Copyleaks

Plagiarism report

MendezMirandaLeonardoAlejandro.d...

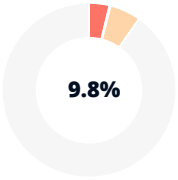
Scan details

Scan time:
May 25th, 2024 at 02:49 UTC

Total Pages:
96

Total Words:
23943

Plagiarism Detection



Types of plagiarism		Words
Identical	3.5%	827
Minor Changes	0.2%	50
Paraphrased	5.2%	1251
Omitted Words	8.9%	2138

AI Content Detection

Text coverage
- AI text
- Human text

Plagiarism Results: (85)

Metodología de ingeniería de software: diseño, desarrollo y evaluación d... 1.8%

https://www.docsity.com/es/ejercicios-de-quimica-ehb/10893203/

Prepara tus exámenes Consigue puntos Orientación Universidad Vende en Docsity Inicia sesión Regístrate Prepara tus e...

Metodologías de desarrollo de software 1.6%

https://repositorio.uca.edu.ar/bitstream/123456789/522/1/metodologias-desarrollo-software.pdf

Maida

Maida, Esteban Gabriel ; Pacienzia, Julián Metodologías de desarrollo de software Tesis de Licenciatura en Sistemas y Computación Facul...

metodologias-desarrollo-software.pdf 1.2%

https://www.slideshare.net/slideshow/metodologiasdesarrollosoftwarepdf/263769682

Submit Search Upload metodologias-desarrollo-software.pdf • 0 likes•22 views E ESTEBAN AULESTIA.ORGFollow metodologias-desarrollo-so...

metodologias-desarrollo-software.pdf 1.2%

https://www.slideshare.net/slideshow/metodologiasdesarrollosoftwarepdf/266252442

Submit Search Upload metodologias-desarrollo-software.pdf • 0 likes•7 views G GermanVargas70Follow proyectoRead less Read more Educ...

Results

metodologias-desarrollo-software - Tecnologías | Studenta — 1%

https://es.studenta.com/content/135915929/metodologias-desarrollo-software

Logo Studenta Iniciar sesión Volver Compartir Leer completo metodologias-desarrollo-software Ingeniería Ingeniería de Software ...

metodologias-desarrollo-software - Tecnologías | Studenta — 1%

https://es.studenta.com/content/135915929/metodologias-desarrollo-software

Logo Studenta Iniciar sesión Volver Compartir Leer completo metodologias-desarrollo-software Ingeniería Ingeniería de Software ...

Metodologías ágiles en el desarrollo de software | PDF — 0.9%

https://www.slideshare.net/slideshow/metodologas-giles-en-el-desarrollo-de-software/19924354

Submit Search Upload Metodologías ágiles en el desarrollo de software • 0 likes•218 views P princeosFollow Report Share ...

EL SOFTWARE Y LA INGENIERÍA DE SOFTWARE – INGENIERÍA DEL SOFTWARE — 0.9%

https://ingsotfwarekarlacevallos.wordpress.com/category/el-software-y-la-ingenieria-de-software/

INGENIERÍA DEL SOFTWARE Portafolio Digital || Karla Cevallos Menú Widgets B...

Articulo agiles metodos | PDF — 0.9%

https://www.slideshare.net/slideshow/articulo-agiles-metodos/73415610

Submit Search Upload Articulo agiles metodos • 0 likes•38 views T Tito Gonzalo Chipana HuarcusiFollow metodologia agil Read less Re...

conceptos, tipos y dimensiones del conocimiento.pdf — 0.8%

https://www.slideshare.net/slideshow/conceptos-tipos-y-dimensiones-del-conocimientopdf/254530470

Submit Search Upload conceptos, tipos y dimensiones del conocimiento.pdf • 0 likes•37 views M MerlethyCoronaFollow conceptos, tipos ...

ConceptoTiposYDimensionesDelConocimiento-2274043.pdf — 0.8%

https://www.slideshare.net/slideshow/conceptotiposydimensionesdelconocimiento2274043pdf/266720464

Submit Search Upload ConceptoTiposYDimensionesDelConocimiento-2274043.pdf • 0 likes•3 views L Luis Zambrano CedeñoFollow Concepto y ...

Results

Metodologias Agiles | PDF — 0.8%

https://www.slideshare.net/slideshow/metodologias-agiles-7464971/7464971

Submit Search Upload Metodologias Agiles • 1 like•740 views P puyol10Follow Report Share Report Share 1 of 8Download ...

Metodologias agiles | PDF — 0.8%

https://www.slideshare.net/ronaljulio347/metodologias-agiles-25426076

Submit Search Upload Metodologias agiles • 5 likes•27,552 views R ronaljulio347Follow Desesarrollo de los principales Metodogias Agi...

Metodologias agiles | PDF — 0.8%

https://www.slideshare.net/ronaljulio347/metodologias-agiles-25426076

Submit Search Upload Metodologias agiles • 5 likes•27,552 views R ronaljulio347Follow Desesarrollo de los principales Metodogias Agi...

Metodologias Agiles | PDF — 0.7%

https://www.slideshare.net/slideshow/metodologias-agiles-7464971/7464971

Submit Search Upload Metodologias Agiles • 1 like•740 views P puyol10Follow Report Share Report Share 1 of 8Download ...

Redalyc.Gestión del conocimiento como apoyo para la mejora de proceso... — 0.7%

https://www.redalyc.org/pdf/643/64328116.pdf

Capote, Joanna; LLanten Astaiza, Carlos Julian; Pardo Calvache, César Jesús; González Ramírez, Alberto de Jesús; Collazos, César Alberto

Ingeniería e Investigación ISSN: 0120-5609 revii_bog@unal.edu.co Universidad Nacional de Colombia Colombia Capote, Joanna; LLanten Astai...

actas.pdf — 0.7%

https://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf

Metodologías Ágiles en el Desarrollo de Software Alicante, 12 de Noviembre de 2003 Organización: Grupo ISSI (Ingeniería del Software y S...

Comparativa del Ciclo de Vida en Desarrollo de Software Híbrido EssUp v... — 0.6%

https://www.docsity.com/es/material-para-trabajo-1/8479515/

Prepara tus exámenes Consigue puntos Orientación Universidad Vende en Docsity Inicia sesión Regístrate Prepara tus e...

Results

Metodologías Ágiles para el Desarrollo de Software y Metodologias Para ... 0.6%

https://www.slideshare.net/jofese/metodologas-giles-y-metodologias-web-apra-el-desarrollo-de-software?ne...

Submit Search Upload Metodologías Ágiles para el Desarrollo de Software y Metodologias Para el desarrollo Web •Download as PPTX, PDF...

METODOLOGÍAS ÁGILES | PPT 0.6%

https://www.slideshare.net/slideshow/metodologas-giles-32383114/32383114

Submit Search Upload METODOLOGÍAS ÁGILES •Download as PPTX, PDF• 0 likes•1,698 views Humbert Ramirez JaramilloFollow Metodologías á...

METODOLOGÍAS ÁGILES | PPT 0.6%

https://www.slideshare.net/slideshow/metodologas-giles-32383114/32383114

Submit Search Upload METODOLOGÍAS ÁGILES •Download as PPTX, PDF• 0 likes•1,698 views Humbert Ramirez JaramilloFollow Metodologías á...

ABC de la gestión de proyectos: W para la gestión del conocimiento en lo... 0.6%

https://parm.com/es/abc-de-la-gestion-de-proyectos-w-para-la-gestion-del-conocimiento-en-los-proyectos/

Down-Menu*/ .nav li ul {position: absolute; padding: 20px 0; width: 300px;} ...

Gestión del conocimiento como apoyo para la mejora de procesos softwa... 0.6%

http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=s0120-56092008000100015

Services on Demand Journal SciELO Analytics Google Scholar H5M5 () ...

T010_45432217_T.pdf.txt 0.5%

https://repositorio.uncp.edu.pe/bitstream/handle/20.500.12894/5584/t010_45432217_t.pdf.txt;jsessionid=cf8...

Facultad de IngenierÃa de Sistemas Desarrollo de un aplicativo mÃ3vil para el acceso a la informaciÃ3n de los procesos judiciales en l...

T010_45432217_T.pdf.txt;jsessionid=5815B8D611FD282A7D887EBBE5FE133... 0.5%

https://repositorio.uncp.edu.pe/bitstream/handle/20.500.12894/5584/t010_45432217_t.pdf.txt;jsessionid=58...

Facultad de IngenierÃa de Sistemas Desarrollo de un aplicativo mÃ3vil para el acceso a la informaciÃ3n de los procesos judiciales en l...

Results

TECNICATURA SUPERIOR EN DESARROLLO DE SOFTWARE ... 0.5%

https://isp4lt-sfe.infd.edu.ar/sitio/wp-content/uploads/2021/12/propedeutico-tecnicatura-2022_completo.pdf

Analia

INSTITUTO SUPERIOR DE PROFESORADO No 63 NATALIA QUESSÚS 2 TECNICATURA SUPERIOR EN DESARROLLO DE SOFTWARE CUADERNILLO PROPEDÉUTICO INGRES...

Gestión del conocimiento en ingeniería de software 0.5%

https://www.recimundo.com/index.php/es/article/download/338/html?inline=1

Miryan Dorila Iza Carate a Gestión del conocimiento en ingeniería de software Knowledge Management in Software E...

(PDF) METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES 0.5%

https://www.researchgate.net/publication/299506242_metodologias_tradicionales_vs_metodologias_agiles

Roberth Figueroa-Diaz

Working PaperPDF AvailableMETODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES February 2007 February 2007 DOI:10.13140/RG.2.1.2897...

QA | Metodologías de Aseguramiento de la Calidad 0.5%

https://es.linkedin.com/pulse/qa-metodolog%c3%adas-de-aseguramiento-la

Servicios de Informática Profesional, S.A. (SIPSA)

Acepta...

Diferencias entre Metodologías Tradicionales y Ágiles #MetodologiasAgil... 0.4%

https://arevalomaria.wordpress.com/2011/11/15/diferencias-entre-metodologias-tradicionales-y-agiles-meto...

Ir al contenido Maria Eugenia Arevalo Lizardo El liderazgo es una oportunidad de s...

(PDF) Adoption of Knowledge Management Practices in Software Enginee... 0.4%

https://www.academia.edu/86457450/adoption_of_knowledge_management_practices_in_software_engineeri...

Dr Neeraj D Sharma

Academia.edu no longer supports Internet Explorer. To browse Academia.edu and the wider internet faster and more securely, p...

Wegrow: Los fundamentos de la gestión del conocimiento | Wegrow, mej... 0.4%

https://es.wegrow-app.com/articles/the-fundamentals-of-knowledge-management

Inicio Recursos Casos prácticosBlogSeminarios web y podcasts Nuestra misión Nuestro equipo Reserva una demo Reserva una demo ...

Results

Información conocimiento inteligencia y apendizaje.docx — 0.4%

https://cmapspublic3.ihmc.us/rid=1prvz3j5y-m35m3k-jmkr/informaci%c3%b3n%20conocimiento%20inteligen...

Mountain

Datos Representaciones de hechos o fenómenos materiales, o ideales -existentes en la psique, que es realidad en sí misma-, no tienen pred...

Cuando utilizar el modelo en espiral? – RESPUESTASRAPIDAS — 0.3%

https://respuestasrapidas.com.mx/cuando-utilizar-el-modelo-en-espiral/

Saltar al contenido RESPUESTASRAPIDAS Tu asistente digital Home Artículos Popular Tendencias Consejos út...

Anexos.pdf?sequence=2&isAllowed=y — 0.3%

http://repositorio.unicauca.edu.co:8080/bitstream/handle/123456789/5974/anexos.pdf?sequence=2&isallow...

Marco Conceptual para la implantación de Gestión del Conocimiento en un Programa de Mejora de Procesos Software en MiPyMEs DS ANEXOS Jo...

¿Qué es la gestión del conocimiento? La guía 2024 — 0.3%

https://www.getguru.com/es/reference/what-is-knowledge-management

Regístrate gratis | Iniciar sesión Producto Gurú de... Búsqueda de IA empresarial Búsqueda de conocimiento empresarial basad...

Introducción a las Metodologías de Desarrollo de Sistemas – Ingeniería d... — 0.3%

https://blogfaissalisi.wordpress.com/2018/11/05/introduccion-a-las-metodologias-de-desarrollo-de-sistemas/

Saltar al contenido Ingeniería de los Sistemas de Información Menú y widgets ...

Redalyc.Aplicando Gestión del Conocimiento en el Proceso de Mantenimi... — 0.3%

https://juliopezblog.wordpress.com/wp-content/uploads/2019/03/7-aplicando-gestic3b3n-del-conocimiento-e...

Vizcaíno, Aurora; Soto, Juan Pablo; García, Félix; Ruíz, Francisco; Piattini, Mario

Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial ISSN: 1137-3601 revista@aepia.org Asociación Española para la ...

Redalyc.Aplicando Gestión del Conocimiento en el Proceso de Mantenimi... — 0.3%

https://www.redalyc.org/pdf/925/92503110.pdf

Vizcaíno, Aurora; Soto, Juan Pablo; García, Félix; Ruíz, Francisco; Piattini, Mario

Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial ISSN: 1137-3601 revista@aepia.org Asociación Española para la ...

Results

(PDF) Adoption of Knowledge Management Practices in Software Enginee... 0.2%

https://www.academia.edu/6667377/adoption_of_knowledge_management_practices_in_software_engineerin...

kawaljeet singh

Academia.edu no longer supports Internet Explorer. To browse Academia.edu and the wider internet faster and more securely, p...

Actividad 1 - Conociendo el contexto del software 0.2%

https://skat.ihmc.us/rid=1z5sln6ps-41lqj7-1d9/actividad%201%20-%20conociendo%20el%20contexto%20del...

WARNING: JavaScript is turned OFF. None of the links on this concept map will ...

Fundamentos Teóricos Del Software - Mind Map 0.2%

https://www.mindomo.com/es/mindmap/fundamentos-teoricos-del-software-add6fb8a2b87431fb65f51836a...

...

Aplicando gestión del conocimiento en el proceso de mantenimiento del ... 0.2%

https://dialnet.unirioja.es/servlet/articulo?codigo=2229139

Ayuda ¿En qué podemos ayudarle? × Buscar en la ayuda Buscar Consultar la ayuda ¿En qué podemos ayudarle? ×...

Administración de desarrollo Flashcards | Quizlet 0.2%

https://quizlet.com/369130599/administracion-de-desarrollo-flash-cards/

hello quizlet HomeStudy tools Subjects Create Generate Log in Sign up Administración de desarrollo F...

(PDF) Gestión del conocimiento como apoyo para la mejora de procesos s... 0.2%

https://www.academia.edu/101709175/gesti%c3%b3n_del_conocimiento_como_apoyo_para_la_mejora_de_pr...

Carlos Alberto Mirabal Julian

Academia.edu no longer supports Internet Explorer. To browse Academia.edu and the wider internet faster and more securely, p...

(PDF) Gestión del conocimiento como apoyo para la mejora de procesos s... 0.2%

https://www.academia.edu/50443176/gesti%c3%b3n_del_conocimiento_como_apoyo_para_la_mejora_de_pro...

Joanna Capote

Academia.edu no longer supports Internet Explorer. To browse Academia.edu and the wider internet faster and more securely, p...

Results

IS__Libro_Pressman_7.pdf — 0.2%

http://artemisa.unicauca.edu.co/~cardila/is__libro_pressman_7.pdf

Ingeniería del software UN ENFOQUE PRÁCTICO SÉPTIMA EDICIÓN Roger S. Pressman, Ph.D. University of Connecticut MÉXICO • BOGOTÁ • BUE...

Ingenieria del Software. Un Enfoque Practico — 0.2%

https://tesuva.edu.co/phocadownload/ingenieria_del_software._un_enfoque_practico.pdf

Roger S. Pressman

00Pressman(i-xxx)prelim.indd ii 2/2/10 11:40:14 Roger S. Pressman, Ph.D. University of Connecticut MÉXICO • BOGOTÁ • BUENOS AIRES • CA...

Ingenieria del Software. Un Enfoque Practico — 0.2%

https://www.javier8a.com/itc/bd1/ld-ingenieria.de.software.enfoque.practico.7ed.pressman.pdf

Roger S. Pressman

www.FreeLibros.me Ingeniería del software UN ENFOQUE PRÁCTICO SÉPTIMA EDICIÓN Roger S. Pressman, Ph.D. University of Connecticut MÉ...

ESCUELA POLITÉCNICA NACIONAL - PDF Free Download — 0.2%

https://docplayer.es/184517730-escuela-politecnica-nacional.html

Iniciar la sesión ...

Metodologías de desarrollo de software | DSpace-CRIS @ UCA — 0.2%

https://repositorio.uca.edu.ar/handle/123456789/522?locale=en

Skip navigation English ...

Metodologías de desarrollo de software | DSpace-CRIS @ UCA — 0.2%

https://repositorio.uca.edu.ar/handle/123456789/522

Skip navigation English ...

Description: Metodologías de desarrollo de software — 0.2%

https://repositoriosdigitales.mincyt.gob.ar/vufind/record/riuca_de9409d2bc65ba527936e938ea5d93a0

Skip to content Argentina.gob.ar Presidencia de la Nación ...

Results

Modelo en Espiral: todo lo que necesitas saber - Ryte Wiki — 0.2%

https://es.ryte.com/wiki/modelo_en_espiral

CategoríasMarketing OnlineSEORedes SocialesUsabilidadMarketing MóvilAnalítica WebDesarrollo ryte.com Publicación de la página Estado...

Ventajas y desventajas del modelo de cascada | Blog Lucidchart — 0.2%

https://www.lucidchart.com/blog/es/pros-y-contras-de-la-metodologia-de-cascada

PINGDOM_CANARY_STRINGSkip to Content Otros blogsOtros blogs CategoríasCategorías Buscar en el blog de Lucidchart Encuentra algo inspi...

Métodos Ágiles de Programación: febrero 2016 — 0.1%

http://angelmedina26.blogspot.com/2016/02/

Métodos Ágiles de Programación ...

Conocimiento pedagogicos generales | PDF — 0.1%

https://www.slideshare.net/josemoises2012/conocimiento-pedagogicos-generales

Submit Search Upload Conocimiento pedagogicos generales •Download as DOC, PDF• 2 likes•17,117 views José Moisés Ramírez SolísFollow...

Libro_Pressman_7.pdf — 0.1%

http://www.javier8a.com/umb/analisis/libro_pressman_7.pdf

Ingeniería del software UN ENFOQUE PRÁCTICO SÉPTIMA EDICIÓN Roger S. Pressman, Ph.D. University of Connecticut MÉXICO • BOGOTÁ • BUE...

UAGRM - DIPLOMADO 2012 - GERENCIA DE PROYECTOS DE DESARROLLO D... — 0.1%

http://oscararuquipacolque.blogspot.com/2012/10/actividades-de-trabajo-1.html

UAGRM - DIPLOMADO 2012 - GERENCIA DE PROY...

Ciclos de vida del software — 0.1%

http://mundogeek.net/archivos/2004/05/20/ciclos-de-vida-del-software/

Ir al contenido Mundo Geek Vida inteligente en la geekosfera ...

Results

ISO 15930 - Electronic document file format for prepress digital data exch... 0.1%

https://diposit.ub.edu/dspace/bitstream/2445/123350/1/pe%cc%81rez-montoro%20(2008)%20gestio%cc%81...

Gestión del conocimiento en las organizaciones Gestión del conocimiento en las organizaciones Fundamentos, metodología y praxis Mario P...

ISO 15930 - Electronic document file format for prepress digital data exch... 0.1%

https://diposit.ub.edu/dspace/bitstream/2445/123350/1/pe%cc%81rez-montoro%20(2008)%20gestio%cc%81...

Gestión del conocimiento en las organizaciones Gestión del conocimiento en las organizaciones Fundamentos, metodología y praxis Mario P...

Modelos de desarrollo de software Flashcards | Quizlet 0.1%

https://quizlet.com/mx/785452137/modelos-de-desarrollo-de-software-flash-cards/

hello quizlet HomeStudy tools Subjects Create Generate Log in Sign up Modelos de desarrollo de software ...

Modelos de desarrollo de software | Quizlet 0.1%

https://quizlet.com/mx/461002821/modelos-de-desarrollo-de-software-flash-cards/

hello quizlet HomeStudy tools Subjects Create Generate Log in Sign up Modelos de desarrollo de software ...

