**RESÚMENES DE ARTÍCULOS**

**JULY DANIELA RAMOS PEÑA**

**SENA**

**ADSO - 2901817**

**Jesús Ariel Gonzales**

**AGOSTO-2025**

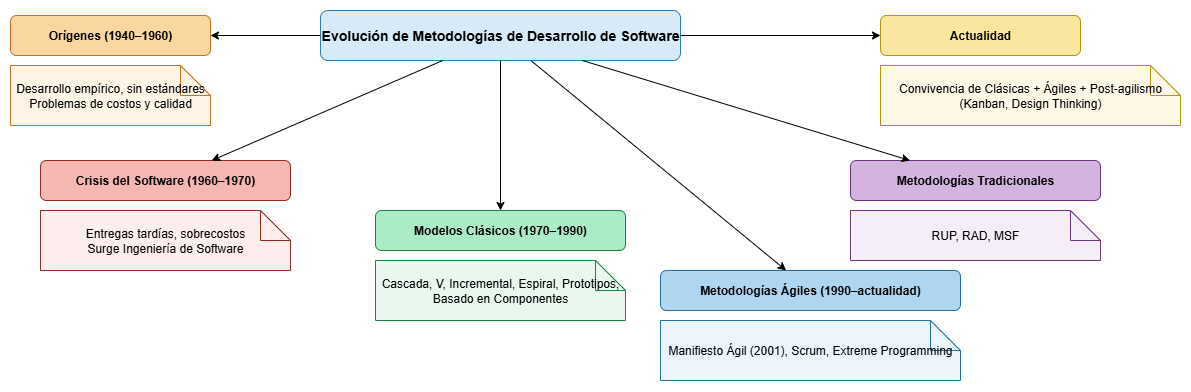
1. **Evolución de las metodologías y modelos utilizados en el desarrollo de software.**

El artículo trata sobre la evolución de las metodologías a través del tiempo junto el desarrollo de software, desde los años 40 hasta la actualidad. Al inicio, los programas se hacían de forma muy improvisada, casi artesanal, lo que daba muchos problemas: software costoso, entregas tarde y sistemas que no cumplían lo que el usuario pedía. Esto llevó a lo que se llamó la “crisis del software”.

Para solucionar esos problemas aparecieron los primeros modelos y metodologías clásicas de desarrollo como el modelo en cascada, el espiral, el incremental o el de prototipos, que buscaban organizar el trabajo en fases, con más control, documentación y planificación. Después se dieron metodologías más estructuradas como RUP, RAD y MSF, que se desarrollaban sobre todo en proyectos grandes y complejos.

Con la llegada de Internet en los años 90, los cambios rápidos y las entregas cortas hicieron que esos métodos fueran poco prácticos. Por eso nacieron las metodologías ágiles, que ponen el interés en las personas, el cliente y la adaptación al cambio. Ejemplos claros son Scrum y Extreme Programming.

En conclusión, el desarrollo de software pasó de ser algo desorganizado y rígido a un proceso más humano y flexible, donde lo importante es entregar valor al usuario en el menor tiempo posible.



**Reflexión:**

Con este articulo me di cuenta que el desarrollo de software ha cambiado mucho, al principio era improvisado, desorganizado, después demasiado rígido y con muchos papeles de por medio, y ahora en la actualidad con las metodologías ágiles se que se sea más práctico, adaptable y con un buen manejo en equipo.

**Bibliografía:**

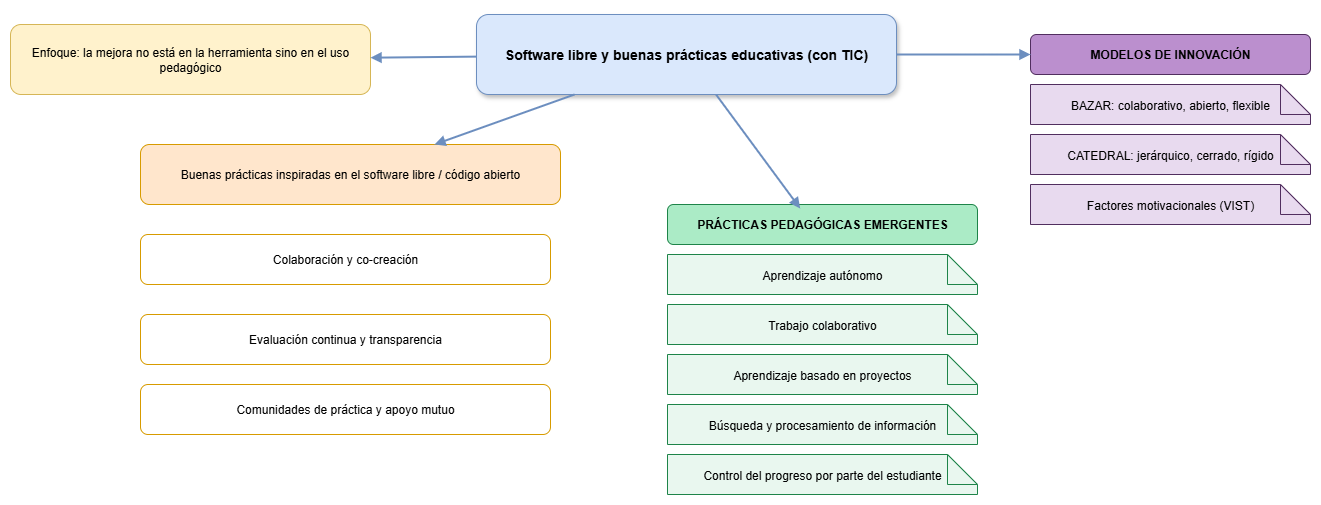
Gamboa, J. Z. (2018). Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. *INNOVA Research Journal*, *3*(10), 20-33.

1. **El software libre y las buenas prácticas educativas con TIC**

El artículo trata sobre cómo el software libre puede transformar las prácticas educativas con tecnologías de la información y comunicación (TIC). El autor explica que la verdadera mejora educativa no está en las herramientas tecnológicas por sí solas, sino en el uso que profesores y estudiantes hacen de ellas para mejorar el proceso de enseñanza con aprendizaje.

Luego se presentan las "prácticas pedagógicas emergentes" identificadas en estudios internacionales, como el desarrollo autónomo en los estudiantes, el aprendizaje colaborativo basado en proyectos, y la personalización del ritmo de aprendizaje. El articulo hace énfasis en el caso exitoso de Extremadura, España, donde se implementó LinEx (sistema operativo libre) en todo su nivel educativo, creando contenidos educativos libres y enseñando a los docentes.

El autor establece una diferencia entre dos modelos de mejora: el "bazar” que es colaborativo, abierto y horizontal contra el “catedral" que es jerárquico, cerrado, y centralizado, apoyando el primero. Finalmente, propone un conjunto de buenas prácticas educativas inspirado en los principios del software libre y código abierto.



**Reflexión:**

Con este articulo pude ver que la educación creativa, innovación con TIC no se trata solo de usar tecnología nueva, sino de cómo los profesores y estudiantes la aplican para aprender o enseñar. Lo que importante como decía el articulo no son las herramientas como tal sino el uso que se les da. De esta manera, el software libre muestra como es el trabajo colaborativo y creativo siendo un buen ejemplo para motivar a aprender y enseñar.

**Bibliografía:**

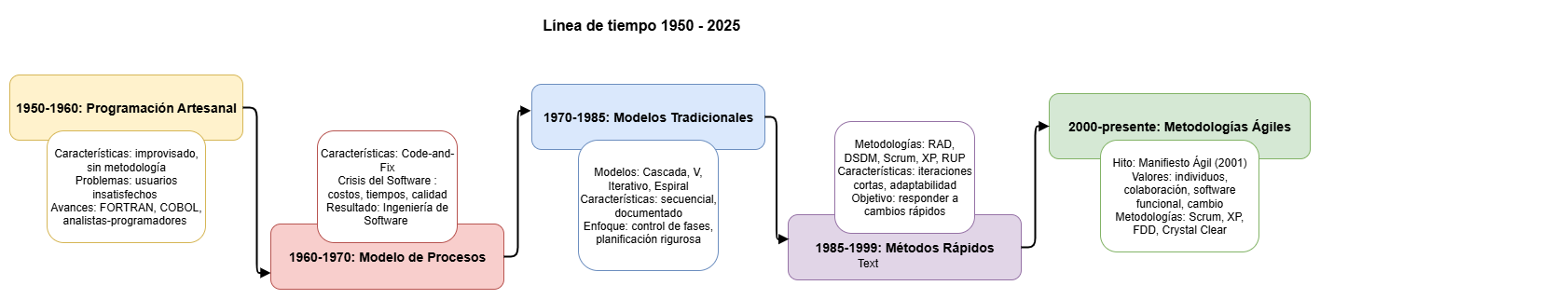
Berrocoso, J. V. (2008). El software libre y las buenas prácticas educativas con TIC. *Comunicación y pedagogía*, *222*(48), 55.

