficas centradas en el usuario.

El documento propone una metodología para diseñar Interfaces Gráficas de Usuario (GUI) centradas en el Usuario, combinando diseño centrado en el usuario pensamiento de diseño desarrollo ágil, usabilidad y experiencia de usuario. Destaca la importancia de las GUI en la interacción humano-computadora y como su calidad impacta la ejecución del sistema. La metodología incluye nueve fases iterativas, desde la estructura hasta el despliegue, involucrando al usuario en cada etapa. Se presenta un caso de estudio basado en una aplicación de comunicación para personas con necesidades complejas, enfrentando retos como limitaciones motrices y cognitivas. Se enfatiza la validación futuro de esta metodología y el desarrollo de herramientas para evaluar su eficacia.

32. Arquitectura de Software con programación orientado a objeto.

El artículo examina la arquitectura de Software desde la programación orientada a objeto (POO) destacando su capacidad para organizar sistemas, mediante componentes, relaciones y principios de diseños que satisfacen requisitos funcionales y no funcionales. La POO utiliza conceptos como clases, objetos herencia y métodos para modelar problemas problemas de manera natural, acercando los conceptos al diseño practico. Lenguaje como Python, Java, PHP y C++ han adoptado características de POO siendo C++ y Java los más usados por su robustez. Entre las ventajas destacan la reutilización de código mejor mantenibilidad y la integración de soluciones preexistentes. Esta evolución permite desarrollar software eficiente rapido y de calidad mejorando la experiencia del usuario.

33. Malelo de amenazas, una ténica de analisis y gestión de riesgos asosiados a Software y aplicaciones

El documento analiza el **Modelado de amenazas** una tecnica clave en la gestión de riesgos para garantizar la seguridad de Software y sistemas. Se destaca su relevancia frente al aumento de vulnerabilidad debido a desarrollos acelerados. El incluye identificar amenazas, evaluar riesgos y aplicar controles para mitigar posibles ataques.

Se detallan cinco etapas esenciales; formar un equipo de analisis, descomponer la aplicación para identificar componentes criticos, determinar amenazas usando metodos como STRIDE asignar valores de riesgo mediante DREAD y definir respuestas priorizadas. Ademas, el texto menciona metodología y herramientas como SDL y Threat Modeling Tool para.

mail como un apoyo pedagógico para mejorar los procesos de enseñanza-aprendizaje.

28. La ingeniería de Software libre y sus herramientas aplicadas a proyectos informáticos.

El documento analiza la gamificación como estrategia para mejorar la motivación, el compromiso y la productividad en el desarrollo de software, identificado y clasificado 49 elementos clave a través de una revisión sistemática de literatura. Entre los más utilizados efectivos destacan puntos, insignias y recompensas que han demostrado su capacidad para motivar a los participantes y optimizar su desempeño. También se consideran otros elementos como retos, retroalimentación y avatares que enriquecen la experiencia de gamificación. Finalmente, se propone un catálogo de estos elementos aplicables tanto en contextos académicos como en la industria con el objetivo de potenciar de la eficacia y compromiso de los equipos de desarrollo.

29. Modelo de casos de uso

El modelo analiza la aplicación de casos de uso en entornos de desarrollo global de software abordando los desafíos de comunicación, coordinación y control en equipos distribuidos. Mediante el uso de desarrollo dirigido por modelos (MDD), se refina la plantilla CUPIDo para adaptarla a contextos globales, permitiendo la documentación automatizada y generación de diagramas. La investigación enmarcada en proyectos de la Universidad Nacional de San Juan, busca mejorar la especificación de requisitos funcionales, integran de requisitos funcionales, integran herramientas colaborativas y generar modelos útiles para diversas etapas del desarrollo. Los resultados preliminares incluyen la versión 1.4 de CUPIDo su extención CUPIDoTPi, destacando su relevancia para la ingeniería de Software en un entorno globalizado.