Como hacer un modelo en espiral? – RESPUESTASRAPIDAS 0.1%

https://respuestasrapidas.com.mx/como-hacer-un-modelo-en-espiral/

Saltar al contenido RESPUESTASRAPIDAS Tu asistente digital Home Artículos Popular Tendencias Consejos út...

(PDF) A Review on Various Software Development Life Cycle (SDLC) Models 0.1%

https://www.researchgate.net/publication/312473242_a_review_on_various_software_development_life_cycle...

Suresh Seema

Home Software Computer Science and Engineering Software Development ArticlePDF AvailableA Review on Various Software Development L...

Proceso ágil para la mejora de procesos de software - Agile SPI - Process.p... 0.1%

http://repositorio.unicauca.edu.co:8080/bitstream/handle/123456789/5964/proceso%20%c3%a1gil%20para%...

Proceso Ágil para la Mejora de Procesos de Software: Agile SPI - Process UNIVERSIDAD DEL CAUCA César Jesús Pardo Calvache Luis Eduardo ...

Results

Gestión del conocimiento como apoyo para la mejora de procesos softwa... 0.1%

https://www.semanticscholar.org/paper/gesti%c3%b3n-del-conocimiento-como-apoyo-para-la-mejora-capote...

Skip to search formSkip to main contentSkip to account menuSemantic ScholarSemantic Scholar's LogoSearch 218,599,696 ...

(PDF) Gestión del conocimiento como apoyo para la mejora de procesos s... 0.1%

https://www.academia.edu/68539464/gesti%c3%b3n_del_conocimiento_como_apoyo_para_la_mejora_de_pro...

CESAR JESUS PARDO CALVACHE

Academia.edu no longer supports Internet Explorer. To browse Academia.edu and the wider internet faster and more securely, p...

Seema Kute - Google Scholar 0.1%

https://scholar.google.com/citations?user=22daguyaaaaj&hl=en

Loading... The system can't perform the operation now. Try again later. Citations per year Duplicate citations The following art...

---

arquitectura de software - Mind Map 0.1%

https://www.mindomo.com/es/mindmap/arquitectura-de-software-0b5c799803f148b7a9b3ed5073d8a2c5

...

Profesiograma del Puesto de Trabajo - Derecho Bancario, Financiero y Tri... 0.1%

https://infobancarios.es/profesiograma-del-puesto-de-trabajo/

Saltar al contenido Términos y Condiciones Política de Privacidad Contacto ...

Metodologías de desarrollo software | Blog Santander Open Academy 0.1%

https://www.santanderopenacademy.com/es/blog/metodologias-desarrollo-software.html

...

Metodologia Segun Pressman - PDFCOFFEE.COM 0.1%

https://pdfcoffee.com/metodologia-segun-pressman-pdf-free.html

Guest

Email: Login Register English Deutsch Español Français Português Hom...

Results

Ingenieria de Software — 0.1%

https://gc.scalahed.com/recursos/files/r161r/w25469w/ingdelsoftwarelibro9_compressed.pdf

Ian Sommerville

SOMMERVILLE La presente obra se actualizó al incorporar los siguientes cambios: >N [illegible] uevos capítulos sobre software ágil y sistemas embe...

Adoption of Knowledge Management Practices in Software Engineering ... — 0.1%

https://www.researchgate.net/publication/254021712_adoption_of_knowledge_management_practices_in_sof...

Neeraj Sharma

ArticleAdoption of Knowledge Management Practices in Software Engineering Organizations: A Survey of Software Engineers' Perception...

Repositorio de GRIAL: La sistematización del proceso de revisión del esta... — 0.1%

https://grialdspace.usal.es/handle/grial/2561?mode=full

Skip navigation ...

(PDF) Metodología para la revisión sistemática de literatura Disponible en — 0.1%

https://www.researchgate.net/publication/364152259_metodologia_para_la_revision_sistematica_de_literatur...

Francisco José García-Peñalvo

Home Accounting Economics Financial Economics Audit PresentationPDF AvailableMetodología para la revisión sistemática de literatur...

Repositorio de GRIAL: Cómo hacer una Systematic Literature Review (SLR) — 0.1%

https://repositorio.grial.eu/handle/grial/2253

Skip navigation ...

Repositorio de GRIAL: Systematic Literature Reviews como método de in... — 0.1%

https://repositorio.grial.eu/handle/grial/2880

Skip navigation ...

Systematic Literature Reviews como método de investigación — 0.1%

https://zenodo.org/records/7830939

García-Peñalvo, F. J.

Skip to main ...

Results

Resumen del taller "Metodología para la revisión sistemática de literatur... 0.1%

https://departamento-psicologia.pucp.edu.pe/noticias/resumen-del-taller-metodologia-para-la-revision-siste...

nosotros PresentaciónDocentesComisionesEquipo de TrabajoInfraestructura y Servicios novedades NoticiasEventos Revista de...

vzv73qvbdzb7zc4fcc2yukjj6m 0.1%

https://scholar.archive.org/work/vzv73qvbdzb7zc4fcc2yukjj6m

beta English عَرَبِيّ ...

Cómo gestionar eficazmente los equipos de proyectos de medición inteli... 0.1%

https://es.linkedin.com/advice/0/what-most-effective-team-management-practices?lang=es

Acepta...

Bioinformatica de laboratorio clinico medicina de precision y bioinforma... 0.1%

https://fastercapital.com/es/contenido/bioinformatica-de-laboratorio-clinico--medicina-de-precision-y-bioinfo...

Inicio Contenido Bioinformatica de laboratorio clinico medicina de precision y bioinformatica clinica reduciendo la brec...

Plagiarism Report Content

Universidad Veracruzana
Región: Coatzacoalcos
Licenciatura en ingeniería de software
La gestión del conocimiento en la ingeniería de software
Monografía
Presenta:
Leonardo Alejandro Méndez Miranda
Directora de trabajo recepcional:
Ma Teresa de la Luz Sainz Barajas
Elija el mes de Elija el año
"Lis de Veracruz: Arte, Ciencia, Luz"
Universidad Veracruzana
Universidad veracruzana
Región Coatzacoalcos
Licenciatura en ingeniería de software
La gestión del conocimiento en la ingeniería de software
Monografía
Presenta:
Leonardo Alejandro Méndez Miranda
Directora de trabajo recepcional:
Ma Teresa de la Luz Sainz Barajas
Dedicatoria
Este trabajo va dedicado a mis padres Osiel y Francelia que con su amor, cariño y apoyo me impulsaron a finalizar esta etapa de mi vida y que en todo momento me acompañaron.

Índice

Índice de Figuras

Resumen
La gestión del conocimiento en la industria del software enfrenta desafíos cruciales debido a la urgencia de la entrega de productos en un entorno de ritmo acelerado. A pesar del reconocimiento de su importancia, la implementación efectiva de la gestión del conocimiento en las organizaciones desarrolladoras de software aún es apenas aceptable. El conocimiento predominante en ingeniería de software es tácito, lo que dificulta su transformación en conocimiento explícito, y la falta de herramientas adecuadas agrava esta situación. Aunque los profesionales muestran disposición para adoptar nuevas tecnologías y compartir conocimientos a través de medios digitales, la falta de sistemas de depósito de conocimiento organizacional es evidente. Investigaciones en Colombia resaltan la necesidad de fortalecer estos procesos para obtener ventajas competitivas sostenibles y superar desafíos en un entorno dinámico. En conclusión, la gestión del conocimiento en la industria del software requiere un impulso significativo para alcanzar su máximo potencial y mejorar su efectividad. Siendo el objetivo mostrar lo valioso de la gestión del conocimiento en la ingeniería de software.
Palabras clave: Gestión del conocimiento (GC), ingeniería de software, proceso de desarrollo de software.
Introducción
Las características que la gestión del conocimiento ofrece como doctrina es que se busca que sea aplicada dentro de un campo como lo es la ingeniería de software, tal y como demuestran las investigaciones anteriores, es una idea que se ha estado buscando implementar de una manera, que, aunque pareciera fácil, ha venido con ciertas problemáticas imprevistas.
En una encuesta realizada por Sharma, Singh y Goyal *(2012)* mencionan que, en cuanto a los grupos organizacionales, en las pequeñas empresas, el 92% de los encuestados dice que sus organizaciones no proporcionan ningún repositorio de conocimiento formal ni sistemas de software (...) De las medianas y grandes empresas, el 93% y el 90% de los encuestados respectivamente afirman que las empresas SE no cuentan con ningún repositorio para almacenar conocimientos.
Aunque pareciera que los ingenieros de software no son conscientes de la existencia de esta disciplina, las encuestas realizadas por Sharma, Singh y Goyal *(2012)* demuestran lo contrario al mencionar que, en organizaciones pequeñas, el 39% de los encuestados conoce hasta cierto punto los conceptos de gestión del conocimiento. De manera similar, en las organizaciones medianas y grandes, el 54% y el 42% respectivamente saben algo sobre gestión del conocimiento.
La gestión del conocimiento debido a su enfoque podría ser fácilmente aplicado dentro del flujo de trabajo de los ingenieros de software, y que estos últimos tienen una leve noción de lo que esto es y de lo que podría ser de utilidad, pero a pesar de ello las empresas no deciden integrarlo dentro de la infraestructura de su empresa para mejorar el rendimiento de sus empleados.
De aquí surge el cuestionamiento, ¿Por qué las empresas no quieren o intentan integrar la gestión del conocimiento dentro de sus infraestructuras? Además ¿La situación de la empresa sería igual o mejor si se llega aplicar?
Siendo el principal objetivo de este escrito el exponer las razones del por qué la gestión del conocimiento está siendo aplicada o no dentro de las empresas de desarrollo de software y las repercusiones tanto positivas como negativas que alguna de estas situaciones se esté dando.
De acuerdo con una encuesta realizada por Sharma, Singh y Goyal *(2012)* a varios ingenieros de software sobre cómo se maneja la gestión del conocimiento dentro de sus empresas, más del 90% mencionan que sus empresas no cuentan con ningún método o medio que sirva para el almacenamiento del conocimiento, ya sea una empresa chica, mediana o grande.
Por lo que esto sería una razón para la investigación de este tema, averiguar por qué las empresas de desarrollo de software no implementan la gestión del conocimiento dentro de su infraestructura organizacional, a pesar de los beneficios que esta metodología podría traer para sus trabajadores y su ritmo de trabajo.
En la misma encuesta elaborada por Sharma, Singh y Goyal *(2012)*, se cuestionó a los ingenieros de software sobre que tanto conocen sobre la metodología de gestión del conocimiento y por lo menos la mitad de ellos tienen una breve noción de lo que trata esta técnica.
Esto sería otro motivo para ahondar en la relación entre la gestión del conocimiento con la ingeniería de software, aunque las empresas de desarrollo de software no han implementado un método o técnicas que ayuden al almacenamiento del conocimiento dentro de su infraestructura, los ingenieros de software ya cuentan con una breve noción sobre lo que este procedimiento trata.

Otra razón que se obtiene es según lo descrito por Khalid, Shehryar y Arshad *(2015)* que la gestión del conocimiento es muy beneficioso para las organizaciones ya que resuelve muchos problemas relacionados con el desarrollo de software y también ayuda a mejorar el proceso de software y la calidad del producto, lo cual da un motivo más para investigar la relación que podría existir entre estos elementos y los beneficios que podría traer una interacción más estrecha.
Es por esto por lo que esta investigación es viable, ya que existen diversos escritos, encuestas o documentos en internet o bibliotecas virtuales que cuentan con bastantes informes sobre la gestión del conocimiento, el desarrollo de software y una posible relación que estos podrían tener, revisando sus beneficios y dificultades que enfrentan las empresas al querer implementarlo en su estructura organizacional.
Además el presente estudio busca abordar el tema de la gestión del conocimiento en la ingeniería de software como sus aplicaciones, beneficios y problemáticas al momento de ser aplicada en empresas, pudiendo ser documentos, encuestas o investigaciones de categoría nacional o global, ya que es una manera más fácil de conseguir información y al ser enfocados de esta manera, se busca un contexto más actual o de años recientes para poder dar con los datos que se desean obtener, con un alcance descriptivo y de un análisis no experimental ya que en este estudio se busca más recabar información más que ponerla en práctica en un ambiente controlado además de que los datos serán manejados de manera cualitativa ya que estamos observando los distintos elementos de una manera objetiva y concisa los datos que se quieren tener y mediante esta investigación pienso que podría ser una metodología que como profesionista podría sugerir aplicar dentro de una empresa en la que vaya a laborar para mejorar su rendimiento y eficiencia.
De igual manera se revisarán algunas de las investigaciones anteriores que se han realizado de esta temática para ver que tanto han aportado cada uno de estos escritos, así como obtener un mejor contexto de lo que se tratara en este escrito.
Una investigación llamada *"Aplicando gestión del conocimiento en el proceso de mantenimiento del software"* realizada por Vizcaíno, A. *(2006)*, en el cual menciona cómo los procesos de desarrollo y mantenimiento de software generan mucha información diariamente, pero esta información no siempre se gestiona de manera efectiva, por lo que puede llevar a problemas como la repetición innecesaria de trabajo y la falta de aprovechamiento del *"capital intelectual"* de una organización.
Del mismo modo Vizcaíno, A. *(2006)* subraya que esta problemática es especialmente grave en el caso del mantenimiento de software, que se considera el proceso más costoso y largo del ciclo de vida del software y que el conocimiento que posee un ingeniero de mantenimiento evoluciona constantemente debido a la naturaleza del mantenimiento, en donde cada fase del proceso genera nueva información que los ingenieros internalizan y a la que añaden su experiencia, transformándola en conocimiento tácito.
Esto con base a que cada ingeniero de mantenimiento posee un conocimiento que puede ser útil para los demás miembros del equipo. Por lo tanto, es esencial fomentar la colaboración y la comunicación entre los ingenieros de mantenimiento, siendo el conocimiento un elemento clave para mejorar el proceso de mantenimiento de un software.
Otra investigación realizada en Colombia por parte de Capote, J.*(2008)*, nombrada como *"Gestión del conocimiento como apoyo para la mejora de procesos software en las micro, pequeñas y medianas empresas"*, en la cual expone la importancia de un programa de mejora de procesos de software (SPI) en empresas de software, con el objetivo de mejorar la satisfacción del cliente, obtener productos de alta calidad y generar lecciones aprendidas. Este programa se basa en el trabajo en equipo, la colaboración y la comunicación efectiva. También se destaca la importancia de la gestión del conocimiento en estos procesos, ya que son intensivos en conocimiento y generan conocimiento y experiencia necesarios para la toma de decisiones.
Adicionalmente el autor Capote, J.*(2008)* de este articulo enfatiza la importancia de las MiPyMEs en Colombia debido a su gran impacto económico y a sus contribuciones al PIB del país. Sin embargo, estas empresas a menudo tienen dificultades para acceder a recursos financieros, procesos innovadores y nuevas tecnologías para mejorar su productividad y la calidad de sus productos.
Por último, este autor agrega a su investigación una forma de buscar desarrollar un marco conceptual que resulte de la selección de un modelo de gestión del conocimiento apropiado y adaptado para ser introducido en un programa SPI en las MiPyMEs, permitiendo la integración de estas dos áreas de una manera ágil.
Otra investigación titulada *"The role of knowledge management in global software engineering", realizada por Khalid (2015) en la cual trata sobre como la Gestión del Conocimiento en la Ingeniería de Software es un campo de estudio que se centra en cómo identificar, organizar, almacenar y difundir información dentro de una organización de software y que puede ser una herramienta valiosa para resolver problemas comunes en el desarrollo de software, como la falta de acceso a información relevante o la dificultad para compartir conocimientos entre los miembros del equipo.*
Así mismo, una investigación realizada Sharma (2012), llamada "Adoption of Knowledge Management Practices in Software Engineering Organizations: A Survey of Software Engineers", en donde menciona que las empresas suelen confiar en que sus empleados poseen conocimientos especializados que pueden aplicar a las tareas que se les asignan. Sin embargo, hay un problema importante: ese conocimiento no es propiedad de la empresa, sino de los empleados. En otras palabras, los empleados son los dueños y controladores de ese conocimiento.
Por ultimo la investigación buscada fue la elaborada por Dingsøyr *(2009)* "What Do We Know about Knowledge Management? Practical Implications for Software Engineering,*", en donde trata sobre que en la última década, ha habido muchos debates sobre la economía del conocimiento, la importancia del trabajo*

*intensivo en conocimiento y cómo las empresas de la sociedad del conocimiento pueden beneficiarse de una mejor gestión de sus conocimientos y que la gestión del conocimiento (KM), según sus principales defensores, Thomas Davenport y Lawrence Prusak, se define como "*un método que simplifica el proceso de compartir, distribuir, crear, capturar y comprender la situación de una empresa".

Durante este escrito se observara la posible unión entre la gestión del conocimiento con la ingeniería de software para ser aplicada como una doctrina o cultura que puede desarrollarse dentro de un ámbito empresarial enfocada al software, teniendo como definición de la gestión del conocimiento por parte de Rodríguez, Pedraja, Muñoz, Araneda *(2022)* como la doctrina que surge con la promesa de aprovechar el capital intelectual de las organizaciones, garantizando un gran impacto en ámbitos estratégicos como la realización de nuevos productos o incluso en la institución de estrategias de innovación.

1. Fundamentos conceptuales de la gestión del conocimiento

Para comprender primero lo que la gestión del conocimiento ofrece como metodología y profundizar en este tópico, es necesario conocer el contexto que la forma así mismo como los conceptos que se relacionan directamente con esta doctrina, permitiendo entender el cómo y dónde surge, así como el fundamento o bases de lo que esta propone como metodología y su función.

1.1 Dato, información y conocimiento

Para poder entrar en contexto en cuanto a la gestión del conocimiento a que se refiere, se deben de entender algunos de sus conceptos básicos que se relacionan con este ámbito, siendo estos el dato, la información y el conocimiento, teniendo que aclarar que se llega a confundir sus conceptos como uno solo, a pesar de que su significado y su uso los hace diferenciar entre ellos, y a pesar de esas diferencias, están estrechamente relacionados.

La relación entre estos tres tópicos se puede explicar perfectamente haciendo una analogía con una escalera, denominado el escalón más bajo como datos, el medio como la información y por ende, el último sería el del conocimiento, por lo que irían de menos a más, en cuanto a su refinamiento.

El primer concepto para revisar, además de ser el primer escalón de la escalera, tomando en cuenta la analogía mencionada anteriormente, sería el del dato, el cual Montuschi *(2001)* menciona en su artículo de la siguiente manera: *"Así se ha dicho que datos son la materia prima en bruto, que pueden existir en cualquier forma (utilizable o no) y que no tienen un significado por sí mismos". (p. 25)*

En este sentido, siendo más específicos sobre el concepto de dato, de acuerdo con Rus y Lindvall *(2002)* afirman lo siguiente: *"Los datos consisten en hechos discretos y objetivos sobre eventos, pero nada sobre su propia importancia o relevancia; es materia prima para crear información." (p. 10).*

Otro concepto que tenemos de datos es el otorgado por Nuñez *(2004)*, en el cual menciona lo siguiente: *"Los datos son representaciones de hechos o fenómenos materiales o ideales. -existentes en la psique, que es realidad en sí misma-" (p. 03).*

Por lo tanto, se entiende como a dato, el conjunto de observaciones objetivas de un determinado hecho o situación, diciendo solamente una parte de la realidad en la que se basan, sin explicar el porqué de las cosas, es algo que va a lo concreto.

Pasando al siguiente concepto, este sería el de información, el cual llega a ser confundido con el del conocimiento, ya que se les suele dar el mismo significado provocando esto confusión al momento de querer entender su función o lugar dentro de esta relación de conceptos.

En lo que se refiere a la información, *(Rus y Lindvall, 2002)* esta se define como datos que se organizan y se transforman de tal manera que puedan ser útiles para los usuarios finales, durante el proceso de toma de decisiones; es decir, la información se puede entender como un mensaje que informa al quien lo recibe, los datos deben generar en el receptor impacto en sus juicios o cambiar la manera en que percibe algo y será decisión de la persona que recibió el mensaje si en realidad se le informó o simplemente fue ruido que no le sirvió para nada.

En su artículo, Núñez *(2004)* define a la información como el resultado del proceso comunicativo que convierte el conocimiento en una forma accesible para otros. Sin embargo, este proceso y su producto, la información, no garantizan la realización completa del acto de comunicación y para que se considere un proceso de comunicación pleno, es importante que el producto fluya de manera bidireccional entre los participantes, permitiendo tanto la transmisión directa como la recepción de mensajes, y no de manera unilateral.