1. **Evolución de las metodologías de desarrollo de la ingeniería de software en el proceso la ingeniería de sistemas software**

El artículo trata sobre la evolución de las metodologías de desarrollo de software desde los años 40 hasta la actualidad. Inicialmente, en los años 50 al 60, la programación era completamente artesanal e improvisada, donde los programadores codificaban sin metodologías formales, lo que producía descontento en los usuarios porque no se entendían bien sus necesidades.

En los años 60 al 70 surgió la "Crisis del Software" debido a proyectos costosos, tardíos e inestables, lo que llevó al surgimiento de la Ingeniería de Software como disciplina. Durante los 70 al 80 aparecieron los modelos tradicionales estructurados como Cascada, Incremental y Espiral, que organizaban el desarrollo en fases secuenciales con mayor control y documentación.

Durante 1980 y 1990 aparecieron los conocidos métodos rápidos como RAD, Scrum y Extreme Programming, creados para adaptarse a ambientes variables donde la rapidez y la adaptabilidad eran fundamentales. Con el transcurso del tiempo, estos conceptos se desarrollaron y, desde el año 2000, se establecieron las metodologías ágiles. Estas centraron su atención en las personas, la cooperación con el cliente y la habilidad para ajustarse al cambio. Las técnicas ágiles se transformaron en una solución casi indispensable para numerosos proyectos que demandan agilidad y flexibilidad.



**Reflexión:**

Aquí en este articulo pasa como el primero, nos damos cuenta que el desarrollo de software ha pasado de ser algo muy improvisado en los 40, a metodologías más estructuradas llenas de control, y ahora hay son metodologías ágiles que benefician mucho más en el desarrollo del trabajo. Hoy en día las dos conviven, ambos estilos existen, pero se destacan mucho más las metodologías ágiles.

**Bibliografía:**

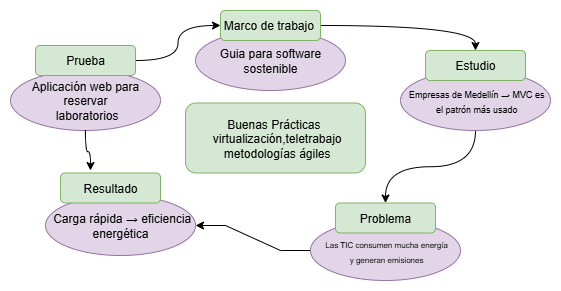
Garcés Guayta, L. R., & Egas, L. M. (2014). Evolución de las metodologías de desarrollo de la ingeniería de software en el proceso la ingeniería de sistemas software.

1. **Buenas prácticas para el desarrollo de software sostenible**

El artículo trata sobre cómo crear software más amigable con el medio ambiente, algo súper importante porque la tecnología consume muchísima energía y contamina bastante. Los autores investigaron esto porque, aunque las TIC nos ayudan a ser más eficientes, también generan un montón de emisiones de gases que dañan el planeta.

Lo que hicieron fue estudiar empresas de software en Medellín para ver qué tecnologías usan más, encontraron que MVC es el patrón más popular, y después revisaron un montón de estudios para identificar las mejores prácticas. Básicamente descubrieron que hay dos tipos de prácticas importantes: las técnicas, como usar virtualización, cloud computing y lenguajes de programación eficientes, y las del proceso, como trabajar desde casa, usar documentos digitales livianos y metodologías ágiles como Scrum.

Al final crearon un marco de trabajo que es como una guía para desarrollar software sostenible y lo probaron haciendo una aplicación web para reservar laboratorios. Los resultados mostraron que su aplicación cargaba súper rápido lo que demuestra que es eficiente.



**Reflexión:**

Este artículo muestra cómo también el software puede ser más amigable con el medio ambiente. Porque, aunque la tecnología nos ayuda muchísimo, también gasta muchos recursos ambientales. Digamos en el artículo en Medellín estudiaron a empresas de software y se dieron cuenta que para reducir el desgaste ambiental se deben utilizar buenas practicas y patrones, y esto es bueno porque ayuda tanto al ambiente como al programa.

**Bibliografía:**

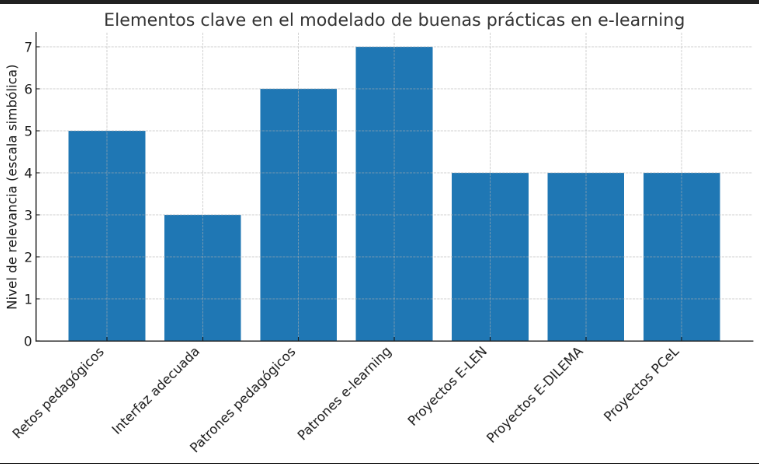
Soto Durán, D. E., Reyes Gamboa, A. X., Giraldo Mejía, J. C., Villamizar Jaimes, A. E., & Vidal Alegría, F. A. (2022). Buenas prácticas para el desarrollo de software sostenible.

1. **Modelado de “buenas prácticas” docentes en experiencias e-learning**

El artículo trata sobre un problema que tienen muchos profesores universitarios: saben mucho de su materia, pero no tienen idea de cómo enseñar bien, especialmente en plataformas virtuales.

Para solucionar esto, existen los "patrones educativos" que son como recetas de buenas prácticas que se pueden reutilizar. El problema es que todos estos patrones están escritos solo en texto, y eso los hace súper ambiguos y difíciles de aplicar en la práctica. Los autores proponen usar las mismas técnicas que se usan para crear software, diagramas y modelos visuales, para representar estos patrones educativos. Específicamente usan una metodología llamada OO-Method que permite crear diagramas súper claros y precisos.

Para probar su idea, tomaron el patrón "DEBATE" que estaba descrito solo con texto confuso y lo convirtieron en diagramas que muestran exactamente cómo funciona: qué tipos de usuarios participan, cómo se relacionan, qué pasos sigue el debate, etc. La idea es que con estos diagramas los profesores puedan aplicar las buenas prácticas más fácilmente, y hasta se podrían generar aplicaciones de e-learning automáticamente a partir de los modelos.



**Reflexión:**

En este artículo pienso en como muestra que muchos profesores dominan su materia, pero no siempre saben cómo enseñarla bien en ambientes virtuales. El que se representen de forma visual facilita su comprensión y aplicación, haciendo así que los docentes mejoren sus prácticas en y logren experiencias de aprendizaje y enseñanza más eficientes.

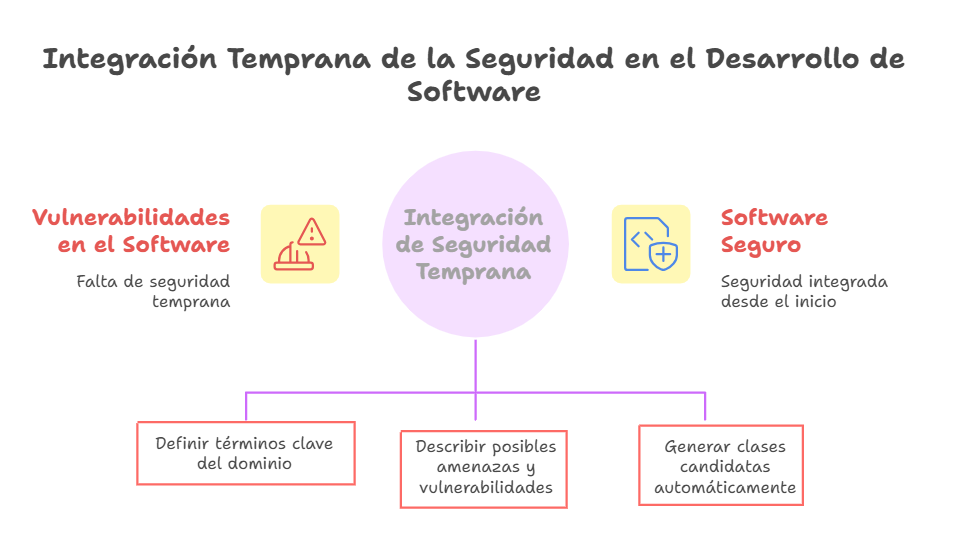
**Bibliografía**

Ballester, J. V., Panach, J. I., Buendía, F., & PASTOR, O. (2011). Modelado de “buenas prácticas” docentes en experiencias e-learning. *Proc. ISELEAR 2011*.

