



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Dirección General de Estudios de Posgrado

Facultad de Ingeniería de Sistemas e Informática

Unidad de Posgrado

**Un modelo de priorización de requisitos mediante
herramientas colaborativas CMS**

TESIS

Para optar el Grado Académico de Magíster en Ingeniería de
Sistemas e Informática con mención en Gestión de Tecnología de
Información y Comunicaciones

AUTOR

Elias Daniel BRAVO LOAYZA

ASESOR

Mg. Percy Edwin DE LA CRUZ VÉLEZ DE VILLA

Lima, Perú

2021



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Bravo, E. (2021). *Un modelo de priorización de requisitos mediante herramientas colaborativas CMS*. [Tesis de maestría, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Unidad de Posgrado]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Elias Daniel Bravo Loayza
Tipo de documento de identidad	DNI
Número de documento de identidad	10455982
URL de ORCID	https://orcid.org/0000-0001-6142-8463
Datos de asesor	
Nombres y apellidos	Percy Edwin De La Cruz Vélez de Villa
Tipo de documento de identidad	DNI
Número de documento de identidad	08583141
URL de ORCID	https://orcid.org/0000-0002-4943-7620
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	León Fernández Cayo Víctor
Tipo de documento	DNI
Número de documento de identidad	07001405
Miembro del jurado 1	
Nombres y apellidos	Rosa Menéndez Mueras
Tipo de documento	DNI
Número de documento de identidad	10246770
Miembro del jurado 2	
Nombres y apellidos	Luis Soto Soto
Tipo de documento	DNI

Número de documento de identidad	22409532
Datos de investigación	
Línea de investigación	C.0.3.22. Ingeniería de software
Grupo de investigación	SOFTGTIC
Agencia de financiamiento	Sin financiamiento.
Ubicación geográfica de la investigación	Perú, Lima, Lima, Lima Av. Universitaria s/n. - Av. Venezuela cdra. 34, Lima - 12°03'30"S 77°05'00"O
Año o rango de años en que se realizó la investigación	2017 - 2021
URL de disciplinas OCDE	2.02.04 - Ingeniería de sistemas y comunicaciones URIS: https://purl.org/pe-epo/ocde/ford#2.02.04



UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
Universidad del Perú. Decana de América
Facultad de Ingeniería de Sistemas e Informática
Vicedecanato de Investigación y Posgrado
Unidad de Posgrado

**ACTA DE SUSTENTACIÓN VIRTUAL DE TESIS PARA OPTAR EL GRADO
ACADÉMICO DE MAGÍSTER EN INGENIERÍA DE SISTEMAS E
INFORMÁTICA CON MENCIÓN EN GESTIÓN DE TECNOLOGÍA DE
INFORMACIÓN Y COMUNICACIONES**

A los treinta (30) días del mes de diciembre de 2021, siendo las 16:20 pm., se reunieron en la sala virtual <https://meet.google.com/vvr-ptbk-dih> el Jurado de Tesis conformado por los siguientes docentes:

Mg. Cayo Víctor León Fernández (Presidente)
Mg. Rosa Menéndez Mueras (Miembro)
Dr. Luis Soto Soto (Miembro)
Mg. Percy Edwin De La Cruz Vélez de Villa (Miembro Asesor)

Se inició la Sustentación invitando al candidato a Magíster **ELIAS DANIEL BRAVO LOAYZA**, para que realice la exposición oral y virtual de la tesis para optar el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Gestión de Tecnología de Información y Comunicaciones, siendo la Tesis intitulada:

“UN MODELO DE PRIORIZACIÓN DE REQUISITOS MEDIANTE HERRAMIENTAS COLABORATIVAS CMS”

Concluida la exposición, los miembros del Jurado de Tesis procedieron a formular sus preguntas que fueron absueltas por el graduando; acto seguido se procedió a la evaluación correspondiente, habiendo obtenido la siguiente calificación:

..... CATORCE (14) - APROBADO

Por tanto, el presidente del Jurado, de acuerdo al Reglamento General de Estudios de Posgrado, otorga al Bachiller **ELIAS DANIEL BRAVO LOAYZA** el Grado Académico de Magíster en Ingeniería de Sistemas e Informática con mención en Gestión de Tecnología de Información y Comunicaciones.

Siendo las. 17:50 horas, el presidente del Jurado de Tesis, da por concluido el acto académico de Sustentación de Tesis.

Mg. Cayo Víctor León Fernández
(Presidente)

Mg. Rosa Menéndez Mueras
(Miembro)

Dr. Luis Soto Soto
(Miembro)

Mg. Percy Edwin De La Cruz Vélez de Villa
(Miembro Asesor)



Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Vicedecanato de Investigación y Posgrado

Unidad de Posgrado

INFORME DE EVALUACION DE ORIGINALIDAD

N° 005-UPG-VDIP-FISI-2020

1. Autoridad Académica que emite el Informe de Originalidad:	Directora de la Unidad de Posgrado
2. Apellidos y Nombres de la autoridad académica:	Mamani Rodriguez Zoraida Emperatriz
3. Operador del programa informático de similitudes:	Mamani Rodriguez Zoraida Emperatriz
4. Documento evaluado:	Tesis para Posgrado Titulo: "Un Modelo de Priorización de Requisitos Mediante Herramientas Colaborativas CMS"
5. Autor del documento:	Elias Daniel Bravo Loayza
6. Fecha de recepción de documento	ultima recepcion: 12/06/2020
7. Fecha de aplicación del programa detector de similitudes:	ultima revision: 15/06/2020
8. Software utilizado:	Turnitin
9. Configuración del programa detector de similitudes:	i. Excluye textos entrecomillados: SI ii. Excluye biografías: SI iii. Excluye cadenas menores a 40 palabras: SI iv. Otro criterio (especificar): NO
10. Porcentaje de similitudes según programa detector de similitudes	nueve por ciento (9%)
11. Fuentes originales de las similitudes encontradas	Se adjuntan en doce (12) fojas al presente informe
12. Observaciones:	Ninguna.
13. Calificación de originalidad i. Documento cumple criterios de originalidad, sin observaciones. ii. Documento cumple criterio de originalidad, con observaciones. iii. Documento no cumple criterios de originalidad.	Documento cumple criterio de originalidad, sin observaciones
14. Fecha del Informe:	15/06/2020

[Firma manuscrita]



Mg. Zoraida Mamani Rodriguez
Directora de la UPG-FISI

Elias Daniel Bravo Loayza

UN MODELO DE PRIORIZACIÓN DE REQUISITOS MEDIANTE HERRAMIENTAS COLABORATIVAS CMS

Tesis presentada a la Universidad
Nacional Mayor de San Marcos
Lima - Perú, para obtener el grado
de Magíster en Ingeniería de
Sistemas con mención en gestión
de Tecnología de Información y
Comunicaciones.

Orientador: Mg. Percy E. De La Cruz Vélez de Villa

Universidad Nacional Mayor de San Marcos
Facultad de Ingeniería de Sistemas e Informática
Unidad de Postgrado

Lima - Perú
Diciembre – 2021

© Elias Daniel Bravo Loayza, 2021
Todos los derechos reservados

FICHA CATALOGRÁFICA

Elias Daniel Bravo Loayza

Lima, Perú 2021

Orientador: Mg. Percy Edwin De La Cruz Vélez de Villa

Disertación: Maestría en Ingeniería de Sistemas e Informática con
mención en gestión de Tecnología de Información y
Comunicaciones

Universidad Nacional Mayor de San Marcos - Facultad de Ingeniería
de Sistemas e Informática - UPG, 2021.

Páginas: 174 páginas.

Dedicatoria

A mi familia.

Agradecimientos

A mi familia, por todo el apoyo y comprensión que he recibido en mi vida.

A mi asesor de tesis, el Mg. Percy E. De La Cruz Vélez De Villa, por sus palabras, sugerencias y recomendaciones.

A la Universidad Nacional Mayor de San Marcos y a la Facultad de Ingeniería de Sistemas e Informática, por haberme permitido realizar y concluir mis estudios de postgrado.

... Y a todos aquellos que no menciono aquí pero que, sin embargo, merecen las siguientes palabras: muchas gracias.

Elias Daniel Bravo Loayza

Resumen

La priorización de requisitos es un tema importante de investigación, sobre el cual se han elaborado numerosos trabajos a lo largo de los años. Sin embargo, las múltiples propuestas solo se centran en un aspecto: el práctico o el teórico. En el presente trabajo, se muestra una propuesta integral, que abarca el planteamiento teórico a través de un modelo, la implementación práctica y, finalmente, la validación.

Actualmente, existen distintas técnicas de priorización de requisitos que permiten obtener las prioridades relativas que cada uno de los *stakeholders* le asignan a fin de obtener una lista unificada y priorizada. En entornos distribuidos, se requiere el apoyo de una herramienta que permita a los *stakeholders* realizar las tareas de priorización. Por otro lado, la priorización de requisitos se realiza dentro de una metodología de desarrollo de *software*; por tanto, todo planteamiento válido deberá estar soportado por alguna metodología.

Se propone un modelo de priorización de requisitos que permitirá al equipo de desarrollo obtener a una lista única y aceptada de requisitos a partir de la priorización de los *stakeholders*, mediante la implementación práctica del marco teórico a través de una herramienta de software colaborativo de amplia difusión.

Esta investigación se realizó bajo el enfoque cuantitativo, mediante el cual se alcanzarán los objetivos y se verificarán las hipótesis propuestas a través de un experimento debidamente pautado y documentado en un caso de estudio.

Se obtuvo como resultado una lista priorizada por todos los *stakeholders* en modo colaborativo, sin reportar alguna dificultad o fallo en su uso.

Palabras claves: requisitos, priorización de requisitos, *CMS*.

Abstract

Requirement prioritization is an important research topic, on which numerous papers have been written over the years. However, the many proposals only focus on either the practical or the theoretical aspect. This work presents a comprehensive proposal, covering the theoretical approach through a model, the practical implementation and, finally, the validation.

At present, there are different requirement prioritization techniques that make it possible to obtain the relative priorities assigned by each stakeholder to obtain a unified and prioritized list. In distributed environments, the support of a tool that allows the stakeholders to perform the prioritization tasks is required. On the other hand, requirement prioritization is performed within a software development methodology; therefore, any valid approach must be supported by some methodology.

This work proposes a requirement prioritization model that will allow the development team to obtain a single and accepted list of requirements based on the prioritization of stakeholders, by means of the practical implementation of the theoretical framework through a widely used collaborative software tool. The research was conducted under the quantitative approach, through which the objectives will be achieved. Furthermore, the proposed hypotheses will be verified through a properly paced and documented experiment in a case study.

As a result, a list prioritized by all stakeholders was obtained in collaborative mode, without reporting any difficulty or failure in its use.

Keywords: requirements, requirements prioritization, *CMS*.

CONTENIDO

CONTENIDO.....	10
LISTA DE CUADROS	13
LISTA DE FIGURAS	14
1. CAPÍTULO I: INTRODUCCIÓN.....	16
1.1 SITUACIÓN PROBLEMÁTICA	17
1.2 FORMULACIÓN DEL PROBLEMA	18
1.2.1 <i>Problema General</i>	19
1.2.1 <i>Problemas Específicos</i>	19
1.3 MOTIVACIÓN.....	20
1.4 OBJETIVOS	21
1.4.1 <i>Objetivo General</i>	21
1.4.2 <i>Objetivos Específicos</i>	21
1.5 JUSTIFICACIÓN DE LA INVESTIGACIÓN	21
1.5.1 <i>Justificación Teórica</i>	21
1.5.2 <i>Justificación Práctica</i>	22
1.6 SUPUESTOS	23
1.7 ORGANIZACIÓN DE LA TESIS.....	24
2. CAPÍTULO II: MARCO TEÓRICO.....	26
2.1 MARCO FILOSÓFICO O EPISTEMOLÓGICO DE LA INVESTIGACIÓN	26
2.2 ANTECEDENTES DE INVESTIGACIÓN	27
2.2.1 <i>Técnicas de Priorización de Requisitos y Diferentes Aspectos de la Priorización</i>	27
2.2.2 <i>Comparación de Métodos de Priorización de Requisitos en la Industria: Un estudio de la Efectividad de los Métodos de Ranking, Árbol de Búsqueda Binaria y la Matriz Wieggers</i>	27
2.2.3 <i>Usando Wikis como Estrategia Colaborativa para Apoyar la Elicitación de Requisitos de Software</i>	28
2.2.4 <i>Un diseño experimental para comparar las Técnicas de priorización de requisitos de software</i> 28	
2.2.5 <i>Usando redes sociales populares para apoyar la elicitación, la priorización y negociación de requisitos</i>	28
2.2.6 <i>ReproTizer: Una Herramienta de Priorización de Requisitos de Software Completamente Implementada</i>	29
2.2.7 <i>MBRP: Un modelo de priorización de requisitos basado en el algoritmo Pagerank</i>	30
2.2.8 <i>iStar-p: Un lenguaje de modelado para la priorización de requisitos</i>	30
2.2.9 <i>Priorización de requisitos de software con el lenguaje de requisitos orientado a objetivos</i>	31
2.2.10 <i>Uso de redes sociales para identificar los requisitos iniciales del usuario para pequeñas empresas de software</i>	31
2.2.11 <i>Una herramienta para el apoyo al proceso de priorización de requisitos en un entorno de desarrollo de software ágil</i>	32
2.2.12 <i>SRPTackle: una técnica de priorización de requisitos de software semiautomatizada</i> 32	
2.3 BASES TEÓRICAS.....	32
2.3.1 <i>Requisito</i>	32
2.3.2 <i>Stakeholders</i>	36
2.3.3 <i>Ingeniería de Requisitos</i>	37
2.3.4 <i>Gestión de Requisitos</i>	39
2.3.5 <i>Documentación de Requisitos</i>	42
2.3.6 <i>Modelo para el Proceso de Desarrollo de Software</i>	43
2.3.7 <i>Software Colaborativo</i>	60
2.3.8 <i>Priorización de Requisitos</i>	62

2.3.9	<i>Técnicas de Priorización de Requisitos</i>	66
2.3.10	<i>Software Libre y de Código Abierto</i>	67
2.3.11	<i>Licencia de Software</i>	68
3.	CAPÍTULO III: METODOLOGÍA	72
3.1	TIPO Y DISEÑO DE LA INVESTIGACIÓN	72
3.2	HIPÓTESIS	73
3.2.1	<i>Hipótesis General</i>	73
3.2.2	<i>Hipótesis Específicas</i>	73
3.3	VARIABLES E INDICADORES	74
3.3.1	<i>Variable Independiente</i>	74
3.3.2	<i>Variable Dependiente</i>	75
3.4	OPERACIONALIZACIÓN DE VARIABLES	76
3.5	MATRIZ DE CONSISTENCIA	77
3.6	METODOLOGÍA	78
3.6.1	<i>Unidad de análisis</i>	78
3.6.2	<i>Población de estudio</i>	78
3.6.3	<i>Tamaño de la muestra</i>	79
3.6.4	<i>Selección de la muestra</i>	79
3.6.5	<i>Técnica de Recolección de datos</i>	79
3.6.1	<i>Técnica de procesamiento y análisis de datos</i>	80
4.	CAPÍTULO IV: ESTADO DEL ARTE	81
4.1	TAXONOMÍA	81
4.2	METODOLOGÍA DE BÚSQUEDA	82
4.3	REVISIÓN DEL ESTADO DEL ARTE	84
4.3.1	<i>Benchmarking de trabajos sobre trabajos relacionados</i>	86
4.3.2	<i>Principales Técnicas de Priorización de Requisitos</i>	89
4.3.3	<i>Críticas a las Técnicas de Priorización</i>	101
4.4	HERRAMIENTAS DE SOFTWARE	102
4.4.1	<i>Herramientas Representativas</i>	102
4.5	PRIORIZACIÓN DE REQUISITOS EN ENTORNOS DISTRIBUIDOS	103
4.6	HERRAMIENTAS DE COLABORACIÓN PARA OBTENER REQUISITOS	104
4.6.1	<i>El Uso de Wikis como Herramienta de Interacción Colaborativa</i>	104
4.6.2	<i>Requisitos basado en Foros</i>	105
4.7	SISTEMAS DE GESTIÓN DE CONTENIDOS - CMS	106
4.7.1	<i>Arquitectura del Diseño del Sistema</i>	106
4.7.2	<i>CMS de Código Abierto</i>	107
4.7.3	<i>CMS Comerciales</i>	107
4.8	PROCESO DE PRIORIZACIÓN	108
4.9	PRIORIZACIÓN DE REQUISITOS EN ENTORNOS DISTRIBUIDOS	108
4.10	PRIORIZACIÓN DE REQUISITOS CON VOTO ACUMULATIVO	109
4.10.1	<i>Investigaciones actuales sobre Voto Acumulativo</i>	110
4.11	SISTEMAS DE GESTIÓN DE CONTENIDOS – CMS	110
4.11.1	<i>Drupal</i>	111
4.11.2	<i>Joomla</i>	111
4.11.3	<i>WordPress</i>	112
4.11.4	<i>Plugins WordPress</i>	112
5.	CAPÍTULO V: APORTE	114
5.1	MODELO PROPUESTO	114
5.2	IMPLEMENTACIÓN DEL MODELO PROPUESTO	116
5.2.1	<i>Técnica de Priorización</i>	116
5.2.2	<i>Modelo de Desarrollo de Software</i>	118
5.2.3	<i>Licencia de Software</i>	119
5.2.4	<i>Herramienta Colaborativa</i>	119
5.2.5	<i>Componente para la Herramienta Colaborativa</i>	120

5.3	CASO DE ESTUDIO.....	121
5.3.1	<i>Importancia de la Gestión Documentaria</i>	122
5.3.2	<i>Un Caso de Prueba Basado en Moreq</i>	122
6.	CAPÍTULO VI: RESULTADOS Y DISCUSIÓN.....	125
6.1	ANÁLISIS, INTERPRETACIÓN Y DISCUSIÓN DE LOS RESULTADOS.....	125
6.1.1	<i>Contexto de la investigación</i>	125
6.1.2	<i>Objetivo de la Investigación</i>	126
6.2	PRUEBAS DE HIPÓTESIS.....	127
6.2.1	<i>Instalación</i>	127
6.2.2	<i>Diseño</i>	127
6.2.3	<i>Estructura</i>	127
6.2.4	<i>Operación</i>	128
6.2.5	<i>Puntos a Validar</i>	128
6.2.6	<i>Entorno</i>	129
6.3	PRESENTACIÓN DE RESULTADOS.....	129
6.3.1	<i>Resultados para los indicadores</i>	131
7.	CAPÍTULO VII: CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS	133
7.1	CONCLUSIONES	133
7.2	RECOMENDACIONES.....	134
7.3	TRABAJOS FUTUROS.....	135
	REFERENCIAS BIBLIOGRÁFICAS.....	136
	ÍNDICE DE PALABRAS.....	149
	ANEXOS	151
	ANEXO A – SELECCIÓN DE LA TÉCNICA DE PRIORIZACIÓN.....	151
	ANEXO B – SELECCIÓN DE LA HERRAMIENTA CMS	154
	ANEXO C – SELECCIÓN DEL PLUGIN PARA LA VOTACIÓN DE LOS REQUISITOS	160
	ANEXO D – IMPLEMENTACIÓN DEL PLUGIN PARA EL CÁLCULO DEL VOTO ACUMULATIVO	163
	ANEXO E – LISTADO DE REQUISITOS ESTÁNDARES PARA LA EVALUACIÓN DEL MODELO.....	168
	ANEXO F – ALGORITMO DE IMPLEMENTACIÓN DE LA TÉCNICA VOTO ACUMULATIVO - VA	171

LISTA DE CUADROS

TABLA 1 OPERACIONALIZACIÓN DE VARIABLES	76
TABLA 2 MATRIZ DE CONSISTENCIA	77
TABLA 3 EVALUACIÓN DEL ESTADO DEL ARTE DE LAS INVESTIGACIONES ACTUALES.	86
TABLA 4 EJEMPLO DE MATRIZ CREADA CON PROCESO ANALÍTICO JERÁRQUICO	91
TABLA 5 TÉCNICAS DE PRIORIZACIÓN MÁS POPULARES.....	110
TABLA 6 APARICIÓN DE LAS TÉCNICAS EN LOS ARTÍCULOS DE INVESTIGACIÓN EN LA INDIA	116
TABLA 7 EVALUACIÓN DE LA FACILIDAD DE USO DE LA IMPLEMENTACIÓN DEL MODELO	132
TABLA 8 COMPARACIÓN DE LAS HERRAMIENTAS CMS CANDIDATAS	157
TABLA 9 COMPARACIÓN DE LOS PLUGINS DE WORDPRESS PARA IMPLEMENTAR EL VOTO ACUMULATIVO.....	162

LISTA DE FIGURAS

FIGURA 1 INTERFAZ DE UNA HERRAMIENTA DE PRIORIZACIÓN	29
FIGURA 2 LISTADO DE STAKEHOLDERS	30
FIGURA 3 MODELO ITERATIVO.....	33
FIGURA 4 REQUISITOS DE USUARIO Y REQUISITOS DEL SISTEMA.....	35
FIGURA 5 ROL DE LA PRIORIZACIÓN DE REQUISITOS EN LA INGENIERÍA DE SOFTWARE.....	38
FIGURA 6 ACTIVIDADES DEL DESARROLLO DE SOFTWARE	43
FIGURA 7 CICLO DE VIDA DEL RUP	48
FIGURA 8 ENTERPRISE UNIFIED PROCESS [EUP].....	49
FIGURA 9 PRÁCTICAS DE PROGRAMACIÓN EXTREMA	52
FIGURA 10 PRODUCT BACKLOG	53
FIGURA 11 ESTRUCTURA DE LA NORMA ISO/IEC 12207, EN SU PROCESO PRINCIPAL DEL CICLO DE VIDA.....	55
FIGURA 12 ESTRUCTURA DE MOPROSOFT	58
FIGURA 13 ASPECTOS DE NEGOCIOS EN LA PRIORIZACIÓN DE REQUISITOS	64
FIGURA 14 ASPECTOS TÉCNICOS EN LA PRIORIZACIÓN DE REQUISITOS.....	66
FIGURA 15 ESQUEMA DE EXPERIMENTO Y VARIABLES	74
FIGURA 16 ÁRBOL DE BÚSQUEDA BINARIA.....	94
FIGURA 17 ORDENAMIENTO BURBUJA	97
FIGURA 18 TÉCNICA DE GRUPOS PRIORITARIOS.....	98
FIGURA 19 TÉCNICA MoSCoW.....	100
FIGURA 20 MATRIZ DE WIEGER	101
FIGURA 21 MODELO-VISTA-CONTROLADOR.....	107
FIGURA 22 LOS CINCO ELEMENTOS DEL MODELO PROPUESTO.....	115
FIGURA 23 FACILIDAD DE USO DE LAS PRINCIPALES TÉCNICAS DE PRIORIZACIÓN.....	117
FIGURA 24 EJECUCIÓN DE LA PRUEBA DE PARTE DE LOS PARTICIPANTES.....	128
FIGURA 25 ACCESO AL COMPONENTE VOTO ACUMULATIVO	130
FIGURA 26 PANEL ‘VOTACIÓN’ DEL COMPONENTE VOTO ACUMULATIVO.....	130
FIGURA 27 PANEL ‘PRIORIZACIÓN’ DEL COMPONENTE VOTO ACUMULATIVO	131
FIGURA 28 LISTA DE LOS QUINCE SOFTWARES COLABORATIVOS MÁS USADOS EL 2019	155
FIGURA 29 INFORMACIÓN ESTÁNDAR DEL PLUGIN WORDPRESS	163
FIGURA 30 PLUGIN INSTALADO DE VOTO ACUMULATIVO	164
FIGURA 31 PESTAÑAS DEL PLUGIN VOTO ACUMULATIVO	164
FIGURA 32 PESTAÑA PRIORIZACIÓN DEL PLUGIN VOTO ACUMULATIVO.....	164
FIGURA 33 CÓDIGO FUENTE DE LA PESTAÑA ‘VOTACIÓN’	166
FIGURA 34 CÓDIGO FUENTE DE LA PESTAÑA ‘PRIORIZACIÓN’	167
FIGURA 35 ALGORITMO DE LA IMPLEMENTACIÓN DEL VOTO ACUMULATIVO.....	173

Abreviaturas

API	<i>Application Programming Interface</i> Interfaz de Programación de Aplicaciones
CASE	<i>Computer Aided Software Engineering</i> Ingeniería de <i>Software</i> Asistida por Computadora
CMMI	<i>Capability Maturity Model Integration</i> Modelo de Madurez de Capacidad Integrado
CMS	<i>Content Management System</i> Sistema de Gestión de Contenidos
CU	<i>Use Case</i> Caso de Uso
ERS	<i>Software Requirements Specification</i> Especificación de Requerimientos de <i>Software</i>
FOSS	<i>Free and Open-Source Software</i> <i>Software</i> Libre y de Código Abierto, sinónimo de <i>software</i> libre
FURPS+	<i>Functional and non-functional requirements</i> Requisitos funcionales y no funcionales
ISO	<i>International Organization for Standardization</i> Organización Internacional de Normalización
MVC	<i>Model View Controller</i> Modelo Vista Controlador
POO	<i>Object-oriented programming</i> Programación Orientada a Objetos
UML	<i>Unified Modeling Language</i> Lenguaje Unificado de Modelado
UNMSM	Universidad Nacional Mayor de San Marcos
UP	<i>Unified Process</i> Proceso Unificado
RUP	<i>Rational Unified Process</i> Proceso Unificado de <i>Rational</i>
SEO	<i>Search Engine Optimization</i> Posicionamiento en Buscadores
TI	<i>Information Technology</i> Tecnologías de Información

1. Capítulo I: INTRODUCCIÓN

La priorización de requisitos es una actividad importante en el desarrollo de *software* (Sher, Jawawi, & Mohammad, 2019). El proceso de implementación de los modelos de priorización de requisitos requiere que la técnica propuesta por el modelo sea soportada por las herramientas de *software* para realizar dicha priorización, es decir, que la herramienta pueda sustentar el modelo. Existen propuestas que implementan soluciones con ayuda de *software*; sin embargo, se sustentan en herramientas difíciles de configurar, con un nivel de complejidad que dificultan su uso práctico y extendido. Otra dificultad es que estas herramientas se encuentran basadas en *software* protegido por derechos de autor; es decir, tienen *copyright*. Se propone, entonces, una solución basada en *software* libre, que soporte un entorno distribuido.

El modelo de priorización de requisitos de *software* en entorno distribuido deberá poderse implementar en una herramienta de amplio uso, como son los sistemas de gestión de contenidos (*CMS*). Las herramientas *CMS* permiten ampliar sus funcionalidades por medio de *plugins*, con los cuales se implementará el modelo.

El modelo propuesto será validado con un caso de prueba con el cual se evaluará si el modelo es viable, es decir, si permite realizar la priorización de los requisitos utilizando la técnica seleccionada en una herramienta *CMS* que permita la priorización de requisitos de forma fácil y amigable al *stakeholder*, además de asegurar que los resultados obtenidos son precisos, confiables, implementables y de bajo costo.

Luego, se muestran las conclusiones, recomendaciones y trabajos futuros sobre el trabajo de investigación. Por último, se incluyen los anexos en los que se muestra la selección de la herramienta *CMS*, la selección del componente para el *CMS*, la implementación del componente que utilice la técnica de priorización seleccionada, la

lista de requisitos estándares para la evaluación del modelo y la implementación de la técnica seleccionada.

1.1 Situación Problemática

En un estudio sistemático, en un rango de 16 años (Okesola, y otros, 2019), identificaron que la priorización de requisitos ocupa el cuarto lugar como tema de investigación en ingeniería de requisitos.

Mediante la implementación de un algoritmo basado en *PageRank* de *Google*, los autores (Shao, Peng, Lai, & Wang, 2017) crearon un método para realizar la priorización de los requisitos de *software*, de acuerdo a las preferencias de los usuarios.

Numerosos autores proponen diversos modelos para la priorización de requisitos, por ejemplo, (Gambo, Ikono, Achimugu, & Iroju, 2018) diseñaron un modelo basado en *rankings* para realizar la priorización de requisitos y llevaron a cabo una evaluación empírica en un entorno real, mediante una aplicación desarrollada por ellos mismos. Comparando su método con otros existentes, al finalizar, demostraron una mayor agilidad.