Dicho de una manera más simple por Montuschi *(2001)* sería de la siguiente manera: *"(...) la información son los datos que tienen "valor"* y que el valor informativo depende del contexto". *(p. 26)*. Por ende, se entiende que la información es un dato que ha adquirido un significado al brindarle un entorno o contexto que lo ayuda a dar más valor a su contenido, de tal manera que hasta podría afectar a la persona que reciba dicha información.

Pero esto no quiere decir que toda la información que existe en el mundo es útil y buena para el individuo que busca aprender algo, más en esta época que la creación de información falsa o mala se puede hacer con mayor facilidad, si no que se debe de tener cuidado con lo que se escoge, dicho de una manera más específica por Montuschi *(2001)* de la siguiente manera:

Se ha señalado con todo acierto que mucha de la información que hoy nos inunda es mala información, información falsa que surge por todos lados, (...) y que se mueve rápido gracias a las nuevas tecnologías y a los manipuladores de la información. El importante problema adicional que esto presenta es el de poder distinguir la buena de la mala información, para lo cual no existen recetas fáciles. *(p. 29)*.

El último concepto para revisar es el conocimiento el cual para Rus y Lindvall *(2002)* establecieron su significado de la siguiente manera:
El conocimiento es más amplio que los datos y la información y requiere la comprensión de la información. No solo está contenido en la información, sino también en las relaciones entre los elementos de información, su clasificación y los metadatos (información sobre la información, como quién ha creado la información). *(p. 10)*.
Otro concepto, que es dado por Núñez *(2004)*, hace mención que el conocimiento es un proceso dinámico y multifacético, que abarca desde el nivel personal hasta el social, organizacional y grupal además incluye la percepción, comprensión, reelaboración creativa, concepción de aplicaciones y transformación de la información con fines comunicativos, derivada de diversas fuentes y soportes.
Luego entonces, se puede concebir como la mezcla de varios elementos, como puede ser la experiencia, los valores, la información y saber hacer las cosas, todo esto lo adquiere una persona y que lo termina desarrollando en su mente, siendo esto parte de la complejidad que hay dentro de la mente humana.
De ahí es que el conocimiento sea un elemento crucial dentro de las organizaciones, tal y como explica Ciprés *(2004)* en su artículo, en el cual establece que este último puede afectar la naturaleza de las decisiones de inversión que se pueden llegar a tomar para los recursos, además de que este cambia la naturaleza del trabajo y la propiedad, permitiendo atraer a nuevos trabajadores que vengan con ideas o proyectos interesantes que beneficien a la empresa en algún apartado de su infraestructura.
La información y los datos son elementos que ayudan al crecimiento del conocimiento como elementos primarios para la nutrición de esta, pero el que exista una gran cantidad de estos elementos no quiere decir que todos los datos e información que se pueda adquirir, sirva para el crecimiento del conocimiento de un individuo, sino que es muy posible que esta se pierda al no ser necesaria, además de consumir valioso tiempo que podría ser utilizado para adquirir conocimiento, tal como dice Montuschi *(2001)* en su artículo.
Además, Núñez *(2004)* también en su artículo resalta las diferencias entre estos dos conceptos en el que comenta que, aunque los datos suelen carecer de significado por sí mismos, este carácter de falta de significado es relativo en el contexto humano por lo que implica que no existe un límite claro entre datos e información, sino un continuo de grados de significado y que dependiendo del contexto, algunos datos pueden adquirir significado y convertirse en información e incluso haciendo que algunos datos sean informativos en ciertas circunstancias y no en otras.
Otras diferencias a resaltar entre conceptos, es la de conocimiento e información, en donde algunos autores los ven como un stock o un flujo respectivamente, como comenta Montuschi *(2001)* en su artículo, donde ven al conocimiento como una estructura compleja y que cuenta con partes conectadas de varias maneras con ataduras diversas, mientras que a la información la identifican como mensajes o señales que *"atacan"* esta estructura de manera continua, y que pueden o no contribuir de manera significativa a esa estructura.
Como se puede observar la diferencia entre datos, información y conocimiento es notoria, se suele confundir sus conceptos ya que es frecuente que en bibliografías especializadas o diccionarios confundan o mezclen los significados en uno solo, tal como menciona en Montuschi *(2001)* en su artículo, por lo que es importante remarcar sus desemejanzas ya que son ideas fundamentales para entender el proceso que se realiza en la metodología de la gestión del conocimiento.
En resumen, el dato, la información y el conocimiento han sido conceptos que su significado se suele mezclar o confundir entre ellos, pero realmente cada uno tiene su propio significado y propósito de ser, sin embargo, eso no quiere decir que sean excluyentes entre sí, al contrario, guardan una estrecha relación entre estas ideas.

1.2 Tipologías de conocimiento

Después de haber estudiado los primeros conceptos, se puede profundizar en el concepto que más resalta entre ellos, siendo el del conocimiento, por lo que es necesario observar los distintos tipos de conocimiento que existen y los beneficios que ofrecen cada uno al adaptarse mejor a cada forma de adquirir los saberes.
El conocimiento se obtiene mediante un individuo que desea entender la realidad, es por ello por lo que surgen distintos tipos de conocimientos con base a la manera en que este se adquiere, siendo algo planteado por Diferenciador *(2022)* en su artículo.
Según cada autor tiene en mente sus propias tipologías o clasificación del conocimiento según el modo de como este se recolecta o mediante que procesos o actividades son necesarias para alcanzar esos saberes, por lo que se revisaran distintas tipologías propuestas por diferentes autores.
El primer tipo de conocimiento a revisar es uno conocido como conocimiento empírico, el cual según es explicado por Diferenciador *(2022)* en su artículo como un tipo de conocimiento que se adquiere mediante la observación y la interacción con el entorno, por así decirlo, surge de la experiencia.
Otra tipología propuesta, es la del conocimiento generativo, el cual es explicado en el artículo de Núñez *(2004)* como el proceso y los resultados de crear nuevos conocimientos mientras se resuelven problemas o se identifican alternativas a nuevas propuestas u oportunidades.
El conocimiento científico es otra tipología propuesta, en el cual Diferenciador *(2022)* menciona que este se relaciona con la lógica y el pensamiento crítico y analítico, el cual se origina en base a hechos verificados y analizados por algún método o metodología.
Otra tipología presentada y que se llega a confundir con el concepto de habilidad, es el conocimiento operacional, el cual trata, según explica Núñez *(2004)* en su artículo, sobre las formas prácticas de aplicar las metodologías y métodos, de igual manera haciendo hincapié en su diferencia con la habilidad ya que el

conocimiento operacional consiste en saber cómo se deben realizar las operaciones, mientras que la habilidad consiste en saber realizar dichas operaciones, en el tiempo requerido, con los parámetros de calidad establecidos y con éxito.
De igual manera, otro tipo de conocimiento que sufre este tipo de confusión es el conocimiento instrumental y como explica Núñez *(2004)* el cual se refiere al dominio de la variedad de instrumentos disponibles para la aplicación de técnicas y operaciones, y que debe distinguirse entre este tipo de conocimiento a las habilidades instrumentales, que consisten en saber elaborar y aplicar los instrumentos, consecuentemente.
Una tipología propuesta es la del conocimiento analítico, en el cual Diferenciador *(2022)* en donde explica en su artículo como este conocimiento surge de la observación y la reflexión lógica para entender un elemento abstracto de la realidad, siendo en contraste al conocimiento intuitivo, el cual está más relacionado a la intuición o a la experiencia, sin haber tenido un razonamiento previo.
Otra categoría propuesta por Núñez *(2004)* es la del conocimiento conceptual el cual cubre el conocimiento de teorías, leyes, regularidades, conceptos y nociones, así como sus interrelaciones de significado.
Una clasificación distinta es el conocimiento filosófico, el cual según lo explicado por Diferenciador *(2022)* este surge de lo que una persona reflexiona sobre las ideas, conceptos, preguntas, entre otros temas o tópicos que sean desconocidos para la persona en sí.
Existe correlación entre estos vocablos, sobre todo en su concepto u obtención, pero a la hora de ponerlos en práctica es donde salen a relucir sus diferencias sustantivas, debido a las características propias de cada contexto.
El anterior listado fue algunas de las clasificaciones que se le da al conocimiento que es de carácter más común, pero varios autores difieren o proponen distintos tipos que puede haber, en base a lo que pueden ofrecer o el medio en cómo se aprenden, siendo una propuesta dada por Ciprés *(2004)*, el cual es la siguiente tabla donde junto varias opciones dadas por distintos autores, habiendo algunas que ya se mencionaron *(Ver tabla 1)*:
Tabla 1.-Taxonomías del conocimiento y ejemplos
Tipos de conocimiento
Definiciones
Ejemplos
Tácito
Conocimiento que está en las acciones, experiencia y forma parte de un contexto especifico.
Formas de relacionarse con un cliente especifico.
Tácito cognitivo
Modelos mentales
Creencias individuales sobre relaciones causas-efecto
Tácito técnico
Know-how aplicable a un trabajo especifico
Habilidades de cirugía
Explicito
Articulado, conocimiento generalizado
Conocimiento sobre los principales clientes de una zona
Individual
Creado por e inherente al individuo
Percepciones conseguidas a través de un proyecto concluido
Social
Creado por e inherente a las acciones colectivas de un grupo
Normas de comunicación entre grupos
Declarativo
Know-about
Que medicamento es apropiado por una enfermedad
De procedimiento
Know-how
Como administrar determinado medicamento
Causal
Know-why
Comprender por qué los medicamentos funcionan
Condicional
Know-when
Comprender cuando prescribe un medicamento
Relacional
Know-with
Comprender como interactúa un medicamento con otros grupos de medicamentos
Pragmático
Utilidad de un conocimiento para una organización
Mejores prácticas, estructura de negocios, experiencia en proyectos, dibujos de ingeniería, informes de mercado

Fuente: Ciprés, M. S., & Llusar, J. C. B. *(2004)*. Concepto, tipos y dimensiones del conocimiento: configuración del conocimiento estratégico. Revista de economía y empresa, 22(52), 175-196.

De igual manera, podemos observar distintitas versiones o propuestas de la tipología del conocimiento, variando de autor en autor y como menciona Ciprés *(2004)* en su artículo, que las variadas clasificaciones dadas pueden dar una idea de disparidad de los tipos de conocimiento y de la falta de unanimidad en su clasificación y características, lo cual invita a la ambigüedad.

En conclusión, existen diversas clasificaciones para el conocimiento, esto basado en el modo de obtenerlo y para el propósito en que será usado, además diversos autores difieren en cuales serían los principales o como referirse a ellos, pero en términos generales, algunos permanecen igual sin importar el autor del que hable de ellos.

2. Gestión del conocimiento en las organizaciones

Una vez entendiendo las bases de lo que conforma como un antecedente para la metodología de la gestión del conocimiento, ya se puede profundizar en lo que esta doctrina ofrece, que la componen, como funciona, donde se puede aplicar, así como las ventajas y desventajas al aplicarla dentro de un ámbito empresarial general, así como entender sus propios conceptos que esta posee.

2.1 Definición de gestión del conocimiento en las organizaciones

Actualmente, el conocimiento se ha vuelto uno de los pilares dentro de la organización de las empresas, siendo la clave del éxito de cualquier empresa la creacion, uso y difusión de este, ya que, gracias a esto, les permite ser innovadores en los caminos, soluciones o servicios que le pueden ofrecer a sus clientes, creando un valor que la diferencian de las otras que conforman su competencia y por ende destacar sobre ellas.

El conocimiento es un elemento que se quiera o no, está presente en una organización, el cual constantemente está en transmisión ya que es algo normal en la vida natural de una empresa, pero a pesar de la gran cantidad de flujo que circula de conocimiento dentro de la misma, este puede que no sea usado o aprovechado en su totalidad.

Esto se debe principalmente a que todo este conocimiento no se encuentra en un solo lugar, si no que está disperso en distintas áreas dentro de la organización y es lo que dificulta su acceso, lo que provoca lentitud, dificultad o doble esfuerzo al querer solucionar una determinada situación, aunado a que la principal fuente, puede radicar en una sola persona.

Para que una empresa pueda superar esta problemática y alcanzar un estado de éxito, existe esta disciplina conocida como la gestión del conocimiento (GC) y como Rodríguez, Pedraja, Muñoz, Araneda *(2022)* mencionan, esta se puede vislumbrar como una doctrina que surge con la promesa de aprovechar el capital intelectual de las organizaciones, garantizando un gran impacto en ámbitos estratégicos como la creacion de nuevos productos o incluso en la institución de estrategias de innovación.

Una definición más exacta de lo que es la gestión del conocimiento, es otorgada por Rodríguez *(2022)*, afirmando que: *"La gestión del conocimiento es un proceso sistemático que permite identificar, crear, adquirir, aprender, compartir y usar el conocimiento y las experiencias para lograr los objetivos organizacionales, lograr eficiencia y eficacia, y para conseguir ventajas competitivas sostenibles." (p. 268).*

La idea del concepto de GC ya lleva un tiempo que se estableció, siendo específicamente entre mediados de los años 80´s, surgiendo debido a que se quería derivar conocimiento del denominado diluvio de información, además en la década de los 90´s, se le empieza a relacionar más con elementos tecnológicos e informáticos, esto debido a las nuevas tecnologías que surgieron en ese tiempo, como la internet.

Esta disciplina, se divide en tres fases durante su proceso de desarrollo, las cuales son nombradas por Rodríguez et. al *(2010)* como 1) crear conocimiento, lo que involucra la exploración, mixtura y la manifestación del conocimiento mediante el hacer; 2) compartir conocimiento, mediante la transferencia y compartición de este a través de los colaboradores que lo poseen; y 3) aplicar conocimiento, lo que implica convertir el mismo en un resultado valioso para la organización.

Después de revisar lo que la gestión del conocimiento es, de lo que trata de manera general y su origen, se hace posible recabar la idea de general de que esta disciplina fue creada con el objetivo de compartir el conocimiento que alguien dispone sobre un tema a otros que carecen de ese conocimiento o no lo poseen, ya sea para dar soluciones a problemáticas antes resueltas u otras situaciones, siendo tan versátil que se puede adaptar a cualquier ámbito o campo.

2.2 El aprendizaje en la gestión del conocimiento

Cuando una persona ignora cómo realizar una determinada acción o actividad para una situación o cuándo no se sabe sobre un tema, se recurre a un proceso de aprendizaje para obtener el conocimiento que hace falta para llevar a cabo las actividades relacionadas a ese ámbito que es desconocido y del cual se busca aprender.

Asimismo, Gutiérrez, *(2008)* afirma que *"es el proceso mediante el cual se consigue adquirir el conocimiento de alguna cosa por medio del estudio o la experiencia". (p. 67).* Mismo que puede dividirse en cuatro etapas, según explica Gutiérrez *(2008)*; siendo la primera la del desconocimiento o la del pre-aprendizaje, donde el sujeto no sabe o no tiene conocimiento de lo que desconoce, aunque suene raro; es en la segunda fase donde ya empieza el aprendizaje, aquí el sujeto ya tiene una pequeña noción del tema que le era desconocido para él. En la tercera, es donde el individuo entra de lleno en el proceso de aprendizaje, donde ya puede realizar tareas con base al tema que le era desconocido, en el que ahora, ya se ha formado y va adquiriendo más habilidades para poder realizar faenas que estén relacionadas con este ámbito, se puede decir que es cuando el individuo es un aprendiz. La última del proceso, es donde el sujeto alcanzó un grado de maestría o de

maestro, es decir, la persona ya es capaz de ejecutar labores más difíciles obteniendo buenos resultados y sin mucho esfuerzo, como consecuencia de que el conocimiento ya lo ha vuelto suyo.
Por otra parte, también puede y debe ser relacionado con el conocimiento organizacional, debido a que como menciona Gutiérrez *(2008)*, al representar uno de los pilares para garantizar el buen funcionamiento y productividad de una empresa, el aprendizaje resulta de gran apoyo para la adaptabilidad de la empresa, ayuda a corregir o compensar problemáticas, sobre todo cuando esta se enfrenta a elementos externos e incontrolables; mientras que su ausencia, puede incidir en que la entidad sufra de episodios de fuga o de obsolescencia
Gutiérrez *(2008)*, se refiere a los episodios de fuga, como el abandono de la empresa por alguno de los colaboradores, justo antes de que sus conocimientos hayan sido convertidos en corporativos y por ende, representan la pérdida de dicho activo y disminución en el inventario del conocimiento organizacional.
La obsolescencia, por su parte, está presente cuando el conocimiento organizacional ya no es óptimo y pierde potencial, debido a que se ha vuelto obsoleto e inoperable, causando así, un detrimento en el valor para la propia empresa *(Gutiérrez, 2008)*.
Cuando en una organización existe una gestión del conocimiento, el papel del aprendizaje se vuelve tres veces más importante de lo que ya es, debido a que al contribuir al conocimiento individual, como efecto dominó y en esa misma proporción, crecerá el conocimiento organizacional; y gracias a los medios de difusión por parte de la GC, será posible que varios integrantes de la empresa puedan acceder a este conocimiento provocando que el nivel de conocimiento individual robustezca las contribuciones a la empresa, haciendo referencia a lo que Gutiérrez *(2008)* menciono en su momento.
En conclusión, se determina que el proceso de aprendizaje es un pilar fundamental dentro de la metodología de la gestión del conocimiento, ya que es la acción que permite recabar la información que uno no conoce para transformarlo en tu conocimiento y en una doctrina enfocada en la obtención de conocimiento es una actividad importante que será realizada por los que busquen aprender algo nuevo y necesario, además de que es algo constante durante todo este proceso.
2.3 Metodología para la gestión del conocimiento
Para poder integrar de manera efectiva un programa de gestión del conocimiento dentro de una empresa, se deben de tener en cuenta seis factores muy importantes, el humano, el cultural, la estrategia, los procesos, los contenidos y el contexto, los cuales son necesarios para poder implementar un plan que se adecue a las necesidades de la empresa y de la mejor manera.
Una propuesta de metodología es la que plantea Gutiérrez *(2008)*, la cual está conformada por tres aspectos, el primero, llamado de análisis, es donde se establecen el conjunto de operaciones a realizar antes de que el programa de GC sea instaurado; en el segundo llamado diseño, se busca realizar tres operaciones que son importantes para la aplicación del programa de GC: la planificación del proceso de gestión de contenidos, el diseño conceptual de los recursos documentales y la estructura de la comunidad de gestión del conocimiento. El último, se designa como implementación, justo aquí es donde se lleva a cabo una prueba piloto, con miras a que posterior se pueda migrar al programa general de GC, aplicándolo a la totalidad de la empresa y al desarrollo integral de las herramientas tecnológicas.
Todas estas fases se enfocan en analizar de manera exhaustiva y clara estructura de la organización, diseñar el plan que sea más adecuado, que ayude a la recolección y distribución del conocimiento de la mejor manera, para garantizar la exitosa aplicación del sistema en la totalidad de la empresa.
3. La ingeniería de software
Una vez entendido el concepto e idea de lo que conforma a la gestión del conocimiento, es momento de revisar el otro concepto importante que conforme a este estudio: la ingeniería de software. La cual ha empezado a ser más destacada y mencionada en estas últimas décadas, la cual está en constante crecimiento y actualización en cuanto información y proceso de aprendizaje requiere, por lo que sería importante de igual manera como se hizo con la metodología seria pertinente revisar lo que es y conforma a esta profesión.
3.1 Concepto e importancia
La ingeniería de software, según nos explica Piattini *(2016)*, es una disciplina que surgió desde hace bastante tiempo, siendo oficialmente nombrada así en el año de 1968, la cual se relaciona con el desarrollo o creación de programas de software, que, para entender su funcionamiento, se debe entender el producto que desarrollan primero, el software.
Una definición del software que es propuesta por Pressman *(2010)*, en el cual explica que es un producto construido por programadores que de igual manera se les encomienda el mantenimiento, este producto contendrá programas que se realizan en un equipo de cualquier tamaño o arquitectura.
El software ya no es un elemento ajeno para las personas en su vida cotidiana, está presente en el teléfono inteligente, en televisores, en computadoras, en la gran variedad de equipos electrodomésticos, por mencionar algunos, y para que quede más claro, Sommerville *(1998)* lo define como el conjunto de programas y documentación asociada, orientados a la creación, diseño y despliegue de productos de software para un cliente particular o para un mercado en general.
Como se puede observar en las distintas definiciones, el software no solo se refiere a los programas ejecutables, sino que también a la documentación que se realiza en ellos y para ellos, estableciendo así, las bases para el desarrollo de un producto de calidad, siendo el ingeniero de software el artífice de ambos quehaceres y mimetizando así, la ingeniería, con el software.