1. **Embebiendo Patrones de Seguridad en Etapas Tempranas del Proceso de Desarrollo de Software**

El artículo trata sobre un problema importante en el desarrollo de software: la mayoría de las aplicaciones son vulnerables porque no se piensa en seguridad desde el principio. Los autores se preguntaron qué tan temprano se puede incorporar la seguridad en un proyecto de software. Su propuesta es bastante interesante: usan tres herramientas que se combinan para detectar necesidades de seguridad desde la fase de requisitos. Primero crean un "diccionario" del dominio, después describen escenarios de lo que puede pasar, y finalmente una herramienta llamada BMW genera automáticamente clases candidatas del sistema. Lo probaron con el patrón de "Autorización” y lo aplicaron a un caso real de seguros de vehículos. El resultado fue que lograron identificar automáticamente las clases necesarias como Cliente, Cuenta de Ahorro y Póliza, que ya incluían los mecanismos de seguridad desde el análisis.

La ventaja más grande es que al detectar estos requisitos de seguridad tan temprano, se evitan los costos enormes de arreglar vulnerabilidades después. Su objetivo es crear un catálogo de patrones de seguridad reutilizables para que otros desarrolladores puedan aplicarlos fácilmente en sus proyectos desde el inicio.



**Reflexión:**

El artículo plantea que la seguridad no debería pensarse al final, sino desde el inicio del desarrollo de software. Proponen un mecanismo para detectar necesidades de seguridad en la fase de requisitos, usando herramientas que permiten modelar el dominio y generar clases con seguridad incorporada. Esto demuestra que añadir patrones de seguridad desde el comienzo no solo evita vulnerabilidades, sino que también ahorra costos y tiempo. Para darnos cuenta que la seguridad sea parte desde el inicio y no algo de última hora.

**Bibliografía:**

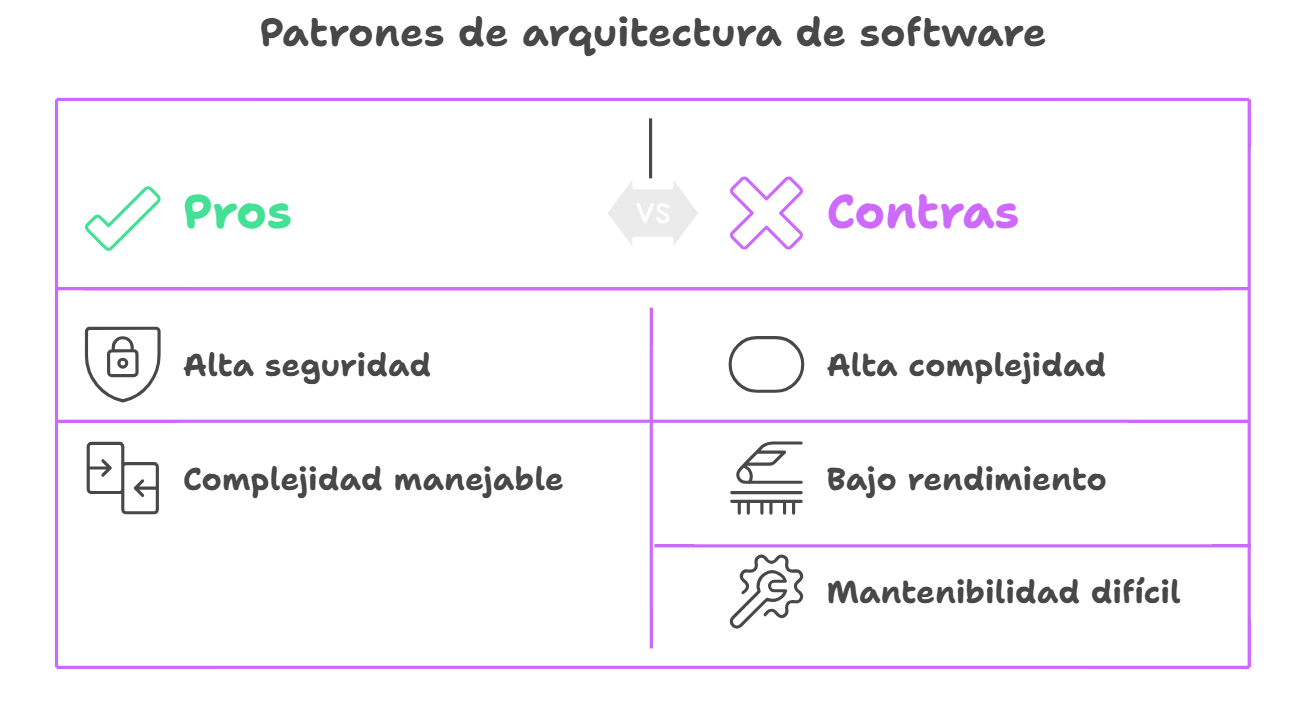
Solinas, M., Fernandez, E. B., & Antonelli, L. (2015). Embebiendo Patrones de Seguridad en Etapas Tempranas del Proceso de Desarrollo de Software.

1. **Evaluación de la calidad en patrones de arquitecturas software para sistemas críticos ferroviarios: un enfoque basado en la mantenibilidad**

El artículo habla sobre cómo elegir la mejor arquitectura de software para sistemas ferroviarios sin que sea imposible de mantener después. El problema es que las arquitecturas más seguras suelen ser más complejas de programar y mantener. Los investigadores probaron 5 patrones diferentes: programación de N versiones, aceptación por votos, bloque de recuperación, maestro-esclavo y redundancia de hardware. Cada uno tiene diferentes niveles de complejidad y seguridad.

Para evaluar esto, pusieron a 5 equipos de estudiantes universitarios a implementar cada patrón durante una semana, midiendo tanto qué tan difícil fue desarrollarlo como la calidad del código final. Usaron herramientas automáticas para medir líneas de código, complejidad y cumplimiento de buenas prácticas.

Los resultados muestran que la redundancia de hardware es la más fácil pero solo sirve para sistemas de baja poco críticos, mientras que el bloque de recuperación ofrece un buen balance entre seguridad alta y complejidad manejable. La aceptación por votos resultó ser la más compleja y difícil de mantener.

****

**Reflexión:**

El artículo muestra que en los sistemas de trenes no es fácil encontrar un equilibrio entre la seguridad y lo fácil que sea mantener el software. Algunas formas de diseñar son muy seguras, pero también muy complicadas de programar y cuidar con el tiempo. Otras, en cambio, logran un buen punto medio, como el bloque de recuperación. Esto nos enseña que no basta con pensar solo en la seguridad, también hay que buscar que el sistema sea práctico y se pueda mantener sin tantas dificultades.

**Bibliografía:**

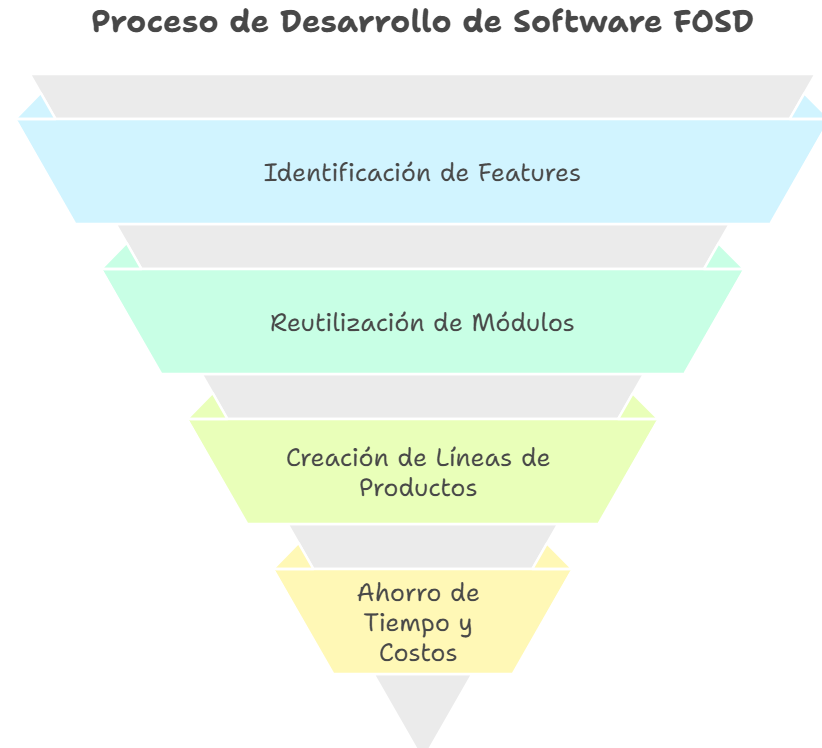
Duprat, J. A., & Irrazábal, E. (2019). Evaluación de la calidad en patrones de arquitecturas software para sistemas críticos ferroviarios: un enfoque basado en la mantenibilidad. In *de XXV Congreso Argentino de Ciencias de la Computación, Río Cuarto*.