(Muhammad, Mustapha, & Ibrahim, 2018) propusieron un método que permita gestionar la priorización de un número grande de requisitos, en vista a que los distintos enfoques se vuelven muy complejos cuando el número de requisitos es muy grande.

Mediante el uso de un *software* desarrollado por ellos mismos, los autores (Sapunkov & Afanasieva, 2019) efectuaron la automatización de la priorización de requisitos de *software*, realizando una estimación de las prioridades en función de las características clave de los usuarios, teniendo como resultado la optimización de los tiempos de gestión de requisitos.

Los distintos enfoques de priorización de requisitos consumen tiempo e implican un esfuerzo en la realización de la tarea (Abbas, Naila Jan, Saadatmand, & Daniel Sundmark, 2019), por tanto, se requiere de una herramienta con la cual realizar el proceso.

Otros autores proponen lenguajes de modelado visuales, pero no describen como trabajar en entornos distribuidos, o gestionar numerosos requisitos (Flório, Lencastre, Pimentel, & Araujo, 2019) y (Leshob, Hadaya, & Renard, 2019).

En una investigación, (Mowad, AbdElazim Muhammad, & Elfakharany, 2020) propusieron un enfoque que permita realizar la priorización de requisitos en un desarrollo de *software* con metodologías ágiles, usando como *software* base un *ERP open-source*. Otros autores (Rahim, Chowdhury, & Das, 2017), incluso, también propusieron una técnica de priorización de requisitos, en un entorno ágil.

1.2 Formulación del Problema

Los errores en la priorización de requisitos pueden provocar el fracaso de todo el proyecto (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). Los *stakeholders* muchas veces tienen diferentes criterios, perspectivas y enfoques sobre las características del sistema a desarrollar. Por eso, para una adecuada gestión de requisitos, se necesita identificar una óptima forma de unificar los distintos criterios, perspectivas y enfoques, con el fin de obtener una única lista de requisitos priorizados y aceptados por todos.

Un aspecto en la implementación de la priorización de requisitos es la selección de la herramienta con la cual se realizará dicha tarea. Existen pocas investigaciones que describen la implementación de las técnicas de priorización, y algunas de ellas se apoyan

en herramientas basadas en *software* colaborativo (como *wikis* o foros); otras realizan toda la implementación de la solución, el cual es un proceso muy complejo y requiere muchos recursos.

También existen herramientas que permiten realizar las tareas de gestión de requisitos, incluyendo la priorización. Sin embargo, la mayoría de ellas se requiere el pago de licencia, por tanto, veces no son accesibles por su costo (Pressman & Maxim, 2019). Por ello, se evaluará el uso de *software* de fuente abierta y licencia libre.

1.2.1 *Problema General*

Se propone como pregunta de la investigación:

- *PG: ¿De qué manera un modelo de priorización de requisitos implementado en una herramienta colaborativa CMS contribuye en la toma de decisiones de los stakeholders?*

1.2.1 *Problemas Específicos*

- *PE1: ¿Cómo estaría diseñado un modelo de priorización de requisitos para que pueda ser implementado con una buena “facilidad de uso” en una herramienta colaborativa CMS?*
- *PE2: ¿De qué manera los stakeholders pueden realizar la priorización de una lista de requisitos mediante una herramienta CMS?*
- *PE3: ¿De qué manera se implementaría la priorización de requisitos con un enfoque de bajos costos en recursos?*

1.3 Motivación

La importancia de la priorización de requisitos se puede apreciar en la cantidad de investigaciones existentes a lo largo de los años (Anand & Dinakaran, 2018).

Las implementaciones actuales de priorización de requisitos, por ejemplo, mediante algoritmos basados en *PageRank* de *Google* (Shao, Peng, Lai, & Wang, 2017) y (Abbas, Naila Jan, Saadatmand, & Daniel Sundmark, 2019), o mediante el desarrollo un sistema funcional (Achimugu, Selamat, & Ibrahim, ReproTizer: A Fully Implemented Software Requirements Prioritization Tool, 2016), implican la necesidad de disponer de bastantes recursos para su desarrollo, lo cual dificulta su adopción por equipos pequeños, o con escasos recursos disponibles.

La priorización de requisitos es el aspecto más importante para la toma de decisiones (Ibriwesh, Ho, Chai, & Tan, 2019) y la planificación del desarrollo de software (Adhim, Hudaib, & Al-Shboul, 2019), y no solo para la ingeniería de *software*, sino también para la gestión de TI y otros ámbitos, como banca comercial (Kravchenko & Bruskin, 2017), productos y servicios (Song, 2017), análisis de relación con el cliente (Hossain, y otros, 2019) o *startups* (Melegati, Goldman, Kon, & Wang, 2019), entre otros ámbitos.

Existen numerosos estudios acerca del citado tema, tanto a nivel teórico como análisis comparativo entre las distintas técnicas, y unos pocos que se enfocan en la implementación práctica.

Por ello, se requiere un estudio que proponga un modelo basado en una técnica de priorización de requisitos que sirva de respaldo teórico y que pueda ser implementado con *software* colaborativo, en un entorno real, y que la implementación incluya medidas de seguridad.

1.4 Objetivos

1.4.1 Objetivo General

- OG: Identificar como un modelo de priorización de requisitos usando una herramienta colaborativa CMS con PE1: *¿Cómo estaría diseñado un modelo de priorización de requisitos para que pueda ser implementado con una buena “facilidad de uso” en una herramienta colaborativa CMS?*

1.4.2 Objetivos Específicos

- OE1: Diseñar un modelo de priorización de requisitos que pueda ser implementado con una buena “facilidad de uso” en una herramienta colaborativa.
- OE2: “Implementar” un modelo de priorización de requisitos con el cual los stakeholders puedan realizar la priorización de una lista de requisitos mediante una herramienta CMS de código libre y usando redes sociales.
- OE3: Implementar un modelo de priorización de requisitos con un enfoque de bajos “costos en recursos”.

1.5 Justificación de la Investigación

1.5.1 Justificación Teórica

La obtención de una lista de requisitos priorizados a partir un conjunto de ellos es una importante actividad en el desarrollo de *software*.

Para las organizaciones, el riesgo es no alcanzar las expectativas de los

stakeholders (Barbosa, Pinheiro, Silveira, & Filho, 2019), se debe considerar que la opinión de los mismos varía para cada requisito (Adhim, Hudaib, & Al-Shboul, 2019). Por ello, es posible reducir el riesgo a través de una adecuada priorización de requisitos. El desarrollo de *software* de calidad dependerá de la adecuada selección de los requisitos y un adecuado ordenamiento basado de los aspectos prioritarios (Sher F. , Jawawi, Mohamad, & Muhammad, 2014).

Los investigadores han propuesto diversas de las técnicas de priorización, y no existe una técnica única que pueda utilizarse para todos los tipos de proyectos (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

Con la implementación de la solución propuesta se podrá disponer de un modelo funcional y eficaz, que permita realizar la priorización de requisitos basado en el marco teórico sin incurrir en sobrecostos para realizar el proceso.

1.5.2 *Justificación Práctica*

Las propuestas de modelos de priorización de requisitos llegan hasta la validación teórica, basados en la bibliografía, la lógica, el análisis y la argumentación.

Para el modelo propuesto en el presente trabajo de investigación se busca llegar hasta el desarrollo de un modelo funcional; es decir, uno cuya implementación y validación sean viables.

La mayoría de los proyectos incluyen una gran cantidad de requisitos de software que deben priorizarse de acuerdo con los recursos limitados en términos de tiempo, presupuesto y satisfacción de los usuarios, el objetivo principal en el desarrollo de *software* (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

Los softwares colaborativos son tipos de *software* son muy conocidos y tienen amplia difusión en la actualidad, haciendo que su uso sea una alternativa válida y viable. Ejemplo de *software* colaborativo son las herramientas de gestión de contenidos – *CMS* (del inglés *Content Management System*).

Existen diferentes herramientas de gestión de contenidos, entre ellos las *wikis*, los foros y los *blogs*. Hay investigaciones que utilizan estas herramientas como parte de su propuesta de solución; sin embargo, los *CMS* basados en *blog* son de uso más extendido (BuiltWith® Pty Ltd, 2019). Por ello como parte del modelo de solución propuesto, se incluye un *CMS*¹ basado en blog.

Los trabajos de investigación estudiados que proponen herramientas colaborativas como parte de una solución en las diferentes etapas de la ingeniería de requisitos, no incluyen bases teóricas, ni método o técnica que las sustente.

1.6 Supuestos

La propuesta se diseñará considerando la aceptación por parte de los *stakeholders* de la lista de requisitos entregada.

Como *stakeholders* se hace referencia a los trabajadores de la organización que son usuarios del *software*.

Se da por aceptado que los *stakeholders* aceptan la técnica de priorización de requisitos utilizado en el modelo propuesto.

¹ *CMS: Content Management System*, Sistema de Gestión de Contenidos.

Se asume la aceptación de los *stakeholders* de la herramienta de priorización de requisitos propuesta.

Se considera que los *stakeholders* participantes se encuentran distribuidos geográficamente durante el proceso de priorización de requisitos.

No se consideran condiciones de presupuesto y limitaciones en los tipos de las tecnologías a adoptar, ni en el tamaño de la empresa u organización.

1.7 Organización de la Tesis

La presente tesis está organizada en siete (07) capítulos.

En el Capítulo I se efectúa la Introducción a tema la tesis propuesta, incluyendo la situación problemática, formulación del problema, la motivación, los objetivos de la tesis (general y específicos), la justificación de la investigación, los supuestos y la organización de la tesis.

En el Capítulo II se presenta el Marco Teórico, los antecedentes de investigación, donde se describe brevemente las actuales investigaciones, luego, las bases teóricas, los conceptos y tecnologías asociadas a los requisitos, *stakeholders*, ingeniería de requisitos, proceso de elicitación, el modelo de requisitos, la elicitación de requisitos, la documentación de los requisitos, la colaboración, la priorización de requisitos y el *software* de código abierto.

En el Capítulo III se presenta la Metodología utilizada en la presente investigación. Se muestra el tipo y diseño de la investigación, la hipótesis, las variables, indicadores, operacionalización de variables, la matriz de consistencia y la metodología.

En el Capítulo IV se detalla el Estado del Arte, referente a la priorización de requisitos de *software*, técnicas de priorización de requisitos, las herramientas de *software*, la priorización de requisitos en entornos distribuidos, la priorización de requisitos en entornos distribuidos, la priorización de requisitos con voto acumulativo y los sistemas de gestión de contenidos.

En el Capítulo V se describe el aporte de la presente investigación, en cada uno de sus elementos y un ejemplo de implementación.

El Capítulo VI presentan los resultados y discusión, para el caso de estudio. Se define el tipo de aporte, se presenta el modelo propuesto, la implementación del modelo propuesto, el caso de estudio, que se divide en: selección del caso de estudio, la planificación del experimento, la realización del experimento y la evaluación de los resultados.

El Capítulo VII es de las conclusiones y recomendaciones finales del proyecto, así como posibles trabajos futuros o temas de investigación surgidos a partir del presente estudio.

Al final, se incluye la sección Anexos. El Anexo A, muestra la selección de la técnica de priorización. El Anexo B, que muestra la selección de la herramienta *CMS*. En el Anexo C, se presenta la selección del *plugin* para realizar la votación de los requisitos propuestos. El Anexo D, presenta la implementación del *plugin* para realizar el cálculo del voto acumulativo. El Anexo E, incluye la lista de requisitos para la evaluación del modelo propuesto. En el Anexo G se presenta la implementación de la técnica Voto Acumulativo – VA.

2. Capítulo II: MARCO TEÓRICO

2.1 Marco Filosófico o Epistemológico de la Investigación

El desarrollo de soluciones se encuentra enmarcado dentro de la filosofía, en la filosofía de la técnica y tecnología (Martín Jiménez, 2018), según las investigaciones del autor, en uno de los primeros usos de los términos relacionados se considera el uso de lenguaje técnico es un lenguaje especializado.

Los problemas relacionados a las investigaciones tecnológicas tienen su origen en las necesidades prácticas (Lavado, 2020). Basado en la taxonomía propuesta por Mario Bunge, señala que la informática y la computación pertenecen a la taxonomía de la ingeniería del conocimiento. La evaluación de los productos tecnológicos no depende de criterios de verdad, sino de eficiencia y confiabilidad. En esa línea, la presente investigación estará orientada a la mejora de una condición existente, mediante un modelo, es decir orientado a eficiencia.

La técnica y tecnología, se encuentran enmarcados dentro de la Teoría del Conocimiento (Martín Jiménez, 2018), por tanto, de la Epistemología (Suárez Ardura, 2017).

La epistemología de la ingeniería está enmarcada en cuatro dimensiones: Ciencias sociales, ciencias básicas, diseño y la práctica (Valdivia Caballero, 2021), la pregunta metodológica tendrá como respuesta un nuevo conocimiento, construido desde la ingeniería. Ingeniero proviene de la palabra ingenio, y este, de otra palabra del latín (Martín Jiménez, 2018).

La epistemología de la ingeniería de sistemas (Escobar Terán, Calderón Serrano, Cevallos Romero, & Escobar Terán, 2017), o ciencias computacionales como otros autores lo describen, se encuentran estandarizados por la *ACM (Association for Computing Machinery)*, incluyendo las carreras y proponiendo un syllabus estandarizado. El rápido avance de las tecnologías, hacen que las distintas disciplinas también evolucionen.

2.2 Antecedentes de investigación

Se identifica en la bibliografía revisada numerosas tesis y artículos de investigación de fuentes de prestigio, que abordan el tema de la gestión de requisitos, en cada uno de sus pasos.

2.2.1 Técnicas de Priorización de Requisitos y Diferentes Aspectos de la Priorización

(Sher F. , Jawawi, Mohamad, & Muhammad, 2014) realizaron una revisión sistemática de la literatura para reconocer la importancia de una correcta selección de los requisitos y una adecuada priorización basada en aspectos de criterios de calidad, seleccionando aquellas investigaciones que evalúan una de las nueve principales técnicas.

2.2.2 Comparación de Métodos de Priorización de Requisitos en la Industria: Un estudio de la Efectividad de los Métodos de Ranking, Árbol de Búsqueda Binaria y la Matriz Wiegers

También está el trabajo de (Kyosev, 2014), en su tesis de maestría de Negocios Informáticos comparó los tres métodos de priorización de requisitos, enfocándose en un entorno académico. Identifica a los requisitos de *software* como los más importantes.

(Kyosev, 2014), quien cita al libro de los autores (Wohlin, y otros, 2012) ‘Experimentación en Ingeniería de Software’, determinó que en el alcance del experimento se establece mediante la definición de sus objetivos. Con el fin de obtener mejores y más generales resultados, recomienda ejecutar el experimento en un proyecto grande, con un *software* real y un *staff* de profesionales, aunque también se puede realizar con practicantes.

2.2.3 Usando Wikis como Estrategia Colaborativa para Apoyar la Elicitación de Requisitos de Software

Sobre el uso de *wikis* como *software* colaborativo, (Sevilla, Zapata, Torres, & Collazos, 2014), afirmaron que «Las *wikis* son un buen camino para coordinar contenido [...]». Asimismo, de acuerdo a los mismos autores, es mayor la tendencia a que los equipos de desarrollo de *software* estén integrados por personas que trabajan distribuidas físicamente.

2.2.4 Un diseño experimental para comparar las Técnicas de priorización de requisitos de software

Para la validación de la presente tesis, se realizará un experimento que determine la adopción o no de dicho método o tecnología en un entorno de trabajo (Mustafa & Zainuddin, 2014).

2.2.5 Usando redes sociales populares para apoyar la elicitación, la priorización y negociación de requisitos

(Seyff, Todoran, Caluser, & Singer, 2015) propusieron el uso de software de redes sociales como herramientas de apoyo en la ingeniería de requisitos, permitiendo a los usuarios una participación más activa. De acuerdo a los autores, un ejemplo sería *SoftWiki*.

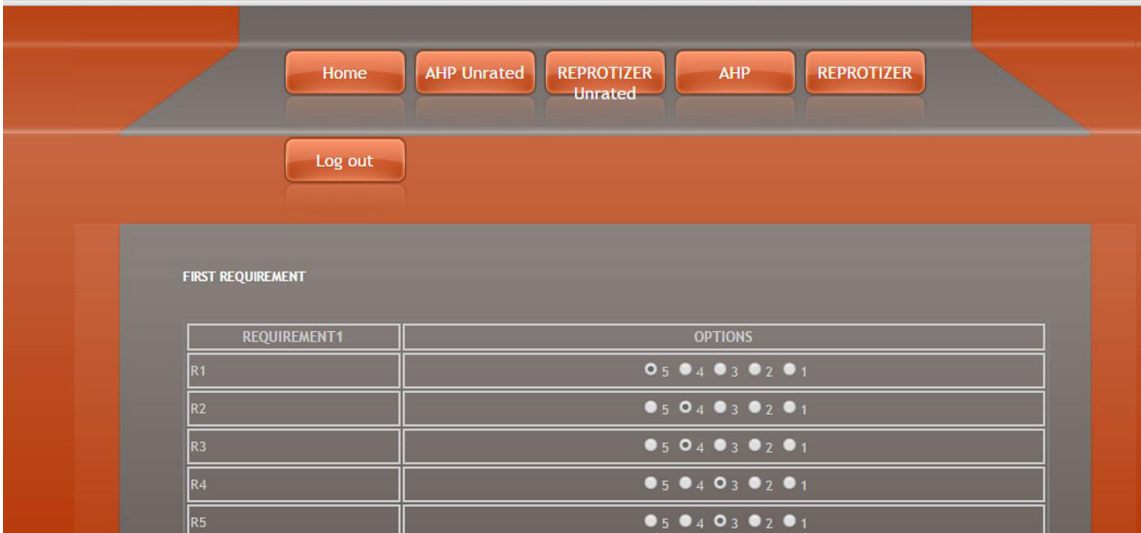
2.2.6 *ReproTizer: Una Herramienta de Priorización de Requisitos de Software Completamente Implementada*

Es una solución integral propia. El sistema implementado es funcional y se encuentra alojado en un servidor con dominio público, es decir es accesible a través de la web, como se puede ver en la Figura 1 y Figura 2.

Figura 1

Interfaz de una herramienta de priorización

www.pachimugu.com/RP/EP.aspx?event=Event5520152575529



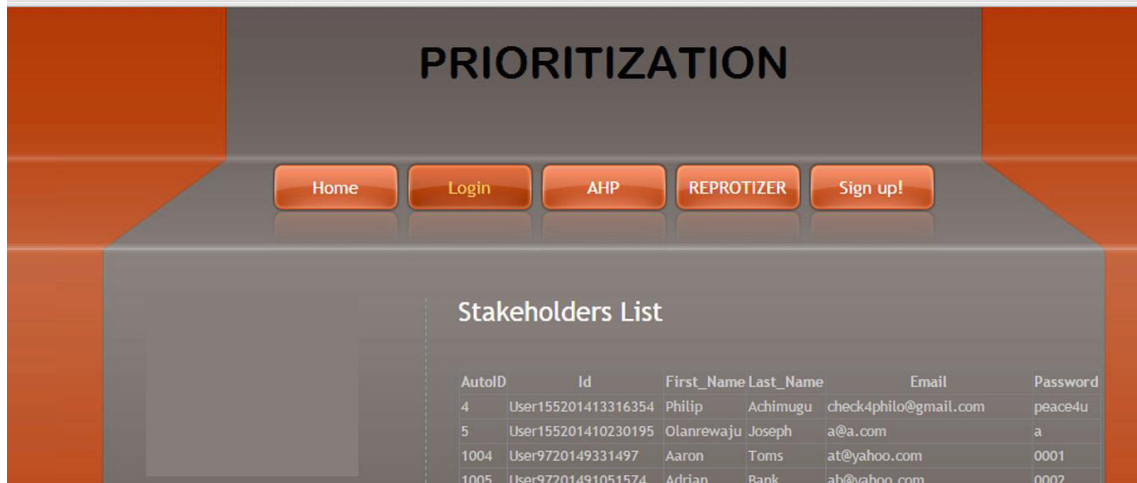
REQUIREMENT1	OPTIONS
R1	☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1
R2	☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1
R3	☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1
R4	☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1
R5	☐ 5 ☐ 4 ☐ 3 ☐ 2 ☐ 1

Nota. Permite a los stakeholders realizar la priorización, (Achimugu, Selamat, & Ibrahim, 2016).

Figura 2

Listado de stakeholders

rw.pachimugu.com/Stakeholders.aspx



AutoID	Id	First_Name	Last_Name	Email	Password
4	User155201413316354	Philip	Achimugu	check4philo@gmail.com	peace4u
5	User155201410230195	Olanrewaju	Joseph	a@a.com	a
1004	User9720149331497	Aaron	Toms	at@yahoo.com	0001
1005	User97201491051574	Adrian	Bank	ab@yahoo.com	0002

Nota. La aplicación muestragra la relación de los stakeholders, (Achimugu, Selamat, & Ibrahim, 2016)

2.2.7 MBRP: Un modelo de priorización de requisitos basado en el algoritmo Pagerank

En su propuesta los autores (Abbas, Naila Jan, Saadatmand, & Daniel Sundmark, 2019) describen la implementación del algoritmo *Pagerank*, sin embargo, una descripción del mismo no está disponible, y solo se puede aproximar su diseño y por tanto los resultados.

Un aporte positivo de la investigación es la identificación y valoración de las ventajas de automatizar el proceso de priorización, criterio que muchas veces se utiliza para valorar los distintos métodos y que si es significativo cuando es el proceso manual.

2.2.8 iStar-p: Un lenguaje de modelado para la priorización de requisitos

Es su investigación, los autores (Flório, Lencastre, Pimentel, & Araujo, 2019) proponen realizar la priorización de requisitos a través de un lenguaje de modelado, el cual incluirá distintos elementos, como íconos para información de los *stakeholders*,

criterios de priorización, valores asignados por los stakeholders. Este enfoque ayuda a visualizar el resultado de la priorización, sin embargo, cuando el número de requisitos es muy grande, se hace muy complejo el enfoque visual, además la tarea de priorización sigue siendo manual en este enfoque, lo que lo hace un proceso tedioso.

2.2.9 Priorización de requisitos de software con el lenguaje de requisitos orientado a objetivos

Los autores (Leshob, Hadaya, & Renard, 2019) proponen el lenguaje de dominio *GRL* o Lenguaje de Requisitos Orientado a Objetivos para la priorización de requisitos, incluyendo la notación y las reglas que permitan realizar la priorización de forma visual. Ofrece como beneficio el enlazar de forma directa los objetivos del negocio y los requisitos. Eligen el lenguaje *GRL* por ser sencillo de comprender para los *stakeholders* y permite describir el proceso mediante pocas instrucciones.

2.2.10 Uso de redes sociales para identificar los requisitos iniciales del usuario para pequeñas empresas de software

En el artículo de (Ali & Hong, 2019) los autores diseñan un modelo basado en redes sociales, motivados por el hecho que la mayoría de empresas de *software* son pequeñas. Consideran importantes el uso de redes sociales para complementar los paradigmas de tradicionales de la ingeniería de software, por ejemplo, medio de desarrollo de aplicaciones para *smartwatch* o *smartphone*. Para su caso de estudio usan *Facebook*, con un grupo de discusión y un moderador. Utilizaron el botón me gusta para la priorización de los requisitos. Al ser una red social, los aspectos de seguridad, acceso al contenido y administración de cuentas se encuentran limitadas por las funcionalidades que ofrece esta. Las redes sociales son entornos distribuidos.

2.2.11 Una herramienta para el apoyo al proceso de priorización de requisitos en un entorno de desarrollo de software ágil

Los autores (Mowad, AbdElazim Muhammad, & Elfakharany, 2020) presentan el desarrollo de una herramienta para la priorización de requisitos en un entorno de desarrollo ágil, usando como *software* base el *ERP open-source Odoo, Python* y como base de datos *PostgreSql*. Implementaron un caso de estudio en un entorno real, en una empresa.

2.2.12 SRPTackle: una técnica de priorización de requisitos de software semiautomatizada

Los autores (Hujainah, Binti Abu Bakar, Nasser, Al-haimi, & Zamli, 2021) desarrollaron una herramienta denominada SRPTackle-Tool, desarrollada para automatizar la priorización en la técnica desarrollada por ellos: SRPTackle. La herramienta está desarrollada en el lenguaje *C#*, con el IDE *.NET, HTML* para la interfaz gráfica de usuario y *Microsoft SQL Server* como motor de base de datos.

2.3 Bases Teóricas

2.3.1 Requisito

Los requisitos de un sistema son las descripciones de los servicios que un sistema debe proporcionar y sus limitaciones de funcionamiento, reflejan las necesidades de los *stakeholders* (Sommerville, 2016). En las primeras fases del desarrollo, los requisitos pueden afectar la viabilidad del proyecto (Bennaceur, Than Tun, Yu, & Nuseibeh, 2019). Los requisitos pueden variar desde declaraciones abstractas de alto nivel, simples bocetos hasta especificaciones formales (matemáticamente rigurosas) (Laplane, 2018).

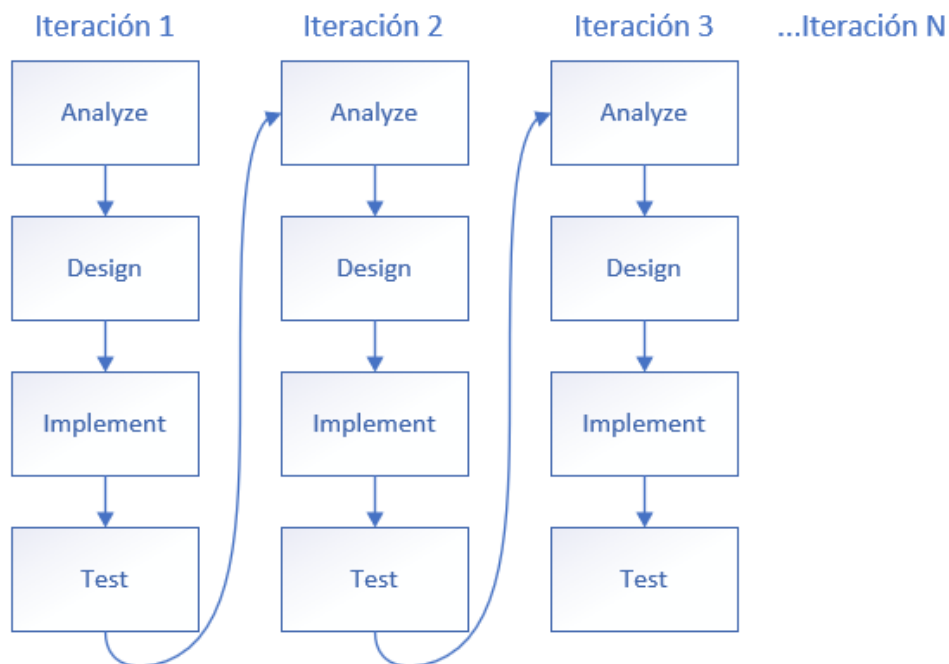
En su libro sobre Ingeniería de *software* (Pressman & Maxim, 2019), señaló que, si el diseño y desarrollo el producto no resolvía el problema especificado, este no satisfaría las necesidades de nadie. Cada proyecto es único, dos sistemas similares, no tendrán idénticos requisitos (Bahill & Madni, 2017).

Un error durante la adquisición de requisitos puede tener consecuencias en el *software* desarrollado y provocar la no aceptación del producto por parte de los *stakeholders*; incluso si se terminó en desarrollar en los plazos establecidos.

Los requisitos se organizan dentro de un modelo iterativo. La idea básica es desarrollar un sistema dentro de ciclos repetidos (iteraciones) y pequeños intervalos de tiempo. En este modelo, el software no es desarrollado en un solo tiempo, ver Figura 3. El primer paso es el análisis, en el cual se obtienen los requisitos (Yaseen, Ibrahim, & Mustapha, 2019).

Figura 3

Modelo iterativo



Nota. Muestra el flujo del modelo iterativo, (Yaseen, Ibrahim, & Mustapha, 2019).

2.3.1.1 Niveles de Requisitos

Hace referencia a la segmentación de los requisitos en dos agrupaciones básicas, requisitos de usuario, para designar los requisitos de alto nivel, y requisitos del sistema a aquellos que describen de forma detallada de lo que el sistema debe realizar.