Los encargados del desarrollo de un producto de software se les conoce como ingenieros de software, como explica Panteleo *(2015)* en su libro, esta disciplina se entiende como el manejo de una perspectiva sistemática, disciplinada y cuantificable para la creacion, operación y mantenimiento de un software.
En este orden de ideas, Sommerville *(1998)*, le da en su libro la siguiente definición: *"La ingeniería de software es una disciplina de ingeniería que se interesa por todos los aspectos de la producción de software, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación." (p. 7).*
En otras palabras, el ingeniero de software es el especialista encargado de crear un sistema de software, abarcando desde el levantamiento de los requerimientos, el diseño del programa, la codificación y despliegue de este, así como su respectiva documentación, hasta que dicho producto está listo para ser utilizado por el o los usuarios finales y/o requiera de mantenimiento *(Sommerville, 1998)*.
Siendo el principal objetivo de esta disciplina, como dice Sommerville *(1998)* en su obra, ya que desea apoyar al desarrollo de software profesional y desplazar la programación individual, ya que se incluyen en el proceso de desarrollo técnicas que apoyan a la especificación, el diseño y la evolución de un programa, aspectos que no son usados en la programación personal.
Una vez adentrado un poco más en el contexto de este ámbito, resulta más sencillo entender la importancia que tiene la disciplina en cuestión en la sociedad actual, en las organizaciones y, sobre todo, en la vida diaria.
La importancia del software, como menciona Pressman *(2010)* en su libro, recae en que este influye a casi todos los aspectos de la vida actual, inclusive ha alcanzado al comercio, cultura y actividades cotidianas, así mismo la ingeniería es software tiene la misma importancia ya que permite construir sistemas complejos en un tiempo adecuado y con buena calidad.
Ahondando en el tema de la relevancia del software, Pressman*(2010)* comenta en su libro que el software alcanzo su grado de importancia gracias al manejo y distribución del producto más importante de la actualidad, la cual es la información, esto gracias a que transforma los datos personales a un modo que puedan ser más útiles en un contexto local, además de que administra la información de negocios para mejorar la competitividad, provee una vía para las redes mundiales de información (la internet) y brinda los medios para obtener información de distintas maneras.
Otro famoso autor que menciona la relevancia actual del software es Sommerville *(1998)* el cual comenta que en el mundo moderno es inaudita la idea de trabajar sin el software, ya que las infraestructuras nacionales o servicios públicos se manejan por sistemas en computadoras, inclusive el sistema financiero, la fabricación y la distribución industrial ya se encuentran completamente computarizadas, alcanzando incluso al entrenamiento con el uso de software en los diversos medios de entretenimiento, siendo la ingeniería de software fundamental para el funcionamiento de las sociedades modernas.
Hablar sobre la ingeniería del software, es hablar del software en sí y como explica Piattini *(2016)*,la industria del software no surgió en esta última década, si no que ya lleva casi setenta años de existencia como industria y conforme ha pasado el tiempo se ha ido mejorando el proceso de desarrollo, para que se adapte a las necesidades de los usuarios y las nuevas tecnologías para poder entregar un producto robusto que permita una adecuada fiabilidad, idoneidad funcional, eficiencia en el desempeño, seguridad, compatibilidad, mantenibilidad, transferibilidad, accesibilidad y usabilidad.
Para concluir con el tema, el software se fue incrustando dentro de la estructura de la sociedad a tal punto, que se volvió en uno de los componentes importantes para el debido funcionamiento de esta misma, ya sea local o global y para mantener un correcto y adecuado despliegue y desarrollo del software, son necesarios los ingenieros de software, expertos documentados en este ámbito para elaborar un buen producto.

3.2 Recorrido histórico

Tanto el software como la ingeniería de software deben su existencia uno del otro, por lo que desde el principio han estado relacionados, teniendo casi setenta años de existencia, como comenta Piattini *(2016)* en su artículo, que durante este tiempo ha avanzado demasiado y una muestra de ello son los lenguajes de programación que ahora son más complejos que los primeros que surgieron, los procesos de desarrollo son más sofisticados y por ende las aplicaciones desarrolladas hoy en día son más complejas.
Durante todo este tiempo, el proceso de desarrollo de software ha pasado por distintas etapas a lo largo de su historia, como pueden ser propuestas o contrapropuestas que se fueron formulando con el paso de los años para mejorar el flujo de trabajo y proponer o formular soluciones a problemáticas o dificultades que fueron surgiendo, por lo que se revisara a continuación la evolución del software a través de las décadas hasta la actualidad.
Comenzamos en la década de los años 40 y 50, en donde al software no se le daba el valor que merecía, ya que se pensaba erróneamente que podrían recibir el mismo tipo de desarrollo que el hardware, ya que este último era más costoso y, por ende, se le daba más importante, según lo comentado por Piattini *(2016)* en su obra.
Incluso durante esta época el proceso de trabajo y de desarrollo de software era distinto, ya que no se contaba con profesionistas encargados enteramente del trabajo del desarrollo del software, si no que los mismos que creaban el hardware diseñaban el software, esto según lo explica Piattini *(2016)* en su texto.
En esta primera década de vida del software, se puede observar que los ingenieros de ese tiempo no dimensionaban la importancia que tendría el software en el futuro, que este era igual de importante y pesado que el hardware y que sería muy difícil para un trabajador encargarse de las dos cosas al mismo tiempo.
En la década de los 60, como hace referencia Piattini *(2016)*, la diferencia entre el hardware y el software se

empieza a notar cada vez más porque cada uno debía tener un tipo de desarrollo más especializado y ya no podían ser tratados de una manera similar, como en el caso del software, que debía ser más organizado y estructurado para su mejor entendimiento, pero en esa época no era común ver código que estuviera bien redactado, lo cual provocaba dificultades cuando se quería entender el código escrito.

Fue en esta época, como comenta Piattini *(2016)* donde surge el código spaguetti que así se le llama al código que está mal escrito y que hace extremadamente difícil leerlo y por ende entender cómo funciona, además de que se empezó hacer común los arreglos apresurados de problemáticas que surgían del software en los últimos momentos antes del plazo de entrega establecido.

De igual manera, Piattini *(2016)* menciona que es en esta época que surgen conceptos importantes dentro del mundo de desarrollo del software así como el análisis de situaciones que se estaban dando, como fue en 1966 en la el NASA/IEEE Software Engineering y en 1969 en las conferencias de la OTAN que se planteó el análisis de la crisis del software, así como las bases para la reutilización o la arquitectura software, igual en ese año Dijkstra público el articulo: Go To Statement Considered Harmful que impulsó la programación estructurada y se empezó a citar el concepto de factoría o fábrica de software.

Sin embargo, aun cuando todos estos conceptos e ideas que estaban dando forma al proceso de desarrollo de software, los profesionales seguían trabajando de manera ad-hoc y se enfocaban más en los sistemas y en la programación, sin desarrollar más lo que verdaderamente debía ser una ingeniería de software.

En el lapso de los 70, las organizaciones comprobaron que los costos del software eran más elevados que los del hardware, esto se debía a que merecía un cuidado especial para el desarrollo de software para poder entregar un buen producto, lo que dio pie al surgimiento de los primeros modelos o estructuras para el análisis y diseño de software, como es el modelo de ciclo de vida en cascada por parte de Royce y el modelo Entidad-Relación por parte de Chen, así como distintas propuestas como la descomposición modular o el concepto de ocultamiento de información, tal y como señala Piattini *(2016)*.

El concepto de la crisis del software fue un evento que se habló a finales de los años sesenta pero que se fue dando entre los primeros años de los setenta, lo cual hace mención Pantaleo *(2015)* en su escrito, el cual se refiere a problemáticas que fueron apareciendo como lo fue el tamaño de los sistemas, la alta complejidad para resolver problemas junto con la falta de madurez en las herramientas y procesos de desarrollo.

Siendo en esta época en la que al software se le dio el valor que siempre debió tener, esto se nota con los conceptos o ideas básicas que se fueron planteando en la década de los 60̋ s, en donde empiezan a tomar una forma y estructura más consistente, así como el surgimiento de modelos de trabajo que puedan ayudar y organizar el flujo de trabajo del desarrollo del software mediante etapas que le den orden al proceso de desarrollo, ya que se buscaba que el software fuera más robusto.

En los 80̋ s, se empezaron a enfocar en medidas que aumentaran la calidad del software y potenciar el proceso de desarrollo, como Piattini *(2016)* señala en su escrito, mencionando la charla dada por Leo Osterweil en la Conferencia Internacional sobre Ingeniería de Software (ICSE) que marcó el comienzo de un nuevo enfoque para método de cómo crear software, como los primeros intentos de formalizar el procedimiento de desarrollo, buscando calidad en los productos desarrollados.

Junto a estas conferencias que se fueron dando, así como los acontecimientos y preocupaciones que fueron apareciendo durante esta época, como comenta Pantaleo *(2015)*, se concluyó que era necesaria una disciplina en la rama de la tecnología que se encargara sistemáticamente del diseño y construcción de productos del software y con esto en mente, se crea la ingeniería de software junto con el Instituto de Ingeniería de Software (SEI) en la Universidad de Carnegie Mellon en 1984.

En lo que se refiere a esta década en cuanto avances, como dice Piattini *(2016)*, se empezaron a automatizar algunas partes del desarrollo con el surgimiento de la primera generación de las herramientas CASE, además de que se empezó a popularizar el uso de lenguajes de programación orientados a objetos, que con anterioridad ya se contaba con algunos lenguajes, como en los sesenta con Simula y en los setenta con Smaltalk, fue en la década de los ochenta que más se difundieron, sobre todo con el surgimiento de C++, Eiffel y Objective-C.

Con el surgimiento de nuevas tecnologías durante esta época, la formación de los ingenieros de software requería aprender a manejar las herramientas de CASE, así como comprender el nuevo paradigma que había surgido de que fue la programación orientada a objetos siendo un cambio muy drástico y de igual manera debía aprender sobre los procesos de software y los modelos de madurez, como dice Piattini *(2016)*.

Como se puede examinar, esta época vino con grandes avances tecnológicos que ayudarían a fortalecer y mejorar el proceso de desarrollo de un software, al presentar nuevas herramientas, paradigmas o lenguajes que ayudasen para alcanzar ese objetivo, pero de igual manera fue dando más carga de aprendizaje a los expertos de desarrollo de software para poder ir a la vanguardia en lo que el software se refiere.

Fue en los 90̋ s, como Piattini *(2016)* explica en su artículo, de un amplio desarrollo en el apartado de técnicas o metodologías, como se muestra con el desarrollo de más modelos relacionados con l a mejora de los procesos de software, como lo fue Ideal o TSP y de igual manera normas o estándares que ayuden a mejorar la calidad de software, como las ISO, al mismo tiempo la orientación a objetos se consolida por lo que aparecen más de cien metodologías que terminan dando como resultado el Lenguaje de Modelado Unificado (UML) y el proceso unificado (UP), elementos tan importantes que a día de hoy siguen siendo utilizados.

De igual manera en esta época la construcción orientada a objetos se fue fortaleciendo y agarrando estructura, esto mediante una multitud de técnicas y conocimientos enfocados en esto, esto provoca una mayor dedicación por parte de los ingenieros de software en profundizar en estos nuevos saberes y tanto

ellos como las empresas buscan desarrollar estas *"buenas prácticas"* para una correcta construcción de software de una calidad elevada, sin embargo siguieron arrastrando problemas del pasado, como es el mantenimiento del software y solucionarlo mediante la externalización, esto según lo explicado por Piattini *(2016)*.

Durante el transcurso de esta década, el desarrollo de métodos, técnicas y procesos para poder crear un software de calidad, siendo tan importantes para el software que estos tópicos siguen estando presentes en la actualidad.

En la década de los 2000ʹ s, como nos explica Piattini *(2016)* articulo, es la época donde se busca disminuir la complejidad de las metodologías que se fueron planteando durante la década pasada esto mediante la firma del Manifiesto Ágil, así mismo aparecen los métodos híbridos, que quieren un equilibrio entre la adaptabilidad de las metodologías agiles con la documentación y formalidad de las metodologías rigurosas, siendo un ejemplo de estos la metodología SCRUM, la cual a día de hoy sigue siendo muy utilizada.

En febrero del 2001 nace el término agil en el ámbito de desarrollo de software en la reunión celebrada en Utah-EEUU, en donde participaron 17 expertos de la industria del software, incluidos algunos creadores o impulsores de metodologías de software, siendo su objetivo el elaborar valores y principios que permitieran a los equipos desarrolladores elaborar software de manera rápida y respondiendo a los cambios drásticos que surjan, pretendiendo ofrecer alternativas a las metodologías rígidas de antaño, después de esta reunión fue que se creó el Agile Alliance, una organización que buscaba promover el uso de estas nuevas metodologías y que alcanzó su punto de partida con la creacion del Manifiesto ágil.

Además, en esta década es donde se ve el crecimiento del desarrollo de software basado en modelos (DSDM), de las líneas o familias de software que serían de utilidad para el mantenimiento de software y del desarrollo de software distribuido, que hace referencia a equipos de desarrollo distribuidos por el mundo, requiriendo ahora no solo que los ingenieros de software se informen en ámbitos tecnológicos si no también en habilidades blandas, como la correcta comunicación o el trabajo en equipo con diferencias de horario, esto comenta Piattini *(2016)*.

Al principio de esta década, la Web comenzó a evolucionar pasando de solo ser un almacén de información global con poco impacto en el desarrollo de software, como menciona Sommerville *(1998)*, pero a estos navegadores se le fueron agregando funcionalidades, significando que los sistemas que se basan en la Web se podían elaborar en los sitios que tuvieran acceso a dichos sistemas usando un navegador Web, en lugar de una interfaz de un usuario de propósito, lo que condujo al desarrollo de una gran variedad de nuevos productos de sistemas que entregaban servicios innovadores, los cuales ingresaban mediante la web.

En estos años se buscaba mejorar, agilizar y disminuir la complejidad, de distintas metodologías de desarrollo que fueron propuestas y usadas en los años 90, inclusive buscando juntar lo mejor de dos metodologías en una nueva para agilizar el proceso de desarrollo, relacionándose con el surgimiento de nuevas ramas del desarrollo de software fueron surgiendo, así como la comunicación entre grupos de trabajo por todo el mundo se fue dando de manera más natural gracias al uso del internet.

Ya en la década del 2010, comenta Piattini *(2016)* en su artículo, se observa una consolidación de las tendencias establecidas en las décadas previas, a la vez que se incrementa la fusión entre la Ingeniería del Software y la Ingeniería de Sistemas, poniendo un énfasis particular en los requisitos no funcionales y la seguridad de los sistemas y con la creciente relevancia de la Ciencia, Gestión e Ingeniería de los Servicios se subraya la necesidad de un enfoque interdisciplinario que incluya disciplinas como la informática, marketing, gestión empresarial, ciencias cognitivas y derecho para el diseño de servicios.

Simultáneamente en el transcurso de esta década, menciona Piattini *(2016)*, se busca la adaptación de los métodos de desarrollo de software para operar en un mundo abierto, esencial en áreas como la inteligencia ambiental, aplicaciones conscientes del contexto y la computación omnipresente, se ha vuelto crucial y de igual manera apareció un desafío significativo en la gestión de proyectos de software siendo por parte de la aparición de los Sistemas de Sistemas Intensivos en Software (SISOS), con características como decenas de millones de líneas de código, diversas interfaces externas, competencia entre proveedores y estructuras jerárquicas complejas.

Así mismo, nos menciona Piattini *(2016)* en el escrito, se observó con la adopción de la Ingeniería del Software Continua y su tecnología asociada, conocida como DevOps, que facilita la reducción del intervalo entre la decisión de implementar un cambio en el sistema y su puesta en producción, este gran cambio requirió un gran cambio cultural, ya que este movimiento promovía la responsabilidad compartida entre el desarrollo y la operación para garantizar la entrega de software de alta calidad al usuario final, además era ya necesario aprender conceptos nuevos, como la infraestructura como código o los microsistemas por parte de los ingenieros de software.

En conclusión, el software ha a travesado un gran recorrido histórico y con el transcurso de los años ha recibido bastantes actualizaciones, mejorías, integración de documentación entre otros elementos para que el proceso de desarrollo fuera más dinámico, ágil, robusto y bien documentado, pero para llegar a eso hubo un proceso de aprendizaje y de adaptación por parte de los expertos en diseño de software.

3.3 Áreas de aplicación

En esta última década, como comenta Piattini *(2016)*, todavía se siguen afianzando los aspectos que se establecieron lapsos atrás, se toman más en cuenta los requisitos no funcionales de los sistemas, así como elementos externos, tales como el marketing, el derecho y la gestión empresarial, dentro del diseño de servicios del software; además del surgimiento de una nueva metodología, cultura y filosofía conocida como

DevOps, la cual, en términos generales busca la automatización del proceso de desarrollo de software para reducir tiempos y aceptar el trabajo y responsabilidad compartida.
La ingeniería de software se puede aplicar en cualquier área que necesite un sistema de software para automatizar actividades que permitan brindar a los clientes un servicio de una manera más eficaz, rápida y confiable, o bien la manufactura de un producto de calidad, algunas de esos contextos que Sommerville *(1998)* nos explica son los siguientes:
Software de sistemas: Son los sistemas o programas que se crean para servir dentro de otros programas.
Software de gestión: Son sistemas o programas que manejan una o varias bases de datos que contienen información comercial.
Software de ingeniería y científico: Son programas caracterizados por el manejo de números dentro de sus funciones, como puede ser simulaciones de sistema o aplicaciones interactivas.
Software de computadores personales: Son los programas más usados, algunos ejemplos serian como Excel, Word, Steam, entre otros.
Software empotrado: Son programas que sirven para controlar productos y sistemas de los mercados, además de que solamente se ubican dentro del disco duro como *"solo lectura"*.
Software de inteligencia artificial: Programas o sistemas que hacen uso de algoritmos no numéricos para poder resolver problemas.
De igual manera, Pressman *(2010)* propone otras áreas de aplicación, aunque él los divide por dominios, siendo según su criterio los más importantes los siguientes, compartiendo algunas con Sommerville:
Software de sistemas: En el cual Pressman *(2010)* establece como un conjunto de programas diseñados para proporcionar servicios a otros programas, además este tipo de software interactúa profundamente con el hardware de la computadora, soportando la operación concurrente, la secuenciación y la administración de procesos, así como el manejo de datos complejos y múltiples interfaces externas.