30. Patrones de interacción para el diseño de videojuegos en smartphones.

El documento "Patrones de interacción para el diseño de videojuegos en smartphones" de Leandro Filigriana, Andrés Solano y Cesar A. Collazos investigan la usabilidad de los videojuegos móviles mediantes patrones de interacción identifica problemas comunes que afectan la experiencia del usuario, como la baja calidad en la física del juego y las limitaciones en la personalización de controles. La investigación aplica patrones de interacción basados en soluciones probadas para problemas recurrentes, como la "Configuración del Área de Juego" y el "Despliegue de información de alta densidad". Se espera que los patrones propuestos mejoren las herramientas y paradigmas en la industria de los videojuegos móviles facilitando una experiencia de usuarios más amigables

21. Desarrollo de framework web para el desarrollo dinámico de aplicaciones.
El documento propone una ontología para estructurar y organizar el conocimiento del diseño de software, abordando diagramas [illegible] patrones esenciales [illegible] ciclo de vida del [illegible] para esta [illegible]

25. Buenas prácticas para el desarrollo de aplicaciones Web través de patrones de diseño usando el Marco de trabajo zend framework.

El documento aborda la implementación de buenas prácticas en el desarrollo de aplicaciones Web utilizando patrones de diseño y el zend framework. Se identifican problemas comunes en aplicaciones Web que afectan su mantenibilidad y escalabilidad proponiendo patrones como adapter, composite, Decorator, y factory Method para mejorar la calidad del código. Además se detallan estándares de codificación específicos de zend framework, como convenciones de nombres, manejos de variables y documentación, que contribuyen a generar código entendible y mantenible. Finalmente se evalúa la aplicabilidad de estas prácticas mediante ejemplos concretos y pruebas, mostrando como estas estrategias facilitan el desarrollo, reducen errores y mejoran la escalabilidad de las aplicaciones

26. Modelo de investigación en Ingeniería de Software: Una propuesta de investigación tecnológica

El documento sobre el "Modelo de Investigación en Ingeniería del Software. destaca que la investigación en esta disiplina busca generar conocimiento aplicable en entornos industriales definiendo métodos y modelos útiles para el desarrollo de software. La estrategia propuesta consta de tres fases: investigación aplicada a través de proyectos reales y transferencia del conocimiento mediante cursos y conferencias. Además, se identifican distintos tipos de investigaciones enfocandose en el tipo de pregunta resultado y validación Los metodos de investigación se clasifican en observación se clasifica, histórico y controlados se concluye que es esencial una estrategia que vincule la investigación academica con las necesidades industriales manteniendo un enfoque metodológico ringuroso.

27. Metodología ágil para el desarrollo de aplicaciones moviles educativas

El documento propone una metodología ágil para el desarrollo de aplicaciones móviles educativas, integrando aspectos pedagógico y tecnológicos. Basada en la programación Extrema (XP) y un framework que articula componente pedagógicos con metodología de software. La propuesta se sentra en un proceso iterativo y flexible que fomentan la colaboración entre pedagogos y desarrolladores, asegura la adaptabilidad a cambios de requisitos y prioriza los aspectos pedagógicos desde el inicio. Incluye etapas de planeación, diseño, codificación y pruebas, orientadas a crear herramientas moviles solidas y efetivas para el aprendizaje. Como ejemplo, se desarrollo una aplicación para prevenir el grooming, un proyecto realizado por estudiantes del profesorado en informática de la universidad Nacional de Santiago del Estero. En definitiva, esta metodología busca aprovechar la tecnología

Nombre: Fecha:

Profesor: Materia:

Institución: Curso: Nota:

23. Buenas prácticas en la construcción de software.

El artículo reflexiona sobre la importancia de las buenas prácticas en la construcción de software en el contexto de la cuarta revolución industrial, donde las organizaciones enfrentan desafíos para desarrollar sistemas de información complejos de forma eficiente y sin exceder costos. Se destacan estándares como clean code, que promueve código simple y fácil de leer, y principios como KISS, DRY y el diseño de funciones pequeñas y específicas. También se resalta la relevancia de una correcta arquitectura para satisfacer requisitos funcionales y no funcionales, abarcando estilo como arquitectura en capas, monolíticos, microservicios y basadas en eventos. Los patrones de diseño se presentan como herramientas claves para solucionar problemas comunes y estructurar el código de forma eficiente. Además, se enfatiza el uso de herramientas tecnológicas y metodología como XP para optimizar procesos, mantener flexibilidad y asegurar entregables funcionales en ciclo iterativos. Por último, el artículo concluye que integrar estas prácticas y estándares no solo mejora la calidad y mantenibilidad del software, sino que también reduce tiempos y costos, beneficiando a las organizaciones y sus equipos de desarrollo.