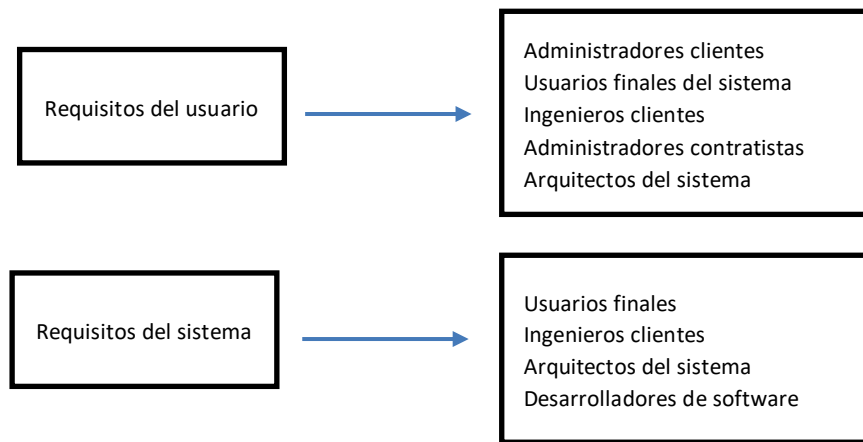
De acuerdo con (Sommerville, Software Engineering, 2016), los requisitos del usuario y del sistema se pueden definir como se muestra a continuación:

- Requisitos del usuario: Son enunciados, descritos en lenguaje natural y representados en diagramas, de las funcionalidades que el sistema debe realizar y de las restricciones que debe cumplir. Describen los requisitos funcionales y no funcionales de forma comprensible para los *stakeholders* sin mayores habilidades técnicas. Debe especificar el comportamiento externo del sistema, sin definir atributos de diseño. Sus requisitos están definidos en forma de formularios o módulos (Yaseen, Ibrahim, & Mustapha, 2019).
- Requisitos del sistema: Describen de forma detallada las funcionalidades, atributos y características del sistema, por tanto, definen de manera precisa la solución a implementar. Los requisitos del sistema describen y explican en forma detallada su implementación. Pueden ser utilizados como parte del contrato donde definen una especificación completa y consistente del sistema entero.

Ambos niveles de especificación del sistema transmiten información sobre el sistema a diferentes grupos de *stakeholders*, como se puede ver en la Figura 4.

Figura 4

Requisitos de usuario y requisitos del sistema



Nota. Quienes gestionan los requisitos de usuario y los requisitos del sistema, (**Sommerville, Software Engineering, 2016**)

2.3.1.2 Clasificación de Requisitos

A menudo, los requisitos de sistemas *software* se clasifican en funcionales y no funcionales, o como requisitos del dominio (Sommerville, Software Engineering, 2016):

- Requisitos Funcionales: Característica que debe ofrecer el *software*. Los requisitos funcionales también se describen con los CU. De manera análoga, pueden describir de manera explícita lo que el sistema no debe hacer. Constituyen el núcleo de los requisitos de los *stakeholders* (Yaseen, Ibrahim, & Mustapha, 2019).
- Requisitos No funcionales: Son restricciones o limitaciones, incluyen, por ejemplo, restricciones de tiempo, de documentación y cumplimiento de estándares. Comúnmente se aplican al sistema en su totalidad.
- Los requisitos no funcionales también son denominados requisitos de calidad, tales como: puntualidad, rendimiento, capacidad de respuesta, seguridad, confiabilidad y facilidad de uso. También llamados requisitos de soporte (Yaseen, Ibrahim, & Mustapha, 2019). Los requisitos no funcionales son igual de

importantes que los funcionales y deben ser evaluados en las etapas iniciales del proyecto (Khush Bakht, Inayat, & Allah Bukhsh, 2019).

2.3.2 Stakeholders

Los *stakeholders* son aquellos que se benefician en forma directa o indirecta del software que se está desarrollando (Pressman & Maxim, 2019) y (Sommerville, Software Engineering, 2016). Una adecuada es importante para la continuidad del proyecto (Barbosa, Pinheiro, Silveira, & Filho, 2019).

Los *stakeholders* tienen diferentes criterios sobre que es un proyecto de *software* exitoso, estas diferencias pueden haber sido influenciadas por sus conocimientos, intereses y rol en el proyecto (Hujainah, Binti Abu Bakar, & Abdullateef Abdulgabber, 2019).

Los *stakeholders* participan en el desarrollo del sistema, durante la priorización de los requisitos de la manera correcta de acuerdo a su importancia, por lo tanto, los requerimientos se pueden ordenar en ejecución (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

2.3.2.1 Gestionar a los Stakeholders

Es muy importante recoger la información relevante de los *stakeholders*: nombre, cargo / posición, información de contacto, los requisitos / expectativas del proyecto, y los grupos de *stakeholders* de tipo interno o externo.

Las reuniones con los grupos de *stakeholders* (de ser posible a nivel personal), deben ser bien planificadas y preparadas. La introducción a la entrevista y las preguntas en sí deben estar preparadas de antemano.

2.3.2.2 **Identificación de los Stakeholders**

Uno de los primeros pasos es la identificación de los *stakeholders* (Bahill & Madni, 2017).

Los *stakeholders* tienen un enfoque diferente sobre cómo debe ser el sistema. En primera instancia, el ingeniero de requisitos o quien cumpla sus funciones elaborará una lista de interesados que lo ayudaran en la captura de requisitos. Esta lista se irá completando conforme conozca mejor a los *stakeholders* (Pressman & Maxim, 2019).

Como los *stakeholders* son seres humanos, traen sus valores y preferencias al proyecto. Además, tienen diferentes experiencias, lo que se refleja en sus conocimientos específicos.

Los usuarios finales necesitan funcionalidades que les permitan realizar sus tareas y sean sencillas de aprender.

2.3.3 **Ingeniería de Requisitos**

La ingeniería de requisitos se encarga de facilitar el proceso de comprender las necesidades de los *stakeholders*, razonar las necesidades, valorar la viabilidad del proyecto, proponer una solución válida para el proyecto, definiendo una la solución, gestionando los requisitos durante todo el proceso de desarrollo del sistema (Pressman & Maxim, 2019).

La ingeniería de requisitos no consiste solo en registra las necesidades de los distintos *stakeholders*, sino que esta cubre el ciclo de vida del sistema. Para que un *software* reciba la aceptación de los *stakeholders*, los requisitos deben ser registrados, analizados y priorizados (Achimugu, Selamat, Ibrahim, & Naz'ri Mahrin, 2014). Cubre

también requisitos de negocio y de TI², en *startups*, aplicando un modelo ágil (Melegati, Goldman, Kon, & Wang, 2019). Las actividades de ingeniería de requisitos son complejas y se entrelazan con otros procesos de desarrollo y gestión de la organización (Heikkil'a, Paasivaara, Lasssenius, Damian, & Engblom, 2017).

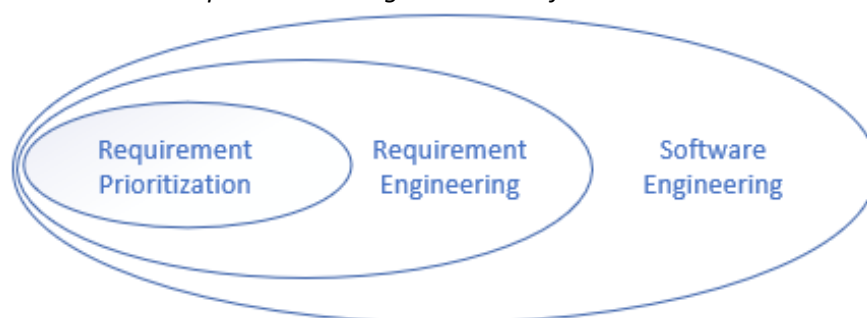
Las investigaciones en ingeniería de requisitos, en su enfoque tradicional como en el ágil ha sido estudiado a lo largo de los años (Okesola, y otros, 2019).

Además, los *stakeholders* tienen opiniones diversas en cuanto a la prioridad de cada requisito. En otras palabras, el acuerdo de las partes interesadas sobre la prioridad de los requisitos debe tenerse en cuenta para el proceso de priorización de requisitos (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

La priorización de requisitos forma parte de la ingeniería de requisitos, que es uno de los campos de estudio de la ingeniería de *software* (Muzamal Liaqat, Azam, Adeel Ahmed, & Mehboob, 2016), ver Figura 5.

Figura 5

Rol de la Priorización de Requisitos en la Ingeniería de Software



Nota. Muestra como está ordenado la priorización de requisitos dentro de la ingeniería de software, (Muzamal Liaqat, Azam, Adeel Ahmed, & Mehboob, 2016)

² Tecnología de Información.

2.3.3.1 Necesidad y Requisito

Se debe distinguir entre necesidad y requisito. 'Necesidad' es la percepción que tienen el cliente, los usuarios y otros *stakeholders* sobre el sistema. 'Requerimiento' es la percepción del sistema desde la perspectiva del *project manager*, el analista y diseñador del sistema.

La priorización de requisitos ayuda a implementar las necesidades de los *stakeholders* en mediante un proyecto de *software* (AbdElazim, Moawad, & Elfakharany, 2020).

2.3.4 Gestión de Requisitos

La gestión de requisitos son todas aquellas actividades que le permiten al equipo del proyecto a identificar, controlar y realizar una trazabilidad de los requisitos y cambios durante el desarrollo del proyecto. Gestionar requisitos involucra actividades como control de cambios y la trazabilidad (Pressman & Maxim, 2019).

La gestión de requisitos comienza con la identificación de los mismos. A cada requisito se le asigna un único identificador (Pressman & Maxim, 2019).

En proyectos de desarrollo pequeños, la implementación de los procesos de desarrollo de la ingeniería de requisitos es fácil de incorporar, sin embargo, en proyectos de desarrollo grandes, la gestión de los requisitos resulta un proceso compleja y difícil (Abdullah Awais, 2016).

En organizaciones grandes, la comunicación informal y simples hojas de anotaciones no son suficientes para la gestión de requisitos (Heikkil'a, Paasivaara, Lasssenius, Damian, & Engblom, 2017). En las organizaciones espera un número de

requisitos de hasta 2,000 (Barbosa, Pinheiro, Silveira, & Filho, 2019) o un documento de 800 páginas (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). En pequeñas empresas, se considera alto un número de 150 requisitos (Ali & Hong, 2019).

2.3.4.1 Fases en la Gestión de Requisitos

Las fases de la gestión de requisitos son la elicitación, el análisis, la especificación y la validación. Otra denominación es 'proceso de ingeniería de requisitos' (Tong, Zhuang, Guo, & Ma, 2014).

2.3.4.1.1 Elicitación

Consiste en identificar las necesidades de los stakeholders y que buscan (Laplante, 2018).

2.3.4.1.2 Análisis

El análisis tiene como propósito evaluar, comprender y modelizar los requisitos; luego, se evalúa en términos de calidad, correctitud, claridad y consistencia. Como consecuencia de ello, se puede determinar su nivel de importancia, y cuáles podrían ser descartadas.

2.3.4.1.3 Especificación

La especificación registra y documenta los requisitos de tal manera que puedan ser utilizadas por los *stakeholders*, y en particular por los diseñadores quienes pueden diseñar y construir el sistema.

La especificación es el producto final que genera la ingeniería de requisitos, y sirve como base a las actividades de ingeniería subsecuentes (Pressman & Maxim, 2019).

2.3.4.1.4 Validación

Se evalúa la calidad de los requisitos en proceso y se verifica que estén representadas las necesidades de los *stakeholders*. La validación realiza un análisis de la especificación para que los productos de trabajo cumplan con los estándares definidos previamente para el proyecto (Pressman & Maxim, 2019).

2.3.4.2 Flujo de Requisitos

Se basa en la especificación de requisitos de *software* (SRS), que contiene un modelo de requisitos y un modelo de casos de uso (CU³).

Tiene como artefactos al modelo de requisitos, el modelo de CU, el modelo del dominio y el modelo del negocio.

2.3.4.2.1 Modelo de Requisitos

El modelo de requisitos es el primero en desarrollarse y sirve de base para la elaboración de los demás durante el desarrollo de *software*. En general, a este nivel y no en otro posterior, los cambios en las funcionalidades del sistema son más fáciles de implementar, y con reduce las posibles consecuencias.

2.3.4.2.1.1 Modelo de Caso de Uso

UML es un estándar de modelado que permite la creación de diagramas estandarizados y sus especificaciones asociadas que mejoran la comunicación entre los

³ Caso de Uso.

stakeholders y aumentan su participación en el proyecto. Con *UML*, los equipos del proyecto crean, revisan y modifican diagramas y especificaciones de manera compartida. Estos diagramas pueden ser diseñados en herramientas de modelado, llamados herramientas *CASE*. Consta de catorce tipos de diagramas (Bernhard, 2017).

UML no proporciona ninguna estructura formal para la especificación de los CU. Las actividades consisten en buscar actores y CU, tanto del modelo del negocio como del modelo de requisitos. Los CU enlazan los flujos de trabajos fundamentales. *UML* está orientado a satisfacer una gran cantidad de requisitos, por lo tanto, está sobrecargado con una gran cantidad de diagramas de modelos diferentes con una semántica amplia y algunos, poco claros (Bernhard, 2017).

Los actuales sistemas tienen un diseño basado en CU (Bahill & Madni, 2017).

2.3.5 Documentación de Requisitos

La forma de documentar se define a nivel de estándares internos, y también a nivel de cada proyecto (Vigneshwar Alavandhar & Nikiforova, 2017).

El documento de requisito del producto es creado en lenguaje natural, y es comúnmente aplicado en formularios para captura de requisitos practicado por muchas organizaciones. El uso de lenguaje natural ofrece ciertas ventajas al no requerir el aprendizaje de una nueva notación. Además, puede usarse para expresar cualquier tipo de requisitos.

La mejor forma de describir requisitos es mediante el uso de lenguaje natural, en modo narrativo, a través de oraciones y frases (Bahill & Madni, 2017).

2.3.5.1 Especificación de requisitos de software – ERS

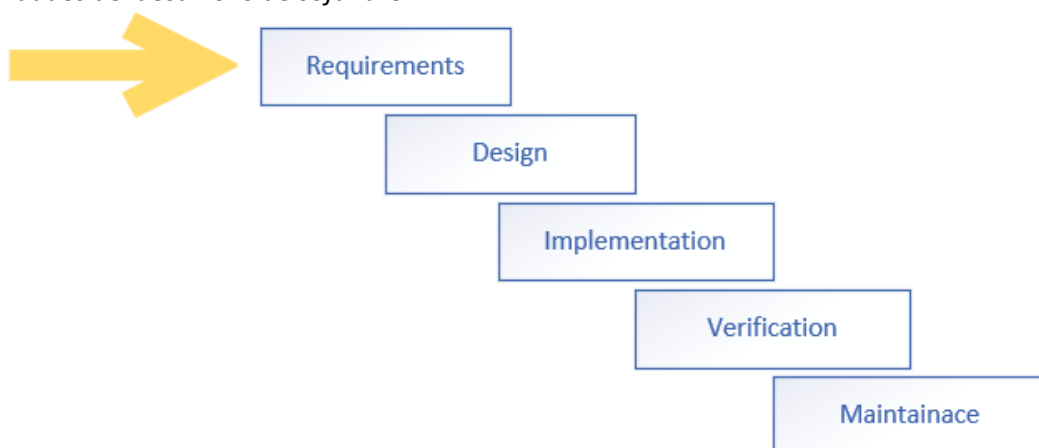
Una especificación de requisitos de *software* (ERS) es un documento en el cual se registra una descripción detallada de aquellas necesidades que deberá satisfacer el software a elaborar, previo a la construcción del *software*. Muchos utilizan plantilla estándar con los elementos a incluir, comúnmente se utiliza un lenguaje natural y diagramas (Pressman & Maxim, 2019). La calidad del *software* dependerá de la especificación de requisitos (Ibriwesh, Ho, Chai, & Tan, 2019).

2.3.6 Modelo para el Proceso de Desarrollo de Software

Abarcan todas las etapas del proceso de desarrollo de *software*. Algunas metodologías son descritas como ágiles, aquella en las que el flujo de trabajo y los modelos son considerados como documentación, con iteraciones breves y frecuentes (Fatima & Mahmood, 2019). El flujo básico que sigue cada metodología se muestra en la Figura 6.

Figura 6

Actividades del desarrollo de software



Nota. Muestra la secuencia de actividades que siguen cada una de las metodologías, (Abdullah Awais, 2016).

Entre las metodologías de estudio podemos identificar a las siguientes:

- *CMMI*.
- *ISO/IEC 15504*.
- *UP*.
- *RUP*.
- *Enterprise Unified Process* - EUP (tiene su origen en el *RUP*).
- *OpenUp* (tiene su origen en el *RUP*).
- *Microsoft Solutions Framework (MSF)*.
- Programación Extrema (*Extreme Programming - XP*).
- *Scrum*.
- NTP-ISO/IEC 12207.
- ISO/IEC 29110.
- *MoProSoft*.
- *Kanban*.
- *Prince2*.

Todos los modelos para el proceso de desarrollo de *software* referidos permiten implementar la priorización de requisitos con la técnica seleccionada.

2.3.6.1 **CMMI**

Es un proceso o modelo orientado a la mejora de la Organización para la mejora de sus procesos, con el fin desarrollar mejores productos y servicios. Es desarrollado y administrado por la Universidad Carnegie Mellon. El *CMMI (Capability Maturity Model Integración)* son un compendio de mejores prácticas que ayudan a las organizaciones a la mejora de sus procesos. Estas colecciones son desarrolladas por equipos de producto con distintas organizaciones de la industria, el gobierno y el *Software Engineering Institute (SEI)*.

De modo independiente al modelo que elija la organización, las prácticas propuestas por *CMMI* deben adecuarse a ellas en relación a sus objetivos de la organización.

El modelo *CMMI* es ampliamente usado para implementar procesos de ingeniería de *software* y aseguramiento de la calidad en todas las empresas, instituciones, organizaciones militares y comerciales (Summers, 2021), defensa y espacio, hospital y atención médica, seguros, consultoría de gestión, subcontratación (Chaudhary & Chopra, 2017). *CMMI* es un modelo de mejora que puede ser adoptado para resolver distintos problemas de performance o rendimiento en cualquier industria (Chaudhary & Chopra, 2017).

Tiene tres áreas de trabajo (Chaudhary & Chopra, 2017):

- Desarrollo de productos o servicios: *CMMI* para desarrollo (*CMMI-DEV*)
- Gestión de servicios: *CMMI* para servicios (*CMMI-SVC*)
- Adquisición de productos o servicios: *CMMI* para adquisición (*CMMI-ACQ*).

Recopila las mejores prácticas para ayudar a los miembros del equipo de software, empresas e instituciones a mejorar el desarrollo de sus procesos y productos. *CMMI* se enfoca en todas las actividades de ingeniería de software para el desarrollo de productos y servicios de software para satisfacer las necesidades de los clientes y usuarios finales (Summers, 2021).

2.3.6.2 **ISO/IEC 15504**

La norma ISO/IEC 15504, conocida como '*Software Process Improvement and Capability Determination*' (*SPICE*), define un modelo de evaluación de procesos. Se trata de un estándar internacional que permite evaluar la capacidad y madurez de los procesos de *software* de una organización, así como el mantenimiento de sistemas de

información y productos de *software*.

2.3.6.3 El Proceso Unificado – UP

El *UP* es iterativo, por tanto, no es necesario obtener la versión final para avanzar a la siguiente fase. Los distintos artefactos son modelados mediante *UML* (Valencia & Arcos, 2018).

Las fases del Proceso Unificado son, resumen de (Valencia & Arcos, 2018):

- Inicio: El desarrollador visualiza la viabilidad del proyecto y los posibles riesgos.
- Elaboración: Se ven con más detalle los requisitos y se establecen los conceptos del problema.
- Construcción: La solución al problema comienza a implementarse en un software, el Modelo de Diseño - conceptos en un entorno de software comienza a ser más importante que el Modelo de Dominio - conceptos en un entorno del mundo real.
- Transición: Se prueba el software para que se comprueben algunos problemas, fallos, cumplimiento de requisitos.

El *UP* se adapta a diversos tamaños de proyectos y equipos para desarrollar un producto de software de alta calidad. *UP* es un *framework*, requiere de un proceso de adaptación al contexto específico por parte de las organizaciones. El ciclo de vida del *UP* consta de cuatro fases (inicio, elaboración, construcción y transición) con hitos y un número de iteraciones que siguen los flujos de trabajo fundamentales: requisitos, análisis, diseño, implementación y pruebas (Alvarez & Hurtado, 2018).

UP tiene 3 características fundamentales: (i) basado en casos de uso: su progreso se mide por casos de uso completados, (ii) centrado en la arquitectura: la arquitectura da una perspectiva de la construcción del software al equipo de desarrollo, y (iii) iterativo e incremental, divide el proyecto en subproyectos más pequeños llamados iteraciones, produciendo incrementos del producto (Alvarez & Hurtado, 2018).

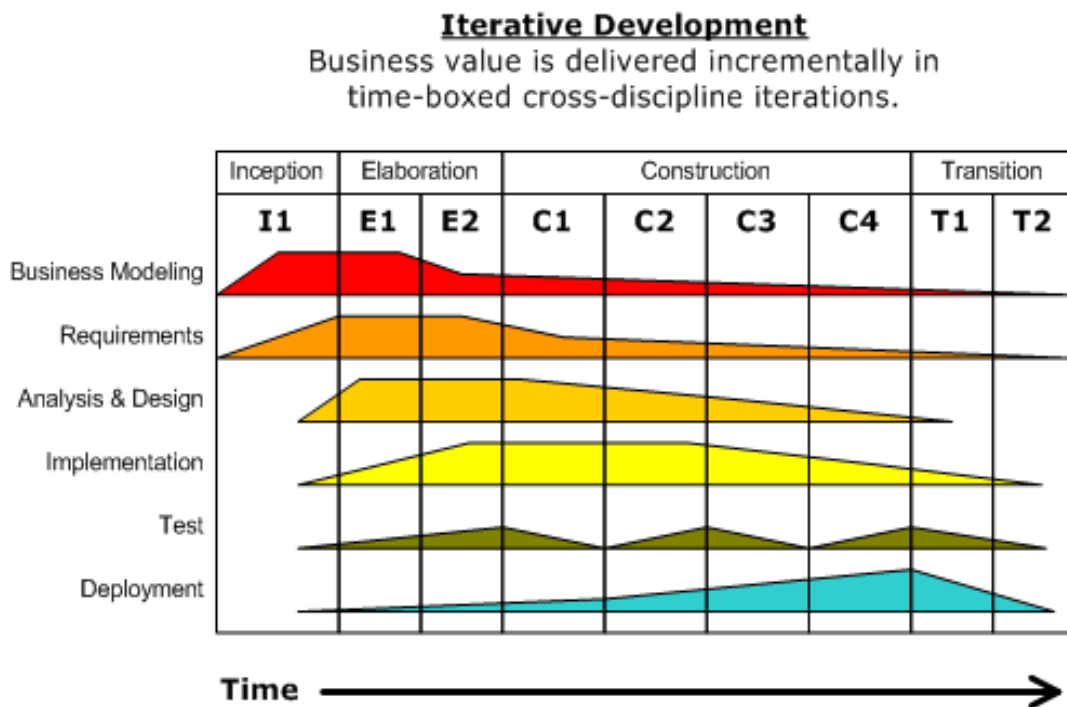
2.3.6.4 **RUP**

RUP consta de tres elementos básicos: actividades, roles y artefactos. El uso de los distintos artefactos dependerá de cada proyecto. Cada proyecto lo integra un grupo de miembros, quienes asumen uno o más roles. Cada rol participa en una o más actividades y cada actividad produce uno o más artefactos. Cada iteración cubre parte o la mayor parte de la fase de desarrollo. Cada iteración se basa en los entregables de la anterior, mejorando el producto hasta obtener el resultado final. El proceso iterativo permite a los miembros del equipo asumir múltiples roles. *RUP* puede ser adoptado por pequeñas empresas, que tienen menos de 50 personas.

En equipos pequeños, la implementación de muchos de los roles se suele omitir. Entre las desventajas de *RUP* se encuentran la falta de apoyo a la gestión de recursos humanos, la gestión de comunicaciones y gestión de contratos. Consta de cuatro fases y nueve flujos de trabajo en cada fase (Karen Tia, Nuryasin, & Maskur, 2019), ver Figura 7.

Figura 7

Ciclo de vida del RUP



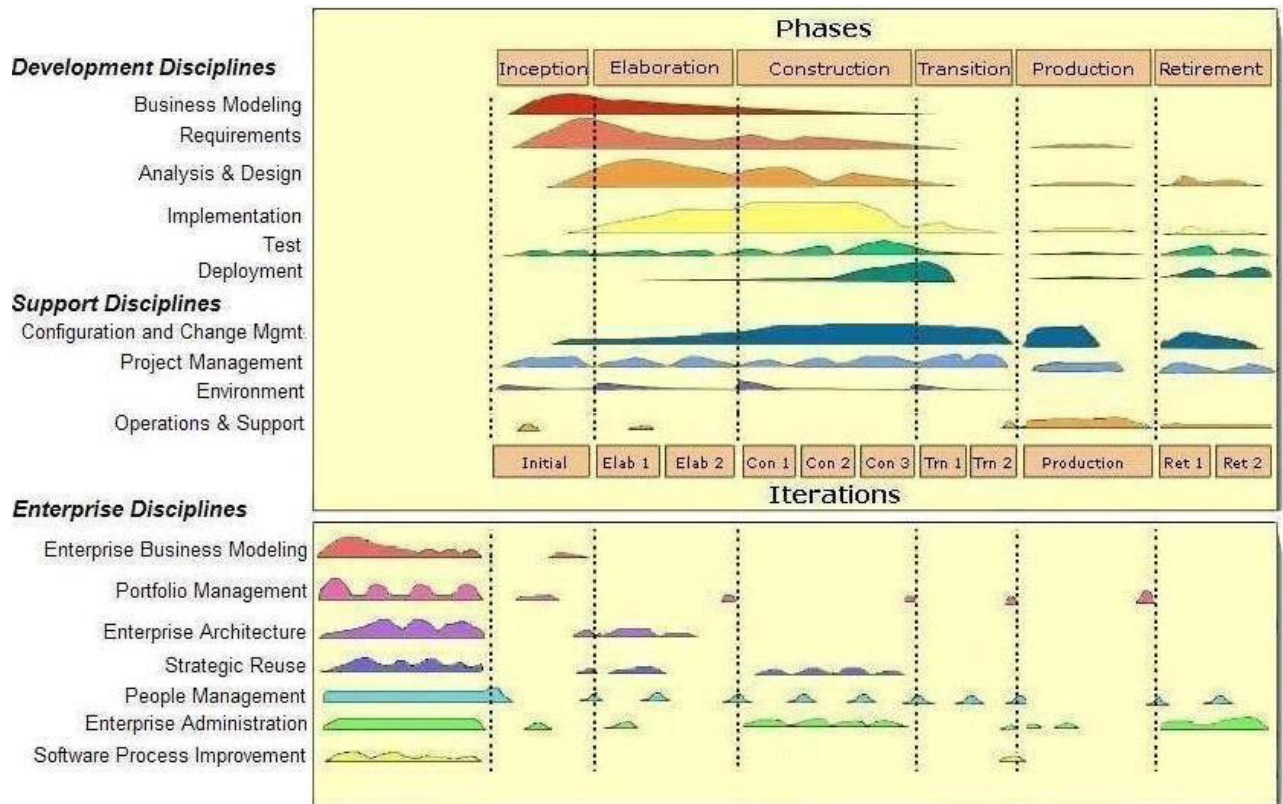
Nota. Se muestra el desarrollo iterativo e incremental, dominio público.

2.3.6.5 Enterprise Unified Process - EUP

EUP es un ciclo de desarrollo de *TI*, es una extensión del *Proceso Unificado Racional (RUP)*. No solo gestiona el ciclo de desarrollo de software, cubre además todo el ciclo de vida de *TI*, agrega la disciplina Gestión Empresarial de acuerdo a las necesidades de la arquitectura empresarial (Mubarok & Tjahjadi, 2019).