Software de aplicación: Se refiere Pressman *(2010)* a que establece a programas específicos diseñados para abordar necesidades comerciales particulares, procesan datos comerciales o técnicos de manera que facilitan operaciones de negocios o decisiones administrativas o técnicas.
Software de ingeniería y ciencias: Se destaca por el uso de algoritmos intensivos en cálculos numéricos, abarcando desde la astronomía hasta la biología molecular y la fabricación automatizada, sin embargo, las aplicaciones modernas en este campo están migrando hacia algoritmos no numéricos, reflejando un cambio hacia soluciones que abordan problemas complejos sin depender exclusivamente de métodos numéricos tradicionales, como según explica Pressman *(2010)* en su libro.
Software incrustado: Como Pressman *(2010)* explica, este se encuentra dentro de un producto o sistema, diseñado para ejecutar funciones específicas y controlar características tanto para el usuario final como para el mismo sistema, además puede realizar tareas limitadas y particulares, como el control de un horno de microondas, o proporcionar una amplia gama de funcionalidades y control de operaciones.
Principio del formulario
Software de línea de productos: Está diseñado, como dice Pressman *(2010)*, para ofrecer capacidades específicas para ser utilizado por una amplia gama de consumidores, el cual puede enfocarse en mercados especializados, como el control de inventario de productos, o dirigirse a mercados masivos que abarcan procesamiento de textos, hojas de cálculo, gráficos por computadora, entre otros.
Aplicaciones web: Las *"webapps"*, o aplicaciones web, representan una categoría de software enfocada en redes que abarca una amplia gama de aplicaciones, siendo en su forma más básica, las aplicaciones web que básicamente son un conjunto de archivos HTML vinculados que muestran información utilizando texto y gráficos limitados, sin embargo, desde la aparición de la Web 2.0, estas han evolucionado hacia entornos de cómputo preferidos que no solo ofrecen características y funciones aisladas al usuario final, sino que también están integradas con bases de datos corporativas y aplicaciones de negocios, expandiendo su alcance y funcionalidad, como explica Pressman*(2010)* en su libro.
Software de inteligencia artificial: Este utiliza, según explica Pressman *(2010)*, algoritmos no numéricos para abordar problemas complejos que son difíciles de resolver mediante cálculos computacionales o análisis directos, lo que incluye aplicaciones en áreas como robótica, sistemas expertos, reconocimiento de patrones en imágenes y voz, redes neuronales artificiales, demostración de teorías y juegos.
En definitiva, el software es tan flexible que su aplicación se puede realizar en una gran cantidad de ámbitos o áreas, si se realiza de la manera correcta, es debido a esta flexibilidad que tiene que actualmente es un pilar fundamental en muchas empresas o ámbitos, como lo son el entretenimiento, financiero, científico, entre otros, al realizar programas que ayuden y facilitan las actividades o acciones que se deben realizar en estas áreas.
3.4 Ciclo de vida de desarrollo de software
Durante el proceso de trabajo para la creacion de un software, entra un elemento conocido como el ciclo de vida de desarrollo de software, en donde se integran metodologías de desarrollo que están compuestas por etapas o fases a cumplir para finalizar el trabajo, teniendo cada una sus ventajas y desventajas, dependiendo además del tipo de proyecto al que quiere ser aplicada.
El autor Shylesh *(2017)* menciona en su paper, que el ciclo de vida de desarrollo de software (SDLC), el cual también es conocido como modelo de proceso de desarrollo de software, es usado para diseñar, desarrollar y producir productos de software de buena calidad, confiable, rentable y que cubra el plazo en la industria del

software.
Como menciona Marulanda *(2012)* en su artículo, acerca del Ciclo de Vida de Desarrollo de Software, se entiende a este como un proceso iterativo y que sin importar la metodología aplicada al flujo de trabajo puede incluir las siguientes etapas, las cuales son las siguientes: Requisitos, Diseño, Desarrollo y Pruebas, siendo estas las principales a tomar en cuenta, por lo que es muy raro que en una metodología falte alguno de ellos.
De igual manera, Shylesh *(2017)* escribió en su artículo que, todos los procesos del SDLC el cual se compone de actividades finitas que son usadas para desarrollar un software, en el cual cada proceso de SDLC vienen con un plan completo para describir como diseñar, desarrollar, mantener y aumentar la eficiencia de un producto de software, y la siguiente ilustración muestra las diferentes fases de un ciclo de vida típica o comunes en un desarrollo de software y que además pueden que se encuentren presentes en distintas metodologías de desarrollo que buscan mejora la calidad y proceso de trabajo en un proceso de desarrollo.
Explicando cada fase del ciclo de vida, empezando por la etapa de planificación y análisis de requisitos, como explica Shylesh *(2017)* en su paper, es la fase más esencial y básica de cada proceso de ciclo de vida, siendo realizado por los miembros más experimentados al tener una plática con el o los clientes.
La etapa de requisitos o planeación, como se menciona antes, es normalmente la primera etapa dentro del ciclo de vida, en esta parte es donde se realizan las entrevistas a los clientes, se levantan los requerimientos funcionales o no funcionales, se establecen cronogramas de trabajo, se dividen partes y se comienza el análisis del software que se quiere diseñar, como se va a crear y que se le integrara para cumplir los requisitos o solicitudes del cliente, siendo la base de todo, que dependiendo como fue planteada puede o no provocar problemas a futuro.
Pasando a la fase de definición que sigue a la de planificación, explica Shylesh *(2017)* en su paper, trata en donde lo planteado con anterioridad, se define de mejor manera, como lo son los requisitos, además de documentar estos últimos, y así obtener la verificación del cliente, siendo algunos resultados de esta fase un documento donde enlistan los requisitos del software a diseñar y desarrollar.
La siguiente etapa es la del diseño de la arquitectura del software, como describe en el paper de Shylesh *(2017)*, esta fase usa como base las especificaciones creadas en los requisitos de software para la arquitectura del software que se está trabajando, usualmente se suelen diseñar más de una posible arquitectura como potenciales opciones a escoger para trabajar en base a ello, pasando por una revisión por parte de los interesados en el producto siguiendo criterios como evaluación de riesgos, robustez, modularidad del diseño, tiempo y costo, para luego seleccionar la mejor opción.
En la etapa de diseño como se nos explica, que usualmente es la segunda etapa por llevar a cabo en las metodologías más simples, es donde se empiezan a trazar o plasmar las ideas, requerimientos o solicitudes recolectadas en la etapa anterior en diagramas, tablas, o prototipados, todo esto para entender de manera grafica el funcionamiento que se le quiere dar al sistema.
La etapa posterior, como explica Shylesh *(2017)* en el paper, es la de desarrollo, o también conocida como la de programación o construcción por algunos autores, siendo donde inicia el verdadero desarrollo real del producto en base a la arquitectura diseñada, en el cual si este diseño se realizó de manera adecuada y exitosa, esta etapa no será tan complicada, y para llevar a cabo esta etapa los desarrolladores utilizan variadas herramientas para trabajar como lo son compiladores, intérpretes y depuradores para escribir código, de igual manera depende el tipo de software será el lenguaje de programación a usar.
La etapa de desarrollo, también conocida como la de codificación en algunas metodologías, es posiblemente la etapa más pesada, es donde todo lo diseñado y planeado en las anteriores etapas se llevarán a cabo, suele ser una etapa crítica y la que mayor tiempo lleva en el ciclo de vida de desarrollo del software.
Las dos últimas etapas del ciclo de vida es la de prueba y despliegue, en el paper de Shylesh *(2017)* se explica que, en la fase de pruebas se corrobora si el producto finalizado cumple con los requisitos del usuario que fueron especificados desde el principio, además los defectos que vayan surgiendo durante esta etapa se van informando, rastreando, reparando y vuelven a ser sometidos a pruebas para confirmar la calidad del producto, de ahí que esta etapa sea superada se pasa a la última fase la cual es la de Implementación y mantenimiento donde el producto ya será desplegado en un entorno real, esto después de la aceptación de las pruebas y de ahí solo quedaría en un futuro irle agregando mejoras adicionales.
La etapa de pruebas o testing, como se describe, suele ser de las últimas fases que se llevan a cabo, aquí se corrobora que lo que se creó en la etapa de desarrollo funcione correctamente y que tenga la menor cantidad de fallos posible, existiendo distintos planes o métodos de pruebas que ayuden a medir un solo componente del sistema por separado o a todo el sistema en conjunto, de ahí se finaliza con la etapa de despliegue, o implementación, donde el software ya es lanzado al público.
Tras después de haber revisado lo que es un ciclo de vida de desarrollo de software y las fases más generales de este mismo, siendo las más comunes las de planificación, diseño, despliegue, desarrollo, pruebas y despliegue, donde en cada fase se realizan determinadas acciones que ayuden poco a poco construir el producto deseado, se pueden pasar a las metodologías de desarrollo que son usadas para la creacion de un producto de software.

3.4.1 Metodologías de desarrollo

Como se ha hablado, en el ciclo de vida se utilizan metodologías de desarrollo que sirvan para dar un orden a las actividades que se planean realizar para crear un producto, estas se han ido evolucionado conforme fue pasando el tiempo, esto para adaptarse a las tecnologías que fueron surgiendo, asimismo mejorar el proceso

de desarrollo, para volverlo más ágil.
Una definición que Maida *(2015)* da sobre lo que son las metodologías, es la siguiente: *"Es el conjunto de métodos que se utilizan en una determinada actividad con el fin de formalizarla y optimizarla. Determina los pasos a seguir y cómo realizarlos para finalizar una tarea." (p. 17)*
Tal como menciona Canós *(2003)* en su artículo, el desarrollo de software es difícil, siendo prueba de esto la enorme cantidad de propuestas metodológicas que abarcan distintas secciones del proceso de desarrollo de software al presentar dos distintas opciones para facilitar el proceso, las tradicionales que se centran mas en el control del desarrollo, mediante una doctrina rigurosa que revisa actividades, herramientas, artefactos producidos y notaciones, en cambio están las agiles que se enfoca en el valor del individuo, más interacción con el cliente y mayores iteraciones de entrega.
Las metodologías se encuentran divididas en dos grandes grupos, las tradicionales y las agiles, siendo el enfoque del primer grupo en las buenas prácticas dentro de la ingeniería del software, siguiendo un marco de disciplina estricto y riguroso, mientras que el segundo grupo se enfoca más en la obtención de una solución a los problemas de una forma rápida en un ambiente flexible y con cambios constantes, haciendo caso omiso de la documentación rigurosa y los métodos formales, esto lo menciona Maida *(2015)* en su escrito.
A continuación, se presenta una tabla con las diferencias más notables entre estos dos grupos, pero aun así con sus diferencias suelen ser muy utilizados cualquiera de los modelos integrados en estos grupos, así como que los tradicionales sirvieron como bases para los agiles al detectar los fallos que los primeros tuvieron al incluirlos dentro de procesos de desarrollo *(Ver Tabla 2)*.
Tabla 2.- Comparación entre Metodología Ágil y Tradicional

Metodologías ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos
Poca documentación	Documentación exhaustiva
Muchos ciclos de entrega	Pocos ciclos de entrega

Fuentes: Maida, E. G., & Pacienzia, J. *(2015)*. Metodologías de desarrollo de software.
Como podemos observar en la tabla presentada, son muy obvias las diferencias que hay entre las metodologías agiles y las tradicionales, esto se debe al contexto y necesidades del momento en que cada tipo de metodología fue propuesto en su momento, siendo las tradicionales enfocadas en una solución más rígida, controlada y que llevaba tiempo en el desarrollo, mientras que las agiles se diseñaron más para estos tiempos modernos en que se prioriza la creacion de software más rápido y sin tanta complicaciones.
Actualmente, las metodologías más usadas son las agiles, aunque esto no quita que algunas empresas, usualmente pequeñas o medianas, sigan trabajando con las metodologías tradicionales permitiendo que sigan un tiempo más vigentes en el ámbito de desarrollo, pero no cabe duda de que, por el contexto actual, la mejor opción son las metodologías agiles.
A continuación, revisaremos las metodologías tradicionales y agiles, revisando las propuestas que cada tipo tuvo en su momento.
3.4.2 Metodologías tradicionales
Profundizando en las metodologías tradicionales o también conocidas como pesadas, Maida *(2015)* nos comenta en su artículo, que esta clase de métodos se rigen por un control riguroso y exhaustivo de todo el proceso de desarrollo, como seria la planificación y control de este último, así como ser especifico con los requisitos, modelado y el plan de trabajo, de igual forma al no ser tan flexible en sus estructuras, las metodologías de este campo no son las más adecuadas para entornos de trabajo donde los requisitos no son

predecibles o que se deben de hacer cambios constantes a fases pasadas ya que puede aumentar el costo.
Este grupo de metodologías fueron creados e implementados en una época donde el proceso de desarrollo era más *"lento"*, así mismo al ser tan riguroso e inflexibles que regresar a etapas anteriores supondría grandes costos a las empresas desarrolladores, por lo que toda tenía que estar diseñado de la manera más minuciosa posible, para evitar problemas en el futuro.
Las grandes desventajas que las metodologías clásicas que comparten, y de las que se han hecho ya mención, es la lentitud en el proceso de desarrollo, los costos, su rigidez y que el producto se prueba al final, dificultando el proceso de corrección, aun así, hoy en día algunos equipos de desarrollo siguen trabajando con algunas de estas metodologías en su proceso de trabajo.
A pesar de todo, estas metodologías son mayormente usadas por equipos de desarrollo para proyectos pequeños y no tan complejos, de igual manera se cuenta con una buena organización en la estructura de trabajo al tener bien definido lo que se quiere realizar desde un principio al crear la documentación necesaria y requisitos bien definidos.
3.4.3 Modelo de cascada
Una de las metodologías más clásicas es la de Waterfall (Cascada), y como comenta Maida *(2015)* en su escrito, esta metodología fue diseñada entre los años de 1966 y 1970, siendo la primera metodología en ser creada y la base para las que siguieron, también fue nombrado por la manera en que sus fases fueron ordenadas, como si *"cayeran"* por la gravedad, imitando a una cascada.
Este enfoque metodológico ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior y solo al final de cada etapa se revisa para corroborar que todo está bien y se pueda avanzar a la siguiente etapa, por añadidura los resultados no se podrían ver hasta muy avanzado el proyecto por lo que cualquier cambio debido a un error puede suponer un gran retraso además de un alto coste de desarrollo, tal y como dice Maida *(2015)*.
Las etapas que la conforman son las siguientes: fase de análisis(requisitos), en donde se analizan y juntan todos los requisitos obtenidos para el producto a desarrollar, en la fase de diseño se hace la estructura del producto de software en base a lo establecido en la etapa anterior , en la etapa de codificación(desarrollo) donde se crean pequeñas unidades mediante código y que se van probando de manera separada según su funcionalidad hasta la siguiente etapa donde se integraron, de ahí la etapa de prueba, donde todas las unidades que se integraron se prueban para comprobar que los objetivos se cumplan y los errores surgidos se tienen que corregir y probar nuevamente, llegando a la última etapa que es la del mantenimiento, que viene en conjunto con la de despliegue, que es cuando se lanza el producto a un entorno real y se siguen agregando funcionalidades al sistema, según como comenta en su paper Shylesh *(2017). (Ver Ilustración 1).*
Como menciona Shylesh *(2017)* en su paper, es imposible que una metodología o modelo SDLC pueda ser adecuado para todos los tipos de aplicaciones de software, por lo que es fundamental seleccionar el modelo que mejor se adapte al software que se quiere diseñar, así que tomando en cuenta esto se enumeran posibles situaciones donde se puede usar la metodología de cascada, siendo los siguientes:
Donde los requerimientos de software son claros, entendibles y bien documentados.
La definición del software es estable.
La tecnología es entendida y no dinámica.
No hay ambigüedad.
El proyecto es pequeño.
Los recursos requeridos están disponibles.
El modelo de cascada se enfoca más para proyectos sencillos, pequeños, concisos, sin tanta ambigüedad y no sea propenso a cambios durante el transcurso del desarrollo, esto se debe a que esta metodología es algo rígida en cuanto al flujo en que sus fases se van desempeñando, por lo que no es recomendable para proyectos muy grandes y que sufren cambios constantes durante su ciclo de vida.
Algunas ventajas y desventajas que este modelo tiene, que son enumeradas por Shylesh *(2017)*, son las presentadas a continuación *(Ver Tabla 3)*.
Tabla 3: Ventajas y desventajas del modelo cascada

Ventajas	Desventajas
Es fácil de entender y seguir.	El software funcional está disponible ya muy tarde durante el ciclo de vida
Cada etapa tiene un entregable y un proceso de revisión específicos.	Tiene bastantes riesgos
Cada fase es procesada y completada una vez por tiempo.	Modelo pobre para proyectos largos
Funciona bien para proyectos pequeños donde los requerimientos están bien comprendidos	No se adecua para proyectos donde los requerimientos cambian frecuentemente
Fases bien definidas.	No puede manejar cambios en los requerimientos
Los procesos y resultados son documentados.	Un ajuste al alcance durante el proceso del modelo puede terminar el proyecto.
Se organizan tareas fácilmente.	Difícil medir el progreso durante las fases.

Fuente: Shylesh, S. *(2017)*. A study of software development life cycle process models. In National Conference on Reinventing Opportunities in Management, IT, and Social Sciences *(pp. 534-541)*.
En resumen, el modelo de cascada se basa en que los equipos sigan una secuencia de pasos y nunca avancen hasta que se haya completado la fase anterior, siendo todo muy rígido, por lo tanto, no deja prácticamente ningún lugar para cambios o revisiones imprevistos, siendo ideal para equipos pequeños y proyectos predecibles.
Ilustración 1.-Modelo de cascada

Fuente: Martínez, M. F. o. A. (s. f.). Metodología de cascada. https://fernandoarciniega.com/metodologia-de-cascada/
3.4.4 Modelo prototipos
Otra metodología es la de prototyping(prototipos), en la cual Maida *(2015)* comenta que esta suele realizar versiones preliminares de un sistema con fines de demostración o validación, siendo construido de manera rápida, para que pueda ser dado a los usuarios o clientes para que ellos puedan experimentar con este y dar retroalimentación sobre lo que les agrado o no.
El prototipo generado usualmente no es el sistema completo e inclusive muchos de los detalles de este todavía no estan construidos en el, siendo la meta final el proporcionar un sistema que ya venga con una funcionalidad general, explica Kute *(2014)* en su artículo.
Este modelo, según la explicación Kute *(2014)*, tiene sus áreas o situcaiones donde puede ser aplicado sin tantas complicaciones, siendo las siguientes:
Debe de ser usado cuando se busca que el sistema deseado necesite tener bastante interacción con los usuarios finales.
Típicamente es usado para sistemas online o interfaces web, los cuales tienen una gran cantidad de interacción con el usuario final, por lo que esta metodología se adapta de buena manera.
Algunas ventajas y desventajas que este modelo tiene, que son explicadas por Kute *(2014)*, son enumeradas en la siguiente tabla *(Ver Tabla 4)*.
Tabla 4: Ventajas y desventajas del modelo de prototipos

Ventajas	Desventajas
Los usuarios están involucrados en el proyecto.	Se implementa la vanguardia y luego se repara el camino de los sistemas constructivos.
Por el modo de trabajo, esta previsto que los usuarios tengan una mejor idea del sistema desarrollado.	Cuando el alcance del sistema puede aumentar más allá de la expectativa se puede aumentar la complejidad
La desilusión puede ser percibida desde antes	Una aplicación incompleta puede causar que la aplicación no pueda ser usada como el sistema completo que se diseñó.
Los comentarios rápidos de los usuarios están disponibles para mejorar	Análisis deficiente o incompleta del problema
Los problemas relacionados con la funcionalidad pueden ser identificados fácilmente.	

Fuente: Kute, S. S., & Thorat, S. D. *(2014)*. A review on various software development life cycle (SDLC) models. International Journal of Research in Computer and Communication Technology, 3(7), 778-779.
En resumen, podemos considerar este modelo más adecuado para proyectos enfocados a un entorno web, una situación donde los usuarios finales interactúen de mayor manera con él, mediante prototipos de este último para recibir comentarios o retroalimentación de lo creado para ir lo mejorando, aunque es un método muy útil para sistemas altamente complejos o de ambigüedad.
3.4.5 Modelo en espiral
El método Spiral (Espiral) es también muy conocida, siendo muy usada en su momento y Maida *(2015)* en su artículo hace mención de que esta toma las ventajas del modelo de cascada y el de prototipos, agregando el elemento de análisis de riesgo dentro de sus fases, representándose en un bucle en espiral, en donde cada cuadrante representa una fase y que conforme avancen las fases van ganado *"madurez"* al aproximarse cada vez más al final deseado, hasta que una de las iteraciones logre lo deseado.
Se encuentra dividido en cuatro fases, siendo la primera la de identificación, donde se busca recopilar los requisitos comerciales, de sistemas y unitarios todo esto mediante una comunicación constante con el cliente, la otra etapa es la de diseño, donde se realiza el diseño arquitectónico, diseño lógico de módulos, diseño físico y diseño final integrados dentro de las espirales, de ahí viene la fase de construcción, en donde comienza el desarrollo real donde con cada modelo creado se le envía al cliente para recibir sus comentarios y la ultima etapa es la de evaluación y riesgos, donde se identifica, estima y monitorea la viabilidad técnica y la gestión de riesgos y el próximo desarrollo comienza con los comentarios del cliente en la siguiente iteración, siendo algo continuo durante el ciclo de vida, según explica Shylesh *(2017). (Ver Ilustración 2).*
El modelo en espiral es usado ampliamente en la industria actual, aunque al igual que el modelo de cascada, tiene su nicho o area donde se puede aplicar de la mejor manera, siendo esas situaciones las siguientes, tal y como describe Shylesh *(2017)*:
Las restricciones al presupuesto y la evaluación de riesgos son importantes.
Proyectos medios o largos.
Cuando el compromiso del proyecto es prolongado debido al entorno de requisitos cambiantes.