1. **Análisis de Desarrollo de Software Orientado a Feature–Línea de Producto de Software para aplicaciones de TVD**

Este artículo habla de hacer software que básicamente no reinvente la rueda cada vez. Se llama FOSD (Desarrollo Orientado a Features) y se da así: en lugar de hacer cada programa desde cero, identificas las "features" o funciones que se repiten en diferentes proyectos y las reutilizas como módulos.

La idea central son las Líneas de Productos de Software, que es como tener una plataforma base de la cual sacas diferentes versiones según lo que necesite cada cliente. Los investigadores se enfocan específicamente en aplicaciones de Televisión Digital Interactiva, que es esa tecnología que te permite no solo ver tele sino interactuar con apps directamente en la pantalla. En Argentina este sistema comenzó en 2010 y estaba programado para reemplazar completamente la TV analógica en 2020.

El enfoque es súper práctico: después de hacer tres productos, ya empiezas a ahorrar tiempo y plata comparada con el desarrollo tradicional. Además, como reutilizas componentes que ya están probados, la calidad mejora y el mantenimiento se vuelve más fácil.



**Reflexión:**

Este artículo muestra lo importante que es no empezar siempre de cero al crear software. El FOSD ayuda a aprovechar lo que ya existe y adaptarlo a diferentes necesidades, algo muy útil en la televisión digital, donde cada usuario busca experiencias distintas. Me parece una forma práctica de trabajar, porque ahorra tiempo, esfuerzo y permite enfocarse más en mejorar lo que realmente necesita la gente.

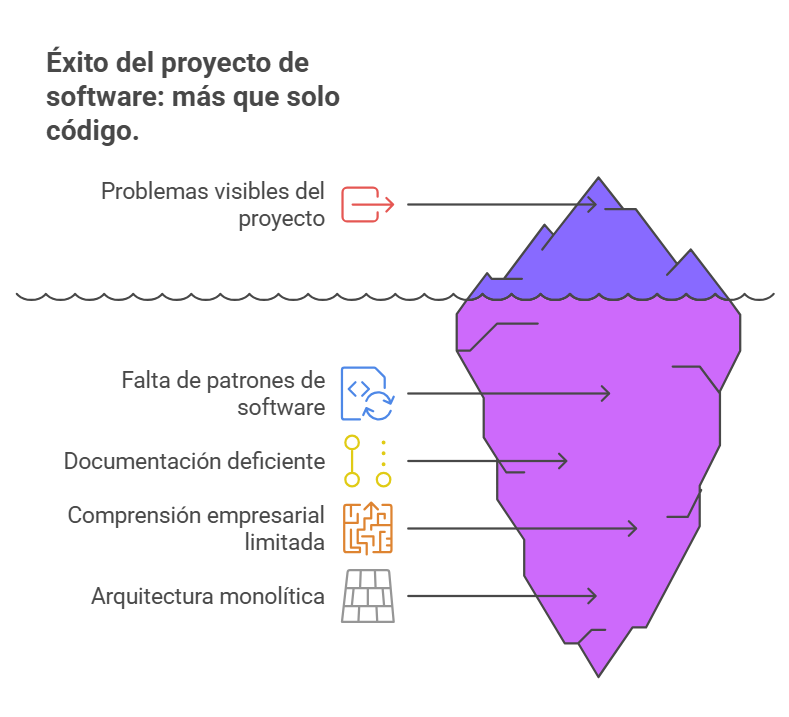
**Bibliografía:**

Miranda, M. F., Casas, S. I., & Marcos, C. A. (2015). Análisis de desarrollo de software orientado a Feature-Línea de producto de Software para Aplicaciones de TVDI. *Informes Científicos Técnicos-UNPA*, *7*(2), 167-195.

1. **La importancia de la estructuración de patrones de Software y buenas prácticas para la consecución exitosa de proyectos de desarrollo de soluciones informáticas integrales**

Este artículo básicamente dice que: hay un montón de empresas haciendo software de una forma muy desorganizada y eso trae problemas dañinos e innecesarios después. El autor cuenta que ha visto muchos proyectos donde los programadores se enfocan demasiado en sacar el producto rápido sin pensar en cómo va a funcionar a largo plazo. Y el resultado son aplicaciones que nadie puede mantener, llenas de errores, y que cuando algo falla nadie sabe cómo arreglarlo tan solo el programador origen.

La solución que propone es bastante razonable: usar patrones de software probados, documentar bien todo desde el principio, y entender realmente el negocio antes de ponerse a programar. También habla mucho de hacer arquitecturas por capas, donde cada parte del sistema tiene su función específica y puede reutilizarse en otros proyectos.



**Reflexión:**

Este artículo muestra cómo muchas veces, por la presión de entregar rápido, se descuida la calidad del software. Eso termina siendo un problema, porque se crean sistemas difíciles de mantener y que solo entiende quien los programó. Se deja algo clave en el artículo el seguir buenas prácticas, usar patrones probados y documentar desde el inicio puede decidir si el proyecto fracasa o que sirva y se desarrolle con éxito.

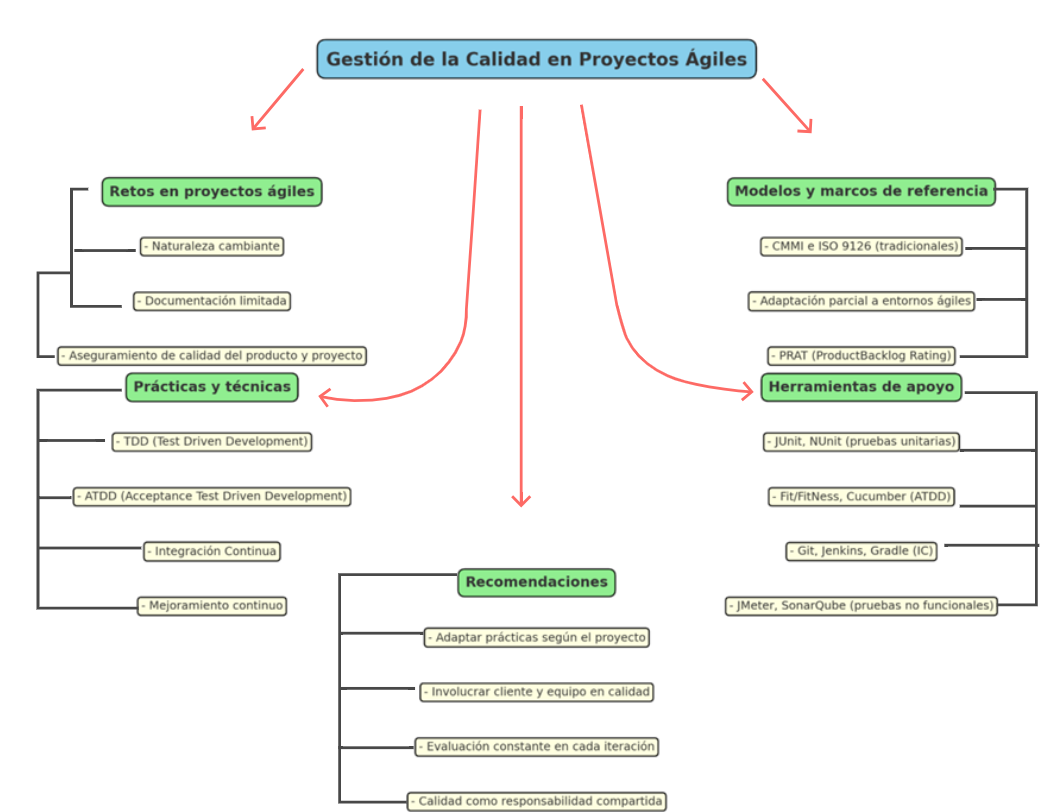
**Bibliografía:**

Mejía, J. A. G. (2015). La importancia de la estructuración de patrones de Software y buenas prácticas para la consecución exitosa de proyectos de desarrollo de soluciones informáticas integrales. *Revista Ontare*, *1*(2), 39-61.