Figura 8

Enterprise Unified Process [EUP]



Nota. Se muestra las disciplinas adicionales al RUP que permite gestionar la metodología,
(AmbySoft Inc, 2015)

2.3.6.6 OpenUP

Es un proceso unificado de enfoque ágil iterativo e incremental. *OpenUP* se enfoca en centra en modo colaborativo el desarrollo de proyecto.

OpenUP es una metodología que aplica un enfoques iterativo e incremental dentro de un ciclo de vida estructurado; aplica un enfoque pragmático y ágil. El proyecto de divide en micro incrementos, a través de los cuales se desarrolla el proyecto. En cada iteración se definen cuáles serán los entregables (Alarcón & Rojas, 2018).

OpenUP se divide en cuatro procesos (o etapas):

- Comienzo: Se definen los entregables, el plan del proyecto y las iteraciones.
- Elaboración: Se define en detalle las necesidades de los *stakeholders*, se define la arquitectura y la gestión de riesgos.
- Construcción: Se desarrollan cada uno de los productos del proyecto, minimizando el costo del desarrollo.
- Transición: Puesta a punto del producto y su despliegue.

2.3.6.7 **Microsoft Solutions Framework - MSF**

MSF es un *framework* altamente escalable, por lo que es compatible con equipos de cualquier tamaño, desde pequeños, grandes hasta complejos. En proyectos grandes, la gestión se realiza a varios niveles. En proyectos complejos, se requerirá un director de proyecto o un equipo de gestión de proyectos. *MSF* cuenta con el proceso de identificación continua de riesgos en el proyecto (Vigneshwar Alavandhar & Nikiforova, 2017). Se caracteriza por su flexibilidad, permitiendo ser adaptado a diferentes tipos de proyectos. Tiene una estructura flexible que cuenta con conceptos, modelos y buenas prácticas; se centra en el modelado de los procesos, no estando vinculado a una tecnología en particular (Turner, 2015).

Microsoft Solutions Framework define un conjunto de actividades, artefactos y roles. Es un modelo de desarrollo de software con elementos bien definidos, con planificación y definición de requisitos. Es una combinación de método en cascada y espiral. Se desarrolla en base a iteraciones, pueden durar de 30 a 90 días. Es considerada una metodología tradicional (Vigneshwar Alavandhar & Nikiforova, 2017).

En *MSF*, hay seis grupos de roles y cada grupo de roles tiene una o más áreas funcionales y cada área funcional puede tener una o más responsabilidades y las

responsabilidades tienen una o más tareas. Un rol puede ser asumido por una o varias personas o depender del tamaño del proyecto, uno o más roles pueden ser asumidos por una o más persona (Vigneshwar Alavandhar & Nikiforova, 2017).

2.3.6.8 Programación Extrema – XP

La Programación Extrema - *XP* (Sommerville, Software Engineering, 2016) es un método de trabajo que se desarrolla bajo la filosofía ágil, propuesto por Kent Beck. *XP* es un método ligero de desarrollo de software (Bernhard, 2017).

Hace un mayor énfasis en la relación entre las personas que aspectos formales de la metodología. Los requisitos se agrupan y denominan escenarios o historias de usuario, y se implementan en tareas. Una de las características más distintivas es que los desarrolladores realizan la programación en parejas y las pruebas son diseñadas previamente a la escritura de código. Todas las pruebas se deben ejecutar en forma satisfactoria cuando el código nuevo se integre al sistema. Las nuevas versiones son publicadas con breves márgenes de tiempo. *XP* descarta la elaboración de documentación superflua y un desarrollo iterativo e incremental (Saleh, Maruful Huq, & Rahman, 2019). Asimismo, los clientes tienen importante participación en la especificación y definición de cada escenario. Otra característica es que promueve no hacer sobre tiempos (Sommerville, 2021). Todas sus características se describen como prácticas, ver Figura 9.

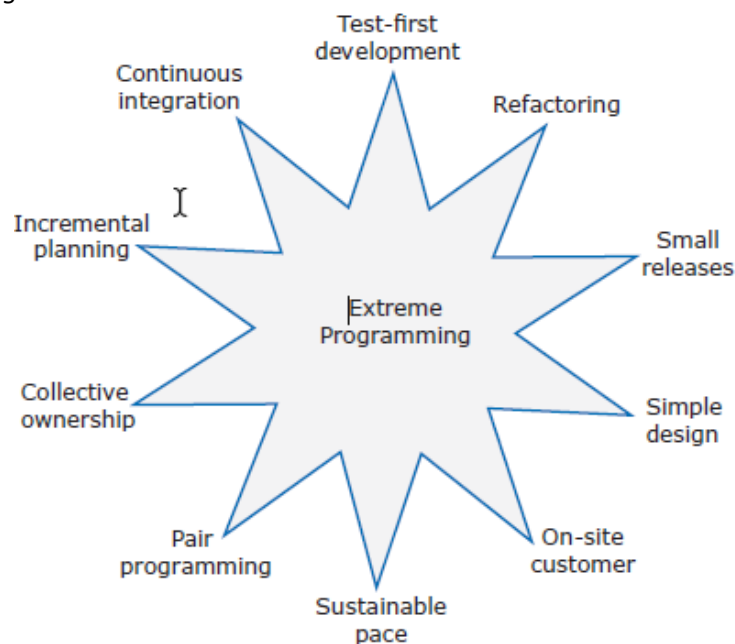
XP asume que la creación de documentación requiere mucho esfuerzo, y esta es a menudo mucho más errónea que el código en sí mismo porque normalmente no se puede analizar y probar automáticamente de manera satisfactoria. Casi no se crea documentación en los proyectos de *XP* (a excepción del código y las pruebas). Para compensar esto, consideran muy importantes los buenos comentarios en el código fuente, basados en estándares de programación y un extenso conjunto de pruebas

(Bernhard, 2017).

Es una de las metodologías ágiles más difundidas (Saleh, Maruful Huq, & Rahman, 2019).

Figura 9

Prácticas de Programación Extrema



Nota. Se encuentran descritos en el manifiesto de *XP*, (Sommerville, 2021).

2.3.6.9 Scrum

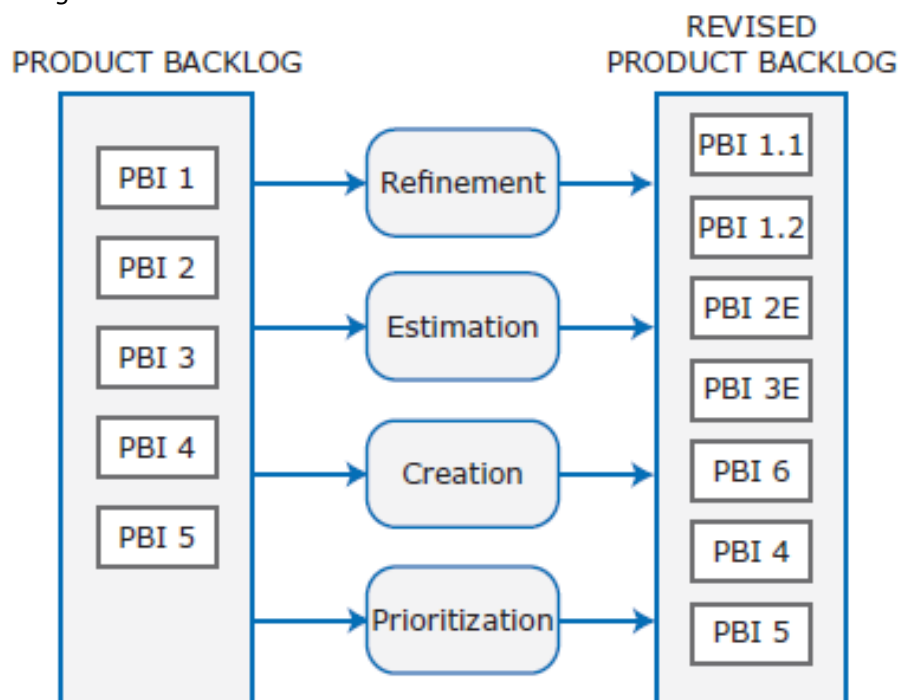
Scrum es una metodología ágil para la gestión de proyectos (Scrum Alliance, 2015). Mediante *Scrum* el proyecto se estructura en tiempos y costos planeado, en un entorno colaborativo, flexible y adaptado al cambio. Se enfoca en entregas parciales con plazos previamente definidos de tiempo, hasta obtener el producto final. Algunos de sus roles y artefactos reciben una denominación propia, así el *scrum master* cumpliría un rol análogo al jefe de proyectos. *Scrum* es un conjunto de buenas prácticas, entregando productos de alto valor (Maximini, 2018). Con *Scrum*, se trata de que los equipos mejoren la colaboración y el desempeño del equipo (Jacobson, Lawson, Ng, McMahon,

& Goedicke, 2019). Se encuentra entre las metodologías ágiles que tiene la mayor aceptación (Sommerville, 2021), (Okesola, y otros, 2019) y (Jacobson, Lawson, Ng, McMahon, & Goedicke, 2019).

La lista de necesidades en *Scrum* se denomina *Product Backlog*, y siempre debe ser priorizado y refinado, ver Figura 10.

Figura 10

Product Backlog



Nota. El *Producto Backlog* se revisa, refina, actualizan las estimaciones, desglosan requerimientos y se prioriza, (Sommerville, 2021).

Scrum consta de tres componentes (Heath & Gupta, 2021):

- El equipo *Scrum*: Son los integrantes del proyecto, personas autoorganizadas y multifuncionales que entregarán el software producido al cliente.
- Eventos *Scrum*: Eventos que ayudan a crear regularidad, brindan retroalimentación y promueven un ciclo de vida iterativo e incremental.

- Artefactos *Scrum*: Elementos producidos que representan trabajo o valor agregado y que reflejan progreso y los logros del trabajo del equipo.

2.3.6.10 NTP-ISO/IEC 12207

La Norma Técnica Peruana NTP-ISO/IEC 12207 fue diseñada por el Comité Técnico de Normalización de Ingeniería de *Software* y Sistemas de Información, y revisada y aprobada por la Comisión de Reglamentos Técnicos y Comerciales del INDECOPI, que tiene el cargo de Organismo Nacional de Normalización, tomo como base la norma ISO/IEC 12207:2016 IEEE (Inacal, 2016).

El estándar agrupa las actividades del ciclo de vida de desarrollo de software en cuatro grupos de procesos. Cada proceso del ciclo de vida dentro de esos grupos se describe en términos de su propósito y resultados esperados con un conjunto de actividades y tareas relacionadas que se pueden realizar para alcanzarlos (Lozano Alarcón & Montenegro Risco, 2020).

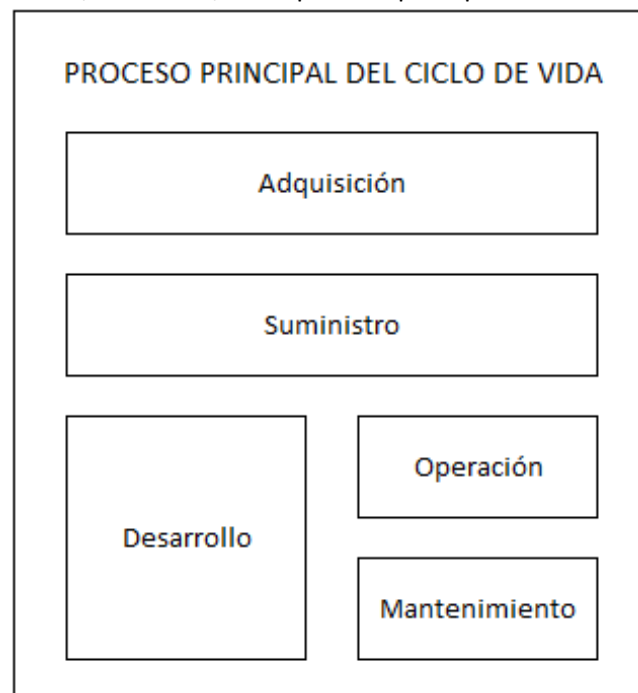
Los principales procesos del ciclo de vida son (Inacal, 2016): proceso de adquisición, proceso de suministro, proceso de desarrollo, proceso de operación y proceso de mantenimiento, ver Figura 11.

La NTP-ISO/IEC 12207 provee una propuesta de arquitectura para los procesos del ciclo de vida del *software*, pero no define el cómo se deben implementar las distintas actividades y tareas incluidas en los procesos. Las organizaciones que decida aplicar la norma deben especificar cual será el grupo de actividades y tareas implementará (Luna Aguinaga & Córdova Llacsahuache, 2019). Fue propuesta con la finalidad de especificar los procesos del ciclo de vida del *software* que permitan adquirir, proporcionar, desarrollar, operar y mantener el software, proporcionando así una hoja de ruta sobre la cual el usuario del pudiera basarse y orientarse para aplicarla (Gómez Checa, 2018).

La versión publicada es de uso obligatorio en el sector público en el Perú, mediante Resolución Ministerial N° 041-2017-PCM. Aun cuando se cumplen sus regularizaciones, no existen certificaciones para dicho estándar.

Figura 11

Estructura de la norma ISO/IEC 12207, en su proceso principal del ciclo de vida



Nota. Los procesos principales apoyan a las partes principales durante el ciclo de vida del software, (Inacal, 2016).

2.3.6.11 ISO/IEC 29110

La norma ISO/IEC 29110 define cuatro perfiles que clasifica el nivel de implementación de buenas prácticas en cualquier empresa, resumen de (Pastrana, Ordóñez, & Cobos, 2019):

- Inicial: La empresa no cuenta con estándares de desarrollo, gestión de proyectos y las distintas etapas de desarrollo se realizan sin lineamientos.
- Básico: La empresa cuenta con ciertos formalismos para cada una de las distintas

etapas, pero no implementan ninguna de las metodologías disponibles ni buenas prácticas.

- Intermedio: La empresa conoce e implementa algunas metodologías y buenas prácticas.
- Avanzado: Las empresas conocen el desarrollo de software que han adoptado. Todos sus artefactos están basados en modelos aceptados y reconocidos internacionalmente y el uso de buenas prácticas es parte de la cultura organizacional.

La norma ISO/IEC 29110 comprende una serie de directrices y documentos técnicos orientados a apoyar a las empresas muy pequeñas a incrementar la calidad de sus productos y servicios, incrementar su productividad y mejorar sus procesos con costos mínimos. Está orientado a minimizar la brecha en el uso de buenas prácticas en pequeñas y micro empresas, las cuales carecen de conocimientos o experiencia en la implementación de modelos o estándares. Los documentos (plantillas) definidos en la norma, pueden ser usados en sus procesos “tal cual”, porque están definidos para pequeñas organizaciones que no tienen procesos establecidos (Muñoz, Mejia, & Laporte, 2018).

El estándar ISO/IEC 29110 propone un conjunto de prácticas probadas o buenas prácticas, enfocadas en reforzar los procesos de gestión del proyecto y desarrollo del *software* para las micro y pequeñas empresas. Proporciona un conjunto de buenas prácticas, cuya implementación permite una mejora en la calidad de sus productos o servicios. El proceso de Implementación de *software* se realiza de forma sistematizada, las actividades del desarrollo de *software* son: análisis, diseño, implementaci[on], integración y pruebas; de acuerdo con los requisitos especificados, ya sea para el desarrollo de un nuevo *software* o su modificación. El proceso está compuesto por seis actividades: inicio, análisis de requisitos, arquitectura y diseño detallado, construcción, integración y pruebas y, entrega del producto (Muñoz, Peralta, & Laporte, 2019). La

norma ISO 29110 ayuda a mejorar el nivel de madurez de una organización joven mediante la implementación de mejores prácticas probadas y el desarrollo de productos de trabajo estandarizados (Laporte, Tremblay, Menaceur, Poliquin, & Houde, 2017).

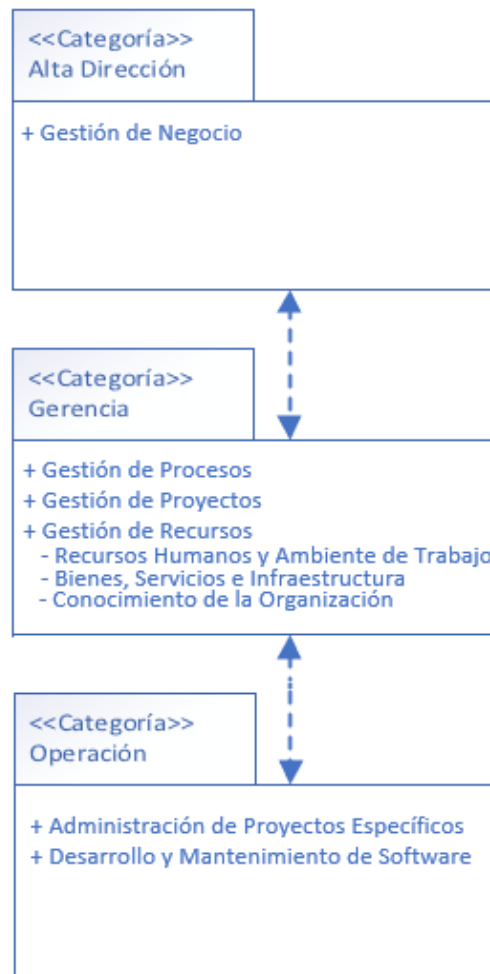
2.3.6.12 **MoProSoft**

MoProSoft es una norma desarrollada en Méjico (Norma NMX-I-059/02-NYCE-2016) que tiene como objetivo mejorar las capacidades de las organizaciones dedicadas al desarrollo de *software* para ofrecer servicios con calidad y alcanzar altos niveles de competitividad. Está diseñado para ser un modelo de referencia que describe un conjunto de buenas prácticas y métodos de gestión de proyectos en ingeniería de *software*, que contribuyen a que las organizaciones mejoren su forma de gestionar los proyectos. *MoProSoft* tiene como motivación la necesidad de alcanzar la madurez de los procesos de desarrollo de software, pues los modelos internacionales como *CMMI* se centran en empresas de gran tamaño.

Es compatible con *CMMI* e ISO/IEC 15504-7 (Darias González, 2020). En la Figura 12 se puede apreciar la estructura típica de *MoProSoft*.

Figura 12

Estructura de Moprosoft



Nota. Proporciona una estructura estándar de una organización, que puede ser grande o pequeña y adaptable a cada caso, **(Darias González, 2020)**.

2.3.6.13 Kanban

Kanban es la metodología que propuso la premisa *Just-in-Time (JIT)* o justo a tiempo. Utiliza un tablero para visualizar el avance del proyecto a través del tiempo (Saleh, Maruful Huq, & Rahman, 2019).

2.3.6.14 Prince2

Prince2 es flexible y puede adaptarse a cualquier tamaño y tipo (Mousaei & Javdani Gandomani, 2018). *Prince2 Agile* proporciona una propuesta de gobierno corporativo y de gestión de proyectos. El *framework Prince2* puede ser aplicado a proyectos de cualquier tipo, tamaño, duración, complejidad y alcance, es aplicable tanto a proyectos TI como a proyectos que no son de TI (Lynn Cooke, 2016).

Prince2 Agile describe cómo configurar y adaptar la metodología para poder aplicarlo de forma más eficaz mediante conceptos, *frameworks* con un enfoque ágil. Proporciona un modelo de procesos para la gestión de proyectos. Los procesos se pueden escalar y adaptar fácilmente para adaptarse a los requisitos de todo tipo de proyecto (Axelos Limited, 2018).

Prince2 se basa en seis principios, resumen de (Alarcón & Rojas, 2018):

- Justificación para el negocio: El proyecto debe ser importante desde el punto de vista del negocio y debe tener un retorno de la inversión (*ROI, return of investment*).
- Lecciones aprendidas: Se debe aprender de proyectos anteriores, y aplicar lo aprendido.
- Definición de roles y responsabilidades: Los miembros del equipo deben conocer cuáles son sus tareas y responsabilidades, y las de los demás.
- Administración por etapas: Se divide el proyecto en partes administrables y entregables.
- Administración por excepciones: Un principio que no se encuentra en otras metodologías. Cuando ocurran problemas en el proyecto, serán gestionado por los responsables de más alto nivel del proyecto.
- Enfocado en productos: Todo el esfuerzo del proyecto está orientado a entregar los productos que requieren los *stakeholders*.

- Adaptación al ambiente del proyecto: Los distintos elementos del proyecto debe adaptarse al entorno en el que se desarrollan.

2.3.7 *Software Colaborativo*

Se conoce como software colaborativo, sistemas *Groupware* o sistemas colaborativos son sistemas de software que apoyan el desarrollo de actividades en las que interactúa un grupo de usuarios para combinar sus habilidades y trabajo para alcanzar una meta en común (Cepero García, Montané Jiménez, Toledo Toledo, Benítez Guerrero, & Mezura Godoy, 2021).

Los *softwares* colaborativos se los puede clasificar en siete (07) tipos:

1. Sistemas de control de versiones:

La ingeniería de *software* distribuido necesita gestión de la configuración de modo sistemático. Un sistema de control de versiones permite a los miembros del equipo compartir componentes de software de una manera controlada.

2. *Trackers*

Se utilizan para gestionar los problemas (o '*tickets*') tales como defectos, cambios o solicitudes de atención. Las funciones de seguimiento que implementan las centralizan en una base de datos a las que todos los integrantes del equipo pueden acceder.

3. Herramientas de desarrollo

Cuando el proyecto de desarrollo de *software* se encuentra en un entorno más distribuido, son mayores los requisitos de seguridad, disponer de un repositorio remoto y gestión de archivos de código fuente y binarios compilados.

4. Modeladores

Permite compartir no solo los archivos, sino además el contenido de los mismos. Permite el modelamiento y diseño de *software*, en un entorno distribuido y colaborativo.

5. Bases de Conocimientos

Son sistemas de gestión de contenidos que permiten a los miembros del equipo comparten el conocimiento explícito en la *web*.

6. Herramientas de Comunicación

Los ingenieros de *software* han adoptado una amplia gama de tecnologías de comunicación del proyecto de corriente cuando la interacción directa no es posible. Las herramientas de comunicación asíncrona incluyen correo electrónico, listas de correo, grupos de noticias, foros *web* (Farinha & Mira da Silva, 2011), y más recientemente los *blogs*. Herramientas síncronas incluyen llamadas telefónicas y conferencias estándar, *chat*, mensajería instantánea, voz sobre *IP* y videoconferencia.

7. Aplicaciones Web 2.0

Las aplicaciones *Web 2.0* extienden los *softwares* de colaboración tradicionales mediante las aportaciones directas de los usuarios, las interacciones ricas y construcción de la comunidad. Algunas aplicaciones clave de la *Web 2.0* son los blogs, como *WordPress* (<http://wordpress.org>); *microblogs*, como *Twitter* (<http://twitter.com>); *wikis*, como *Portland Pattern Repository* (<http://c2.com/cgi/wiki>). Recientemente las aplicaciones *Web 2.0 open-source* se han vuelto más populares en proyectos de código abierto y de alcance global. En la *Web 2.0*, el contenido es generado por los propios usuarios, mediante la inteligencia colectiva basada en la agregación de aportes individuales, que no son sistematizados ni guiados (Sánchez Medero & Pastor Albaladejo, 2020).

2.3.8 Priorización de Requisitos

En general, existen más requisitos que los recursos disponibles (limitación de recursos) y las restricciones contractuales o presupuestales (Bennaceur, Than Tun, Yu, & Nuseibeh, 2019).

Existen algunos métodos de clasificación de requisitos que se pueden utilizar.

La priorización de requisitos de software es una tarea importante y difícil que se realiza durante la fase de gestión de requisitos de la ingeniería de requisitos (Yaseen, Mustapha, Ur Rahman, Khan, & Kamal, 2019). Un jefe de proyecto tiene que tomar numerosas decisiones relativas a los recursos disponibles, opiniones contradictorias de los *stakeholders*, oportunidades de mercado, riesgo, costos, entre otros aspectos (Kyosev, 2014).

Para un gran número de requisitos, típico en los proyectos grandes, se requiere un método de priorización que destaque los factores de precisión, tiempo-eficiencia y el fácil aprendizaje (McZara, Sarkani, Holzer, & Eveleigh, 2014) y (Kyosev, 2014); aunque también se aplica en *startups*, las cuales aplican un enfoque ágil y bajo su propio modelo (Melegati, Goldman, Kon, & Wang, 2019). Asimismo, en su investigación destacaron la importancia que viene adquiriendo la priorización de requisitos.

La priorización de requisitos es una actividad importante dentro de los procesos que forma parte de la ingeniería de requisitos. Los requisitos priorizados permiten realizar los procesos en cada una versión, realizándose una adecuada estimación de recursos a asignar en todo el ciclo de vida del producto. En su investigación (Lorentz, 2009) denominó a la priorización como racionalización. Una univoca lista de requisitos de *software* es un factor crítico de éxito para asegurar un eficiente proceso de ingeniería

de *software* (Achimugu, Selamat, Ibrahim, & Naz'ri Mahrin, 2014).

2.3.8.1 Proceso de Priorización de Requisitos

La priorización de requisitos es mayormente realizada al inicio de cada iteración (Jarzębowicz & Sitko, 2020).

2.3.8.2 Aspectos de la Priorización

Se encuentran divididos en dos grandes categorías:

- Aspectos del negocio.
- Aspectos técnicos.

2.3.8.2.1 Aspectos del negocio

Incluyen aspectos de negocios como ventas, marketing, competitividad, estrategia, retención de clientes, simplicidad, innovación, ingenio, centrado en el cliente y disponibilidad (Sher, Jawawi, & Mohammad, 2019), ver Figura 13. Estos aspectos son muy importantes y tienen mucha preponderancia en la priorización.

Figura 13

Aspectos de negocios en la priorización de requisitos



Nota. Los aspectos del negocio implican aspectos comerciales, del cliente e innovación, por (Sher, Jawawi, & Mohammad, 2019).

2.3.8.2.2 Aspectos técnicos

Los aspectos más comunes son la importancia, la penalidad, el costo, el tiempo y el riesgo. Cuando se priorizan requisitos basados en un solo aspecto, el proceso de definir cuál es el más relevante es simple. Sin embargo, cuando se involucran otros aspectos como el costo, los clientes pueden cambiar sus ideas y aquellos requisitos de mayor importancia pueden ser menos importantes si son costosos de implementar. A menudo los aspectos interactúan y los cambios en un aspecto podrían impactar en otro aspecto.

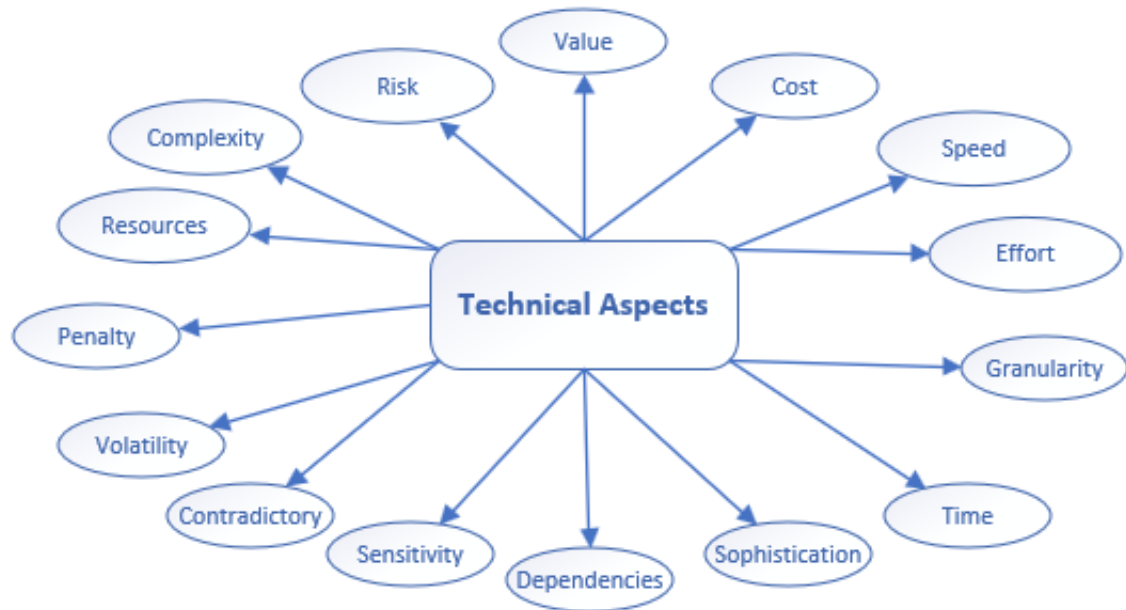
Los principales aspectos a considerar son:

- Penalidad: Es posible evaluar la penalidad cuando se introduce un requisito que no se cumple. Por ejemplo: al no cumplir con una norma, el sistema podría incurrir en una penalidad; incluso si esta es de poca importancia para el cliente

- Costo: El costo de implementación es estimado por lo general por el equipo de desarrollo de la organización. Medidas que influyen en el costo son: complejidad del requisito, capacidad de reutilizar el código existente, cantidad de pruebas y documentación necesaria, etc. El costo es a menudo expresado en términos de horas-hombre (esfuerzo), ya que el costo principal de desarrollo de *software* es por lo general relacionado con el número de horas ejecutadas.
- Tiempo: El tiempo de desarrollo de *software* está relacionado con el número de horas-trabajo realizado. Sin embargo, el tiempo está influenciado por muchos otros factores tales como el paralelismo en el desarrollo de *software*, la capacitación, el desarrollo de infraestructura de desarrollo, el cumplimiento de estándares, etc.
- Riesgo: Todo proyecto conlleva cierto nivel de riesgo. La gestión de riesgos se utiliza para mitigar los riesgos internos y externos, sobre los cuales la organización tiene un mínimo de control o no lo tiene. Estos riesgos podrían incluir, por ejemplo: riesgos de desempeño, riesgos de procesos, riesgos de horario, etc.
- Volatilidad: La volatilidad en los requisitos afecta la estabilidad y la planificación de un proyecto, e incrementa los costos de un proyecto.