Cuando el cliente no conoce sus requisitos completos.
Los requerimientos son complejos y necesitan una evaluación para que queden claro.
Cuando los cambios son esperados durante el ciclo de vida
Algunas ventajas y desventajas presentadas por Shylesh *(2017)* sobre el uso de este modelo son las presentadas en la siguiente tabla *(Ver Tabla 5)*.
Tabla 5: Ventajas y desventajas del modelo de espiral
Ventajas
Desventajas
Cambios continuos en los requerimientos pueden ser manejados
Gestión compleja
Usar muchos prototipos está permitido
La duración para completar el proyecto llega a ser desconocida
Los usuarios logran ver el sistema tempranamente
Proceso complejo
Los requerimientos son recolectados de manera más precisa
La espiral puede seguir indefinidamente
El proceso de desarrollo se divide en partes y las partes más riesgosas son desarrolladas tempranamente para una mejor gestión de estas
Un largo número de fases requiere una documentación más pesada
Fuente: Shylesh, S. *(2017)*. A study of software development life cycle process models. In National Conference on Reinventing Opportunities in Management, IT, and Social Sciences *(pp. 534-541)*.
En resumen, el modelo en espiral es un enfoque de desarrollo de software que puede ser considerado como una respuesta a los inconvenientes del desarrollo en cascada, describiendo el ciclo de vida de un software por medio de espirales, que se repiten hasta que se puede entregar el producto terminado, minimizando los riesgos en el desarrollo de software.
Ilustración 2.-Método en espiral

Fuente: Ciclo de vida en Esperial. (s. f.). INGENIERIA DE SOFTWARE. https://ingsoftware.weebly.com/ciclo-de-vida-en-esperial.html
3.4.6 Metodología incremental
La metodología incremental trata sobre como poder construir el proyecto en etapas incrementales en donde cada etapa agrega una funcionalidad a la estructura, de igual manera estas etapas están conformadas por requerimientos, diseño, codificación, pruebas y entrega, así mismo se puede entregar al cliente un producto más rápido en comparación con el modelo en cascada, dicho en el artículo de Maida *(2015)*.
El modelo incremental es similar al modelo de prototipos y otros enfoques evolutivos, es iterativo por naturaleza, pero se diferencia al prototipo al enfocarse en la entrega de un producto operativo con cada incremento, siendo los primeros versiones simples de lo que seria el producto final pero que otorgan capacidades que sirvan al usuario y le proporciona de igual manera una plataforma para evaluar el producto por parte del usuario, explicado en el artículo de Kute *(2014)*.
Esta metodología es usada en determinadas situaciones o proyectos, ya que no para cualquier tipo de trabajo puede ser aplicado esta metodología, esto según como lo describe Kute *(2014)* en su columna, siendo las siguientes situaciones:
Puede ser usado cuando los requerimientos de un sistema completo estan claramente definidos y entendidos.
La mayoría de los requerimientos deben de estar definidos, sin embargo, algunos detalles pueden evolucionar con el tiempo.
Existe una necesidad de llevar el producto al mercado lo más pronto posible.
Una nueva tecnología está siendo usada.
Los recursos con las habilidades necesarias no están disponibles.
Hay algunas características y objetivos de alto riesgo
Unas ventajas y desventajas de este modelo que salen a relucir por parte de Kute *(2014)* son las presentadas a continuación *(Ver Tabla 6)*.
Tabla 6: Ventajas y desventajas del modelo de incremental
Ventajas
Desventajas
Este modelo es mas adaptativo y menos costoso al realizar cambios en los requisitos
Es necesario una alta calidad en la planeación y diseño
Es más fácil probar y depurar durante las pequeñas iteraciones
Es necesaria una interpretación clara de todo el sistema antes de que pueda ser descompuesto y se desarrolle progresivamente
En este modelo el cliente puede responder en cada construcción
El costo de entrega del producto es menor
Manejar los riesgos es más fácil porque los riesgos son manejados en cada iteración
Fuente: Kute, S. S., & Thorat, S. D. *(2014)*. A review on various software development life cycle (SDLC) models. International Journal of Research in Computer and Communication Technology, 3(7), 778-779.

En resumen, el modelo incremental tiene como objetivo un crecimiento progresivo de la funcionalidad, en donde el producto va evolucionando con cada una de las entregas previstas hasta que se amolda a lo requerido por el cliente o destinatario siendo útil sobre todo cuando el personal necesario para una implementación completa no está completo.

Después de haber analizado uno por uno cada una de las metodologías tradicionales más usadas o conocidas, se procederá a un proceso de comparación entre ellas, en base a sus características o sus actividades, siendo explicados de mejor manera en la siguiente tabla *(Ver Tabla 7)*.

Tabla 7: Comparativa de varios modelos de las metodologías tradicionales

Características
Modelo de cascada
Modelo espiral
Modelo de prototipo
Modelo incremental
Especificación de requisitos
Al comienzo
Al comienzo
Frecuentemente cambian
Al comienzo
Comprensión de requisitos
Bien entendidos
Bien entendidos
No están bien comprendidos
No están bien comprendidos
Costos
Bajo
Costoso
Alto
Bajo
Disponibilidad para reutilizar componentes
No
Si
Si
Si
Complejidad del sistema
Simple
Complejo
Complejo
Simple
Análisis de riesgo
Al principio
Si
No hay
No hay
Participación del usuario en todas las fases del SDLC
Solo al principio
Alto
Alto
Intermedio
Garantía de éxito
Menor
Alto
Bueno
Alto
Fases superpuestas
No hay
Si
Si
No hay
Tiempo de implementación
Largo
Depende del proyecto
Menor
Menor
Flexibilidad
Rígida

Flexible
Altamente flexible
Menos flexible
Incorporar cambios
Difícil
Fácil
Fácil
Fácil
Experiencia requerida
Alto
Alto
Medio
Alto
Control de costos
Si
Si
No
No
Control de recursos
Si
Si
No
Si
Fuente: Elaboración propia basado en las obras de Kute, S. S., & Thorat, S. D. *(2014)* y Shylesh, S. *(2017)*.

Como podemos observar en la tabla, a pesar de que todos los modelos son tradicionales, comparten características así como tienen diferencias entre si, esto se debe a que cada uno fue diseñado bajo el contexto y necesidades del momento para dar solución a un determinada situación al momento de desarrollar un producto de software, y con el paso del tiempo se fueron diseñando más para cada determinada situación, siendo imposible que uno sea usado para cualquier proyecto, cada modelo tienen un proyecto al que pueden adaptarse y ser aplicados.

Existen muchos más modelos tradicionales, pero estos son los más conocidos, compartiendo etapas o inspiración de entre ellos, compartiendo ventajas y desventajas, también no todos pueden ser usados a los mismos proyectos e incluso sería mejor ya no usar algunas opciones que no son tan recomendables hoy en día.

3.4.7 Metodologías agiles

En 2001 hubo una reunión para revisar las metodologías de desarrollo de ese momento, y lo que se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas, naciendo así las metodologías agiles, como menciona Maida *(2015)* en su artículo.

Mediante estas nuevas metodologías, como menciona Maida *(2015)* en su artículo, se buscaba valorar más al equipo desarrollador y las iteraciones que a las herramientas a utilizar, de igual manera integrar más al cliente en el proceso de desarrollo para que su colaboración sea mayor y sobre todo solución a problemas ante un cambio a realizar que el plan estricto a seguir y que no está preparado para cambios imprevistos.

La filosofía de lo que son las metodologías agiles, sobre todo de la palabra *"ágil"*, surgen de un escrito conocido como el *"Manifiesto ágil"*, el cual serviría como un punto de inicio para poder entender esta nueva propuesta además de comprender la cultura que buscaba promover.

Los principales valores que se establecieron en este manifiesto, como comenta Canós *(2003)* en su redacción, al individuo y la relación del equipo de desarrollo sobre el proceso y las herramientas, ya que se enfocan que la llave del éxito para un proyecto de software es la de construir un buen equipo que el entorno, ya que se suele cometer el fatal error de que se diseña el entorno primero y que el equipo de desarrollo se adapte a este, que debería ser al revés, siendo que el equipo configure su propio entorno en base a sus necesidades.

Otro valor en que se enfocan en este manifiesto, como explica Canós *(2003)* en su redacción, es el desarrollar software que responda de manera correcta a tener una buena documentación, siguiendo la regla de no producir documentos al menos que sean necesarios de manera inmediata para tomar una decisión relevante, además estos deben de ser cortos y enfocarse en lo importante.

Un valor más que se enfoca este escrito, que expresa Canós *(2003)* en su artículo, es la colaboración estrecha con el cliente mas que hacer el proceso de negociación para un contrato, proponiendo una interacción constante entre el cliente y el equipo de desarrollo, sugiriendo que esta interacción más constante será la que marque el comienzo del proyecto y asegurando el éxito de este.

El ultimo valor que se explica por parte de Canós *(2003)* y que se menciona en el manifiesto, es la de contestar a las modificaciones de manera adecuada que seguir un plan estricto ya que se considera mejor la habilidad de responder a cambios inesperados que puedan surgir en el transcurso del proyecto, ya que esto determina el éxito o el fracaso de este, por lo que la planificación debe de ser flexible y abierta per no estricta.

Estos valores que se mencionaron explican Canós *(2003)*, sirvieron de inspiración para los doce principios del

manifiesto, siendo las características que los diferencian con las metodologías tradicionales, siendo los dos primeros principios generales y el resto resumen gran parte de lo que el método ágil trata, como es el modo de trabajo y el equipo de desarrollo.
El primer principio como lo explica Canós *(2003)*, trata de que la mayor prioridad es la de satisfacer al cliente con entregas constantes y oportunas del producto y la segunda trata sobre darle la bienvenida a los cambios que permitirán darle un valor agregado al producto y tenga mayor competitividad.
Otros principios que Canós *(2003)* menciona, es en el que establece que las entregas frecuentes de software funcional en intervalos cortos de tiempo, otro principio que se menciona es que el cliente y los desarrolladores deben trabajar en conjunto en todo momento, además de construir un proyecto que gire en torno a individuos motivados al darles el apoyo y entorno que se necesitan para finalizar de manera adecuada el proyecto.
Unos principios que igual se mencionan, como dice Canós *(2003)*, se establece que el dialogo cara a car es el mejor método de comunicación en un equipo de desarrollo, otro es que la principal medida de progreso es el software funcional, así mismo como los procesos agiles impulsan un desarrollo sostenible.
Los últimos principios mencionados, como explica Canós *(2003)*, son donde la atención constante a la calidad y al buen diseño mejora la agilidad, otro donde se establece que la simplicidad es importante, de los equipos organizados surgen las mejores arquitecturas, diseños y principios y el ultimo principio es que el equipo reflexiona como se puede mejorar cada cierto intervalo de tiempo.
Varios vieron esto como una *"revolución"* al producir software al enfocar su filosofía en el factor humano o el producto del software, tal como menciona Canós *(2003)* en su artículo, demostrando su efectividad en proyectos que cambian seguidamente y que llegan a sufrir reducciones de tiempo drásticas, pero manteniendo una alta calidad, aun así cuenta tanto con seguidores como detractores que no las ven como las mejores opciones para reemplazar a las metodologías tradicionales.

3.4.8 Programación Extrema (XP)

Uno de los modelos es el de programación extrema (XP) siendo el más destacado de todos ellos por los cambios que supuso en su momento y Maida *(2015)* en su artículo hace mención que al igual que otros modelos agiles, se va más por la adaptabilidad que la previsibilidad, ya que se considera mejor el poder aptarse a los cambios de requisitos que surgen sobre la marcha que gastar esfuerzos en controlar esos cambios para no afectar lo que se estableció al inicio.
De la misma forma, Canós *(2003)* menciona en su escrito, que es una metodología que se centra en las relaciones interpersonales como clave para el éxito en el desarrollo, promoviendo el trabajo en conjunto, se busca el aprendizaje de los desarrolladores y un buen clima de trabajo, contando con una retroalimentación continua entre el cliente y el equipo de desarrollo mediante una comunicación fluida, definiéndose como la ideal para cuando hay proyectos con requisitos imprecisos y con alto riesgo técnico.
Además, Maida *(2015)* agrega de este modelo que se centra en dos aspectos sociales, siendo las relaciones interpersonales del equipo de desarrollo al querer potenciarlas para poder llevar al éxito el proceso de desarrollo como asegurar el aprendizaje de los desarrolladores y el otro aspecto sería el de una buena y constante contaminación entre los integrantes del equipo como con el cliente para mantener una retroalimentación continua, siendo el más adecuado para proyectos inciertos y cambiantes.
Este modelo se estructura en tres capas que agrupan las doce prácticas básicas de XP, para lograr el objetivo de software como solución ágil, siendo esas capas las siguientes: metodología de programación (diseño sencillo, testing, refactorización y codificación con estándares), metodología de equipo (propiedad colectiva del código, programación en parejas, integración continua, entregas semanales e integridad con el cliente) y metodología de procesos (cliente in situ, entregas frecuentes y planificación), como dice en su artículo Maida *(2015). (Ver Ilustración 3).*
Además, Canós *(2003)* aporta en su artículo, que las iteraciones de cada ciclo todos los involucrados aprenden en el proceso, el cliente y el programador, igualmente se busca que este ultimo no se sienta presionado al darle mas trabajo del estimado provocando que la calidad del producto baje mientras que el cliente tiene la obligación de manejar la entrega para que el producto tenga mayor valor en el mercado.
En resumen, la programación extrema tiene como gran ventaja el de la programación organizada y planificada para que no haya errores durante todo el proceso. Los programadores suelen estar satisfechos con esta metodología. Es muy recomendable efectuarlo en proyectos a corto plazo.
Ilustración 3.-Capas de la programación extrema

Fuente: Lean. (2019, 22 marzo). Programación Extrema (xp). WordPress.com. https://ingsoft1unne.wordpress.com/2019/03/21/programacion-extre

3.4.9 Metodología SCRUM

Otro modelo conocido es el de scrum y Maida *(2015)* lo describe como un proceso en que de manera regular se lleguen aplicar un conjunto de buenas prácticas para el trabajo colaborativo y asegurar el éxito del proyecto, realizando entregas parciales del producto, todo esto girando en torno a los equipos productivos.
Esta metodología, como comenta Canós *(2003)* en su escrito, fue desarrollado por Ken Schwaber, Jeff Sutherland y Mike Beedle, el cual define un marco para el manejo de proyectos y que ha sido exitosa en esta ultima década, la cual preferentemente se usa para proyectos con un rápido cambio en los requisitos, y cuenta con reuniones a lo largo del proyecto, destacando reuniones diarias de 15 minutos del equipo de

desarrollo para coordinación e integración.
Este modelo como menciona Maida *(2015)*, es el indicado para proyectos en entornos complejos, de requisitos cambiantes y donde se requieren resultados lo más pronto posible, agregando también la innovación, la competitividad, la flexibilidad y la productividad como elementos fundamentales. *(Ver Ilustración 4)*.
Scrum también se puede utilizar para resolver situaciones críticas de desarrollo, y Maida *(2015)* menciona que estas situaciones pueden ser cuando al cliente le están quedando mal y no se le está entregando lo que el solicito, las entregas se alargan de más, los costos se han disparado y se ha perdido calidad en el proceso, siendo casi como el último aliento para sacar adelante el proyecto.
La manera en que un proyecto en scrum se ejecuta, según lo escrito por Maida *(2015)*, es de la siguiente manera, mediante la ejecución de bloques temporales cortos y fijos (iteraciones), siendo que cada iteración debe proporcionar un resultado completo, para que cuando el cliente solicite una entrega final se haga con el mínimo esfuerzo.
En resumen, la metodología Scrum es una metodología ágil que hace énfasis en el trabajo en equipo donde la claridad de los objetivos es crucial para avanzar hacia una versión cada vez mejor y el éxito de su aplicación dependerá, en gran medida, de contemplar estas ventajas y desventajas según lo requiera el proyecto.
Ilustración 4.-Metodología scrum

Fuente: Modelo SCRUM - Entrando al modelo - Planificando la iteración — Steemit. (s. f.). Steemit. https://steemit.com/hive-148441/@fullcolorpy/modelo-scrum-entrando-al-modelo-planificando-la-iteracion
3.4.10 Método Kanban
La metodología Kanban tiene su origen por parte de las empresas automovilísticas, siendo específicos de Toyota, así mismo Maida *(2015)* agrega que este modelo se basa en la idea de que le trabajo en curso se debe limitar y solo se debe avanzar en un nuevo bloque cuando el trabajo anterior haya sido finalizado o se ha pasado a otra función en la cadena, haciendo uso de un mecanismo visual para hacer seguimiento del flujo de trabajo.
(Ver Ilustración 5).
El principal aporte que este modelo otorga a las metodologías agiles y que Maida *(2015)* resalta en su artículo es que a través de la implementación de las tarjetas visuales da transparencia al proceso, ya que deja en exposición todas las cosas que impactan en el rendimiento de la organización, así como los efectos de las acciones o la falta de estas por parte de los interesados en el proyecto.
Kanban no genera una revolución radical, sino que es un cambio gradual en la forma en que las personas trabajan y se desenvuelven en su ámbito el cual surge del entendimiento y del consenso de entre todos los participantes del proyecto o así lo ve Maida *(2015)* en su artículo.
Las principales ventajas de este modelo que Maida *(2015)* hace mención en su artículo es la facilidad de aplicación, actualización y asumir por parte de un equipo, así mismo destaca por el uso de técnicas de gestión de tareas de una manera visual y práctica, permitiendo ver el estado de los proyectos de una manera más directa y pautar el desarrollo del trabajo de manera efectiva.
En resumen, Kanban nos ayuda a tener una gestión de trabajo más fluida de manera general en el desarrollo de las tareas gracias a la visualización del trabajo por fases, en donde se logra evitar la sobrecarga y se puede medir el tiempo estimado en el cual se deberían completar las tareas.
Ilustración 5.-Método Kanban

Fuente: Rodriguez, D. (2022, 4 noviembre). Te damos 10 razones para usar la metodología Kanban en tu organización. TecnoSoluciones.com. https://tecnosoluciones.com/10-razones-para-usar-la-metodologia-kanban-en-tu-organizacion/
Cada metodología de desarrollo cuenta con sus propias características, ventajas y desventajas, permitiendo a los equipos desarrolladores o empresas de software elegir el que más adecue a sus necesidades y que les facilite el proceso de desarrollo, para poder garantizar un producto de calidad y que satisfaga las necesidades del cliente.
3.5 Equipo de desarrollo de software
Los equipos de desarrollo de software son fundamentales en la industria del software, ya que son la columna vertebral de cualquier proyecto de desarrollo. Estos equipos se componen de profesionales con diversas habilidades y conocimientos que trabajan juntos para crear soluciones de software. Según el estudio realizado por K. J. Rayner y M. J.Rayner *(2010)*, los equipos de desarrollo de software pueden variar en tamaño, estructura y composición, pero todos comparten el objetivo común de producir software de alta calidad.
La dinámica y la colaboración dentro de los equipos de desarrollo de software son cruciales para el éxito de cualquier proyecto, esto según el estudio realizado por Carvalho y Santos *(2012)* , en donde mencionan que la dinámica del equipo, que incluye la comunicación, la cooperación y la resolución de conflictos, juega un papel crucial en la eficiencia y la calidad del software producido sugiriendo que los equipos con una buena dinámica de equipo tienden a ser más productivos y a producir software de mayor calidad.
Además, la diversidad dentro de los equipos de desarrollo de software es valorada por su capacidad para aportar diferentes perspectivas y soluciones y el estudio de A. K. Sarker y S. K. Sarker *(2016)*, establece que la diversidad en términos de género, edad, experiencia y antecedentes culturales puede enriquecer el proceso