1. **Revisión de herramientas y buenas prácticas implementadas para la gestión de la calidad en proyectos de desarrollo de software con metodologías ágiles**

El artículo trata sobre un problema que enfrentan muchas empresas hoy en día: cómo mantener la calidad del software cuando se usan metodologías ágiles. Los autores hicieron una investigación revisando estudios científicos para encontrar qué herramientas y prácticas funcionan mejor. Descubrieron que no hay una forma única, sino que cada proyecto necesita una combinación específica de prácticas.Entre las mejores herramientas encontraron el Desarrollo Dirigido por Pruebas, donde escribes las pruebas antes que el código, la Integración Continua para automatizar procesos, y las pruebas de aceptación que se enfocan en lo que realmente necesita el cliente.

Lo más interesante es que llegaron a la conclusión de que la calidad en proyectos ágiles no depende solo de herramientas técnicas, sino también de factores humanos como la experiencia del equipo, la comunicación con el cliente, y qué tan bien se adapten las prácticas al contexto específico del proyecto.



**Reflexión:**

Este artículo me recordó que en el desarrollo con metodologías ágiles no es suficiente con enfocarse en la rapidez, también hay que cuidar la calidad en cada paso. No existe un sistema único: lo que funciona en un proyecto puede no servir en otro. La clave está en adaptar las prácticas al contexto del proyecto, entendiendo que la calidad no es tarea de unos pocos, sino de todo el equipo.

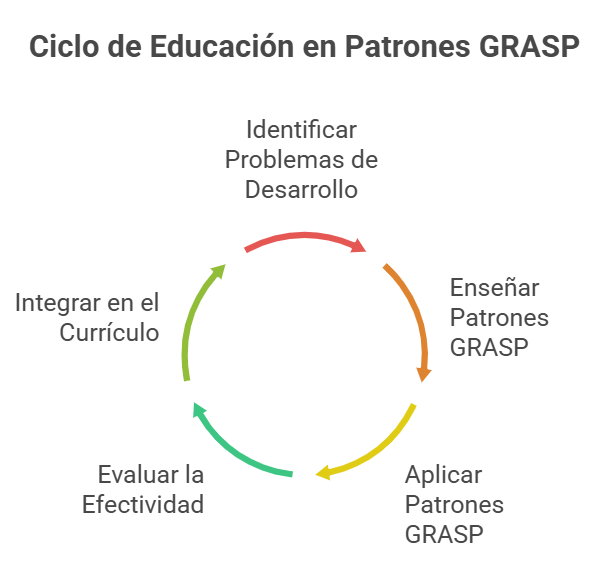
**Bibliografía:**

Mercado Ramos, V. H., & Zapata Arbeláez, J. (2014). Revisión de herramientas y buenas prácticas implementadas para la gestión de la calidad en proyectos de desarrollo de software con metodologías ágiles.

1. **Lineamientos para el diseño de aplicaciones web soportados en patrones GRASP**

Este artículo trata sobre un problema real que enfrentan muchas universidades: los estudiantes de ingeniería de sistemas salen sin saber realmente cómo desarrollar software de calidad. El autor propone enseñar los patrones GRASP como solución.

Los patrones GRASP son como "recetas probadas" para resolver problemas comunes en programación orientada a objetos. Son 8 patrones que ayudan a decidir qué clase debe hacer qué cosa en el programa. El punto clave es la "asignación de responsabilidades": decidir inteligentemente qué funciones va a tener cada clase. Los autores usan el ejemplo de un cajero automático para mostrar cómo aplicar estos patrones paso a paso. Hicieron un estudio con 40 estudiantes y encontraron que efectivamente hay vacíos en estos conocimientos. Su propuesta es que las universidades incluyan estos patrones en el pensum para formar programadores que realmente sepan hacer software bien estructurado y mantenible.

****

**Reflexión:**

En este articulo muestra como muchos estudiantes de sistemas, software o computación, salen al trabajo sin saber bien cómo organizar su código. Los patrones GRASP ayudan porque enseñan a repartir mejor las tareas dentro de un programa, haciendo el software más claro y fácil de mantener. Por eso es importante que en la universidad se usen estas ideas para preparar mejor a los estudiantes.

**Bibliografía:**

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

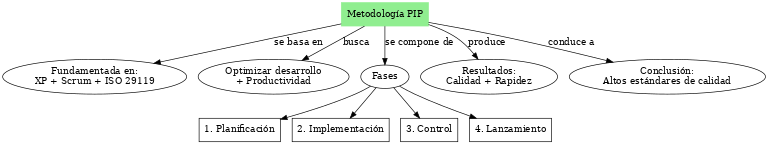
1. **Nueva metodología PIP orientada a las Buenas Prácticas para el desarrollo de software**

Este artículo presenta una nueva metodología llamada PIP (Programación Integrada Patricio) creada para mejorar cómo se desarrolla el software. Los autores vieron que hay un problema real: muchos equipos de desarrollo no tienen una metodología común que todos puedan seguir, lo que causa desorden y proyectos que salen mal.

Su solución fue crear PIP mezclando lo mejor de tres metodologías que ya funcionan bien: Programación Extrema (XP), Scrum, y estándares ISO. PIP tiene 4 fases claras: primero planificas todo, tiempo, recursos, objetivos, luego desarrollas el software, después controlas que todo vaya bien, y finalmente entregas el producto al cliente.

Lo chévere de esta metodología es que es flexible, se adapta según el tipo de proyecto que se tenga. Si se necesita algo más estructurado, se puede hacer. Si necesita algo más ágil, también sirve.

Los autores hicieron pruebas y comparaciones, y encontraron que PIP realmente mejora los tiempos de entrega y la calidad del software. Su objetivo es que las empresas tengan estándares más altos de calidad y puedan responder más rápido a lo que necesitan sus clientes.



**Reflexión:**

Este articulo dice que la metodología **PIP** ayuda a que los equipos de software trabajen de forma ordenada. Para esto toma ideas de Scrum, XP y las normas ISO para organizar las fases del proyecto. Con esto, los programas se hacen más rápido y con mejor calidad.

**Bibliografía:**

Peralta, M. C. H. P., Carnero, M. H. F. P., & Osco, O. F. A. (2022). Nueva metodología PIP orientada a las Buenas Prácticas para el desarrollo de software. *Ciencia Latina Revista Científica Multidisciplinar*, *6*(4), 5032-5040.

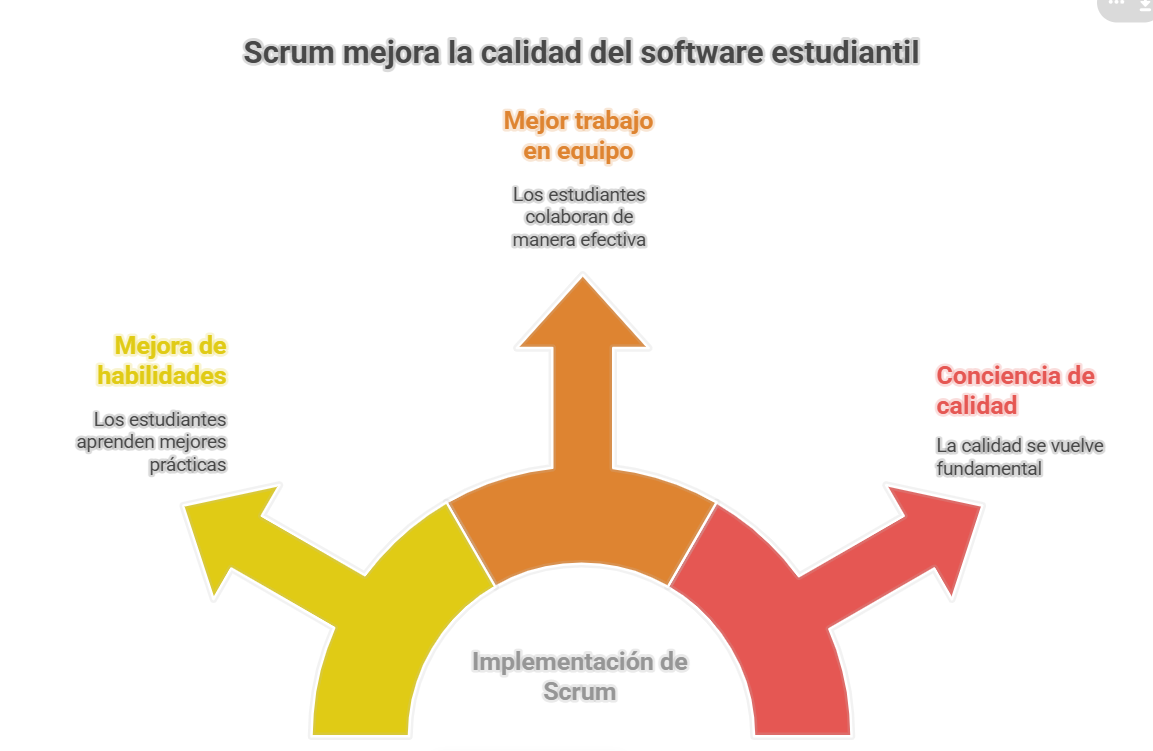
1. **Mejores prácticas de calidad en el desarrollo de software integradas al conocimiento de la ingeniería**

El artículo habla sobre cómo mejorar la calidad en el desarrollo de software en el ambiente universitario. Básicamente, los profesores de una universidad se dieron cuenta de que los estudiantes de ingeniería de software cometían los mismos errores una y otra vez cuando desarrollaban proyectos.