24. Lineas de productos Software Generando código a partir de Modelos y patrones.

El artículo explora las líneas de productos software (LSP), un enfoque que fomenta la reutilización de componentes para mejorar la eficiencia en el desarrollo. Basado en Model Driven Development (MDD) y Model Driven Architecture (MDA), LSP permite crear sistemas con características compartidas desde un núcleo común, optimizando tiempos y productividad. MDA emplea modelos como CIM, PIM para representar diferentes niveles de abstracción y usa procesos de transformación que convierte modelos independientes en específicos, generando código adaptable a diversas tecnologías. Un prototipo desarrollado ilustra este enfoque utilizando PHP, SQL y el patrón MVC para generar interfaces CRUD. Concluye que LSP incrementa la calidad y reutilización del software, mientras que la automatización reduce significativamente los tiempos de desarrollo.

21. Desarrollo de framework web para el desarrollo dinámico de aplicaciones.

El documento propone una ontología para estructura y organizar el conocimiento del diseño de software, abordado diagramas y patrones esenciales para esta etapa del ciclo de vida del desarrollo. Se define ontología como una representación formal y compartida de conceptos que mejora la comunicación y reutilización de conocimiento. La propuesta incluye diagramas estructurales (clases, componentes) y de comportamientos (secuencia, actividades) además de patrones de diseño como GRASP y GOF que abarcan creacionales estructurales y de comportamiento. Esto permite resolver problemas recurrentes en múltiples niveles de abstracciones. La ontología pretende sistematizar conceptos que suelen estar dispersos facilitando su enseñanza y práctica profesional. Se destaca la utilidad de estas herramientas en la mejora de la calidad del software y la promoción de buenas prácticas. Asimismo, aborda el diseño de sistemas colaborativos para enseñar estos conceptos a estudiantes de ingeniería ayudando conocimientos y fomentar una mejor aplicación en proyectos de software.

22. Tipificación de dominios de Requerimientos para la Aplicación de patrones Arquitectónicos.

El artículo aborda la complejidad de la arquitectura de software y propone una metología para facilitar la selección de patrones arquitectónicos en el desarrollo de sistemas. Los investigadores Analizaron 68 proyectos de software para identificar "requisitos funcionales comunes, creando un dominio de requerimientos" que permite asociar patrones arquitectónicos específicos con diferentes tipos de requisitos.

El propósito principal es reducir la brecha entre ingenieros en formación y profesionales experimentados, proporcionando una guía sistemática para seleccionar patrones arquitectónicos. La investigación busca transformar un proceso tradicionalmente basado en la experiencia individual en un metódo más estructurado y reproducible.

Como validación, aplicaron su metodología en un proyecto real de evaluación docente para una institución educativa, desarrollando una aplicación móvil que demostró la utilidad del enfoque propuesto. El resultado es una herramienta que ayuda a los desarrolladores a tomar desiciones arquitectónicas más informadas y consientes.

Su uso facilita la comunicación en los equipos y optimiza los procesos. La agilidad en el desarrollo de software busca reducir riesgos y garantizar la calidad y el cumplimiento de plazos y costes, promoviendo equipos auto organizados. Los patrones y la agilidad comparten el objetivo de minimizar el esfuerzo y asegurar calidad, y se integran en marcos ágiles como XP y Scrum. Este trabajo propone una investigación para desarrollar un marco de trabajo de patrones ágiles que apoyen la auto-organización de equipos a lo largo de su evolución.

16 Sistema para el control y seguimiento de la implementación de la metodología de gestión de procesos de negocio sustentadas en el uso de patrones

El artículo propone un sistema para gestionar y controlar métodos, herramientas y técnicas en empresas de desarrollo de software bajo el paradigma de Gestión de Procesos de Negocio, con enfoque en la reutilización de componentes y la trazabilidad para garantizar la calidad del producto. Basado en la metodología del Dr. Pedro Bonillo, utiliza Intalio BPMS 6.0 y NetBeans IDE con JSF para modelar, diseñar y ejecutar procesos de negocio adaptados a mejores prácticas organizacionales, validando la calidad en cada fase mediante la reutilización de experiencias exitosas.