Figura 14

Aspectos técnicos en la priorización de requisitos



Nota. Los aspectos técnicos son numerosos, es importante gestionarlos adecuadamente, por (Sher, Jawawi, & Mohammad, 2019).

2.3.9 Técnicas de Priorización de Requisitos

Los investigadores han desarrollado diferentes técnicas de priorización de requisitos para la correcta selección y ordenación de los requisitos de usuario (Kyosev, 2014) y (Adhim, Hudaib, & Al-Shboul, 2019).

Para realizar la priorización de los requisitos, los *stakeholders* deben de realizar la comparación de los mismos para determinar su importancia relativa a través de un sistema puntuación o calificación. Estas comparaciones se convierten en un proceso complejo con el aumento en el número de requisitos (Achimugu, Selamat, Ibrahim, & Naz'ri Mahrin, 2014). En entornos reales de trabajo, numerosos stakeholders participan en la toma de decisiones con diferentes opiniones y perspectivas (Ibriwesh, Ho, Chai, & Tan, 2019). Hay que tener en consideración que algunas técnicas funcionan bien con un número pequeño de requisitos, pero para proyectos medianos o grandes no funcionan muy bien (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019).

En su tesis de maestría, (Kyosev, 2014) llamó ‘métodos’ a las técnicas de priorización.

2.3.10 *Software Libre y de Código Abierto*

Software libre y de código abierto es aquel que cumple con ambas condiciones. Es decir, cualquiera es libre de revisar, copiar, estudiar y modificar el *software* según considere, el código fuente se comparte abiertamente para que la gente se sienta libre de mejorar o corregir el *software*. Esto contrasta con el *software* propietario, donde el *software* está bajo un *copyright* restrictivo y cuyo código fuente está por lo general oculto a los usuarios. En inglés, el término es: *Free and open-source software* (FOSS). Asimismo, *The Free Software Definition* recomienda usar los términos y conceptos sobre licencia de *software* en su versión traducida (FSF, 2019).

Es todo *software* de código abierto y gratuito (*Free and open-source software: FOSS*), que ofrecen la mayoría de soluciones eficaces para la transferencia de tecnología en los programas de *software*. Por lo tanto, el *software* libre tiene un potencial importante en la transferencia de conocimiento en los países en desarrollo y la transferencia de conocimiento y práctica entre estudiantes e investigadores. Los *FOSS* han demostrado ser de alta calidad en comparación con el *software* propietario y una de sus ventajas es la continua revisión del modelo de desarrollo.

En 1984 se creó un nuevo paradigma en la industria del *software* cuando Richard Stallman propuso en ese año la definición de *open-source* (código abierto).

Sus principios incluyen:

- Código fuente disponible gratuito.
- De uso libre para cualquier propósito.

- Gratis para modificar y para personalizar.
- Gratis para redistribuir.
- Gratis para crear trabajos que se deriven.
- Libre de unirse al desarrollo y cooperación.

El concepto de *software* libre se ha convertido una nueva forma de aprendizaje que se encuentra en la *web* en la actualidad. Las comunidades de *software* libre colaboran para suministrar y distribuir programas de calidad de la manera más ágil y continua. El principio fundamental es que debe haber libre acceso al código de los programas, permitiendo que cualquiera pueda usar, modificar y distribuir tales modificaciones sin tener que pagar por ello.

2.3.11 *Licencia de Software*

Una licencia de *software* es un acuerdo entre el usuario o adquiriente y el propietario o autor del *software*, en el cual el segundo les otorga al primero ciertos permisos que de otra forma infringirían la ley de derechos de autor. Por lo general, una licencia de *software* contiene en conjunto de reglas que un usuario tiene que cumplir mientras use el *software* (Zahoor & Sarwar Bajwa, 2014).

Las licencias de *software* de los elementos del modelo seleccionados deberán permitir la copia, el acceso al código fuente de los mismos, permitir su modificación y su distribución junto con otros componentes sin una autorización explícita del autor, pero si previamente descritas en la licencia, y todo ello sin implicar costos.

2.3.11.1 Categorías de software Libre y No Libre

Existe una serie de definiciones sobre los tipos o categorías de *software*, tanto libres como propietarios, descritas por (FSF, 2019).

2.3.11.1.1 **Software Libre**

Software Libre es aquel al cual el autor autorizó para que cualquier usuario pueda hacer uso, copiar o distribuir, y en ocasiones permite su modificación. Esto último cuando el autor publica el código fuente.

2.3.11.1.2 **Software de Código Abierto – Open-Source**

Software es aquel que da libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el *software* (Stallman, 2020). Otorga cuatro clases o niveles de libertad a los usuarios de software:

- Libertad 0: Puede ejecutar el programa sea cual sea el propósito.
- Libertad 1: Puede estudiar o revisar el funcionamiento del *software* y si considera, adaptarlo a tus necesidades — permitir el acceso al código fuente es obligatorio.
- Libertad 2: Puede redistribuir copias y ayudar así a tu vecino.
- Libertad 3: Puede mejorar o corregir el *software* y realizar la publicación en la comunidad, permitir el acceso al código fuente obligatorio.

2.3.11.1.3 **Software Propietario**

Software propietario es una forma de denominar al *software* no libre. Es aquel en el cual los desarrolladores o propietarios se reservan todos los derechos y libertades. Estas libertades y derechos no les permiten a los usuarios: la libertad de analizar el *software* ni realizar cambios en él o de ser compartido o la libertad de ejecutarse con algún propósito que no esté descrito en los términos de licencia. El código fuente generalmente no está incluido.

La licencia de *software* privativo recibe el nombre de contrato de licencia de

usuario final (CLUF) o por sus siglas en inglés *End User License Agreement (EULA)*.

2.3.11.1.4 **Freeware**

Freeware es aquel *software* que se publica a título gratuito y está disponible para su uso por tiempo ilimitado. No se permite la modificación ni está accesible su código fuente. El propietario incluye restricciones adicionales, tales como que el *software* debe ser solo para uso personal, académico y no comercial.

2.3.11.1.5 **Software Privado**

El *software* privado o personalizado es desarrollado para un usuario (típicamente una organización o compañía). Dicho usuario lo mantiene y lo utiliza, y no lo libera al público, ya sea como código fuente o como binarios.

2.3.11.1.6 **Software Comercial**

Es aquel fabricado por una persona o empresa con fines de obtener regalías económicas. La mayoría de *softwares* comerciales son privativos, pero también existen *softwares* libres que son comerciales, y *softwares* privativos que no son comerciales. Comercial y privado no son lo mismo.

2.3.11.1.7 **Software con Licencia Copyleft**

Es un tipo de licencia de derecho de autor, que incluye el *software*, cuyos términos de publicación permiten que las copias tengan los mismos términos de distribución, o en caso de ser modificadas, el autor de la modificación lo puede distribuir gratuitamente o con una licencia comercial. Es decir, las licencias *copyleft* no permiten que un tercero le agregue restricciones adicionales al software e inciden en la publicación del código fuente.

2.3.11.1.8 **Software con Licencia GPL**

La Licencia Pública General de GNU (*General Public License* - GNU GPL) son unas cláusulas para publicar software con permisos *copyleft*. Si el autor de un *software* decide publicar su obra, utiliza *software* base con licencia *GPL*, también deberá publicar su *software* con licencia *GPL* e incluir su código fuente. Aunque, si el autor utiliza su *software* elaborado con *software* base *GPL* en un entorno privado, no requerirá distribuir el código fuente de su obra. Una facilidad que proporciona la licencia *GNU GPL* es que ya se encuentra redactada, revisada, con múltiples aportes y evaluaciones, y es de acceso libre.

2.3.11.1.9 **Licencia BSD**

Es una licencia de *software* libre, que permite el uso de código fuente con licencia *BSD* en *software* no libre (propietario), lo que la diferencia de la licencia *GPL*. No obliga al desarrollador a publicar el código fuente, aun si el autor del *software* distribuido le otorga un valor comercial, por tanto, cobra por él. Es la licencia con menos restricciones.

3. Capítulo III: METODOLOGÍA

Para validar el modelo propuesto, se implementó un ejemplo de aplicación, que se desarrolla a través del siguiente caso de estudio:

Se hizo una prueba de campo de tipo experimental con participantes voluntarios a estudiantes universitarios que asumieron de manera ficticia los roles de *stakeholders* involucrados en el dominio del problema. El estudio experimental busca determinar la validez de la técnica de priorización propuesta en un entorno distribuido, usando una herramienta *CMS*.

La estructura del experimento se basa en la propuesta de (Mustafa & Zainuddin, 2014), quien realizó un diseño experimental para comparar distintas técnicas de priorización de requisitos con *software*.

Para evaluar la viabilidad del modelo propuesto, llevará a cabo una evaluación con un *test* de viabilidad que determine la validez del modelo propuesto, empleando una métrica.

3.1 Tipo y diseño de la Investigación

La presente investigación tiene un enfoque cuantitativo, mediante el cual se alcanzarán los objetivos y se verificarán las hipótesis propuestas. En un enfoque cuantitativo, la investigación representa a un conjunto de procesos organizados en forma secuencial para comprobar las hipótesis propuestas (Hernández Sampieri & Mendoza Torres, 2018). Este enfoque permite recabar opiniones sobre un problema en particular, pueden ser experimentales o no experimentales.

El tipo de la presente investigación es experimental.

El diseño es transversal. En este tipo de diseño, las variables son medidas en una sola vez en el tiempo (Corona Lisboa, 2016), estudiándose su comportamiento a través de los valores de sus indicadores.

3.2 Hipótesis

La hipótesis es una enunciado o proposición que requiere ser probado y se define como una explicación provisional del fenómeno a investigar. Se apoyan en el marco teórico existente y se define mediante una proposición. Por tanto, se constituye en una respuesta provisional que será validada como verdadera o falsa (Hernández Sampieri & Mendoza Torres, 2018).

3.2.1 Hipótesis General

Se formula la siguiente hipótesis general:

- *HG: Un modelo de priorización de requisitos implementado en una herramienta CMS contribuye de manera favorable en la toma de decisiones de los stakeholders.*

3.2.2 Hipótesis Específicas

El objetivo del experimento es investigar las siguientes hipótesis específicas:

- *HE1: Es posible diseñar un modelo de priorización de requisitos que pueda ser implementado con una adecuada “facilidad de uso” por los stakeholders.*

- HE2: Es posible “implementar” un modelo de priorización de requisitos utilizando softwares CMS de código libre y redes sociales.
- HE3: Es posible implementar un modelo de priorización de requisitos con un enfoque de bajos “costos en recursos”.

3.3 Variables e Indicadores

Al formular una hipótesis, se debe definir los términos o variables incluidos (Hernández Sampieri & Mendoza Torres, 2018). La relación entre variables se puede ver en la **Error! Reference source not found..**

Figura 15

Esquema de experimento y variables



Nota. Visualización o esquema gráfico, (Hernández Sampieri & Mendoza Torres, 2018).

3.3.1 Variable Independiente

Las variables independientes son aquellas que se pueden manipular durante el experimento (Kyosev, 2014).

La variable independiente es:

- VI1: Modelo de Priorización de Requisitos.

3.3.1.1 Indicador

VI1-I1: Presencia - Ausencia

Descripción

NO: No existe un “Modelo de Priorización de Requisitos”, y se encuentra en la situación actual del problema.

SI: Si se ha implementado en “Modelo de Priorización de Requisitos” en una herramienta y se espera contribuye en la toma de decisiones de los stakeholders.

3.3.2 Variable Dependiente

La variable dependiente es (resultado de la investigación):

- VD1: Toma de decisiones de los stakeholders.

3.3.2.1 Indicadores

- VD1-I1: Facilidad de Uso

Descripción

Nivel de usabilidad percibido por los stakeholders al realizar la priorización de requisitos en una herramienta *CMS*.

- VD1-I2: Implementabilidad

Descripción

Facilidad o dificultad en la implementación del modelo propuesto en una herramienta *CMS*, bajo el enfoque de bajos recursos.

- VD1-I3: Costos en recursos.

Descripción

Costo generado en la implementación del Modelo de Priorización de Requisitos en una herramienta *CMS*.

3.4 Operacionalización de Variables

Tabla 1

Operacionalización de Variables

Variable	Tipo de Variable	Definición conceptual	Indicadores	Descripción del indicador
Modelo de Priorización de Requisitos.	Variable independiente	Modelo de priorización de requisitos de software en entorno distribuido que deberá poderse implementar en una herramienta de amplio uso, como son los sistemas de gestión de contenidos (CMS) y validado con un caso de prueba, se evaluará si el modelo es viable.	Presencia - Ausencia	Aplicación del Modelo de Priorización de Requisitos en una herramienta CMS.
Toma de decisiones de los stakeholders	Variable dependiente	Toma de decisiones de los stakeholders para realizar la priorización de una lista de requisitos.	Facilidad de Uso	Nivel de usabilidad percibido por los stakeholders al realizar la priorización de requisitos en una herramienta CMS.
			Implementabilidad	Facilidad o dificultad en la implementación del modelo propuesto en una herramienta CMS, bajo el enfoque de bajos recursos.
			Costos en recursos	Costo generado en la implementación del Modelo de Priorización de Requisitos en una herramienta CMS.

Elaboración propia.

3.5 Matriz de Consistencia

Tabla 2

Matriz de consistencia

PROBLEMAS	OBJETIVOS	HIPÓTESIS	VARIABLES E INDICADORES	METODOLOGÍA
Problema General	Objetivo General	Hipótesis General	Variable Independiente	Método de Investigación
- PG: ¿De qué manera un modelo de priorización de requisitos implementado en una herramienta colaborativa CMS contribuye en la toma de decisiones de los stakeholders?	- OG: Identificar como un modelo de priorización de requisitos usando una herramienta colaborativa CMS contribuye en la toma de decisiones de los stakeholders.	- HG: Un modelo de priorización de requisitos implementado en una herramienta CMS contribuye de manera favorable en la toma de decisiones de los stakeholders.	VI1: Modelo de Priorización de Requisitos.	Enfoque cuantitativo.
			Indicador VI1-I1: Presencia – Ausencia. Descripción NO: No existe un “Modelo de Priorización de Requisitos”, y se encuentra en la situación actual del problema. SI: Si se ha implementado en “Modelo de Priorización de Requisitos” en una herramienta y se espera contribuye en la toma de decisiones de los stakeholders.	Tipo de Investigación
				Experimental
				Tipo de diseño
				Transversal
				Metodología
				Cuantitativa
			Variable Dependiente	Población de Estudio
			- VD1: Toma de decisiones de los stakeholders.	Colaboradores de una entidad pública realizando sus prácticas preprofesionales.
Problemas Específicos	Objetivos Específicos	Hipótesis Específicas	Indicadores	
- PE1: ¿Cómo estaría diseñado un modelo de priorización de requisitos pueda ser implementado con una buena “facilidad de uso” en una herramienta colaborativa CMS?	- OE1: Diseñar un modelo de priorización de requisitos que pueda ser implementado con una buena “facilidad de uso” en una herramienta colaborativa CMS.	- HE1: Es posible diseñar un modelo de priorización de requisitos que pueda ser implementado con una adecuada “facilidad de uso” por los stakeholders.	- VD1-I1: Facilidad de Uso. Descripción Nivel de usabilidad percibido por los stakeholders al realizar la priorización de requisitos en una herramienta CMS.	Muestra
				Seis (06) colaboradores.
				Instrumento
- PE2: ¿Cómo se puede “implementar” modelo de priorización de requisitos en el cual los stakeholders puedan realizar la priorización de una lista de requisitos mediante una herramienta CMS de código libre y redes sociales?	- OE2: “Implementar” un modelo de priorización de requisitos con el cual los stakeholders puedan realizar la priorización de una lista de requisitos mediante una herramienta CMS de código libre y redes sociales.	- HE2: Es posible “implementar” un modelo de priorización de requisitos utilizando softwares CMS de código libre y redes sociales.	- VD1-I2: Implementabilidad. Descripción Facilidad o dificultad en la implementación del modelo propuesto en una herramienta CMS, bajo el enfoque de bajos recursos.	Encuesta.
- PE3: ¿De qué manera se implementaría la priorización de requisitos con un enfoque de “bajos costos” en recursos?	- OE3: Implementar un modelo de priorización de requisitos con un enfoque de bajos “costos en recursos”.	- HE3: Es posible implementar un modelo de priorización de requisitos con un enfoque de bajos “costos en recursos”.	- VD1-I3: Costos en recursos. Descripción Costo generado en la implementación del Modelo de Priorización de Requisitos en una herramienta CMS.	

Elaboración propia.

3.6 Metodología

3.6.1 Unidad de análisis

Se uso como unidad de análisis al área de informática de una entidad del sector público.

3.6.2 Población de estudio

Se uso como población de estudio a grupo a los colaboradores quienes al momento de la prueba eran estudiantes quienes se encontraban realizando sus prácticas preprofesionales en el área de informática de una entidad el sector público.

3.6.2.1 Sujetos

Es frecuente realizar los estudios de investigación empíricos con estudiantes en lugar de profesionales. La razón es porque el trabajar con profesionales resulta imposible por razones de costo; además, los estudiantes están dispuestos a participar como parte de los cursos a los que asisten. El enfoque de realizar la investigación con estudiantes se realiza utilizando la técnica del muestreo.

En este experimento, la muestra se tomará como *stakeholders* a los estudiantes de pregrado de ingeniería de sistemas. Para asegurar que la muestra es representativa, se tomó como participantes a aquellos estudiantes que hayan estudiado los cursos que incluyan en sus tópicos temas referentes a la Ingeniería de Requisitos.

Para que la muestra tenga el menor sesgo, se pidió que participaran todos los estudiantes para la prueba entre quienes reunían los requisitos para poder realizarla, quienes aceptaron el pedido y participaron todos.

Tomando como referencia a (Mustafa & Zainuddin, 2014), participarán seis (06) estudiantes quienes realizarían la tarea de priorización de requisitos con la lista de requisitos propuesta. Esta es la cantidad de *stakeholders* que permitirá realizar un cálculo de verificación de los resultados obtenidos.

3.6.3 *Tamaño de la muestra*

Se utilizó como tamaño de la muestra a todos los colaboradores como practicantes preprofesionales, que eran un numero de seis (06).

3.6.4 *Selección de la muestra*

Para no implicar algún criterio de selección o exclusión que pudiesen influir en los resultados se invitó a participar a todos trabajadores que cumplieran con el perfil propuesto, participando todos los colaboradores (06) en la presente investigación.

3.6.5 *Técnica de Recolección de datos*

La recolección de datos se realizó de la siguiente manera:

El experimento fue realizado en sesiones breves, de cinco (05) minutos o menos, luego de haberles realizado una breve explicación verbal sobre la realización de la priorización de requisitos en la herramienta *CMS* en la cual se implementa el modelo. En ella, los participantes le asignan un puntaje de 0 (no asigna un puntaje) a 5 (máximo valor asignable) a cada requisito. Internamente, la herramienta *CMS* realiza el cálculo y priorización de los requisitos.

La evaluación del instrumento en la cual los colaboradores que participaron en el experimento evalúan la usabilidad (facilidad de uso) de la implementación del modelo propuesto. La escala de clasificación fue de 1 al 5 (1= difícil, 5=fácil).

3.6.1 Técnica de procesamiento y análisis de datos

Los valores de las variables se expresaron en valores numéricos, por corresponder con un análisis de tipo cuantitativo.

Los valores de la lista de los requisitos ordenados en cuanto a su priorización son realizados por el plugin que implementa el modelo, instalado en la herramienta *CMS*.

El puntaje asignado por cada uno de los participantes es recolectado en una hoja de *Excel* y a través de una formula se obtiene el valor promedio.

4. Capítulo IV: ESTADO DEL ARTE

La priorización de requisitos es una de las fases más importantes en la gestión de requisitos en el proceso de desarrollo de *software*, constituye un factor importante en el éxito o el fracaso del desarrollo de software (Rida Asghar, Tabassum, Nazir Bhatti, & Ali Shah, 2017).

Para permitir una eficiente colaboración, se podrían utilizar los Sistemas de Apoyo a Grupos (SAG).

La priorización de requisitos es un tema importante, tanto para investigadores como para profesionales (Jarzębowicz & Sitko, 2020). La mayor parte de estudios sobre priorización de requisitos de *software* también son de años recientes, y con una orientación a las técnicas y métodos.

Un conjunto de requisitos grande, como, por ejemplo, en un sistema de planificación de recursos empresariales (*ERP*), resulta difícil de administrar e implementar por un solo desarrollador. Los requisitos se deben distribuir en miembros del equipo en paralelo (Yaseen, Mustapha, Ur Rahman, Khan, & Kamal, 2019).

4.1 Taxonomía

La presente tesis se encuentra clasificada bajo las líneas de investigación de las instituciones de tecnologías *Association for Computing Machinery - ACM*, del *Institute of Electrical and Electronics Engineers* - Instituto de Ingeniería Eléctrica y Electrónica - IEEE y de la Universidad Nacional Mayor de San Marcos - UNMSM.

- De acuerdo la clasificación taxonómica del (ACM, 2019), el presente trabajo se clasifica en:
 - *SOFTWARE AND ITS ENGINEERING*
 - Software creation and management*
 - Designing software*
 - Requirement's analysis*
- De acuerdo la clasificación taxonómica del (IEEE, 2019), el presente trabajo se clasifica en:
 - *SYSTEMS ENGINEERING AND THEORY*
 - Requirement's engineering*
- De acuerdo a documento vigente "Programas y Líneas de Investigación en la Universidad Nacional Mayor de San Marcos", aprobado en setiembre de (UNMSM, 2014), el presente trabajo clasifica en:

AREA C: INGENIERÍAS

C.0.3. TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

C.0.3.22. Ingeniería de *software*

4.2 Metodología de Búsqueda

Para realizar la búsqueda se utilizaron plataformas de base de datos de documentos variados (tesis de maestría y doctorado, revistas científicas, libros y otros), entre ellas:

- *IEEE Xplore.*
- *ACM Digital Library.*
- *ScienceDirect.*
- *Web of Science.*
- *Springer.*

- *Scopus*.
- IJSCE - Internactional Journal of Computer Science and Engineering.
- IJSR - International Journal For Research In Advanced Computer Science.
- *Proquest*.
- *Directory of Open Access Journals (DOAJ)*.
- *SCIELO*.
- Elsevier ScienceDirect.
- HINDAWI Publishing Corporation.
- Journal of computer Science.
- International Journal of Software.
- Journal of Networks.
- International Journal of Computer Applications.
- Chiang Mai Journal of Science.
- CLEI Electronic Journal.
- ArXive.

Luego de acceder a estos repositorios, se filtró la búsqueda mediante algunos criterios tales como:

- Criterios de Selección:
 - Documentos que contienen información sobre los temas mencionados.
- Criterios de Inclusión:
 - Fecha: Se restringió la búsqueda a documentos que daten del año 2016 a la actualidad.
 - Tipo de artículos: Incluyendo tesis de maestría y doctorado y artículos científicos.
 - Palabras clave (keywords): Se utilizaron diversas palabras o expresiones claves tales como: "*requirements*", "*requirements prioritization*", "*cms*",

"forum", "requirements engineering", "requirements prioritization", "requirements techniques", "stakeholder", "software requirements engineering", "software requirements elicitation", "prioritization", "content management system", "prioritization methods" entre otras relacionadas al problema propuesto.

- Criterios de Exclusión:
 - Fecha: Se excluyeron aquellos documentos que sean anteriores al año 2014 para los antecedentes del problema (a excepción de los libros).
 - Tipos de artículos: Se excluyeron de los criterios de búsqueda trabajos que no estén publicados en revistas científicas, así como también artículos de tipo comercial.
- Evaluación de los contenidos: Exactitud, objetividad, cobertura, eficiencia, Se investiga en la cadena de autores que han realizado contribuciones a los artículos y temas analizados.

4.3 Revisión del Estado del Arte

Las investigaciones de priorización de requisitos estudiados tienen o un enfoque teórico o un enfoque práctico, pero no ambos aspectos.

Las investigaciones realizadas desde un enfoque teórico solo son validadas en base al marco teórico, la bibliografía, la lógica y la argumentación que elaboran los autores. Sin embargo, no es posible conocer la validez de la propuesta en un entorno real, porque los autores no han realizado la implementación de su propuesta.

En las investigaciones realizadas desde un enfoque práctico, desarrollan una propuesta basada en *software* colaborativo y apto para trabajar en un entorno distribuido funcional, sin embargo, no es posible conocer la validez de la propuesta,

porque los autores no incluyen bases teóricas, ni método o técnica que las respalde como válidas.

Diversas propuestas del enfoque práctico utilizan las wikis como software base, sin embargo, las *wikis*, según (BuiltWith® Pty Ltd, 2019, pág. CMS technologies Web Usage Statistics), representan el 0.51% de las plataformas instaladas a nivel mundial.

4.3.1 Benchmarking de trabajos sobre trabajos relacionados

Tabla 3

Evaluación del Estado del Arte de las investigaciones actuales.