Los principales problemas que encontraron fueron: que los estudiantes no sabían hacer bien las entrevistas a los clientes, diseñaban mal las bases de datos, no hacían pruebas adecuadas y odiaban hacer documentación. Todo esto pasaba porque desarrollaban software de manera muy improvisada.

Para solucionarlo, aplicaron Scrum como metodología ágil en tres proyectos reales de la universidad. El profesor actuaba como Scrum Master y los estudiantes formaban equipos pequeños. Implementaron mejores prácticas como hacer que el cliente participara más, planificar mejor, hacer pruebas entre equipos diferentes y obligar a documentar desde el principio.

Los resultados fueron positivos: los estudiantes fueron mejorando en cada proyecto, desarrollaron un mejor trabajo en equipo y se volvieron más conscientes de la importancia de la calidad. Al final, lograron que entendieran que la calidad no es algo opcional sino fundamental en el desarrollo de software.

****

**Reflexión:**

Este artículo me hace pensar que en el desarrollo de software no basta con saber programar. También es importante planear, comunicarse bien y el trabajo en equipo. Con Scrum y buenas prácticas, los estudiantes aprendieron que la calidad no es un detalle extra, sino la base para que un proyecto tenga éxito.

**Bibliografía:**

del Roció Veloz-Remache, G., Menéndez-Verdecia, J. A., & Aguilar-Moncayo, L. N. (2021). Mejores prácticas de calidad en el desarrollo de software integradas al conocimiento de la ingeniería. *Polo del Conocimiento*, *6*(1), 656-668.

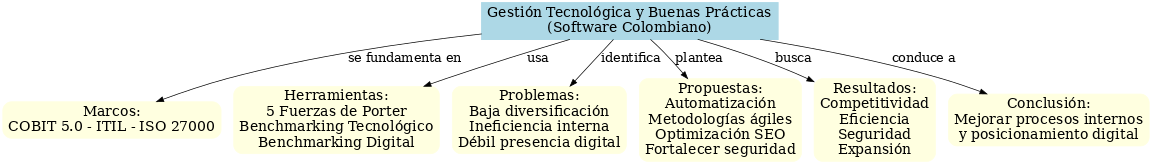
1. **Gestión de tecnología y buenas prácticas empresa de desarrollo de software colombiana**

Este artículo habla sobre cómo una empresa colombiana de desarrollo de software logró mejorar su competitividad usando herramientas de gestión tecnológica reconocidas internacionalmente. Técnicamente, la empresa tenía problemas serios: sus procesos internos eran muy manuales y desorganizados, no tenía buena presencia en internet, y la competencia le estaba ganando clientes dando productos más baratos y variados.

Los investigadores hicieron un diagnóstico completo usando tres enfoques principales. Primero, aplicaron la matriz de Porter para entender contra qué se enfrentaban en el mercado. Segundo, compararon la empresa con sus competidores tanto en tecnología como en presencia digital, y los resultados fueron bastante malos. Tercero, usaron estándares internacionales para proponer mejoras.

Los principales problemas encontrados fueron que tenían muy pocas visitas en su página web, casi no aparecían en Google, sus redes sociales estaban abandonadas, y internamente todo se hacía de forma manual y desorganizada. Para solucionarlo, propusieron automatizar procesos, mejora las redes sociales, adoptar metodologías ágiles y ampliar sus productos.

Al final, la empresa logró desarrollar estos cambios siguiendo un plan estructurado con responsables claros para cada tarea, mejorando su posición competitiva en el mercado tecnológico colombiano.



**Reflexión:**

Este articulo nos muestra que muchas veces las empresas no pierden clientes solo por el precio, sino porque se quedan atrás en organización y en tecnología. La empresa de Colombia su experiencia demuestra que modernizar los procesos, usar metodologías ágiles y mejorar la presencia digital no es un lujo, sino una necesidad. La lección es clara: en un mundo tan competitivo y que avanza cada vez más, quien no se adapta, se queda.

**Bibliografía:**

Espinosa, R. D. C., & Eraso, J. C. C. (2024). Gestión de tecnología y buenas prácticas empresa de desarrollo de software colombiana. *REVISTA DELOS*, *17*(62), e3210-e3210.

1. **Buenas prácticas para la gestión del conocimiento en el proceso de desarrollo de software**

Este artículo habla de un problema que tienen muchas empresas de software en Cuba: cuando un trabajador se va, se lleva consigo todo lo que sabe, y eso deja a la empresa medio perdida con sus proyectos. Los investigadores vieron que la clave está en **gestionar bien el conocimiento** para que no dependa solo de las personas.

Antes, en los 90, la gestión del conocimiento era básicamente guardar archivos y documentos, pero hoy se trata más de crear una cultura donde la gente comparta lo que sabe y aprenda en equipo. Para entender mejor cómo hacerlo, los autores revisaron cinco modelos internacionales y de ahí sacaron 11 buenas prácticas pensadas especialmente para las empresas cubanas. Entre ellas están cosas como identificar qué conocimientos son más valiosos, tener planes para compartir la información, proteger lo que es crítico y aprovechar la tecnología para organizar todo.

Estas ideas no se quedaron en teoría: se probaron en empresas reales, se validaron con expertos y funcionaron tan bien que hasta se convirtieron en una norma oficial en Cuba. Gracias a esto, muchas empresas pudieron retener mejor su conocimiento, incluso cuando había cambios de personal.

****

**Reflexión:**

El artículo muestra que en las empresas el conocimiento es muy valioso y no puede quedarse solo en una persona. En Cuba aplicaron buenas prácticas para que todos puedan compartir y aprovechar lo que saben. Así, el trabajo se vuelve más fácil y la empresa no se detiene cuando alguien se va.

**Bibliografía:**

Lazo Alvarado, Y., Tamayo Oro, L., & Pineda Gutiérrez, A. (2023). Buenas prácticas para la gestión del conocimiento en el proceso de desarrollo de software.

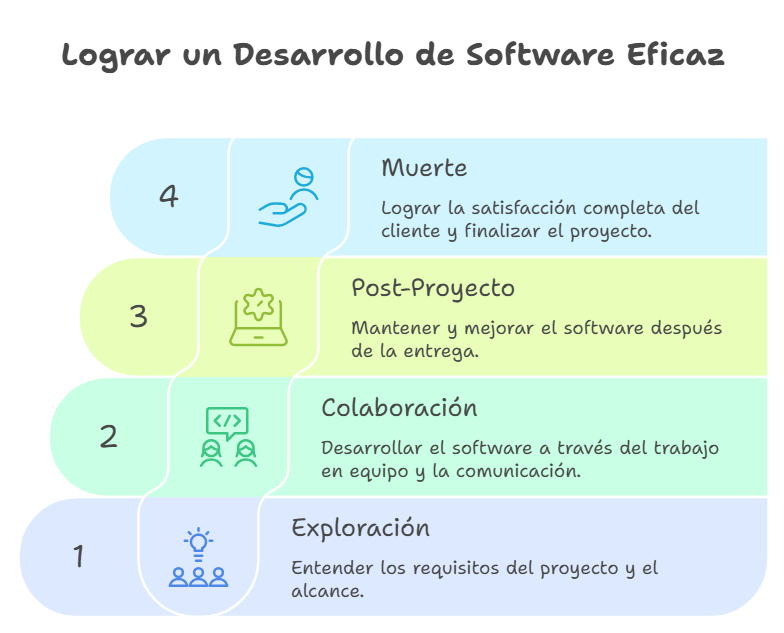
1. **Buenas prácticas, una solución para un mejor desarrollo de software**

El artículo habla sobre cómo crear mejores prácticas para desarrollar software, tomando lo mejor de las metodologías ágiles que ya existen. Esencialment0065, los autores se dieron cuenta de que, aunque hay muchas metodologías ágiles como XP, SCRUM, Crystal Clear y otras, a veces ninguna se adapta perfectamente a un proyecto específico.

Por eso proponen una especie de "metodología híbrida" que combina lo mejor de cada una. Su propuesta incluye ítems importantes como la comunicación constante, la responsabilidad compartida y entregas frecuentes.