17 Generación de la interfaz de usuario de negocio a partir de patrones de negocios basada en los fundamentos metodológicos de TD-MBUID

El documento propone un proceso basado en modelos para desarrollar interfaces de usuario de negocio, utilizando la metodología TD-MBUID. A través de la aplicación de patrones de datos, plantillas de presentación y modelos de interacción, busca interpretar mejor los modelos mentales de los usuarios sobre los datos. Este enfoque mejora la calidad y el tiempo de diseño de prototipos de interfaces utilizando el lenguaje DataForm y herramientas de Eclipse. Su objetivo es optimizar la creación de interfaces alineadas con las necesidades del usuario final, aplicando patrones de diseño y metodologías basadas en modelos para el desarrollo eficiente de software empresarial.

18 Aplicación de patrones de diseño estructurales para el modelamiento de clases de los sistemas empresariales.

La tesis aborda la complejidad de los sistemas empresariales y propone el uso de patrones de diseño estructurales para facilitar el modelamiento de datos en sistemas administrativos. Mediante una investigación cuantitativa aplicada con 36 estudiantes de ingeniería de sistemas se concluyó que la aplicación de estos patrones mejora significativamente el modelamiento de datos destacando la utilidad de soluciones probadas para resolver problemas específicos de diseño.

19 Diseño e implementación de un lenguaje para modelar autómatas deterministas en

Se propone un lenguaje de dominio específico (DSL) para modelar y especificar autómatas finitos deterministas, mejorando su comprensión en entornos educativos. El DSL ofrece una representación formal. Integra mecanismos de visualización y utiliza técnicas de análisis léxico y sintáctico para asegurar su funcionalidad. Concluye que este lenguaje facilita el aprendizaje de la teoría de autómatas, proporcionando una herramienta efectiva para su representación y análisis.

20 Biologística: Estrategias computacionales bioinspiradas para la gestión de las redes de suministro complejas

El artículo propone un modelo bioinspirado para gestionar redes de suministro como sistemas colectivos. Frente a las limitaciones de los enfoques tradicionales basados en control y predicción, plantea aprovechar características emergentes como flexibilidad, robustez y resiliencia. El estudio sugiere que métodos bioinspirados sugieren mejorar la adaptabilidad y adaptabilidad de las redes de suministro en entornos cambiantes.

reutilización reduce automatización de desarrollo

Por ejemplo, Template method es ideal para reutilizar algoritmos con pequeñas variaciones. MVC es popular para separar bien la lógica, la vista y el modelo (aunque suele ser difícil al principio). MVP separa la interfaz y la lógica (ideal para apps grandes) pero es más complejo. Front Controller gestiona todas las solicitudes desde un solo punto, mejorando así la seguridad, aunque escalar la app puede llegar a ser más complicado. MVVM es excelente para mantener modularidad y hacer pruebas, pero puede llegar a ser difícil para principiantes.

4 Marco de trabajo para seleccionar un patrón Arquitectónico en el Desarrollo de Software.

La tesis aborda los desafíos de seguridad en arquitecturas de microservicios y proponen el patrón **Microservice Security Pattern API Gateway (MSPAG)**, que utiliza JWT y el algoritmo H256 para centralizar la autenticación y proteger los endpoints mediante una API Gateway. Incluye un marco teórico, análisis del estado del arte e implementación en C#, Python y Java, evaluando su eficacia con pruebas. Concluye que patrones como MSPAG fortalecen la seguridad de los microservicios, garantizando integridad y confidencialidad. Reflexionar a la hora de elegir patrones arquitectónicos adecuados nos optimiza tiempos, costos y calidad, estableciendo así bases sólidas para los proyectos.

5 Propuesta metodológica para migración de sistemas web con arquitectura monolítica hacia una arquitectura basada en microservicios.

El documento propone la metodología SMMicro para migrar sistemas web monolíticos a microservicios, destacando beneficios como escalabilidad y modularidad, y subrayando la importancia de una descomposición cuidadosa. SMMicro incluye análisis organizacional, descomposición de sistemas y construcción de un ecosistema con componentes como Discovery Server y Load Balancing. Validado en un caso de estudio en la escuela Politécnica Nacional, muestra mejoras en flexibilidad y escalabilidad. Concluye que los microservicios son ideales para sistemas grandes y dinámicos, pero requieren un enfoque estratégico para superar costos iniciales y resistencia al cambio.

6 ¿Son los microservicios la mejor opción? Una evaluación de su eficacia y eficiencia frente a los monolitos.