Trabajo de Investigación	Autor(es)	Aspecto Teórico	Aspecto Práctico	Validación	Aspectos de Seguridad	Comentario
Usando <i>Wikis</i> como Estrategia Colaborativa para Apoyar la Elicitación de Requisitos de <i>Software</i> .	(Sevilla, Zapata, Torres, & Collazos, 2014)		X	X		Utilizó como software base la herramienta <i>SoftWiki</i> .
Elicitación de Requerimientos Distribuidos utilizando <i>Spatial HypertextWiki</i> .	(Seyff, Todoran, Caluser, & Singer, 2015)		X	X		El objetivo de los autores es investigar cómo las redes sociales populares pueden apoyar, a través de preguntas y tablas de validación.
<i>ReproTizer</i> : Una Herramienta de Priorización de Requisitos de <i>Software</i> Completamente Implementada.	(Achimugu, Selamat, & Ibrahim, 2016)	X	X	X		En su propuesta implementan una plataforma completa, incluyendo un algoritmo propio, sin basarse en una técnica de priorización. Le realizaron pruebas de escalabilidad, computacional complejidad, precisión, tiempos de respuesta. Sin embargo, no incluyeron funcionalidades en aspectos de seguridad. Los análisis de vulnerabilidades son importantes.
<i>MBRP</i> : Un modelo de priorización de requisitos basado en el algoritmo <i>Pagerank</i> .	(Abbas, Naila Jan, Saadatmand, & Daniel Sundmark, 2019)	X		X		En su propuesta describen la implementación del algoritmo <i>Pagerank</i> , sin embargo, una descripción del mismo no está disponible, y solo se puede

						aproximar su diseño y por tanto los resultados.
Herramienta de inspección de <i>software</i> para la clasificación de defectos en documentos de requisitos utilizando un enfoque colaborativo.	(Mani Subedi, 2019)	X	X			En su artículo de investigación para obtener el grado de maestría, el autor propone una herramienta la que desarrolla hasta el nivel de prototipo. En su propuesta se enfoca en la evaluación documentaria, pero pueden ser usada en otras etapas. En su propuesta no incluye el código fuente, además, no incluyeron funcionalidades en aspectos de seguridad. Los análisis de vulnerabilidades son importantes.
iStar-p: Un lenguaje de modelado para la priorización de requisitos.	(Flório, Lencastre, Pimentel, & Araujo, 2019)	X	X	X		En la investigación, los autores proponen un lenguaje de modelado para realizar la priorización. Sin embargo, no definen sobre que herramienta CASE se realizará el uso del lenguaje, o cual es el enfoque que aplicará en entornos distribuidos.
Priorización de requisitos de <i>software</i> con el lenguaje de requisitos orientado a objetivos GRL.	(Leshob, Hadaya, & Renard, 2019)	X	X	X		Los autores proponen el uso del lenguaje DRL para realizar la priorización de requisitos. Incluyen un completo set de ejemplos y documentación sobre el uso del lenguaje y como realizar cálculo de la información ingresada por los usuarios. No incluye o describe como sería la captura de los datos con múltiples usuarios.
Uso de redes sociales para identificar los requisitos	(Ali & Hong, 2019)	X	X	X		En su investigación, los autores proponen el uso de redes sociales para

iniciales del usuario para pequeñas empresas de software.						adecuar los métodos de ingeniería de software a las pequeñas empresas. El caso práctico lo realizan con <i>Facebook</i> , mediante un grupo de discusión y un moderador. Sólo participaron un 33% de los <i>stakeholders</i> . La votación lo realizaron a través del 'me gusta'. Por tratarse de una red social, las cuentas de los usuarios no se encuentran administradas por la empresa.
Desarrollo de una herramienta para el apoyo al proceso de priorización de requisitos en un entorno de desarrollo de software ágil.	(Mowad, AbdElazim Muhammad, & Elfakharany, 2020)	X	X	X		Los autores usaron como <i>software</i> base al <i>ERP open-source Odoo</i> , el cual tiene muy poca difusión, representando una seria dificultad en la implementación de nuevas funcionalidades u otras técnicas de priorización.
<i>RPTackle</i> : una técnica de priorización de requisitos de <i>software</i> semiautomatizada.	(Hujainah, Binti Abu Bakar, Nasser, Al-haimi, & Zamli, 2021)	X	X	X		Los autores desarrollaron una solución propia que automatizó una técnica de priorización adoptada a sus necesidades. Sin embargo, por usar tecnologías propietarias como <i>C#</i> y <i>SQL Server</i> resulta dificultosa su adopción, además, al no usar un <i>software</i> base, necesariamente se deberá crear toda la plataforma, y no solo lo referente a la implementación del algoritmo, encareciendo el costo total del proyecto.

Elaboración propia.

Como se puede observar, se han identificado dos trabajos que realizan un enfoque teórico. Cuatro realizan un enfoque práctico. Y uno, de los autores (Achimugu, Selamat, & Ibrahim, 2016) quienes realizan una propuesta teórico y práctico, con su herramienta *ReproTizer*, habilitado para ser usada en un entorno real, por no contar con mecanismos de seguridad en su implementación, que son requisitos no funcionales muy importantes y se encuentran implementados en los CMS como *WordPress* (Nimesh Ratnayake, 2015). Las propuestas con lenguajes de modelado de (Flório, Lencastre, Pimentel, & Araujo, 2019) y (Leshob, Hadaya, & Renard, 2019), solo describen el uso del modelado, pero no indican como sería su aplicación en entornos distribuidos y con múltiples usuarios.

La seguridad en el *software* siempre debe ser una tarea de alta prioridad para desarrolladores y *stakeholders* (Sommerville, 2021).

Todos los autores realizaron una validación de su propuesta.

4.3.2 Principales Técnicas de Priorización de Requisitos

Las técnicas evaluadas son:

1. Asignación Numérica – AN.
2. Proceso Analítico Jerárquico – PAJ.
3. Priorización de Valor Orientado – PVO.
4. Voto Acumulativo – VA.
5. Árbol de Búsqueda Binaria – ABB.
6. Planificación del Juego – PJ.

4.3.2.1 **Asignación Numérica - AN**

Consiste en clasificar los requisitos en tres niveles: 'Obligatorio', 'Deseable' y 'No Esencial'. Cada *stakeholder* clasifica los requisitos especificados de acuerdo a su personal criterio. La Asignación Numérica se basa en la agrupación de los requisitos en diferentes categorías: alta, media y baja (Achimugu, Selamat, Ibrahim, & Naz'ri Mahrin, 2014).

También se puede usar categorías como 'Bajo', 'Medio' y 'Alto'. También se puede usar términos como críticos, estándar y opcionales (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018). Diferentes *stakeholders* tienen diferentes percepciones, agregar una descripción exacta de lo que para un grupo significa disminuye estos problemas (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). La prioridad de los requisitos dentro de ese grupo será similar (Kothawar & Guptha Vajrapu, 2018) y (Rida Asghar, Tabassum, Nazir Bhatti, & Ali Shah, 2017). Es una de las técnicas más populares (Rahim, Chowdhury, & Das, 2017).

Esta técnica está definida en IEEE Std. 830-1998 (Mustafa & Zainuddin, 2014).

4.3.2.2 **Proceso Analítico Jerárquico – PAJ**

El Proceso Analítico Jerárquico es una de las más estudiadas técnicas de priorización de requisitos (Abbas, Naila Jan, Saadatmand, & Daniel Sundmark, 2019). Además, es la técnica de priorización de requisitos más popular (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018). También se le denomina técnica de comparación por pares.

La idea básica es calcular las prioridades mediante la comparación de todos los pares únicos de requisitos para estimar su importancia relativa. En otras palabras, el analista que realiza la comparación tiene que decidir qué requisito es más importante, y en qué medida mediante una escala del 1 al 9. La comparación por pares no es

exclusiva de esta técnica, sino también se utiliza en otras técnicas como el ordenamiento burbuja y la lista de prioridades binarias (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). Como se puede ver en la **Tabla 4**, la parte necesaria de la matriz es la resaltada; lo demás es redundante.

Tabla 4

Ejemplo de Matriz creada con Proceso Analítico Jerárquico

	SR-1	SR-2	SR-3	SR-4	SR-5
SR-1	1	8	1/5	3	1
SR-2	1/8	1	1/5	1/7	1/7
SR-3	5	5	1	1	2
SR-4	1/3	4	1	1	1/2
SR-5	1	7	1/2	2	1

Nota. Se muestra una matriz de comparaciones por pares respecto al criterio, (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

Los pasos para su elaboración son los siguientes:

- Se crea una matriz de $n \times n$ ($n = \#$ de requisitos), ubicándose los requisitos en filas y columnas.
- Para cada par, se coloca un valor relativo a su importancia. En la celda transpuesta se pone a la inversa ($AB=4$, $BA=1/4$).
- Finalmente, se obtiene la relativa prioridad de cada requisito, hacer cálculos de matrices para obtener el vector resultante (por ejemplo: $A=50\%$, $B=12\%$ y $C=38\%$). En esta técnica la cantidad de resultados se incrementa exponencialmente debido al número de comparaciones. Estudios demuestran que no son los más adecuados para un número grande de requisitos. Asimismo, otros investigadores afirmaron que esta técnica presenta limitaciones si los *stakeholders* se encuentran en entornos distribuidos.

Para n requisitos, se necesitarán $n(n-1) / 2$ comparaciones. Por ejemplo, si el número de requisitos es diez, AHP realizará cuarenta y cinco comparaciones (Yaseen, Mustapha, Ur Rahman, Khan, & Kamal, 2019). En esta técnica la cantidad de resultados

se incrementa exponencialmente debido al número de comparaciones.

En resumen, esta técnica utiliza una matriz de comparación de pares para calcular la importancia relativa de cada uno de los requisitos (Achimugu, Selamat, Ibrahim, & Naz'ri Mahrin, 2014).

4.3.2.3 Priorización de Valor Orientado – PVO

En esta técnica, los requisitos se priorizan según las calificaciones de los *stakeholders* y se combinan con los valores *core* asignados por la organización. Los valores *core* son asignados por los ejecutivos de la organización (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019).

La priorización se logra mediante la atribución de valores preferenciales a requisitos en forma de matriz de decisión, y representada en un gráfico de resultados (Achimugu, Selamat, Ibrahim, & Naz'ri Mahrin, 2014).

4.3.2.4 Voto Acumulativo – VA

El Voto acumulativo - VA, también conocido como técnica de los cien puntos o prueba de los cien dólares, es un método simple y directo. El concepto implica que los *stakeholders* que participan en el establecimiento de prioridades utilizando un número de unidades imaginarias (100 puntos, 100 dólares, etc.), las cuales son distribuidas entre los requisitos a priorizar. Es una técnica simple de usar y se considera una de las más precisas y rápidas (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018). Es utilizado en diversos estudios de priorización en ingeniería de *software*.

Cuando todos los participantes han asignado sus puntos, una persona calcula y suma los puntos que cada requisito tiene, y calcula los resultados de la votación

acumulativa. De esta manera, la cantidad de puntos asignados a un requisito representa la preferencia relativa del *stakeholder* (y por tanto el establecimiento de su prioridad) en relación con otros. Los puntos pueden ser distribuidos de la manera que el *stakeholder* decida: asignando todos los puntos a un requisito, distribuyéndolos de manera equitativa, etc. El número de unidades que se le asigna a cada requisito representa la prioridad para este requisito (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018). Al Voto acumulativo a veces se lo conoce como Voto Proporcional, debido a que la cantidad de puntos asignados a un requisito representa la prioridad relativa de ese específico requisito con relación a otros. El término ‘proporcional’ refleja el hecho de que, si la cantidad de unidades asignadas a un problema se divide por el constante número de unidades disponibles para cada una de las partes interesadas, el resultado se convierte en una proporción entre cero y uno. Para obtener una lista resultante de y priorizada de los requisitos, se utiliza el mecanismo de ‘votación’ para identificar su prioridad de unos frente a otro. Así, a los *stakeholders* se les asigna el mismo número de puntos de prioridad con los cuales se pueden ‘gastar’ en los distintos requisitos, obteniéndose una lista de requisitos y la importancia relativa que le asignó a cada requisito (desde su punto de vista) al darle más o menos puntos de prioridad (Pressman & Maxim, 2019). Los requisitos con más alta votación reciben mayor prioridad (Yaseen, Ibrahim, & Mustapha, 2019). Otro autor (Kyosev, 2014) denominó a esta técnica Sistema Multivotos.

En resumen, la técnica consiste en darles a los *stakeholders* US\$ 100 para ser gastados en los requisitos. En comparación con la técnica de proceso analítico jerárquico (PAJ), es más rápida y más fácil de aprender y utilizar.

4.3.2.4.1 **Ejemplo de Voto Acumulativo con varios *stakeholders***

Se tiene a *stakeholders* que buscan comprar algunos alimentos. Con diferentes

preferencias de compra, deciden usar la técnica de VA para definir que adquirir. A cada uno de los amigos se le asigna cien puntos para que elijan cuatro alimentos (leche, té, café, jugo). Así, cada uno puede asignar diferente cantidad de puntos en su compra. La compra final se decide sumando las prioridades de cada *stakeholder*.

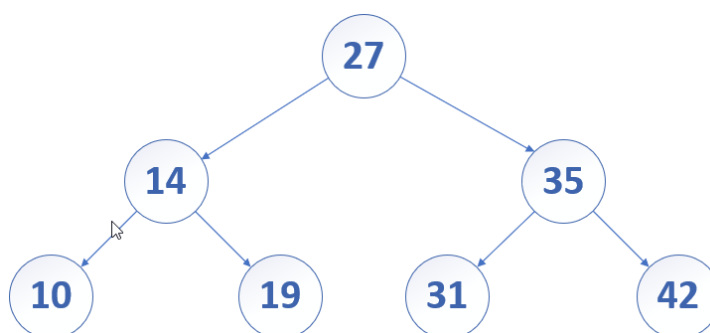
4.3.2.5 Árbol de Búsqueda Binaria – ABB

Un árbol binario – ABB, es un árbol en el que cada nodo tiene como máximo dos hijos un nodo está vacío, tiene uno o dos nodos secundarios. Se deben considerar a los elementos de un conjunto como los requisitos candidatos. Cada nodo hijo es en sí mismo un nodo raíz para sus nodos hijos (Saghir & Mustafa, 2016).

La creación de un ABB con requisitos que representan los elementos de un conjunto se constituye en una técnica para priorizar requisitos de *software*. Es una técnica usada para priorizar un proyecto con un número de requisitos grande (Ibriwesh, Ho, Chai, & Tan, 2019), ver **Figura 16**.

Figura 16

Árbol de Búsqueda Binaria



Nota. Ningun nodo tienen más de dos subnodos, (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

En resumen, esta técnica analiza los requisitos capturados y evalúa los requisitos en un orden jerárquico -relación padre-hijo- (Achimugu, Selamat, Ibrahim, & Naz'ri Mahrin, 2014).

4.3.2.6 Planificación del Juego – PJ

Es una variación de la técnica de Asignación Numérica, donde los *stakeholders* agrupan los requisitos en tres grupos: ‘Aquellos sin los cuales el sistema no funcionará’, ‘Aquellos menos esenciales, pero proveen un significativo valor al negocio’ y ‘Aquellos que sería bueno tener’ (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). Dado que cada requisito se prioriza por separado y no se hacen comparaciones, se adapta bien a proyectos con grandes números de requisitos. Esta técnica es usada en Programación Extrema – XP (Yuan & Hengyuan, 2019). Permite a los *stakeholder* categorizar a los requisitos en tres niveles o grupos (Achimugu, Selamat, Ibrahim, & Naz’ri Mahrin, 2014).

4.3.2.7 Otras técnicas de priorización de requisitos

Además de las evaluadas, existen otras técnicas de priorización de requisitos pero que están consignadas en la bibliografía de muchas investigaciones o son implementados a través de casos de aplicación y son las siguientes:

- Diez Mayores Requisitos: Cada *stakeholder* selecciona sus diez más importantes requisitos. la técnica elige entre todos los requisitos disponibles, los diez requisitos más importantes, en lugar de priorizando cada uno internamente. Este enfoque es adecuado cuando los stakeholders tienen similares prioridades (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). También se llama La técnica de los Diez (Rahim, Chowdhury, & Das, 2017). Tiene como limitaciones que pueden surgir conflictos o insatisfacción en los *stakeholders* (Tong, Zhuang, Guo, & Ma, 2014).
- Despliegue en Función de la Calidad: Analiza y evalúa los requisitos y los registra

en una base de datos con las características de los productos, componentes, procesos y calidad. Para aplicarse, se convierten los originales requisitos en especificaciones del producto asignándole a cada ítem y sus características un valor escalar.

Proporciona un marco de referencia que garantiza una adecuada comunicación entre *stakeholders* y desarrolladores (Yuan & Hengyuan, 2019).

Resulta dificultoso hacer cambios a los requisitos, ya que esto implicaría realizar una nueva conversión y un recálculo de los resultados; además, no permite gestionar la dependencia entre requisitos (Tong, Zhuang, Guo, & Ma, 2014).

- Ordenamiento Burbuja: Está muy relacionado con el HCV, ya que ambas técnicas utilizan la ordenación por pares y requieren $n \times (n-1) / 2$ comparaciones. Sin embargo, en el ordenamiento burbuja solo tiene que determinarse cuál de los dos requisitos es más prioritario, ordenándolos por la prioridad, continua con el requisito siguiente, replicando el proceso hasta haber comparado todos los requisitos (Rida Asghar, Tabassum, Nazir Bhatti, & Ali Shah, 2017).

Es uno de las técnicas más sencillas para ordenar elementos (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018), ver **Figura 17**.

Figura 17

Ordenamiento Burbuja

54	26	93	17	77	31	44	55	20	Exchange
26	54	93	17	77	31	44	55	20	No Exchange
26	54	93	17	77	31	44	55	20	Exchange
26	54	17	93	77	31	44	55	20	Exchange
26	54	17	77	93	31	44	55	20	Exchange
26	54	17	77	31	93	44	55	20	Exchange
26	54	17	77	31	44	93	55	20	Exchange
26	54	17	77	31	44	55	93	20	Exchange
26	54	17	77	31	44	55	20	93	Exchange

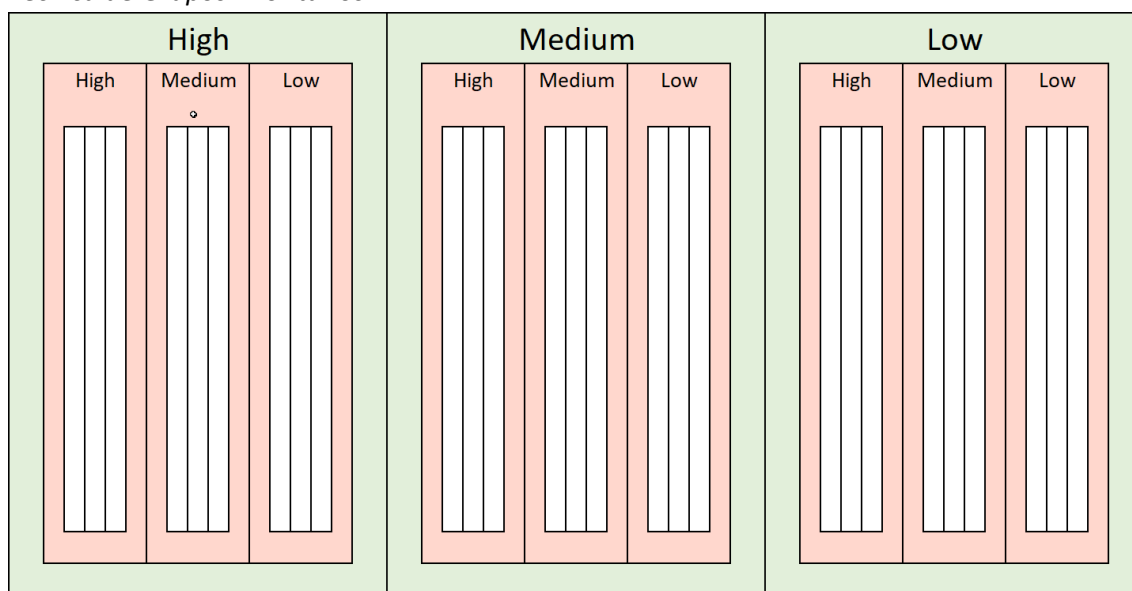
Nota. Si los elementos comparados no se encuentran ordenados, se realiza la permutación, (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

- Voto Acumulativo Jerárquico – VAJ: De manera similar a VA, la priorización con VAJ está orientada a asignar puntos a los requisitos. Sin embargo, cuando se utiliza el VAJ, no todos los requisitos se priorizan al mismo tiempo. En lugar de ello, los requisitos se clasifican en diferentes niveles de jerarquías, y la priorización se realiza solo dentro de cada jerarquía. El Voto Acumulativo Jerárquico (AVC) es un método híbrido, que amplía el CV con algunas características del PAJ. Esta técnica no se puede personalizar (Rahim, Chowdhury, & Das, 2017).
- Árbol de Expansión mínima – AEM: El enfoque de árbol de expansión mínimo se considera la técnica más rápida (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018). La priorización apoya el proceso de toma de decisiones para el plan de ejecución del proyecto en una variedad de campos que van desde la economía, el ejército, la política médica, etc., hasta las actividades del día a día (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019).

- Grupos Prioritarios: La técnica como resultado final sino una lista clasificada de requisitos, a pesar de su nombre, no produce grupos de requisitos (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018). El principio básico es asignar a cada requisito una categoría: alta, media y baja prioridad. Como se puede ver en la **Figura 18**, los grupos se dividen a su vez a los subgrupos (de forma iterativa). La técnica de grupos prioritarios es bastante lenta de realizar y difícil de utilizar. La clasificación se realiza de acuerdo con las preferencias de los *stakeholders* y el juicio de expertos. (Hujainah, Binti Abu Bakar, Nasser, Al-haimi, & Zamli, 2021).

Figura 18

Técnica de Grupos Prioritarios



Nota. Se realiza un ordenamiento de cada subgrupo, (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

- Top 10 requisitos: Los *stakeholders* pueden elegir solo 10 requisitos, los cuales tendrán la misma prioridad. La técnica permite cierta flexibilidad, así, si hubiese algún requisito de importante interés, para algunos *stakeholders*, estos pueden coordinar ente si para no salir afectados (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019).

- MoSCoW: Para el éxito de la técnica se debe precisar las definiciones con los *stakeholders* (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). Es usada en desarrollo ágil (Yuan & Hengyuan, 2019). Según el mecanismo de *MoSCoW*, la lista de requisitos se puede clasificar en las siguientes cuatro categorías o grupos de prioridad (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018) y (Rida Asghar, Tabassum, Nazir Bhatti, & Ali Shah, 2017), ver **Figura 19**:
 - M: Debe tener. Los requisitos deben estar incluidos en el proyecto. No cumplir con estos requisitos significa que todo el proyecto sería un fracaso.
 - S: Debería tener. Una funcionalidad de alta prioridad que no es fundamental para su lanzamiento. Pero se supone que es importante y de gran valor para los usuarios. Tales requisitos ocupan el segundo lugar en la lista de prioridades.
 - C: Podría tener. Este grupo contiene el requisito deseable pero no necesario. Estos requisitos son menos importantes que el del grupo "debería tener".
 - W: no se hará (o tendría). Un requisito que no se implementará en el desarrollo actual pero que puede incluirse en una etapa futura.

Figura 19

Técnica MoScow

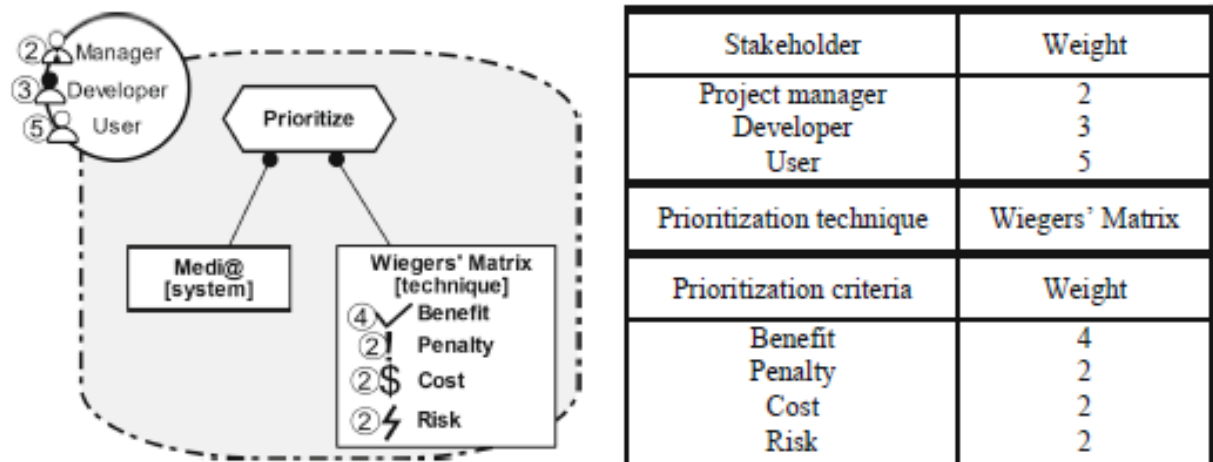
Mo	Must Have The Most vital things you can not live without it
S	Should Have Thing you consider as important, but not vital
Co	Could Have The nice to have
W	Won't Have Thing that provides little to no value you can give up on

Nota. La clasificación de los elementos se realiza en un único paso, (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018).

- Técnica *Wieger*: El método o matriz de *Wieger* calcula la prioridad de los requisitos mediante dividir el valor del requisito entre la suma del costo y riesgos técnicos involucrados. El valor de un requisito, es estimado por el *stakeholder* y la penalización (se produce cuando no se implementa el requisito). Se acostumbra que distintos stakeholders tengan distinto peso al momento de priorizar, ver **Figura 20** (Flório, Lencastre, Pimentel, & Araujo, 2019). Los atributos de un requisito suelen ser valor, costo, riesgos y penalización se evalúan en una escala [1-9] (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019). Este modelo admite varias docenas de requisitos antes de que sea inmanejable (Rahim, Chowdhury, & Das, 2017).

Figura 20

Matriz de Wieger



Nota. Para obtener los valores relativos, se debe asignar pesos a cada uno de los criterios de priorización, (Flório, Lencastre, Pimentel, & Araujo, 2019).

- Ranking Costo – Valor: En la técnica, los requisitos se priorizan en función de su costo y se le asignan valores relativos. Aunque fácil de usar y requiere menos tiempo, ignora otros aspectos como el riesgo y la volatilidad. Además, no admite personalización (Rahim, Chowdhury, & Das, 2017). Aunque es un enfoque relativamente fácil de implementar, pero es relativamente costoso en tiempo y no es de alta precisión (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019).

4.3.3 Críticas a las Técnicas de Priorización

Los problemas identificados pueden resumirse en:

- Las técnicas existentes no proporcionan una solución escalable cuando se tiene un numero grande de requisitos, como en proyectos grandes (de orden de varios cientos o miles).
- La mayoría de las técnicas requieren de mucho tiempo.
- Los resultados son factibles de contener errores o tener márgenes de

imprecisión.

- Los resultados no suelen concordar, cuando se vuelve a repetir la prueba.
- Permiten realizar la priorización a pequeña escala o proyectos de alcance limitado, pero dependiendo de la técnica, con errores.

4.4 Herramientas de Software

Las Herramientas de *Software CASE* de la Ingeniería de requisitos a la captura, análisis, modelado, gestión, pruebas, validación y de gestión de trazas de requisitos.

El flujo de trabajo de las distintas de las herramientas varía. Permiten construir distintos modelos gráficos, en determinado lenguaje grafico como *UML* o algún lenguaje de dominio, que muestran aspectos dinámicos y estáticos del sistema. Mediante los modelos obtenidos, se realizarán las siguientes actividades del ciclo de vida de desarrollo de *software* (Pressman & Maxim, 2019).

Las herramientas *CASE* permiten automatizar el proceso de análisis y diseño. Actualmente existen bastantes productos, de diferentes categorías. Estas herramientas abarcan desde modelado de productos de ciclo de vida, gestión de proyectos, documentación, simulación, programación, diagnóstico de enfermedades, análisis anatómico, seguridad, motor de juegos, educación, aeronáutica, capacitación en simulación, diseño de interfaces y otras categorías. (Udjaja, 2018).

4.4.1 Herramientas Representativas

Las herramientas presentadas a continuación se enfocan en la gestión de requisitos:

- *CaseSpec*: De Goda Software Inc., Virginia, EEUU (<http://www.analysttool.com/>).

- *EasyRM*: Desarrollado por *Cybernetic Intelligence GMBH* (www.easy-rm.com), elabora un tesoro específico del proyecto con la descripción y características detalladas de los requisitos.
- *Enterprise Architect*: De *Sparx Systems*, incorpora la gestión de requisitos (*Sparx Systems*, 2021).
- *IBM Telelogic DOORS (Dynamic Object Oriented Requirements System)*: Constituye un repositorio de requisitos.
- *IBM Rational Requirements Composer*: El cual es una aplicación web (<http://www-01.ibm.com/software/awdtools/rrc/>).
- *IRqA (Integral Requisite Analyzer)*: De *Visure Solutions*, se define como una herramienta para la definición, gestión y pruebas, incluye soporte gráfico.
- *OnYourMark Pro*: Promovido por *Omni-Vista* (www.omni-vista.com), incluye una base de datos de requisitos y su relación entre ellos, y permitiendo a los *stakeholders* analizando su relación y el manejo de tiempos y costos.
- *Rational RequisitePro*: Desarrollado *IBM* (www.rational.com), permite a los *stakeholders* desarrollar un repositorio de requisitos en archivos de *MS Word*, representando las dependencias entre estos y los organiza, priorizando y haciendo una trazabilidad.