Lo interesante es que proponen un ciclo de vida con cuatro etapas: primero la Exploración donde se entiende qué se va a hacer, luego la Colaboración donde se construye todo el software, después el Post-proyecto para hacer mantenimiento y mejoras, y finalmente la Muerte cuando el cliente ya está completamente satisfecho.

La idea es que estas prácticas se puedan usar tanto en proyectos pequeños como grandes, siempre manteniendo la filosofía ágil de adaptarse a los cambios y mantener al cliente involucrado durante todo el proceso.



**Reflexión:**

Este artículo muestra que no siempre una sola metodología ágil sirve para todos los proyectos. A veces es mejor combinar lo mejor de varias para adaptarse mejor a cada caso. La propuesta de un ciclo de vida con etapas claras ayuda a que el equipo y el cliente trabajen juntos de principio a fin, logrando buenos resultados sin importar si el proyecto es grande o pequeño.

**Bibliografía:**

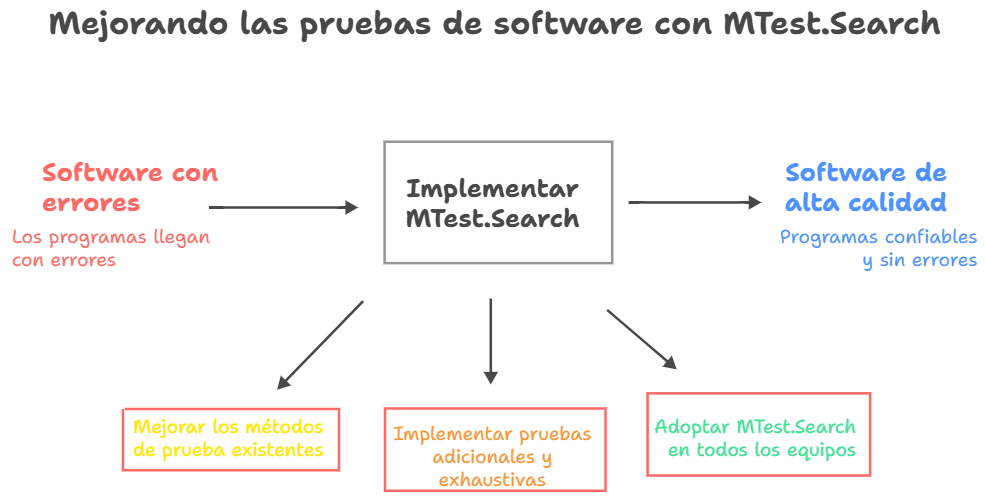
del Carmen Ramírez, L., & Fuentes, A. S. F. (2014). Buenas prácticas, una solución para un mejor desarrollo de software. *Mundo Fesc*, *4*(8), 37-45

1. **Buenas prácticas de ingeniería de software: pruebas de software**

Este artículo habla de cómo una empresa cubana de software decidió mejorar la forma en que prueba sus programas antes de entregarlos a los clientes. La empresa se dio cuenta de que tenía problemas en su software, sus programas a veces llegaban con errores a los usuarios finales, y eso no estaba bien.

Los investigadores de una universidad les propusieron usar un modelo nuevo llamado MTest.Search, que básicamente es una forma más inteligente de probar software. En lugar de que las personas tengan que hacer todas las pruebas manualmente, proponen usar herramientas automáticas que hagan gran parte del trabajo.

La idea es implementar esto en tres fases: primero mejorar lo que ya están haciendo, después agregar pruebas que no estaban haciendo antes, y finalmente que todos los equipos trabajen con este nuevo sistema. Además, explican que esto está conectado con las metodologías ágiles, que son formas modernas de hacer software donde se prueban las cosas constantemente en lugar de esperar hasta el final.

****

**Reflexión:**

Este artículo muestra la importancia de mejorar la forma en que se prueban los programas antes de entregarlos. La empresa cubana entendió que hacer las pruebas solo de manera manual no era suficiente, así que empezó a usar un modelo más moderno y automático llamado MTest.Search. Con esto se busca encontrar errores más rápido, tener mayor calidad en el software y trabajar de manera más ágil.

**Bibliografía:**

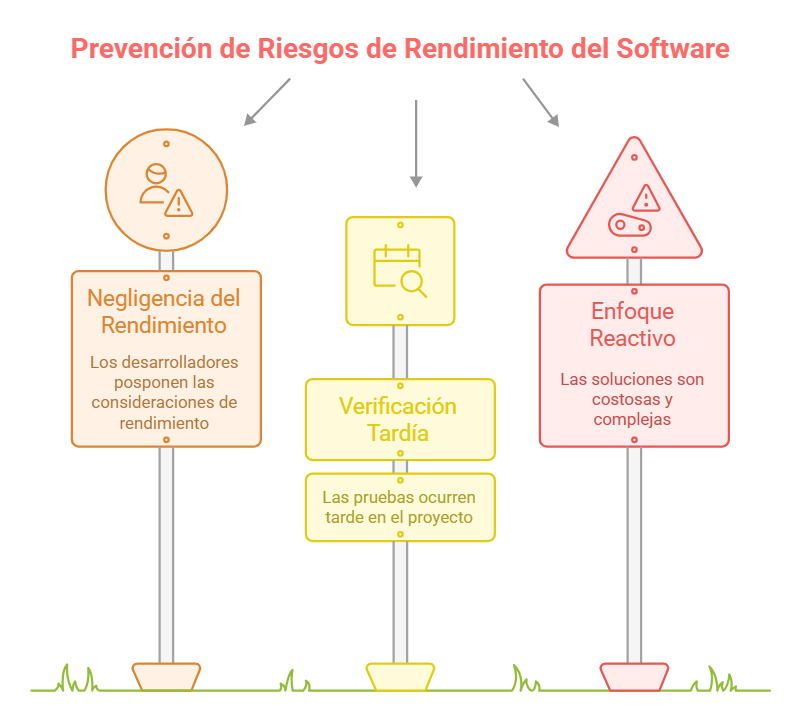
García, A. G., Hernández, M. S., Marcos, S. V., & Dapena, M. D. D. (2023). Buenas prácticas de la ingeniería de software: pruebas de software. *Revista Cubana de Transformación Digital*, *4*(2), 205-1.

1. **Buenas prácticas para prevenir los riesgos de la eficiencia del desempeño en los productos de software**

Este artículo habla de cómo a veces se usa una app que va súper lenta, un programa que se cuelga, o una página web que nunca carga. Los investigadores se propusieron entender por qué pasa esto y cómo evitarlo.

Hicieron una investigación en Cuba preguntando a gente que hace software si se preocupan por que sus programas funcionen rápido y bien. La respuesta fue "sí, es súper importante", pero cuando preguntaron si realmente hacen algo para verificarlo, resultó que más de la mitad no lo hace. Y los que sí lo hacen, muchas veces esperan hasta el final del proyecto, cuando ya es complicado y costoso arreglar los problemas.

Los autores proponen algo que es lógico pero que muchas empresas no hacen: pensar en el rendimiento desde el primer día del proyecto, no al final. Para demostrar que su idea era buena, preguntaron a 15 expertos de diferentes empresas y universidades. Todos estuvieron de acuerdo en que es necesario preocuparse por el rendimiento desde el principio.

****

**Reflexión:**

Cuando se desarrolla un software, no solo se trata de que funcione, sino de que lo haga bien. Si un sistema es lento o falla seguido, el usuario se frustra y deja de confiar en él. En cambio, cuando el rendimiento se cuida desde el inicio, el trabajo diario se vuelve más ágil y agradable. Al final, un buen desempeño no es un lujo, es lo que hace que la tecnología realmente ayude.