de desarrollo, fomentando la innovación y la creatividad, además se destaca en este estudio que los equipos más diversos tienden a ser más innovadores y a producir software de mayor calidad.
En resumen, los equipos de desarrollo de software son esenciales para el éxito de cualquier proyecto de software. La diversidad, la dinámica del equipo y la adaptabilidad a diferentes enfoques de desarrollo son factores clave que contribuyen a la eficiencia y la calidad del software producido.
3.6 Calidad del software
La calidad es un aspecto fundamental en cualquier producto que sea desarrollado por una empresa y organización, ya que esto repercute positiva o negativamente en la opinión que el cliente tendrá de este producto a la hora de usarlo, y como comenta Crosby *(1979)*, pionero en el campo de la gestión de la calidad, enfatiza la importancia de la calidad intrínseca en su obra, comentando lo siguiente: *"La calidad significa hacerlo bien cuando nadie está mirando." (p.23).*
El software también se ve atado a la calidad por lo que es un elemento para tomar en cuenta por los ingenieros de software, siendo un aspecto fundamental en el desarrollo de aplicaciones y sistemas informáticos, ya que influye directamente en la satisfacción del usuario, la eficiencia del producto y la reputación de la empresa desarrolladora es por ello que surge el concepto de calidad del software.
La IEEE Computer Society define la calidad del software en su estándar como un concepto amplio que abarca requisitos del cliente, estándares de la industria y características de mercado: *"Calidad de software es la conformidad de un producto o sistema de software con los requerimientos explícitos e implícitos del cliente, los estándares establecidos por la industria y las características deseadas de mercado." (IEEE Computer Society, 1991, p. 12).*
De igual manera, la calidad no solo se mide en base a los parámetros de una empresa desarrolladora de software, si no que tan satisfecho queda el cliente con el producto entregado, tal y como menciona Humphrey, reconocido por su trabajo en la mejora de procesos de software, el cual destaca la relación entre calidad y satisfacción del cliente en en su obra: *"La calidad de un producto o servicio es definida por la percepción del cliente. [...] La calidad no es una cuestión de conformidad con especificaciones, sino de satisfacción del cliente." (Humphrey, 1989, p. 45).*
Además, se piensa que un software será de calidad únicamente al eliminar los errores que surgen dentro del producto, pero Boehm da una opinión sobre eso, el cual es reconocido por su trabajo en ingeniería de software, en donde resalta la importancia de la detección temprana de defectos en su escrito: *"La calidad en el software se logra no solo eliminando errores, sino también a través de la detección y prevención temprana de defectos." (Boehm, 1981, p. 67).*
Existen distintos parámetros dentro de la industria para poder definir que producto es de calidad y cual no La normativa ISO/IEC 9126-1 establece un marco para evaluar la calidad del software, definiendo seis atributos principales: *"La calidad del software se compone de seis atributos principales: funcionalidad, fiabilidad, usabilidad, eficiencia, mantenibilidad y portabilidad." (2001, p. 34).*
En conclusión, la calidad del software es un concepto que abarca aspectos técnicos y perceptuales, y su evaluación es fundamental para garantizar la satisfacción del cliente y el éxito del producto en el mercado. Citas de expertos como Crosby, Humphrey y Boehm, junto con estándares como el de la IEEE Computer Society y normativas como la ISO/IEC 9126-1, proporcionan un marco sólido para comprender y mejorar la calidad del software.
4. La gestión del conocimiento en la ingeniería de software
El conocimiento en la ingeniería de software es diverso y sus proporciones son inmensas y crecientes, aunque un aspecto negativo de este incremento desmesurado es que las organizaciones tienen problemas para dar seguimiento a qué es este conocimiento, dónde está y quién lo tiene, esto se debe a que el negocio de la ingeniería de software es complejo al tener muchas personas y en ocasiones de diversas culturas, trabajando en distintas áreas y fases del desarrollo de software, convirtiendo en un verdadero desafío recolección de todo ese cúmulo de conocimiento, esto según lo planteado por Rus y Lindvall *(2002)*.
Reforzando lo anterior, con un punto de vista organizacional, Viscaino *(2006)* menciona que: *"En toda organización es conveniente que la información y el conocimiento se procesen y almacenen de forma que estos se puedan reutilizar" (p. 91-98).* Siendo esto de vital importancia en empresas o organizaciones enfocadas en el software, en donde la información y el conocimiento aumentan con el paso del tiempo de una forma que no pueden ser abarcadas en su totalidad e inclusive desconociendo la existencia de fragmentos de información que pudieron ser útiles para el desarrollo de un proyecto.
La ingeniería de software como se ha descrito es un campo que requiere un manejo constante de conocimientos especializados, y lo planteado por Rus y Lindvall *(2002)*, lo que indica que podría beneficiarse de las estrategias de la gestión del conocimiento, lo que conlleva a una posible relación estrecha entre estos dos conceptos. Sin embargo, la cuestión clave es: ¿dónde se almacena y se accede al conocimiento en la ingeniería de software? Siendo evidente que la ingeniería de software involucra una amplia gama de tareas que requieren un alto nivel de conocimiento, como lo son:
Analizar las necesidades de los usuarios para el desarrollo de nuevos sistemas de software y seleccionar las mejores prácticas.
Aplicar técnicas de desarrollo de software y recopilar experiencia en la planificación de proyectos y gestión de riesgos.
Gestionar proyectos y otros aspectos relacionados.
Cada producto y proceso del software, en base a lo que Rus y Lindvall *(2002)* plantea es que son diferentes en

términos de objetivos y contextos, por lo que el enfoque de desarrollo no puede ser el mismo para todos los proyectos, por ejemplo, no es igual el proceso de desarrollo de software para la NASA a uno que necesite McDonalds, debido a que son procesos, actividades y conocimientos totalmente distintos entre sí, además menciona que la diferencia que existe entre estos productos es lo que provoca que los ingenieros de software estén expuestos a una gran diversidad de soluciones y que sea obligatorio el ganar experiencia con cada trabajo que se lleve a cabo para poder desarrollar las mejores soluciones.

Es por ello, que sin duda, lo mejor es que la experiencia obtenida en todos y cada uno de los proyectos, se aplicara en los desarrollos futuros, con la finalidad de disminuir los errores y lograr que el proceso sea más eficaz; lamentablemente esto no es algo que sea llevado a la práctica de manera muy común, ya que a veces ese conocimiento no es almacenado, lo que provoca que los equipos de desarrollo tarden más en dar resultados favorables, estos pueden ser similares a bocetos que ya se han trabajado con anterioridad.

Este tipo de situaciones son comunes entre desarrolladores neófitos y/o que acaban de ingresar a una empresa o que se les ha movido de su usual área de trabajo, debido a que tienen una mayor probabilidad de no contar con el conocimiento requerido, lo cual provoca errores en el desarrollo, a pesar de que ya se cuente con registros de proyectos similares, que podrían haber ayudado para evitar los traspiés.

Justo en este contexto, es donde resulta imperante aplicar la GC en los equipos y organizaciones desarrolladoras de software, ya que esta disciplina busca abordar los problemas que conlleva la captura y compartición del conocimiento.

Siguiendo con el contexto de la gestión del conocimiento, los equipos de desarrollo de software juegan un papel crucial en la captura, organización y compartición de conocimientos. Según A. K. Sarker y S. K. Sarker, *(2018)* la gestión efectiva del conocimiento dentro de los equipos de desarrollo de software es esencial para mejorar la eficiencia, la calidad del software y la satisfacción del cliente, además este estudio destaca la importancia de implementar prácticas de gestión del conocimiento, como la documentación de conocimientos, el intercambio de experiencias y la formación continua, para asegurar que los conocimientos y experiencias acumulados por los miembros del equipo sean reutilizables y aplicables en futuros proyectos.

Una manera fácil de transmitir el conocimiento de un experto a un novato puede ser mediante la captura, almacenamiento y organización de la información en documentos, lo cual dentro del ámbito de desarrollo de software no sería tan complicado, ya que gran parte de los artefactos del proceso de desarrollo se pueden representar mediante documentos, haciendo la gestión documental uno de los pilares dentro del modelo de GC aplicado en el ámbito del software.

No obstante, no basta con tan solo conocer sobre el dominio del desarrollo de software, si no también es necesario empaparse del dominio para el cual se está desarrollando, ya que un contexto nuevo, implica aplicar nuevas técnicas de gestión de proyectos o formarse un desconocido lenguaje de programación, por lo que la curva de aprendizaje de estas herramientas llevaría más tiempo.

Es por ello, que, mediante la gestión del conocimiento, que se puede organizar, empaquetar, almacenar y compartir el conocimiento nuevo que se va adquiriendo, así como la experiencia que ya existía dentro de la organización desde antes, debido a que se puede convertir en una cultura o disciplina enfocada en compartir conocimiento sobre un ámbito de importancia dentro de una organización, como podría ser el surgimiento de nueva tecnología de utilidad para la empresa.

Es así que esta metodología en la ingeniería de software es crucial para mejorar la eficiencia y calidad del trabajo. Como menciona Viscaino *(2006), "facilita el aprendizaje continuo de los profesionales al proporcionar acceso a información valiosa para sus funciones y proyectos".* Esta práctica también promueve la reutilización de conocimientos y experiencias previas, contribuyendo así al desarrollo de soluciones más eficaces y al mantenimiento de estándares de calidad.

Sin embargo, la implementación efectiva de la gestión del conocimiento en este campo enfrenta varios desafíos. Gran parte del conocimiento en ingeniería de software es tácito y difícil de formalizar, lo que dificulta su documentación y transferencia. En palabras de Viscaino *(2006), "gran parte del conocimiento en este campo es tácito y difícil de formalizar".* Esta dificultad en la conversión de conocimiento tácito en explícito dificulta su gestión efectiva y limita el potencial de mejora continua en las organizaciones.

Además, aunque se reconoce la importancia de la gestión del conocimiento, su implementación efectiva aún está en un estado apenas aceptable en muchas organizaciones desarrolladoras de software. Según Galvis, González y Torres *(2016), "la implementación efectiva de la gestión del conocimiento en las organizaciones desarrolladoras de software aún está en un estado apenas aceptable".* La falta de herramientas y metodologías adecuadas para convertir conocimiento tácito en explícito agrava esta situación.

En resumen, aunque la gestión del conocimiento promete mejorar considerablemente los procesos y resultados en la ingeniería de software, su efectividad plena demanda soluciones innovadoras y adaptadas al cambiante entorno de la industria del software. Al superar estos desafíos, las empresas desarrolladoras de software podrán cosechar enormes beneficios, no solo en términos de eficiencia y calidad, sino también en su capacidad para mantenerse competitivas y adaptarse a las demandas del mercado en constante evolución.

4.1 Tipos de conocimiento en la ingeniería de software

En la ingeniería de software y del desarrollo, se detectan dos tipos de conocimiento, según nos explica Rus y Lindvall *(2002)*, los cuales son los siguientes:

El conocimiento que se incrusta dentro de los productos (artefactos), ya que estos son resultados de ideas muy creativas e inteligentes.

El metaconocimiento, el cual es el conocimiento sobre productos o procesos.
Es relevante mencionar que ambos tipos de conocimientos giran alrededor de elementos conocidos como productos o procesos, los cuales dentro del desarrollo de software son muy importantes, siendo los productos (artefactos), los elementos que se crean para un sistema de software, ya sea documentación, código, pruebas, etc., mientras que los procesos, representan los pasos para poder construir el producto.
Adicional, cabe menciona que ambos tipos de conocimiento son muy importantes dentro del desarrollo de software, ya que, con base a estos, es que se puede crear, desarrollar y lanzar un software al mercado de una manera rápida y eficaz, debido a ello es que cada aporte de información nueva es de gran valor, ya que puede facilitar el proceso de desarrollo de software y sobre todo, coadyuvar en la calidad del mismo.

4.2 Gestión del conocimiento para actividades de ingeniería de software

Implementar la GC en cualquier organización es un desafío debido al lapso y esfuerzo que se necesitan antes de que se comience a recuperar la inversión y en las empresas de software se tiene menos tiempo que otras empresas debido al ritmo vertiginoso que conlleva el trabajar en desarrollo de productos, lo cual representa una verdadera amenaza debido a que los individuos cambian sus prioridades, de tal manera que para ellos resulta preminente la entrega a tiempo, más que realizar una búsqueda exhaustiva de conocimiento que les reditúe en una producción de mayor calidad y sobre todo que abone en al aprendizaje organizacional y por ende, a la gestión del conocimiento.
El vital desafío para la GC es que la mayor parte del conocimiento en ingeniería de software es tácita y es poco probable que se vuelva explícito, debido a la falta de tiempo para tal efecto, ya que hay muy pocos enfoques y herramientas que apoyen en la conversión de tácito a explícito, aunado a que gran parte del conocimiento tácito, es tácito de la manera más extrema posible, debido a que el conocimiento es propiedad y está controlado por los hacedores de este.
A pesar de todos estos retos, el implementar la GC en la ingeniería de software debería ser más fácil que en empresas con actividades económica diferente, ya que esta disciplina se respalda por tecnologías de información que ayudan a compartir el conocimiento y es de esperarse, que los ingenieros de software estarán en la disposición de aprender y adoptar nuevas herramientas tecnológicas que le sean de ayuda para realizar un mejor trabajo.
Otro aspecto que favorece a la fertilidad de la GC en la industria del software es que todos los artefactos ya están de manera digital, facilitando así su distribución a través de una amplia gama de medios, en muchas ocasiones sin la necesidad de erogación alguna, además de la disposición de los profesionales de software, para compartir sus conocimientos a través de distintos foros o blogs donde comunican sus saberes sin ningún tipo de compensación monetaria.
El éxito o el fracaso de los sistemas y las diversas prácticas de GC que afirman ser adoptadas por las organizaciones de ingeniería de software se pueden juzgar mejor desde el punto de vista de los ingenieros de software, ya que son los que tienen el conocimiento de primera mano, siendo ellos la cara de la organización y que pueden reflejar la imagen real del estado de cosas en el área de GC en la industria del software, además de que son ellos los más interesados y comprometidos con el intercambio y la gestión de su conocimiento; sin embargo, de acuerdo con Sharma y Goyal *(2017)*, el 92% de ingenieros de software, afirman la ausencia de sistemas de depósito de conocimiento organizacional.
Esto es algo curioso, que a pesar de que los desarrolladores de software sean conscientes de los conceptos, herramientas y técnicas para gestionar el conocimiento, las empresas no las aplican dentro de su configuración, debido a que no cuentan con espacios de almacenamiento donde se pueda guardar dicho conocimiento, provocando que los desarrolladores recurran a otros medios para acceder al conocimiento que necesitan, esto en empresas.
De igual manera se realizó una investigación en Colombia en las compañías desarrolladores de software, por parte de Galvis, González y Torres *(2016)*, en el que menciona que durante los últimos años, se han llevado a cabo múltiples investigaciones que caracterizan a las organizaciones de la industria del software en Colombia y que ha revelado aspectos significativos sobre la industria del software en el país, como lo es la falta de inversión en la innovación y desarrollo prefiriendo un negocio más tradicional, por lo que surge un interés especial en la gestión del conocimiento manifestando un creciente interés en obtener pruebas basadas en datos empíricos sobre el nivel actual de implementación de procesos de Gestión del Conocimiento en las Organizaciones Desarrolladoras de Software (ODS) del país.
Los resultados de este estudio elaborado por Galvis, González y Torres *(2016)*, revelo que la mayoría de las ODS reconocen la relevancia de los procesos de Gestión del Conocimiento (GC), y en una considerable cantidad de ODS, se han incorporado elementos asociados con la GC. No obstante, se observó que la implementación de los procesos de GC en las ODS estudiadas, evaluada en términos de alcanzar los resultados esperados y cumplir sus objetivos, se encuentra en un estado apenas aceptable. Esto remarca la necesidad de reforzar estos procesos para que las empresas de la industria del software en Colombia puedan obtener ventajas competitivas sostenibles y superar desafíos negativos, como un bajo nivel de especialización, una dedicación casi exclusiva a líneas de negocio tradicionales, y una escasa o nula atención a actividades de investigación, desarrollo e innovación.
A diferencia del primer estudio, en este se puede observar mayor interés por parte de las empresas desarrolladoras de software en implementar esta metodología dentro de su infraestructura organizacional, ya que reconocen la importancia de este, sin embargo, la implementación efectiva de estos procesos en sus organizaciones se encuentra en un estado apenas aceptable, indicando una necesidad clara de fortalecer

estos procesos para que las empresas de la industria del software en Colombia puedan lograr ventajas competitivas sostenibles y superar desafíos negativos.
La flexibilidad que GC posee es tal, que se ha considerado aplicar esta metodología en algunas de las etapas del ciclo de vida del desarrollo de software, como puede ser en la etapa de mantenimiento y que Viscaino *(2006)* menciona que en el mantenimiento de software es crucial que la información y el conocimiento se gestionen para ser reutilizados, ya que estos provienen de diversas fuentes y etapas del ciclo de vida del software, sin embargo, actualmente hay una escasez de trabajos que se centren en aplicar técnicas de gestión del conocimiento en el mantenimiento de software.
La razón de ser de esta metodología es que se busca facilitar el aprendizaje de los profesionistas en el desarrollo de sus proyectos al acceder a informacion que les sea de utilidad en sus funciones dentro de la empresa y en las etapas de desarrollo donde deben participar, teniendo como ejemplo a Viscano *(2006)* el cual enfoco su estudio con el siguiente objetivo en mente: *"El objetivo principal es mejorar la reutilización de la información mediante técnicas de razonamiento basado en casos, permitiendo a los ingenieros de mantenimiento aprovechar la experiencia y las lecciones aprendidas de otros profesionales" (p. 91-98).*
En conclusión, la gestión del conocimiento (GC) en la industria del software representa un desafío vital, donde el tiempo y el esfuerzo requeridos para implementarla se contraponen con la urgencia de entregar productos en un entorno de ritmo acelerado. Esta situación genera una amenaza real, ya que los individuos priorizan la entrega a tiempo sobre la búsqueda exhaustiva de conocimiento que podría mejorar la calidad y abonar al aprendizaje organizacional. Además, la mayor parte del conocimiento en ingeniería de software es tácito, lo que dificulta su conversión en conocimiento explícito, y la falta de herramientas adecuadas agrava esta situación. A pesar de estos desafíos, la GC en la industria del software se beneficia de la disposición de los profesionales para adoptar nuevas tecnologías y compartir conocimientos a través de medios digitales. Sin embargo, investigaciones muestran que, aunque se reconoce la importancia de la GC, su implementación efectiva en las organizaciones desarrolladoras de software aún se encuentra en un estado apenas aceptable. Esto resalta la necesidad de reforzar estos procesos para obtener ventajas competitivas sostenibles y superar desafíos negativos en un entorno tan dinámico como el de la industria del software.
4.3 Memoria organizacional
La memoria organizacional es un elemento fundamental dentro de la GC y como esta metodología se junta con la ingeniería de software pero antes hay que entender de que trata esta idea, siendo por parte de Nonaka*(1995)* que la define como un conjunto de conocimientos, experiencias, habilidades y competencias acumuladas por los miembros de una organización a lo largo del tiempo, siendo esencial para el funcionamiento y desarrollo de la organización, ya que permite la reutilización de información y conocimientos, facilita la resolución de problemas y mejora la toma de decisiones, siendo que la gestión del conocimiento busca capturar, organizar, compartir y aplicar esta memoria organizacional de manera efectiva, promoviendo un ambiente de aprendizaje continuo y colaborativo dentro de la organización.
Otro concepto dado para definir lo que es la memoria organizacional, es el presentado por Martin *(2011)* en donde explica que este concepto se centra en la captura, organización, y aplicación de conocimientos y experiencias acumuladas por los miembros de una organización a lo largo del tiempo.
Siendo la memoria organizacional el conjunto de conocimientos e información que una empresa ha reunido a lo largo de los años y la gestión del conocimiento el medio para poder repartirlo entre los miembros de la empresa, y los beneficios que esta relación podrían darle a la ingeniería del software serian gratos, tal y como menciona Martin*(2011)* en donde una gestión efectiva de la memoria organizacional seria crucial para mejorar la calidad del software y reducir los costos asociados con la reingeniería y la corrección de errores. Estos elementos son críticos a la hora de trabajar con productos orientados al software que definen la viabilidad de este producto y la facilidad o dificultad que este desarrollo supondrá al equipo de trabajo.
Para poder asegurar una gestión efectiva de la memoria organizacional existen distintas maneras para lograrlo y una de las formas de hacerlo es a través de sistemas de recomendación basados en ontologías y casos, y como explica Martin *(2011)* sobre estos métodos, los que utilizan técnicas de razonamiento basado en casos para identificar patrones y soluciones a problemas comunes en el desarrollo de software así como propone un sistema de recomendación que utiliza ontologías para representar el conocimiento acumulado sobre el aseguramiento de calidad en software, por lo que permite a los ingenieros de software acceder a recomendaciones basadas en casos previos que han resuelto problemas similares, promoviendo la reutilización de conocimientos y experiencias.
La implementación de este sistema de recomendación basado en ontologías y casos representa un avance significativo en la gestión del conocimiento en la ingeniería de software, ya que facilita el acceso a la memoria organizacional, ayudando a los ingenieros a evitar errores comunes y a mejorar la calidad del software. Además, al promover un ambiente de aprendizaje continuo, este sistema contribuye a la mejora de las prácticas de ingeniería de software en la organización.
4.4 Conocimiento empaquetado que soporta la aplicación del conocimiento
En la búsqueda de maximizar la efectividad del conocimiento, el concepto de conocimiento empaquetado emerge como una herramienta fundamental. Según O'Reilly y Tushman *(2004)*, el conocimiento empaquetado se define como *"el proceso de codificación del conocimiento en formas que se pueden compartir, almacenar y aplicar"* (p. xx). Esta definición resalta la importancia de transformar el conocimiento en recursos accesibles y aplicables en diversas situaciones.
Al respecto, Leonard y Sensiper *(1998)* afirman que *"El conocimiento empaquetado puede incluir manuales, guías*

de mejores prácticas, bases de datos estructuradas y cualquier otro recurso que condense y organice la información de manera efectiva" (p. xx). Esta afirmación subraya la versatilidad de las formas en que el conocimiento puede ser empaquetado para facilitar su aplicación en contextos prácticos.