4.5 Priorización de Requisitos en Entornos Distribuidos

Hay enfoques que fomentan la participación de los *stakeholders* (como los usuarios finales) en la obtención de requisitos, la priorización [...] Sin embargo, estos enfoques no funcionan en forma adecuadas en contextos no tradicionales, tales como los dispositivos móviles o el *cloud computing* [...]. Esto se debe al hecho de que, en estos proyectos, necesitamos involucrar a un gran número de *stakeholders* de diferentes perfiles, distribuidos globalmente y potencialmente anónimos entre sí. Aunque la idea subyacente de involucrar a *stakeholders* críticos sigue siendo válida en esos entornos, hay una falta de novedosos y adecuados métodos y enfoques de ingeniería de requisitos

para usuarios finales.

La gestión, el desarrollo y el mantenimiento de productos de *software* se han dispersado desde una única ubicación de desarrollo a varias ubicaciones en todo el mundo, la mayoría de las veces reconocidas como Ingeniería de *Software Global* (GSE). El desarrollo de software está geográficamente disperso y permite que los *stakeholders*, desarrolladores, gerentes y otros, permanezcan en lugares distantes durante todo el proceso de desarrollo de software, incluyendo personas de países y culturas distintos (Saghir & Mustafa, 2016).

4.6 Herramientas de Colaboración para obtener Requisitos

Las herramientas de colaboración permiten la cooperación de todas las partes interesadas en varias fases del proceso de *software*, incluyendo en la ingeniería de requisitos. La intensa comunicación de la elicitación de requisitos exige un alto nivel de colaboración.

4.6.1 El Uso de Wikis como Herramienta de Interacción Colaborativa

Una *wiki* es un sitio web cuyas páginas pueden ser editadas por múltiples usuarios a través del navegador. En una *wiki* cualquier usuario puede acceder a la edición del contenido y participar en la discusión comentando sobre las distintas secciones, lo que lo hace vulnerable en cuanto a la seguridad y garantía de la veracidad de la información se refiere (Nivela Cornejo, Echeverría Desiderio, & Sánchez Villarroel de Gallardo, 2021). Las *wikis* son una buena herramienta para coordinar contenido [...]» (Sevilla, Zapata, Torres, & Collazos, 2014).

El proceso de interacción colaborativa de las *wikis* ha sido aplicado en todo el ciclo de vida de desarrollo de *software*. Con las *wikis*, existen problemas para procesar información, como: datos no estructurados, reutilización del conocimiento dificultosa, limitadas funciones de búsqueda (las búsquedas se limitan al título de la página). Herramientas basadas en *wikis*, como *SoftWiki*, son muy adecuadas para los usuarios finales que no están familiarizados con las prácticas de la ingeniería de requisitos, tales como la elicitación y la priorización (Seyff, Todoran, Caluser, & Singer, 2015).

Los *wikis* se pueden utilizar para crear documentos que reflejen el conocimiento compartido por un grupo de aprendizaje (Sánchez Medero & Pastor Albaladejo, 2020).

4.6.1.1 Wikis en un Entorno Distribuido y Colaborativo

Las *wikis* son herramientas para trabajar en entorno distribuido y en modo colaborativo, y sirven para resolver problemas detrás de la obtención de requisitos en dicho entorno. Un buen ejemplo del éxito del concepto de *wiki* es *Wikipedia*. Las *wikis* se han utilizado en el desarrollo de *software* en proyectos globales para comunicarse, coordinar, rastrear, discutir, probar y documentar.

La Ingeniería de Requerimiento Distribuida necesita una plataforma que soporte la colaboración en forma efectiva y eficiente entre los *stakeholders* (Sevilla, Zapata, Torres, & Collazos, 2014).

4.6.2 Requisitos basado en Foros

Permiten la discusión o intercambio de opiniones entre los *stakeholders*. También incluye un sistema de encuestas o votación. Se asume que los *stakeholders* aprovechan de la comunicación asíncrona, lo que les permite analizar el problema,

analizar otros puntos de vista y proponer nuevas ideas para contribuir nuevamente. Los foros se publican, almacenan y conservan en un repositorio, se ordenan siguiendo temas de discusión específicos. Están disponibles en varias plataformas. Se utilizan para aplicaciones (Biasutti, 2017).

4.7 Sistemas de Gestión de Contenidos - CMS

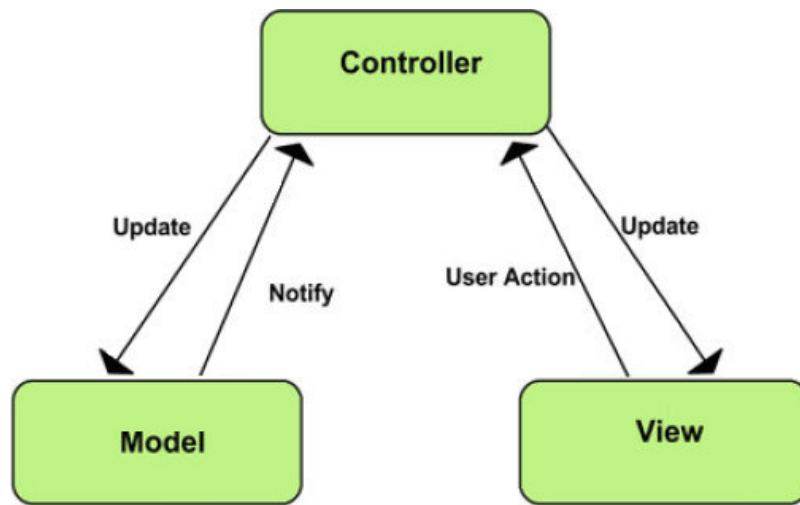
Los CMS permiten la creación y administración de contenidos en páginas *web*. La mayoría de CMS son de libre distribución, bajo licencia GPL (Licencia Pública General de GNU - *General Public Licenses*), por lo general se apoyan en grandes comunidades de software (Nivela Cornejo, Echeverría Desiderio, & Sánchez Villarroel de Gallardo, 2021). Definen procedimientos para gestionar el flujo de trabajo en un entorno colaborativo, estos procedimientos pueden estar en modo manual o con un proceso automatizado. Los CMS, *en principio*, no han sido creados para permitir realizar las actividades de un proceso de priorización de requisitos.

4.7.1 Arquitectura del Diseño del Sistema

La arquitectura de software más utilizada es Modelo – Vista – Controlador (MVC). Es un concepto básico para desarrollar interfaces de usuario, en todas las plataformas y aplicaciones que tienen *front-end*. En la estructura MVC el usuario ingresa la URL en el navegador, el controlador recibe la solicitud, luego se transfiere al modelo para procesar todas las solicitudes y los resultados del proceso se envían de vuelta al controlador, después de que el usuario puede ver el servicio solicitado (Udjaja, 2018).

Figura 21

Modelo-Vista-Controlador



Nota. Como interactúan los componentes del patrón, (Mani Subedi, 2019).

4.7.2 CMS de Código Abierto

El término código abierto hace referencia a la factibilidad que se tiene para acceder a su código fuente y modificarlo, aunque con algunas consideraciones. Mientras el *software* propietario es creado, distribuido, y mantenido por una empresa desarrolladora, en el *software* de código abierto estas tareas de creación, modificación y actualización son manejadas por la misma comunidad de desarrolladores y usuarios.

4.7.3 CMS Comerciales

El *software* comercial no tiene la desventaja de los CMS de código abierto. En los CMS comerciales el vendedor tiene el control sobre los módulos. De ese modo, crea un entorno de usuario más amistoso y seguro, debido a que son CMS comerciales la comunidad no tiene acceso al código fuente; además, no está permitido realizar modificaciones. Por lo general, tienen un menor número de actualizaciones de seguridad que los CMS de código abierto. Algunos CMS comerciales proporcionan el código fuente, bien diseñado y con buena documentación, con una arquitectura

extensible que permite a desarrolladores independientes crear extensiones y módulos que puedan ser incorporados.

Algunos *CMS* comerciales son buenos para propósitos específicos. Por ejemplo, *Microsoft SharePoint* ha sido diseñado para desarrollar de forma rápida una Intranet, especialmente para clientes que utilizan productos *Microsoft*. De otro lado, usar *SharePoint* no es adecuado para desarrollar sitios de Internet públicos, ya que sus herramientas son orientadas a la Intranet.

4.8 Proceso de Priorización

La priorización de requisitos consiste en determinar qué conjunto de requisitos serán implementados, en qué orden y cuales no se desarrollarán. Después de determinar los requisitos, se repite el ciclo según los nuevos requisitos o los cambios en los antiguos. (AbdElazim, Moawad, & Elfakharany, 2020).

Un proyecto de software puede fallar o fallar si no se realiza la priorización de requisitos de la forma adecuada (Rida Asghar, Tabassum, Nazir Bhatti, & Ali Shah, 2017).

4.9 Priorización de Requisitos en Entornos Distribuidos

Existe una tendencia a que los equipos de proyectos de *software* estén constituidos por personas que residen y trabajan distribuidos geográficamente. Existen diversos motivos: optimización de costos, disponibilidad de recursos humanos en lugares geográficamente distantes, se dispone de periodos más largos de tiempo disponibles en función a las zonas horarias, crecimiento de la demanda de *software* a nivel mundial, entre otros.

En la revisión de la bibliografía, algunos autores abordan acerca del rol del

software colaborativo en estos procesos en entornos distribuidos y tratan sobre aspectos referentes a la comunicación (Sevilla, Zapata, Torres, & Collazos, 2014).

4.10 Priorización de Requisitos con Voto Acumulativo

El voto acumulativo es una técnica de priorización usada en la toma de decisiones en varias áreas, como ingeniería de requisitos, análisis de impacto o mejoras de procesos.

La priorización es realizada por los *stakeholders* (usuarios, desarrolladores, consultores, representantes de *marketing* o clientes), quienes responden de cuestionarios diseñados adecuadamente, bajo diferentes perspectivas o posiciones.

Estos aspectos hicieron que se requiera la adopción de un esquema donde los *stakeholders* expresan sus preferencias relativas sobre los requisitos de un modo sistemático y controlado. Los *stakeholders* pueden asignar cero puntos a un requisito, indicando que este ítem no tiene importancia alguna. En estudios empíricos con el fin de investigar los diferentes enfoques que realizan los *stakeholders* en la priorización de requisitos, los resultados obtenidos se codifican y analizan estadísticamente con el fin de encontrar los puntos de vista de los *stakeholders* hacia los requisitos. El análisis estadístico de estos datos puede revelar diferencias significativas, tendencias, desacuerdos y coincidencias entre los *stakeholders*, y, por tanto, una herramienta para la toma de decisiones. En términos estadísticos, cada *stakeholder* produce un vector de valores escalares con un resultado de suma fija. Como ya se ha mencionado, con una simple división cada vector de datos es transformado a un vector de composición de proporciones que suman uno, por lo que se debe utilizar una especial técnica de transformación de datos y métodos estadísticos diseñados para el uso de las proporciones.

4.10.1 Investigaciones actuales sobre Voto Acumulativo

En la actualidad, existen numerosas investigaciones que tienen como objeto de estudio la técnica del Voto Acumulativo. En la Tabla 3 se muestra las técnicas más populares identificadas por los usuarios, según la investigación de (Noor Bor Hazlini, Zulzalil, Hassan, & Mohd Ali, 2019).

Tabla 5

Técnicas de Priorización más populares

Técnica	Porcentaje de Popularidad
Proceso Analítico Jerárquico	50.00%
Voto Acumulativo	41.67%
Modelo Kano	16.67%
Priorización de Valor Orientado	16.67%

Nota. Resultados obtenidos luego de realizar una revisión sistemática de la literatura, (Noor Bor Hazlini, Zulzalil, Hassan, & Mohd Ali, 2019).

Luego de una investigación en la que realizan una comparación de ocho principales técnicas, incluyendo Voto Acumulativo, los autores (Hudaib, Masadeh, Haj Qasem, & Alzaqebah, 2018) la clasifica como cuarta con más facilidad de uso.

4.11 Sistemas de Gestión de Contenidos – CMS

Los CMS son aplicaciones que gestionan todos los elementos de la gestión de contenidos, desde la base de datos, plantillas, la presentación de la página y sus recursos (So, 2018). El "contenido" incluye cualquier cosa, desde una noticia, una publicación de blog, un video o una fotografía, hasta un podcast, un artículo o una descripción de un producto que está vendiendo. En términos más generales, el contenido es cualquier combinación de texto, gráficos, fotografías, audio y video que representa algo que los visitantes de su sitio leerán, verán y escucharán. (Tomlinson, 2017). Parte del presente

trabajo de investigación consistió en elegir las herramientas *CMS* como alternativas a los foros y las herramientas basadas en *wiki*. Para una adecuada selección, se desarrolló el ANEXO B, en el cual se presenta una evaluación. Previamente, se eligieron tres herramientas candidatas en base al número de sitios web activos.

Como resultado de la evaluación, se eligió *WordPress* como herramienta *CMS* para la priorización de los requisitos.

4.11.1 *Drupal*

Drupal es un *CMS* desarrollado en *PHP*. *Drupal* es un *software* libre, de código abierto, con licencia *GPL (GNU General Public License)* (Tomlinson, 2017). Tiene una amplia comunidad de desarrolladores y usuarios que participan en su mantenimiento. *Drupal* permite fácilmente publicar, gestionar y organizar complejos y diversos contenidos de uno o más sitios *web*. Permite el uso de plantillas y módulos, además, permite la interacción con *Apis*. Cuenta con diversos mecanismos de seguridad (Sipos, 2020). Posee una comunidad muy activa, y la aplicación con frecuencia libera nuevas versiones y actualizaciones (Mayekar, 2017).

4.11.2 *Joomla*

Joomla es uno de los más populares sistemas de gestión de contenidos. La instalación por defecto de *Joomla* está lista para gestionar contenidos, permite el uso de extensiones. Está diseñado para que los usuarios publiquen contenido en línea: noticias, blogs, fotos, productos, documentos, eventos (Burge, 2017). Existen cinco extensiones básicas disponibles para *Joomla*: componentes, módulos, *plugins* y plantillas.

4.11.3 WordPress

WordPress es un sistema de gestión de contenidos de distribución libre desarrollado en *PHP*, usa la base de datos *MySQL* y *Apache*, bajo licencia *GPL (GNU General Public License) v2.0* (Cabot, 2018). Entre sus características incluyen una arquitectura y un sistema de *plugins*. Se encuentra preparado para dispositivos móviles. Está optimizado para *SEO*⁴ y para que el sitio aparezca en buscadores como *Google*, *Bing*, *Yahoo!* y otros. El diseño o presentación puede ser cambiado fácilmente mediante temas. Un tema le permite cambiar el aspecto y la funcionalidad, sin alterar el contenido del sitio. Cada sitio *WordPress* requiere de al menos un tema y este debe ser diseñado respetando los estándares con un estructurado *PHP*, *HTML* válido y *CSS*.

La arquitectura de *plugins* de *WordPress* permite a los usuarios extender las características y funcionalidades del sitio *web*. Cada *plugin* provee funcionalidades personalizadas y hace posible que los usuarios satisfagan sus necesidades específicas. Asimismo, existen soluciones nativas para móviles en *WebOS*, *Android*, *iOS* (*iPhone*, *iPod Touch*, *iPad*), *Windows Phone* y *BlackBerry*, permitiendo un conjunto de funcionalidades como agregar nuevos post y páginas, comentarios, moderar comentarios, entre otros (WordPress Foundation, 2019).

4.11.4 Plugins WordPress

Un *plugin* es un programa con una o más funciones, escritos en un lenguaje de programación que añaden un conjunto específico de funcionalidades o servicios y se integran a la aplicación a través de puntos de acceso proporcionadas a través de una *API* de la aplicación (WordPress Foundation, 2021, pág. Plugin API). En *WordPress* están escritos en *PHP*. Una *API* es una interfaz de programación con que cuenta la aplicación. Los *plugins* son una forma de extender, mejorar o añadir nuevas funcionalidades a las

⁴ *Search Engine Optimization*, Posicionamiento en buscadores.

ya existentes en *WordPress* (WordPress Foundation, 2021, pág. Plugins) y (WordPress Foundation, 2021, pág. Managing Plugins).

La mayoría de los usuarios no requieren *plugins*, o requieren unos pocos, como para manejar el *spam* o personalizar listas de *posts*. Otros usuarios requieren variedad de opciones que los *plugins*, proveen tales como reportes del clima, contadores de palabras, sistemas de ratings y más. El uso de *plugins* es opcional, basados en las necesidades del usuario, y es por ello que no se incorporan en el núcleo de *WordPress* (WordPress Foundation, 2021, pág. Managing Plugins).

En el sitio *web* de *WordPress* existen más de 59,000 *plugins* gratuitos para descargar (WordPress Foundation, 2021, pág. Plugin Directory). Los *plugins* ofrecen funciones personalizadas y características que cada usuario puede crear una nueva versión o mejora de acuerdo con sus propias necesidades.

Los *plugins* en *WordPress* están disponibles desde diferentes fuentes. Han sido diseñados por voluntarios y por lo general son gratuitos para el público. Los *plugins* alojados en el sitio *web* de *WordPress* son evaluados extensivamente y son considerados seguros (WordPress Foundation, 2021, pág. Plugin Directory).

5. Capítulo V: APORTE

En la presente tesis, se propone la elaboración de un modelo de priorización de requisitos, uniendo diferentes enfoques teóricos que incluían una técnica de priorización⁵. Se utiliza el término ‘modelo’, por ser la integración de los elementos teóricos y metodológicos.

Se realizará la integración de distintos elementos heterogéneos, tales como: el desarrollo de una técnica de priorización basada en técnicas de priorización existentes, una relación de modelos para el proceso de desarrollo de *software*, la identificación de una herramienta de *software* colaborativo, la identificación de un componente que permita capturar las prioridades individuales de los *stakeholders* y que soporte la técnica propuesta, y el desarrollo de una herramienta que implemente la técnica de priorización desarrollada. Al integrar estos elementos, se obtendrá un modelo que tenga como propósito general proponer una solución al problema identificado en el presente trabajo de investigación.

5.1 Modelo Propuesto

La propuesta incluye elementos diversos, heterogéneos entre sí. Todos ellos son necesarios para que el modelo que se propone sea factible en su implementación, en un ambiente real; es decir, que sea viable y funcional. Los elementos son:

1. Técnica de priorización: También llamada método, son un conjunto de procedimientos que permiten realizar una determinada tarea.
2. Modelo de desarrollo de *software*: Los distintos modelos para el proceso de desarrollo de un *software*, que son compatibles con las técnicas de priorización,

⁵ En su investigación (Kyosev, 2014) los denomina métodos.

es decir, que permiten su implementación.

3. Licencia de *software*: Que permita hacer uso de una herramienta colaborativa que trabaje en un entorno se requiere de una licencia de *software* que le otorgue al usuario ciertos permisos. Si no, se infringiría la Ley de derechos de autor.
4. Herramienta colaborativa: El modelo propuesto en la presente tesis requiere de una herramienta *CMS* alternativa a las propuestas basadas en *wikis* y foros.
5. Componente para la herramienta colaborativa: La propuesta inicial contempla el uso de una herramienta *CMS*, sin embargo, para disponer de ciertas funcionalidades, se recurre al uso de los denominados componentes. Estos componentes permiten extender las funcionalidades de las herramientas *CMS*, haciendo posible realizar la priorización de requisitos, usando la técnica seleccionada.

Figura 22

Los cinco elementos del modelo propuesto



Nota. Todos los elementos deben ser validados, elaboración propia.

5.2 Implementación del Modelo Propuesto

A continuación, se describe y sustenta cada uno de los elementos que constituyen el modelo de priorización de requisitos con herramientas CMS.

5.2.1 Técnica de Priorización

La técnica de priorización permite establecer el orden relativo entre los distintos elementos de un conjunto; sin embargo, se debe seleccionar la técnica adecuada para formar parte del modelo, entre las técnicas identificadas.

Se deben evaluar los métodos de priorización presentados en el estado del arte para realizar la selección.

Basados en la investigación de (Kothawar & Guptha Vajrapu, 2018), se tomará como técnicas candidatas a las identificadas en dicha investigación, ver **Tabla 6**.

Otro criterio de preselección es el referente a la facilidad de uso, para identificar cuál será la lista de técnicas candidatas, lista corta o preselección. Para ello se utiliza como criterio el resultado de la investigación de (Saghir & Mustafa, 2016).

Tabla 6

Aparición de las técnicas en los artículos de investigación en la India

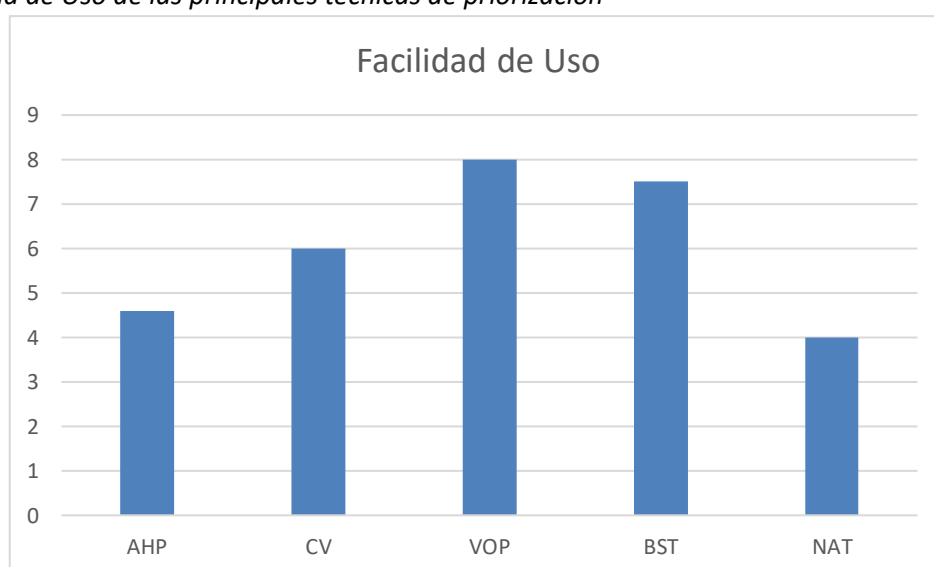
1.	Analytical hierarchy Process (AHP)	45.98%
2.	Quality Functional Deployment (QFD)	17.24%
3.	Planning Game	21.84%
4.	Binary Search Tree (BST)	14.94%
5.	Cumulative Voting (\$100 test)	18.39%
6.	Cost-Value Approach	14.94%
7.	Top Ten Requirements	5.75%
8.	Win Win	9.20%
9.	Pair-wise Comparisons	10.34%

10.	Priority Groups	8.05%
11.	Numerical Assignment	13.79%
12.	Ranking	5.75%
13.	Value-oriented prioritization	1.15%
14.	Software Engineering Risk Understanding and Management	17.24%
15.	Minimal Spanning Tree	9.20%
16.	Wiegeiv Matrix approach	14.94%
17.	Value-based Prioritization	3.45%

Nota. Resultados obtenidos luego de analizar 87 estudios de priorización de requisitos, (Kothawar & Guptha Vajrapu, 2018).

Figura 23

Facilidad de Uso de las principales técnicas de priorización



Nota. Resultados obtenidos a partir de la investigación de los autores, (Saghir & Mustafa, 2016).

Las técnicas de proceso analítico jerárquico (AHP), voto acumulativo (CV) y asignación numérica (NAT) son las técnicas de más utilizadas (Rashdan, 2021).

A pesar de que los métodos AHP y CV demostraron ser técnicas muy confiable y ampliamente utilizadas entre los investigadores, presentan dificultades de escalabilidad y requiere mucho tiempo, debido a que necesitan más comparaciones cuando aumenta el número de requisitos (Noor Bor Hazlini, Zulzalil, Hassan, & Mohd Ali, 2019).

Teniendo como referencia el Anexo A «Selección de la Técnica de Priorización», se selecciona la técnica de Voto Acumulativo.

5.2.2 *Modelo de Desarrollo de Software*

Tal como se documentó en el marco teórico y sobre la base del trabajo de investigación de (Vigneshwar Alavandhar & Nikiforova, 2017), existen numerosos modelos de desarrollo de software que soportan, además se incluyeron otras que son muy difundidas:

- *CMMI*.
- *ISO/IEC 15504*.
- *MoProSoft*.
- *UP*.
- *RUP*.
- *Enterprise Unified Process (EUP)*, tiene su origen en el *RUP*.
- *OpenUp* (tiene su origen en el *RUP*).
- *Microsoft Solutions Framework (MSF)*.
- Programación Extrema (*Extreme Programming - XP*).
- *SCRUM*.
- *NTP-ISO/IEC 12207*.

Como los modelos de desarrollo identificados permiten la incorporación de una técnica de priorización de requisitos, no se especifica un modelo en particular, lo que permite ser seleccionado según sus criterios, por el equipo de desarrollo.

5.2.3 *Licencia de Software*

Con el fin de hacer uso de una plataforma con la cual se seleccione la técnica a aplicar, se requiere de una licencia de *software* que autorice permiso de uso a los usuarios, ya que de otra forma se estaría infringiendo los derechos de autor. Una licencia de *software* esta constituido por un conjunto de reglas que un usuario tiene que cumplir, mientras esté utilizando el *software* (Zahoor & Sarwar Bajwa, 2014).

De las licencias de *software* estudiadas en el estado del arte, las compatibles con el modelo propuesto en la presente tesis son el *software* con licencia GPL.

Tanto la herramienta colaborativa como el componente de herramienta colaborativa seleccionados, deberán ser compatibles con las licencias identificadas, lo cual permitirá hacer uso de las mismas sin infringir las regulaciones.

5.2.4 *Herramienta Colaborativa*

Las herramientas de priorización también son importantes (Rashdan, 2021). La falta de automatización durante el proceso de Priorización de Requisitos influye en la eficiencia de la técnica porque la Priorización de Requisitos se vuelve complicado debido al esfuerzo requerido (Hujainah, Binti Abu Bakar, Nasser, Al-haimi, & Zamli, 2021).

La presente tesis propone el uso de un sistema de gestión de contenidos (*CMS*) como parte de la solución. En el Anexo I se describen las principales herramientas usadas en la actualidad.

Las herramientas CMS más difundidas y gratuitas identificadas son: *WordPress*, *Joomla*, *Drupal*, tal como ya se describió. Los criterios utilizados para seleccionar la herramienta adecuada entre las *CMS* candidatas fueron:

- Popularidad *CMS* open source.
- *Websites* activos.
- Resultados en *Google Trends*.
- Temas gratuitos.
- *Plugins* gratuitos.
- Tiempo de instalación.
- Dificultad para programar *plugins*.
- Tiempo de respuesta.
- Tiempo de respuesta con *cache*.
- Nivel de seguridad *web*.

Como resultado de la evaluación se selecciona la herramienta *CMS WordPress*, que utiliza licencia *GLP*, versión 2.

5.2.5 *Componente para la Herramienta Colaborativa*

Los componentes son pequeños módulos de *software* con los que se puede extender las características o funcionalidades de las herramientas *CMS*. Estos componentes no forman parte de la herramienta, están diseñados para añadir ciertas funciones específicas, permitiendo la máxima funcionalidad y minimizando el exceso de código que, de otra forma, saturaría a la herramienta *CMS* con un gran número de funciones que es posible no se utilicen o los utilicen cierto tipo de perfil de usuario y harían innecesariamente compleja a la herramienta. Los componentes ofrecen características y funciones específicas que permiten a los usuarios adicionar funcionalidades y personalizar la herramienta *CMS* de acuerdo con sus necesidades específicas.

Parte de la propuesta de la presente tesis es utilizar una herramienta *CMS* de amplio uso; sin embargo, esta por sí sola no puede otorgar funcionalidades requeridas,

por lo que se debe recurrir al uso de componentes, como lo de acceso libre y compatibles con alguna de las licencias de *software* que permitan acceder a código fuente y los datos del mismo (en primer lugar).

Dado que la implementación de un algoritmo de priorización de requisitos no es una funcionalidad estándar o de uso muy común, será necesario realizar el desarrollo de un componente que permita realizar la priorización de requisitos de *software* utilizando la técnica de priorización de requisitos seleccionada y los datos ingresados por el usuario en la herramienta *CMS* con otro *plugin*.

El componente para la herramienta colaborativa seleccionado es el Simple Rating, que utiliza la licencia GPL, versión 3. Este resultado se sustenta en el Anexo - C «Selección del Plugin para la Votación de los Requisitos».