**Bibliografía:**

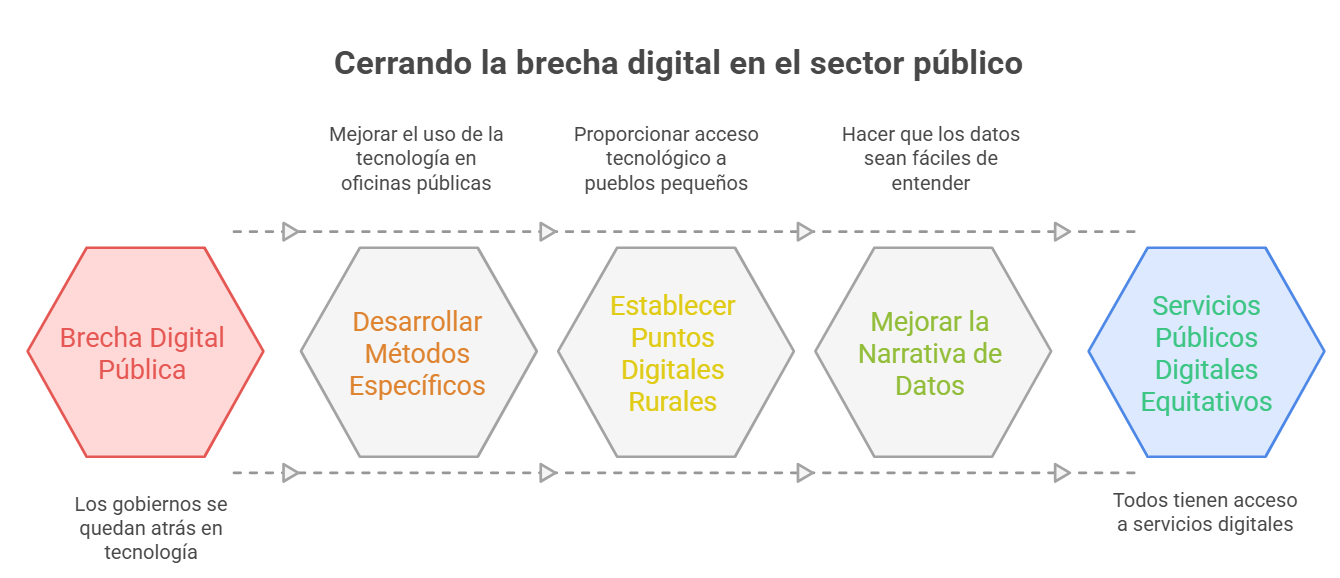
Piñero González, M., Marin Diaz, A., Trujillo Casañola, Y., & Buedo Hidalgo, D. (2021). Buenas prácticas para prevenir los riesgos de la eficiencia del desempeño en los productos de software. *Revista Cubana de Ciencias Informáticas*, *15*(1), 89-113.

1. **Buenas prácticas de calidad de software en el ámbito público**

Este artículo trata sobre un proyecto de investigación que busca ayudar a las instituciones públicas a digitalizarse mejor. Básicamente, los investigadores se dieron cuenta de que mientras las empresas privadas adoptan tecnología súper rápido, los gobiernos y municipios se quedan atrás con métodos antiguos y papeleos.

El proyecto tiene tres partes principales. Primero, están creando métodos especiales para que las oficinas públicas puedan usar mejor la tecnología, trabajando con empresas yerbateras, trenes y sistemas de cuarentena. Segundo, desarrollan "Puntos Digitales Rurales", que son como centros de ayuda tecnológica para pueblos chicos que no tienen buena internet o conocimiento digital, donde la gente puede hacer trámites online y aprender a usar tecnología. La tercera parte es sobre cómo contar historias con datos, o sea, hacer que los gráficos y estadísticas sean más fáciles de entender para que los funcionarios tomen mejores decisiones. Probaron esto con sistemas del COVID-19 y funcionó bien.

El objetivo final es disminuir la diferencia digital entre las ciudades grandes y los pueblos chicos, haciendo que todos tengan las mismas oportunidades de acceso a servicios digitales del gobierno.

****

**Reflexión:**

El artículo muestra que el gobierno todavía tiene mucho por mejorar en lo digital. La idea de los “Puntos Digitales Rurales” es muy buena, porque ayuda a que la gente de los pueblos también pueda acceder a servicios y oportunidades. Al final, se trata de usar la tecnología para que la vida de todos sea un poco más fácil.

**Bibliografía:**

Irrazabal, E., Dapozo, G. N., Greiner, C. L., Lezcano Airaldi, A., Carruthers, J. A., Ferraro, M. D. L. A., ... & González, J. (2023). Buenas prácticas de calidad de software en el ámbito público. In *XXV Workshop de Investigadores en Ciencias de la Computación (Junín, 13 y 14 de abril de 2023)*.

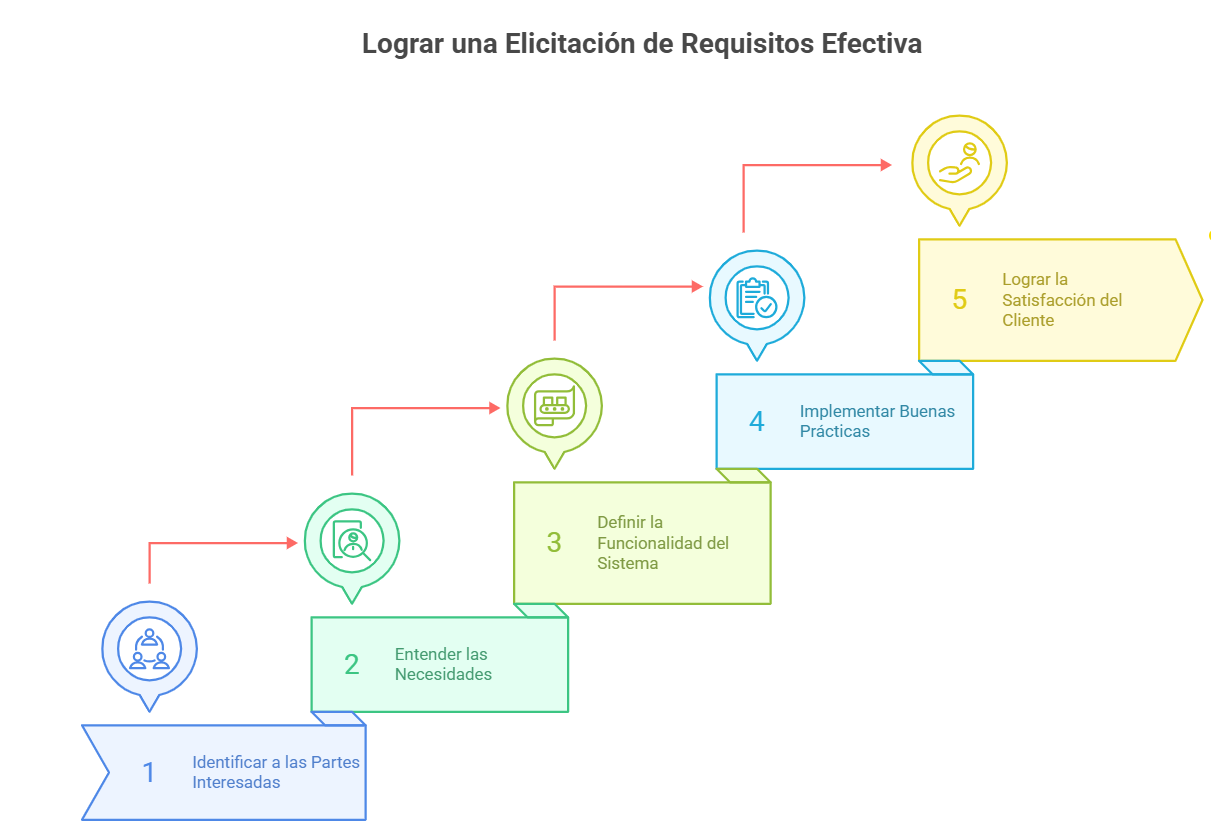
1. **Caracterización de buenas prácticas en la elicitación de requisitos de software referidas en el estándar ISO/IEC/IEEE 29148**

Este artículo habla sobre cómo mejorar una parte súper importante del desarrollo de software que se llama "elicitación de requisitos", que básicamente es cuando los programadores se sientan con el cliente para entender exactamente qué quiere que haga el programa.

Los investigadores se dieron cuenta de que esta etapa es donde se cometen muchos errores que después hacen que el software salga mal, no sirva para lo que el cliente necesitaba, o cueste más tiempo y dinero del esperado. Para solucionarlo, estudiaron las propuestas de varios expertos famosos en ingeniería de software y las compararon con un estándar internacional llamado ISO/IEC/IEEE 29148.

Lo que encontraron es que, aunque cada experto le pone nombres diferentes a las fases, todos están de acuerdo en que obtener bien los requisitos es clave. El estándar internacional sugiere tres actividades principales: identificar a todas las personas involucradas en el proyecto, entender qué necesitan exactamente, y definir cómo debe funcionar el sistema.

La idea es usar estas buenas prácticas como una guía para que los equipos de desarrollo puedan trabajar de forma más organizada y entregar software que realmente satisfaga las necesidades del cliente desde el principio.

****

**Reflexión:**

Este artículo muestra lo importante que es empezar un proyecto de software con requisitos claros. Muchas veces los problemas aparecen porque no se entendió bien lo que el cliente quería desde el inicio. Seguir buenas prácticas, como las que propone el estándar ISO/IEC/IEEE 29148, ayuda a que el trabajo sea más organizado y se eviten errores costosos después.

**Bibliografía:**

Parra-Quero, G., Palma-Urdaneta, O., Torres-Samuel, M. E., & Durán-Garrido, F. (2020). Caracterización de buenas prácticas en la elicitación de requisitos de software referidas en el estándar ISO/IEC/IEEE 29148. *Publicaciones en Ciencias y Tecnología*, *14*(2), 91-99.