El documento compara la eficacia y eficiencia de arquitecturas monolíticas y de microservicios, destacando que los monolitos ofrecen mejor rendimiento en contextos simples debido a su integración directa, mientras que los microservicios destacan en modularidad, escalabilidad y despliegue independiente, aunque implican mayor complejidad y consumo de recursos.

resultados inconsistentes y desafíos con interfaces dinámicas. Big Data y dispositivos móviles. Destaca de las CT son esenciales para detectar defectos críticos tempranamente, aunque persisten retos como la cobertura limitada de pruebas no funcionales y en la nube. Se concluye que, pese a avances, se necesita investigación adicional para integrar eficientemente las pruebas continuas en los procesos de CD, asegurando calidad y confiabilidad.

10 Estudio de métricas y patrones de seguridad en Microservicios.

La tesis titulada "Estudio de metricas y patrones de seguridad en Microservicios" de Juana Victoria Sanchez aborda los desafíos de seguridad en esta arquitectura emergente, destacando la falta de estudios específicos y sistemáticos. Mediante un mapeo sistemático, analiza métricas y patrones de seguridad, identificando 49 patrones y la adaptación de métricas de otras arquitecturas, aunque muchas carecen de validación en entornos reales. Como contribución, propone una guía práctica basada en estándares de calidad (ISO/IEC 9126), con métricas clave y ejemplos aplicables. Concluye que la seguridad en microservicios es un área incipiente que requiere más investigación y validación práctica.

11 Metodología para la identificación de sistemas a través de patrones

La tesis presenta una metodología que sistematiza el conocimiento experto en la identificación de sistemas para controladores industriales. Basada en la teoría de patrones, la metodología organiza el proceso en cuatro etapas: experimentación, preprocesado, atención de parámetros y validación, complementadas por un catálogo y un lenguaje de patrones que guían de manera estructurada el desarrollo de modelos. Contribuye a reducir la curva de aprendizaje, mejorar la comunicación entre expertos y principiantes, y facilitar la aplicación de patrones en áreas técnicas complejas como la ingeniería de control.

12 Marco de trabajo para seleccionar un patrón Arquitectónico en el Desarrollo de Software

El artículo presenta un marco de trabajo para seleccionar patrones arquitectónicos en el desarrollo de software, abordando problemas de desacoplamiento y falta de conocimiento arquitectónico que afectan la calidad de los productos. Basado en cinco fases, identifica los patrones más relevantes (como MVC, Microservicios y Arquitectura en la nube) y define reglas para elegir patrones según el tipo de desarrollo y características requeridas.

Utilizando herramientas como Docker y Mocker, las pruebas mostraron que los monolitos tienen tiempos de respuestas más rápidos y menos tasa de error, mientras que los microservicios son más adecuados para sistemas grandes y dinámicos, especialmente con un API Gateway para optimizar su desempeño. Se concluye que la elección de la arquitectura depende de las necesidades y el contexto del sistema.

7 Modelado y Verificación de Patrones de Diseño de Arquitectura de Software para Entornos de Computación en la Nube

El documento analiza la optimización de consultas SQL en bases de datos relacionales, destacando cómo consultas mal diseñadas afectan el desempeño de las aplicaciones. Resalta el uso de herramientas como PL/SQL Developer para mejorar el rendimiento mediante planes de ejecución. Describe las fases del procesamiento SQL (parseo, optimización, generación de registros y ejecución) y aborda técnicas como el uso de índices y semplificación de consultas. Concluye que la optimización debe ser constante, destacando el valor de herramientas gráficas para identificar mejoras y la necesidad de un conocimiento sólido en SQL y la arquitectura de bases de datos.

8 Solución para gestión y control de información para las bases de datos

El documento propone un sistema para gestionar solicitudes de datos en bases de datos, orientando el crecimiento de información en organizaciones modernas impulsado por tecnologías como IA e IoT. La solución captura, analiza y entrega datos de forma estructurada y segura, evitando la saturación de los sistemas. Implementa metodologías de Ingeniería de Software y la arquitectura de AWS para ofrecer servicios escalables y seguros, utilizando PHP con Laravel y MySQL.

Sus funcionalidades incluyen gestión de solicitudes asignación de tareas, autenticación de usuarios y entrega eficiente de datos. Concluye que la herramienta mejora la interacción entre usuarios y administradores, optimiza tiempos de respuesta y reduce errores, recomendando pruebas adicionales y futuros desarrollos para ampliar capacidades.