La aplicación del conocimiento empaquetado se ve reforzada por la diversidad de fuentes y perspectivas. En palabras de Wenger *(1998), "El conocimiento no reside únicamente en individuos, sino que se construye y se comparte en comunidades de práctica"* (p. xx). Esta noción destaca la importancia de los entornos colaborativos y la participación en la aplicación efectiva del conocimiento empaquetado.

En resumen, el conocimiento empaquetado desempeña un papel crucial en la aplicación del conocimiento al proporcionar recursos accesibles y aplicables, y al fomentar la colaboración y el intercambio de ideas. Adoptar una mentalidad de aprendizaje continuo y estar abiertos a diversas perspectivas son elementos clave para aprovechar al máximo el potencial del conocimiento empaquetado en la resolución de problemas y la toma de decisiones en diferentes contextos.

Conclusión

La gestión del conocimiento en la ingeniería de software presenta una serie de ventajas notables que pueden transformar positivamente la manera en que se desarrollan y mantienen los sistemas de software. En primer lugar, al implementar prácticas efectivas de gestión del conocimiento, las organizaciones pueden mejorar significativamente la eficiencia y la calidad de sus procesos de desarrollo de software. Esto se logra al permitir la reutilización de conocimientos y experiencias previas, evitando la repetición de errores comunes y facilitando la adopción de mejores prácticas.

Además, la gestión del conocimiento en la ingeniería de software fomenta la innovación y el aprendizaje continuo dentro de las organizaciones. Al crear mecanismos para compartir ideas, lecciones aprendidas y enfoques exitosos, se crea un ambiente propicio para la generación de nuevas soluciones y el desarrollo de capacidades técnicas y creativas en el equipo.

Otra ventaja importante es la capacidad de tomar decisiones más informadas y fundamentadas. Al disponer de acceso rápido y estructurado a la información relevante, los líderes y los equipos de desarrollo pueden evaluar mejor las opciones disponibles y seleccionar la estrategia más adecuada para abordar los desafíos técnicos y comerciales.

Sin embargo, la implementación efectiva de la gestión del conocimiento en la ingeniería de software no está exenta de desafíos. Uno de los principales obstáculos radica en la necesidad de crear una cultura organizacional que valore y promueva el intercambio de conocimientos. Esto requiere un cambio cultural significativo, que puede encontrarse con resistencia por parte de aquellos que están acostumbrados a un enfoque más tradicional de trabajo.

Además, la identificación y captura de conocimiento relevante puede resultar complicada, especialmente en entornos donde el conocimiento está disperso en diferentes equipos, sistemas y documentos. La efectividad de los sistemas y herramientas de gestión del conocimiento puede verse comprometida si no se integran adecuadamente con los procesos existentes y si no se adaptan a las necesidades específicas de la organización y su contexto operativo.

Superar estos desafíos requiere un enfoque estratégico y un compromiso continuo por parte de la alta dirección. Es fundamental invertir en la capacitación del personal, desarrollar sistemas de incentivos que fomenten la colaboración y el intercambio de conocimientos, y utilizar tecnologías de la información y la comunicación de manera efectiva para facilitar la captura, almacenamiento y distribución de conocimientos dentro de la organización.

En resumen, si bien la gestión del conocimiento en la ingeniería de software ofrece numerosas ventajas, su implementación exitosa requiere un enfoque holístico que aborde tanto los aspectos culturales como tecnológicos de la organización. Sin embargo, los beneficios potenciales, como la mejora de la eficiencia, la promoción de la innovación y la toma de decisiones más fundamentadas, hacen que valga la pena el esfuerzo invertido en su implementación y desarrollo continuo.

RSL

En función al objetivo que se persigue con la presente investigación, se diseñó una indagatoria de carácter universal, que con base a su propósito es considerada como básica, pura, teórica o dogmática, debido a que se origina en un marco teórico y permanece en él, buscando solo incrementar el nivel de conocimiento de los interesados, absteniéndose de pruebas empíricas; en lo que se refiere a los medios para obtener la información, es posible indicar que es de tipo documental, porque se fundamenta en la búsqueda de producción científica alojada en bases de datos y documentos primarios *(Muntané, 2010)*.

La metodología a utilizar para la obtención de los documentos primarios, será la denominada Revisión Sistemática de Literatura (RSL), debido a que permite estar al día con los diversos temas de interés, tiene como objetivo recopilar y analizar críticamente múltiples estudios o trabajos de investigación a través de un proceso sistemático, garantizando que la información seleccionada cuente con validez o veracidad, calidad metodológica y confiabilidad o reproductividad de resultados (Manterola et. al, 2013).

El proceso de búsqueda que se llevó a cabo fue al establecer los elementos que servirían para poder realizar una investigación que nos aporte información que sea de utilidad para lo que se planea hacer y que además sea de calidad, esto mediante el uso de palabras clave para facilitar el proceso de investigación, (ver Anexo A), siendo en una columna las palabras clave principales y en la otra las palabras o términos que se relacionan con estas, además de sinónimos de estas.

Posteriormente se establecieron los criterios de exclusión e inclusión que fueron aplicados a los resultados

surgidos en la investigación que salieron de las búsquedas, que sirvieron como normativas que limitaron que elementos deben de ser descartados o aceptados dentro de la revisión sistemáticas (ver Anexo B), una vez con estos criterios de búsquedas se realizaron indagaciones en distintos motores de búsqueda así como bibliotecas virtuales para obtener los documentos que sean fiables para el documento a desarrollar.
Después de la depuración de los documentos obtenidos en la búsqueda, se sometieron a unas preguntas de calidad (ver Anexo C) para determinar finalmente si estos documentos obtenidos serian de utilidad para el trabajo que se tenía planeado desarrollar (ver Anexo D), los cuales se colocaron en una tabla donde se marcaba con Si o No si respondían esas preguntas de calidad.
Después de este último filtro de calidad, gran parte de los artículos presentados tenían claro los objetivos de por que realizan dicha investigación, así como el manejo, análisis y recopilación de datos para fundamentar lo presentado, siendo muy contados los documentos que no llegaron a cumplir dichos lineamientos, en donde la gran mayoría no tenía eran donde mostraran datos negativos sobre la investigación además de que no hablaban al final de la veracidad de los datos recopilados, siendo donde más flaquearon las investigaciones escogidas.
Siendo con esas investigaciones las que se escogieron para elaborar este documento, al haber pasado con éxito la mayoría de los filtros de calidad o de utilidad que se establecieron mediante este método de investigación.
Anexo
Anexo A:
Gestión del conocimiento
knowledge management
KM
Conocimiento
Datos
Ingeniería de software
Ingenieros de software
Calidad del software
Ciclo de vida de desarrollo del software
Systems Development Life Cycle
Software
software engineering
Equipos de desarrollo
Anexo B:
Criterios de inclusión
Criterios de exclusión
CI1: Los artículos tienen de título una de las palabras claves
CE1: Artículos que sean antes de los años 2000 o que no se actualicen con el paso del tiempo
CI2: Los artículos se enfocan en temas relacionados a la gestión del conocimiento o la ingeniera de software
CE2: Fuentes de información que sean de paga y no de acceso público o gratuito.
CI3: Los artículos se relacionan con las preguntas propuestas
CE3: Los artículos no vienen de fuentes confiables.
CI4: Artículos en español o ingles
CE4: Artículos que estén incompletos.
CI5: La información puede venir de artículos de investigación o libros
CE5: Artículos que no tengan derechos de autor
Anexo C:
Identificador
Pregunta
P1
¿Están claramente especificados los objetivos de la investigación?
P2
¿El estudio fue diseñado para lograr objetivos?
P3
¿Se describen claramente las técnicas utilizadas y se justifica su selección?
P4
¿Se miden adecuadamente las variables consideradas por el estudio?
P5
¿Se describen adecuadamente los métodos de recopilación de datos?
P6
¿Se describen adecuadamente los datos recopilados?
P7
¿Es claro el propósito del análisis de datos?
P8
¿Se utilizan técnicas estadísticas para analizar los datos adecuadamente descritas y su uso está justificado?
P9

¿Se presentan resultados negativos (si los hay)?
P10
¿Los investigadores discuten algún problema con la validez/confiabilidad de los resultados?
P11
¿Se responden adecuadamente todas las preguntas de investigación?
P12
¿Qué tan claros son los vínculos entre los datos, la interpretación y las conclusiones?
Anexo D:
Nombre del estudio
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
Gestión del conocimiento en las organizaciones: Fundamentos, metodología y praxis
Y
Y
Y
P
P
P
Y
P
N
Y
P
Y
Introduction: Knowledge Management in Software Engineering
Y
Y
Y
P
Y
P
Y
N
N
Y
P
Y
Gestión del conocimiento y cultura organizacional en instituciones de educación superior chilenas
Y
Y
P
P
P
Y
Y
N
N
N
P
Y
Adoption of Knowledge Management Practices in Software Engineering Organizations: A Survey of Software Engineers' Perceptions
Y
Y
Y

Y
P
Y
Y
Y
P
N
P
Y
The role of knowledge management in global software engineering
Y
Y
Y
Y
Y
P
P
Y
N
N
P
P
Datos, información y conocimiento. De la sociedad de la información a la sociedad del conocimiento
P
N
N
Y
P
Y
P
N
N
N
Y
Y
Concepto, tipos y dimensiones del conocimiento: configuración del conocimiento estratégico
P
N
N
Y
P
Y
Y
P
N
N
P
Y
Ciclo de vida de desarrollo ágil de software seguro
P
P
P
P
Y
Y
Y
P
N
N
P
Y
Una revisión de metodologías seguras en cada fase del ciclo de vida del desarrollo de software
Y
P
P

Y
P
Y
N
N
N
N
P
Y
La Gestión del Conocimiento en las Organizaciones
Y
Y
Y
Y
Y
P
Y
Y
P
N
N
Y
Y
Gestión del conocimiento, liderazgo, diseño e implementación de la estrategia: un estudio empírico en pequeñas y medianas empresas
Y
Y
Y
Y
Y
Y
Y
Y
P
P
N
Y
Y
Gestión del conocimiento y cultura organizacional en instituciones de educación superior chilenas
Y
Y
Y
Y
Y
Y
Y
Y
N
N
N
Y
Y
Evolución de la Ingeniería del Software y la formación de profesionales
Y
N
N
Y
P
P
Y
P
N
N
Y
Y
Ingeniería de Software
N
N

Y
Y
Y
Y
Y
N
N
N
Y
Y
Gestión del conocimiento en ingeniería de software
Y
Y
P
Y
P
Y
Y
P
N
N
Y
Y
Adoption of Knowledge Management Practices in Software Engineering Organizations: A Survey of Software Engineers' Perceptions
Y
Y
Y
Y
Y
Y
Y
Y
P
N
Y
Y
What Do We Know about Knowledge Management? Practical Implications for Software Engineering
Y
Y
Y
Y
Y
Y
Y
P
P
N
Y
Y
Introduction: Knowledge Management in Software Engineering
Y
Y
P
Y
Y
Y
Y
P
N
N
Y
Y
Gestión del conocimiento como apoyo para la mejora de procesos software en las micro, pequeñas y medianas empresas

Y
Y
Y
Y
Y
Y
Y
P
N
N
Y
Y
Aplicando gestión del conocimiento en el proceso de mantenimiento del software
Y
Y
P
Y
P
Y
P
N
N
N
Y
Y
Software Engineering Worldwide
P
P
P
Y
P
Y
Y
P
N
N
Y
Y
Software Engineering: A Hands-On Approach
Y
P
P
P
Y
Y
Y
Y
N
N
Y
Y
Exploring knowledge management in agile software development organizations
Y
Y
Y
P
P
Y
Y
Y
P
N
Y
Y
The Analysis of obstacles and solutions for software enterprises to implement knowledge management

Y
Y
Y
Y
P
P
P
Y
Y
N
Y
Y

Referencias

Boehm, B. W. (1981). Software Engineering Economics. Prentice-Hall.

Canós, J. H., Letelier, P., & Penadés, M. C. (2003). Metodologías ágiles en el desarrollo de software. Universidad Politécnica de Valencia, Valencia, 1-8.

Capote, J., Astaiza, C. J. L., Calvache, C. J. P., Ramírez, A. D. J. G., & Collazos, C. A. (2008). Gestión del conocimiento como apoyo para la mejora de procesos software en las micro, pequeñas y medianas empresas. Ingenieria e investigacion, 28(1), 137-145.

Carvalho, J. M., & Santos, M. A. (2012). The Role of Team Dynamics in Software Development. Journal of Systems and Software.

Ciprés, M. S., & Llusar, J. C. B. (2004). Concepto, tipos y dimensiones del conocimiento: configuración del conocimiento estratégico. Revista de economía y empresa, 22(52), 175-196.

Crosby, P. B. (1979). Quality is Free: The Art of Making Quality Certain. McGraw-Hill.

Galvis-Lista, E. A., González-Zabala, M. P., & Sánchez-Torres, J. M. (2016). Un estudio exploratorio sobre el estudio de implementación de procesos de gestión del conocimiento en organizaciones desarrolladoras de software en Colombia. Revista Escuela de Administración de Negocios, 80, 73-90. Universidad del Magdalena y Universidad Nacional de Colombia.

Gutiérrez, MF (2008). Gestión del conocimiento en las organizaciones: Fundamentos, metodología y praxis. En Ediciones Trea eBooks . http://diposit.ub.edu/dspace/bitstream/2445/123350/1/Pe%cc%81rez-Montoro%20%282008%29%20Gestio%cc%81n%20del%20conocimiento%20en%20las%20organizaciones.

Hernández Bejarano, M., & Baquero Rey, L. E. (2020). Ciclo de vida de desarrollo ágil de software seguro. Editorial Los Libertadores.

Humphrey, W. S. (1989). Managing the Software Process. Addison-Wesley.

I. Rus and M. Lindvall, "Guest Editors' Introduction: Knowledge Management in Software Engineering" in IEEE Software, vol. 19, no. 03, pp. 26-38, 2002.
doi: 10.1109/MS.2002.1003450

IEEE Computer Society. (1991). IEEE Standard Glossary of Software Engineering Terminology (ANSI). IEEE Computer Society Press.

ISO/IEC. (2001). ISO/IEC 9126-1: Information technology – Software product quality – Part 1: Quality model. ISO/

Iza Carate, M. D. (2018). Gestión del conocimiento en ingeniería de software. RECIMUNDO, 2(4), 32-47. https://doi.org/10.26820/recimundo/2.(4).octubre.2018.32-47

Kute, S. S., & Thorat, S. D. (2014). A review on various software development life cycle (SDLC) models. International Journal of Research in Computer and Communication Technology, 3(7), 778-779.

Leonard, D., & Sensiper, S. (1998). The role of tacit knowledge in group innovation. California Management Review, 40(3).

Maida, E. G., & Pacienzia, J. (2015). Metodologías de desarrollo de software.

Martín, M. (2011). Memoria Organizacional Basada en Ontologías y Casos para un Sistema de Recomendación en Aseguramiento de Calidad. 10.13140/RG.2.1.1388.2483.

Marulanda L., C., & Ceballos H., J. (2012). Una revisión de metodologías seguras en cada fase del ciclo de vida del desarrollo de software. Ingenierías USBMed, 3(1), 15–22. https://doi.org/10.21500/20275846.260

Montuschi, L. (2001). Datos, información y conocimiento. De la sociedad de la información a la sociedad del conocimiento. Serie Documentos de Trabajo de la Universidad del CEMA, 192(6), 2-32.

N. Sharma, K. Singh and D. P. Goyal, "Adoption of Knowledge Management Practices in Software Engineering Organizations: A Survey of Software Engineers' Perceptions," 2012 Second International Conference on Advanced Computing & Communication Technologies, Rohtak, India, 2012, pp. 24-29, doi: 10.1109/ACCT.2012.17.

Nonaka, I., & Takeuchi, H. (1995). The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. Oxford University Press.

Núñez Paula I.(2004).La gestión de la información, el conocimiento, la inteligencia y el aprendizaje organizacional desde una perspectiva socio-psicológica. Acimed 2004; 12(3). Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1024-94352004000300004&lng=es&nrm=iso&tlng=es

O'Reilly, C. A., & Tushman, M. L. (2004). The ambidextrous organization. Harvard Business Review, 82(4).

Pantaleo, G., Rinaudo, L.(2015). Ingeniería de software. https://books.google.es/books?id=rjxyEAAAQBAJ&lpg=PR3&ots=5lfDt_8A1N&dq=concepto%20del%20software&lr&hl=es&pg=PA4#v=onepage&q=concepto%20del%20software&f=false

Piattini, M. (s.f). Evolución de la Ingeniería del Software y la formación de profesionales. REVISTA INSTITUCIONAL DE LA FACULTAD DE INFORMÁTICA | UNLP, 15–17.http://sedici.unlp.edu.ar/bitstream/handle/10915/57358/Documento_completo.pdf-PDFA.pdf?sequence=1

Pressman, R.,S.(2010). Ingeniería de software: Un enfoque practico. McGraw-Hill. Recuperado en: https://www.javier8a.com/itc/bd1/ld-Ingenieria.de.software.enfoque.practico.7ed.Pressman.PDF

Rayner, K. J., & Rayner, M. J. (2010). Software Development Teams: Characteristics and Effectiveness. Journal of Systems and Software.

Rodríguez-Ponce, E., Pedraja-Rejas, L., Delgado, M. E., & Rodríguez-Ponce, J. (2010). GESTIÓN DEL CONOCIMIENTO, LIDERAZGO, DISEÑO E IMPLEMENTACIÓN DE LA ESTRATEGIA: UN ESTUDIO EMPÍRICO EN PEQUEÑAS Y MEDIANAS EMPRESAS. Ingeniare. Revista Chilena De Ingeniería, 18(3), 373–382. https://doi.org/10.4067/s0718-33052010000300011

Rodríguez-Ponce, E., Pedraja-Rejas, L., Muñoz-Fritis, C., & Araneda-Guirriman, C. (2022). Gestión del conocimiento y cultura organizacional en instituciones de educación superior chilenas. Ingeniare. Revista Chilena De Ingeniería , 30 (2), 266–278. https://doi.org/10.4067/s0718-33052022000200266

Rus and M. Lindvall, "Guest Editors' Introduction: Knowledge Management in Software Engineering" in IEEE Software, vol. 19, no. 03, pp. 26-38, 2002.
doi: 10.1109/MS.2002.1003450

S. Khalid, T. Shehryar and S. Arshad, "The role of knowledge management in global software engineering," 2015 International Conference on Industrial Engineering and Operations Management (IEOM), Dubai, United Arab Emirates, 2015, pp. 1-5, doi: 10.1109/IEOM.2015.7093908.

Sarker, A. K., & Sarker, S. K. (2018). Knowledge Management in Software Development: A Systematic Literature Review. Journal of Information Technology.

Shylesh, S. (2017, April). A study of software development life cycle process models. In National Conference on Reinventing Opportunities in Management, IT, and Social Sciences (pp. 534-541).

Sommerville, I. (1998). Ingeniería de Software. Addison Wesley Iberoamericana, Wilnington (EEUU). https://gc.scalahed.com/recursos/files/r161r/w25469w/ingdelsoftwarelibro9_compressed.pdf

T. Dingsøyr, F. O. Bjørnson and F. Shull, "What Do We Know about Knowledge Management? Practical Implications for Software Engineering," in IEEE Software, vol. 26, no. 3, pp. 100-103, May-June 2009, doi: 10.1109/MS.2009.82.

Vargas-Sánchez, A., & Moreno-Dominguez, M. J. (2005). La Gestión del Conocimiento en las Organizaciones. ResearchGate.
https://www.researchgate.net/publication/277203089_La_Gestion_del_Conocimiento_en_las_Organizaciones

Vizcaíno, A., Soto, J. P., García, F., Ruiz, F., & Piattini, M. (2006). Aplicando gestión del conocimiento en el proceso de mantenimiento del software. Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, 10(31), 91-98.

Wenger, E. (1998). Communities of practice: Learning, meaning, and identity. Cambridge University Press.