9 Problemas que afectan la calidad de software en Entrega Continua y Pruebas Continuas

El documento examina los problemas que afectan la calidad del software en en "entrega continua (CD)" y "Pruebas Continuas (CI)" en entornos ágiles, como pruebas largas,

09/12/2024 18:34

Nombre: Fecha:
Profesor: Materia:
Institución: Curso: Nota:

2

continuación...

4) marca, validado mediante un prototipo, ahorra tiempos, costos, calidad, escalabilidad y mantenibilidad. Concluye que la solución es una herramienta práctica que guía a arquitectos y desarrolladores en la selección de patrones adecuados, estableciendo estructuras correctas desde el inicio del proyecto.

13 Concepto de Arquitectura Software y Principios de Diseño.

El documento define la AS como el diseño de sistemas complejos basado en componentes, forma y lógica, destacando principios como modularidad y bajo acoplamiento. Explica el uso de patrones (creacionales, estructurales y de comportamiento) y estilos arquitectónicos como MVC y Cliente-Servidor para estructurar sistemas. Introduce el estándar "ISO 42010" para describir sistemas intensivos y presenta herramientas de gestión del conocimiento arquitectónico (AKM). También analiza aplicaciones en la "Industria 4.0" como IoT y Big Data, y el uso de diagramas UML para documentar arquitecturas. Concluye que la AS combina decisiones creativas y herramientas modernas para adaptarse a los desafíos actuales de la industria.

14 Patrones de Alexander a la tecnología de Objetos.

Nos explica los "patrones de diseño" en software, que son soluciones reutilizables a problemas comunes en el desarrollo de aplicaciones. Estos patrones surgieron a partir de las ideas del arquitecto Christopher Alexander, adaptadas al campo del software, especialmente en la programación orientada a objetos. El concepto se popularizó en 1995 con el libro "Design Patterns" de las "Cuatro de la Banda" (GoF) que presentó 23 patrones para resolver problemas recurrentes de diseño. Los patrones ayudan a crear sistemas más flexibles y fáciles de mantener, describiendo el contexto del problema, las fuerzas involucradas y la solución.

15 Patrones y Antipatrones en equipos ágiles: una revisión sistemática

En la ingeniería de software, los patrones son soluciones reutilizables a problemas recurrentes, aceptados por los profesionales para mejorar la calidad y reducir costes y errores.

Nombre: Patricia del Pilar Sarmiento R. Fecha: dia mm año

Profesor: Materia:

Institución: Curso: Nota:

1 Revisión de elementos conceptuales para la representación de las arquitecturas de referencia de software.

Resumen

El artículo resalta cómo la Arquitectura de software es esencial en el desarrollo en el desarrollo, definiendo la estructura general de un sistema y mostrando cómo interactúan sus componentes. Según el Instituto de Ingeniería de Software, es como un mapa que muestra las partes y sus conexiones sin meterse en detalles. La investigación encontró que los elementos más importantes en estas arquitecturas son los componentes y los conectores, que se repiten en muchos modelos y lenguajes. También se descubrió que las "vistas arquitectónicas" son la herramienta clave para ayudar a los equipos a entenderse mejor, aunque tienen la desventaja de necesitar un lenguaje formal para automatizar procesos, lo que complica reutilizar diseños. Estos hallazgos no se quedan solo en la teoría; se usan para proponer una manera de representar el conocimiento arquitectónico y documentar las decisiones tomadas, superando las limitaciones actuales de las vistas arquitectónicas.

2 Lenguajes de patrones de Arquitectura de Software: Una aproximación al estado del Arte.

El artículo destaca cómo los lenguajes de patrones han transformado la arquitectura de software, tomando inspiración de los conceptos de Christopher Alexander sobre patrones en arquitectura de edificios. Desde la programación estructurada en los años 80 hasta la actualidad, estos lenguajes han facilitado soluciones repetibles y clave de problemas complejos. Son especialmente útiles en áreas como la seguridad de la información y el diseño de sistemas e-Business, acelerando el desarrollo y mejorando la calidad del software. En conclusión, los lenguajes de patrones son fundamentales para resolver problemas arquitectónicos de forma eficiente y efectiva.

3 Análisis comparativo de Patrones de Diseño de Software.

El documento compara cinco patrones de diseño de Software (Template Method, MVC, MVP, Front Controller y MVVM), explica cómo ayudan a organizar y mantener el código, evitando duplicaciones. No hay un "mejor patrón", cada uno tiene sus pros y contras.