5.3 Caso de Estudio

Como caso de estudio se propuso al área de estudio de la gestión documentaria. En esta área de conocimiento existen numerosas propuestas de estandarización.

Se eligió uno de los modelos de mayor aceptación a nivel global, el *MoReq* - Modelo de requisitos para la gestión de documentos electrónicos de archivo. Es un estándar elaborado por *DLM Forum* y financiada por la Comisión Europea, a diferencia de otros estándares, de alcance solo nacional. El objetivo de usar este estándar es no distraer la atención en la lista de requisitos por encima del modelo a evaluar. Es de acceso gratuito, a diferencia de otras como la norma ISO 15489-1, también un conjunto de normas e informes técnicos que abarcan aspectos sobre la creación, captura y gestión de documentos (Quispe Chucya, 2021), pero es de pago.

5.3.1 *Importancia de la Gestión Documentaria*

La conservación y manipulación de los datos es una tarea esencial para cualquier entidad u organización. Los sistemas de información deben permitir a los usuarios consolidar y acceder a los registros requeridos con facilidad y eficiencia.

La Gestión Documentaria es el conjunto de procedimientos o flujo de trabajo administrativos y técnicos, orientados a permitir planificar, manejar y organizar la documentación producida y recibida por las entidades, desde su origen hasta su destino final con el objeto de facilitar su utilización y conservación (ISO, 2016).

5.3.2 *Un Caso de Prueba Basado en Moreq*

Para la elaboración de la prueba se utilizaron como referencia los criterios aplicados de (Seyff, Todoran, Caluser, & Singer, 2015).

La evaluación se llevó a cabo utilizando el *WordPress* como herramienta seleccionada de *software* colaborativo.

Para permitir a los *stakeholders* que prioricen los requisitos, se utilizó un *plugin* de *WordPress* que permita realizar dicha tarea, con ayuda de la técnica de Voto Acumulativo (porque era la única de las evaluadas que podía ser soportada por alguno de los *plugins* disponibles en la *web*). Se determinó disponer de una *web* habilitada y configurada con los elementos necesarios para realizar la prueba. Los requisitos se ingresarán a la herramienta *CMS*, como un post.

El experimento se realizará en un contexto universitario, en donde los sujetos del experimento serán estudiantes.

A los *stakeholders* se les hará una breve descripción del experimento, pero no una demostración del uso de la herramienta de priorización. Cada uno de ellos realizará el proceso de priorización de los requisitos propuestos de manera individual, para simular que estuvieran trabajando en un entorno distribuido.

Se pidió a cada uno de los colaboradores que no comunicará verbalmente o por algún medio que no fuera el *CMS* alguna información u opinión, para no alterar los resultados de la prueba. El aspecto colaborativo requerido será soportado por la capacidad de la herramienta colaborativa *CMS*, que permitirá la realización de comentarios en cada uno de los requisitos listados.

No se requerirá de realizar algún proceso de capacitación, sensibilización o inducción en el uso de herramientas *CMS*, debido a que parte de la propuesta consiste en que aquellas son muy conocidas en su uso, o su comprensión es muy rápida debido al carácter intuitivo de su interfaz. Los estudiantes no podrán ingresar nuevos requisitos, o modificar o eliminar los existentes, debido a que dichas opciones no forman parte del alcance de la prueba.

No se hará uso del *email*, videoconferencia, mensajería instantánea o algún otro medio que pueda apoyar el proceso de priorización de requisitos. Solo está permitido el uso de comentarios para cada requisito. Estos comentarios están asociados a los requisitos, que serán ingresados por los estudiantes en su rol de *stakeholders*.

Por tratarse de una votación, luego de haber asignado los puntos correspondientes a cada requisito, estos no podrán ser modificados. No se considera el tiempo de duración del experimento como parte de la investigación o como criterio de evaluación de los resultados de la prueba.

Durante el proceso de priorización de requisitos, no se realizará algún tipo de orientación que podría alterar el resultado de la prueba.

En entornos reales, el número de requisitos que se presentan es alrededor de 150 (Ali & Hong, 2019), sin embargo, manejar ese número sería poco práctico en el entorno de un experimento.

6. Capítulo VI: RESULTADOS Y DISCUSIÓN

Luego de haber realizado una revisión del estado del arte, en el presente capítulo se define la propuesta de solución del presente trabajo de investigación. El modelo propuesto, además de contemplar el enfoque teórico, como el propuesto por (Kyosev, 2014) y un enfoque práctico como el definido por (Sevilla, Zapata, Torres, & Collazos, 2014) y (Seyff, Todoran, Caluser, & Singer, 2015).

Antes de proponer un nuevo modelo en la presente tesis, es necesario definir cuál es el tipo de aporte en el cual consistirá el modelo propuesto.

6.1 Análisis, Interpretación y Discusión de los Resultados

6.1.1 Contexto de la investigación

En concordancia con el enfoque usado con (Kyosev, 2014) y (Giraldo, Chicaiza, España, & Pastor, 2021), en el alcance del experimento se establece mediante la definición de sus objetivos. Con el fin de obtener mejores y más generales resultados, afirmaron que se debe ejecutar el experimento en un proyecto grande, utilizando *software* real, y con un *staff* de profesionales.

El contexto del experimento se especifica con las siguientes cuatro dimensiones:

- Offline vs. online.
- Estudiante vs. profesional.
- Simulado vs. problema real.
- Especifico vs. general.

En este contexto, el presente trabajo de investigación puede ser descrito como:

- *Online*: El experimento se relaciona directamente con un producto cuyos

requisitos de *software* son reales. Las priorizaciones realizadas por los *stakeholders* son en línea y vía *web*.

- Estudiantes: Los participantes en el experimento son estudiantes de Ingeniería de Sistemas e Informática, responsables de la priorización de requisitos quienes realizan sus prácticas preprofesionales en una entidad del sector público.
- Problema real: Los requisitos a priorizar son reales, tomados de una lista de requisitos estandarizados.
- Específico: Los resultados del experimento pueden ser aplicados en un contexto similar, tal como se define en el planteamiento del problema y los objetivos de la presente investigación.

6.1.2 *Objetivo de la Investigación*

El objetivo de la investigación es verificar la validez del modelo propuesto, expresados en la viabilidad, funcionalidad y facilidad de uso de la herramienta.

La funcionalidad del modelo indica que la propuesta sí permite realizar la tarea de priorización de requisitos; es decir, es viable. La facilidad de uso desde el punto de vista de los *stakeholders* en el contexto que el rol lo están asumiendo estudiantes de pregrado realizando prácticas preprofesionales de ingeniería de sistemas, informática y software. En una investigación, los autores (Allah Bukhsh, Allah Bukhsh, & Daneva, 2020) realizaron una investigación sistemática de la literatura de estudios empíricos en priorización de requisitos, destacando la importancia de la validación.

Previo a implementar un método o tecnología, se debe verificar a través de un experimento para comprobar si es válida o no su adopción en un entorno de trabajo (Mustafa & Zainuddin, 2014).

6.2 Pruebas de Hipótesis

Para la prueba de evaluación del modelo se utiliza un subconjunto de especificaciones funcionales del *MoReq*. Se eligieron diez requisitos de los disponibles (se utilizó esta cantidad por ser un número fácil de expresar en porcentaje y cantidad, a la vez que no requiere de mayores cálculos).

6.2.1 Instalación

El experimento será conducido en una sesión, en un entorno cerrado (ambiente en el cual realizan sus prácticas preprofesionales), en donde los participantes podrán realizar la priorización de requisitos con un mínimo de interferencia externa. Los participantes deberán realizar el experimento en un entorno bajo control (es decir, mientras dure el experimento se les pedirá que apaguen sus teléfonos móviles y se restringirá el uso de correo electrónico), dado que podría afectar el tiempo dedicado en la realización del experimento.

6.2.2 Diseño

Cada *stakeholders* participante del experimento debe realizar la priorización del mismo set de requisitos; a cada uno se les proporcionará la información sobre el uso de la técnica implementada en la herramienta propuesta.

6.2.3 Estructura

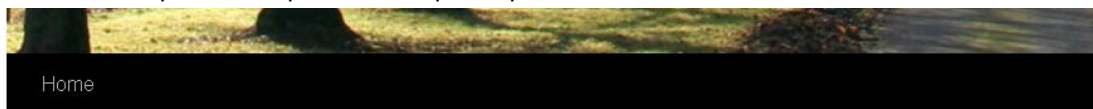
En este experimento no se estructuran los requisitos en términos de dependencia o nivel de complejidad; los *stakeholders* priorizan los requisitos sobre la base de sus propios criterios de prioridad.

6.2.4 Operación

El experimento fue realizado en sesiones de cinco (05) minutos. A los participantes se les realiza una breve explicación verbal de cuál será el experimento y cómo realizar el proceso de votación.

Figura 24

Ejecución de la prueba de parte de los participantes



← Presentación: Impresión

Firmas electrónicas →

Flujos de tareas

Posted on [December 4, 2019](#) by [admin](#)

★★★★★ 0 votes

La función de flujo de tareas del SGD debe ofrecer flujos de tareas formados por una serie de pasos, cada uno de los cuales sería, por ejemplo, el traslado de un documento o expediente de un participante a otro con vistas a la realización de la acción correspondiente.

This entry was posted in [Gestión Documentaria](#). Bookmark the [permalink](#).

← Presentación: Impresión

Firmas electrónicas →

Elaboración propia.

6.2.5 Puntos a Validar

1. Validez del constructo: Hace referencia a la correlación entre las bases teorías y la verificación de los resultados. Sería ideal que al inicio de la sesión se ofrezca una breve introducción para explicar el significado de los requisitos propuestos. Esto es para evitar cualquier interpretación incorrecta de los requisitos por los *stakeholders*, que puedan afectar los resultados.
2. Validación externa: En este punto, los *stakeholders* realizarán las tareas de priorización individualmente.

3. Como los requisitos que serán utilizados en este experimento son independientes, puede ser fácil realizar la tarea de priorización. Sin embargo, en un entorno real puede ser más dificultoso, debido a la dependencia entre alguno de los requisitos.

6.2.6 Entorno

Se realizó el experimento, ejecutado en un entorno en el cual los estudiantes al momento de la prueba están realizando sus prácticas preprofesionales. Se evaluaron los beneficios de utilizar herramientas de *software CMS*, y el uso de sus características colaborativas en el proceso de priorización de requisitos de *software* en un ambiente distribuido.

6.3 Presentación de Resultados

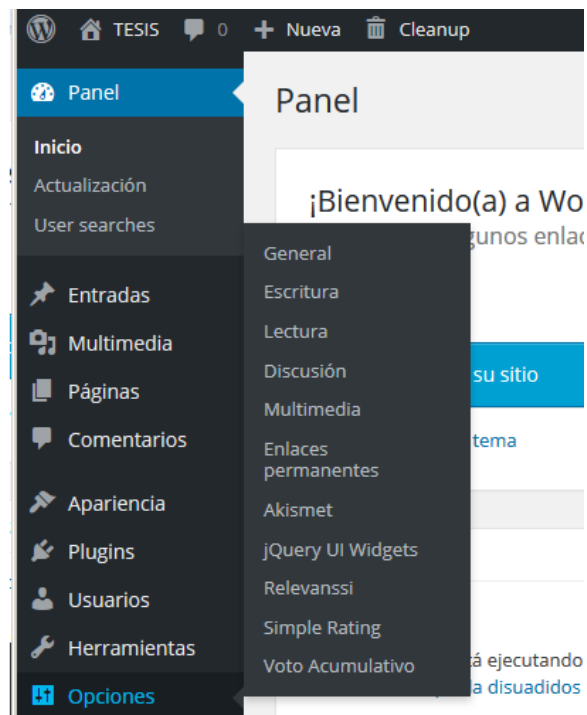
Para realizar el experimento, el sitio *web* está alojado en una *web* pública para poder acceder a él desde cualquier lugar con acceso a internet.

Para visualizar los resultados, se ingresa al Panel > Opciones > Voto Acumulativo, tal como se muestra en la **Figura 25**. En la pestaña «Visualización» se pueden ver los resultados de la priorización asignada por los alumnos a los requisitos propuestos (véase la **Figura 27**).

Con fines de evaluación, se puede ver en la pestaña «Votación» los votos que realizaron cada uno de los alumnos, tal como muestra la Figura 24. Como se puede apreciar, todos los alumnos completaron la realización del experimento de priorización y votación.

Figura 25

Acceso al componente Voto Acumulativo



Elaboración propia.

Figura 26

Panel 'Votación' del componente Voto Acumulativo

VOTACIÓN

PRIORIZACION

VOTO ACUMULATIVO - PANEL

El concepto involucra que los stakeholders participan en el establecimiento de prioridades utilizando un número de unidades imaginarias (100 puntos) las cuales son distribuidas entre los requisitos a priorizar.

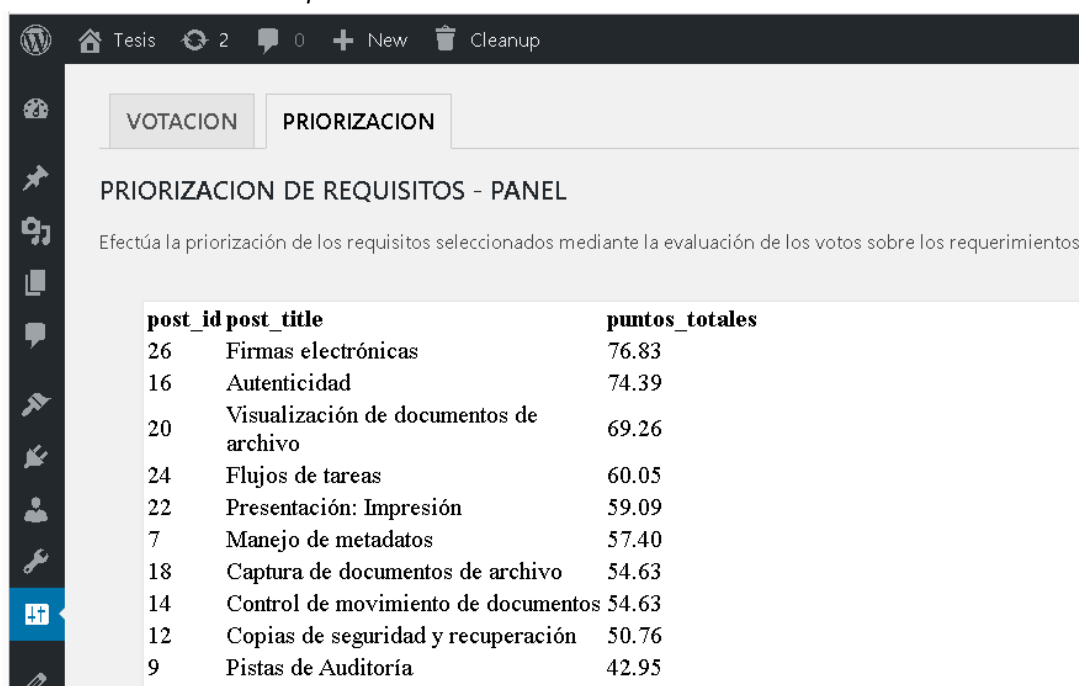
Seleccionar user_id: — Seleccionar todos —

post_id	user_id	points	votos
6	190.102.155.240	3	8.33
12	190.102.155.240	4	11.11
14	190.102.155.240	4	11.11
16	190.102.155.240	3	8.33
18	190.102.155.240	5	13.89
22	190.102.155.240	3	8.33
34	190.102.155.240	4	11.11
36	190.102.155.240	2	5.56
44	190.102.155.240	4	11.11
46	190.102.155.240	4	11.11
6	190.102.155.231	5	13.16
12	190.102.155.231	5	13.16
14	190.102.155.231	3	7.89

Elaboración propia.

Figura 27

Panel 'Priorización' del componente Voto Acumulativo



post_id	post_title	puntos_totales
26	Firmas electrónicas	76.83
16	Autenticidad	74.39
20	Visualización de documentos de archivo	69.26
24	Flujos de tareas	60.05
22	Presentación: Impresión	59.09
7	Manejo de metadatos	57.40
18	Captura de documentos de archivo	54.63
14	Control de movimiento de documentos	54.63
12	Copias de seguridad y recuperación	50.76
9	Pistas de Auditoría	42.95

Elaboración propia.

De acuerdo con los resultados, el uso del *CMS* permitió aplicar la técnica de priorización de requisitos Voto Acumulativo a través de *plugins* ya existentes.

6.3.1 Resultados para los indicadores

Los resultados del experimento presentado validan la hipótesis principal y específicas de la presente investigación.

6.3.1.1 Indicador de la variable independiente

VI1-I1: Presencia – Ausencia:

Se verifica que un “Modelo de Priorización de Requisitos” implementado en una herramienta *CMS* contribuye a la toma de decisiones de los *stakeholders*, verificado a través de los indicadores de las hipótesis específicas.

6.3.1.2 Indicadores de las variables dependientes

- VD1-I1: Facilidad de Uso

Luego de realizar la prueba, se entrevistó a los participantes sobre su opinión acerca del proceso de priorización de requisitos que acababan de realizar, se les realiza una encuesta en la cual se les pregunta cuál es su opinión sobre la facilidad de uso o de usabilidad. La escala de clasificación fue de 1 al 5 (1= difícil, 5=fácil), obteniéndose un resultado de 3.50; es decir, se tuvo un valor positivo o de aprobación.

Tabla 7

Evaluación de la facilidad de uso de la implementación del modelo

Alumno 1	Alumno 2	Alumno 3	Alumno 4	Alumno 5	Alumno 6	Promedio
3	3	4	4	4	3	3.50

Elaboración propia.

- VD1-I2: Implementabilidad.

Mediante el despliegue del “Modelo de Priorización de Requisitos”, en una herramienta *CMS*, cumpliendo con las restricciones especificadas para cada uno de sus componentes.

- VD1-I3: Costos en recursos.

La implementación del “Modelo de Priorización de Requisitos” se realizó en una herramienta *CMS* de código libre, demostrando que no es necesario disponer de bastantes recursos para su desarrollo, permitiendo su adopción por equipos pequeños, o con escasos recursos disponibles.

7. Capítulo VII: CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

7.1 Conclusiones

- Se demostró como un modelo de priorización de requisitos pudo ser implementado en una herramienta colaborativa *CMS* y que contribuye en la toma de decisiones de los *stakeholders*.
- Se demostró que es posible diseñar un modelo de priorización de requisitos que pueda ser implementado con una buena o adecuada “facilidad de uso” en una herramienta colaborativa *CMS* de código libre y redes sociales.
- Se demostró que es posible diseñar e implementar un modelo de priorización de requisitos en software *CMS* de código libre, donde cada uno de los elementos cumplen con las condiciones establecidas.
- Se demostró que es posible implementar un modelo de priorización de requisitos con un enfoque de bajos “costos en recursos”, enfocado en el uso de una herramienta *CMS* y en los componentes que permiten desplegar la técnica de priorización en la referida herramienta.
- Se investigó e identificó varias herramientas que permitirían a los *stakeholders* realizar la priorización de requisitos, que permita implementar alguna de las técnicas de priorización de requisitos en entornos distribuidos.
- Se demostró la validez del modelo propuesto en la herramienta de *software* colaborativo por medio de la evaluación teórica realizada. Asimismo, se

identificó que, tan importante como el disponer un modelo válido a nivel teórico, es importante contar de un modelo que sea válido a nivel práctico; es decir, que también pueda ser implementado en una herramienta de software.

- Se implementó y evaluó un conjunto de criterios para la identificación de una adecuada herramienta de *software* colaborativo, la cual aplicó los principios del *software* libre en la implementación del modelo propuesto.
- Se identificó y desarrolló un complemento para la herramienta de *software* colaborativo, el cual permitió a los *stakeholders* realizar la priorización de requisitos. Y, como parte del trabajo de investigación, se desarrolló un complemento adicional, el cual permitió la visualización de los resultados obtenidos durante la priorización de requisitos en la herramienta de *software* colaborativo seleccionada. Así, se estableció un caso de estudio que permitió evaluar y demostrar la validez del modelo propuesto.

7.2 Recomendaciones

- Las herramientas de *software* colaborativo, permiten aplicar una técnica de priorización, sin utilizar criterios subjetivos o personales para priorizar.
- La implementación de modelos como solución a un problema, se debe comprobar su validez mediante la implementación y evaluación del mismo en un caso de prueba.
- Con el fin de tener un modelo o propuesta válida, esta debe ser verificada desde un enfoque teórico y práctico.
- Para validar un modelo hacerlo mediante la participación de los usuarios. Esta

validación también se puede realizar con estudiantes, con la condición que tengan conocimientos teóricos sobre la materia, a través de los cursos que la imparten y de preferencia cuenten con prácticas preprofesionales.

7.3 Trabajos Futuros

- Se propone la identificación de otros modelos de priorización de requisitos que se apoyen en el uso de herramientas colaborativas aún no utilizadas, tales como redes sociales, tecnologías de dispositivos móviles, mensajería instantánea, y otras que emerjan.
- Realizar investigaciones en otras áreas del conocimiento⁶, además de la ingeniería de *software*, la gestión de requisitos en general y la priorización de requisitos en particular es importante, por ejemplo, en la gestión de TI y otros ámbitos.
- Diseñar modelos que involucren otros aspectos de la gestión de requisitos: elicitación, especificación o validación, basados en herramientas de *software* colaborativo.
- Identificar en qué aspectos se pueden implementar *software* de gestión de contenidos (CMS) como herramientas de soporte.
- Con el fin de reforzar y complementar los resultados de la investigación, se sugiere utilizar como sujetos del estudio a otros grupos de *stakeholders*, tales como docentes o profesionales.

⁶ La (ACM and IEEE, 2021) en su reciente compendio *Computer Science Curricula 2020*.

REFERENCIAS BIBLIOGRÁFICAS

- Abbas, M., Naila Jan, I., Saadatmand, M., & Daniel Sundmark, E. (2019). *MBRP: Model-based requirements prioritization using pagerank algorithm*. IEEE. Obtenido de <https://itea3.org/project/xivt.html>
- AbdElazim, K., Moawad, R., & Elfakharany, E. (Febrero de 2020). A Framework for Requirements Prioritization Process in Agile Software Development. *Journal of Physics: Conference Series*, 1454. doi:<https://doi.org/10.1088/1742-6596/1454/1/012001>
- Abdullah Awais, M. (31 de Julio de 2016). Requirements Prioritization: Challenges and Techniques for Quality Software Development. *ACSII Advances in Computer Science: an International Journal*, 5(20), 14-21. Obtenido de <http://www.acsij.org/acsij/article/view/458>
- Achimugu, P., Selamat, A., & Ibrahim, R. (2016). ReproTizer: A Fully Implemented Software Requirements Prioritization Tool. En *Transactions on Computational Collective Intelligence XXII* (Vol. 9655, págs. 80-105). Berlin, Alemania: Springer-Verlag Berlin Heidelberg. doi:10.1007/978-3-662-49619-0_5
- Achimugu, P., Selamat, A., Ibrahim, R., & Naz'ri Mahrin, M. (11 de Febrero de 2014). A systematic literature review of software requirements prioritization research. *Information and Software Technology*, 56(6), 568–585. doi:<http://dx.doi.org/10.1016/j.infsof.2014.02.001>
- ACM. (22 de noviembre de 2019). *Computing Classification System*. Obtenido de <http://www.acm.org/publications/article-templates/sig-alternate-05-2015.cls>
- ACM and IEEE. (2021). *Computer Science Curricula 2020*. New York. doi:<https://doi.org/10.1145/3467967>
- Adhim, K., Hudaib, A., & Al-Shboul, B. (2019). *Efficient requirement prioritization based on enhanced multi-verse optimizer* (Vol. 97). Amman, Jordania: Journal of Theoretical and Applied Information Technology.
- Alarcón, C., & Rojas, S. (Enero-Junio de 2018). Comparación entre Prince2 y OpenUp para desarrollo de software. *Tecnología Investigación y Academia*, 6(1), 77-83. Obtenido de <https://revistas.udistrital.edu.co/index.php/tia/article/view/8655>
- Ali, N., & Hong, J.-E. (26 de Abril de 2019). Using Social Network Service to determine the Initial User Requirements for Small Software Businesses. *International Journal of Applied Business and Economic Research*, 8. Obtenido de <https://arxiv.org/pdf/1904.12583v1>

- Allah Bukhsh, F., Allah Bukhsh, Z., & Daneva, M. (Marzo de 2020). A systematic literature review on requirement prioritization techniques and their empirical evaluation. *Computer Standards & Interface*, 69, 1-40.
doi:<https://doi.org/10.1016/j.csi.2019.103389>
- Alphabet . (09 de Noviembre de 2021). *Preguntas frecuentes sobre los datos de Google Trends*. Obtenido de <https://support.google.com/trends/answer/4365533?hl>
- Alvarez, J., & Hurtado, J. (19 de Agosto de 2018). UP-VSE: A Unified Process - Based Lifecycle Model for Very Small Entities. *Colombian Conference on Computing*, 260-275. doi:https://doi.org/10.1007/978-3-319-98998-3_21
- AmbySoft Inc. (14 de Junio de 2015). *EUP*. Recuperado el 19 de Agosto de 2015, de EUP: <http://enterpriseunifiedprocess.com/>
- Anand, R., & Dinakaran, M. (2018). WhaleRank: an optimisation based ranking approach for software requirements prioritisation. *Int. J. Environment and Waste Management*, 21(1), 1 - 21.
doi:<https://dx.doi.org/10.1504/IJEW.2018.091307>
- Axelos Limited. (2018). *PRINCE2 Agile*® (3ra. ed.). Londres, Inglaterra: The Stationery Office.
- Bahill, A., & Madni, A. (2017). *Tradeoff Decisions in System Design*. Cham, Switzerland: Springer. doi:10.1007/978-3-319-43712-5
- Barbosa, P., Pinheiro, P., Silveira, F., & Filho, M. (15 de Julio de 2019). Selection and Prioritization of Software Requirements Applying Verbal Decision Analysis. (R. Natella, Ed.) 2019(2306213), 20. doi:<https://doi.org/10.1155/2019/2306213>
- Bennaceur, A., Than Tun, T., Yu, Y., & Nuseibeh, B. (2019). Requirements Engineering. En *Handbook of Software Engineering*. Springer.
doi:https://doi.org/10.1007/978-3-030-00262-6_2
- Bernhard, R. (2017). *Agile Modeling with UML* (1era. ed.). Cham: Springer.
- Biasutti, M. (2017). A comparative analysis of forums and wikis as tools for online collaborative learning. (Elseiver, Ed.) *Computers & Education*, 111, 158-171.
doi:<https://doi.org/10.1016/j.compedu.2017.04.006>
- BuiltWith® Pty Ltd. (20 de Diciembre de 2019). *CMS technologies Web Usage Statistics*. Recuperado el 20 de Diciembre de 2019, de <http://trends.builtwith.com/>
- Burge, S. (2017). *Joomla 3 Explained*. OSTRaining.
- Buytaert, D. (10 de Mayo de 2015). *Drupal*. Obtenido de <http://drupal.org/es/drupal>

- Cabot, J. (Mayo-Junio de 2018). A Content Management System to Democratize Publishing. *Impact*, 35(3), 89-92. doi:<https://doi.org/10.1109/MS.2018.2141016>
- Cepero García, M. T., Montané Jiménez, L., Toledo Toledo, G., Benítez Guerrero, E., & Mezura Godoy, C. (2021). Heuristics for Awareness Support in Groupware Systems. *DYNA New Technologies*, 8(1), 1-11. doi:<https://doi.org/10.6036/NT9980>
- Chaudhary, M., & Chopra, A. (2017). *CMMI for Development, Implementation Guide*. (N. Karkal, Ed.) New York, EE.UU.: Apress. doi:<https://doi.org/10.1007/978-1-4842-2529-5>
- Corona Lisboa, J. (09 de Enero de 2016). Apuntes sobre métodos de investigación. *Medisur*, 14(1), 81-83. Obtenido de <http://scielo.sld.cu/pdf/ms/v14n1/ms16114.pdf>
- Darias González, H. (2020). *Herramienta para la Gestión de la Base de Conocimiento de la Implementación de la Norma NMX-I-059/02-NYCE-2016 Moprosoft Nivel II*. División de Estudios de Posgrado e Investigación. Villahermosa: Instituto Tecnológico de Villahermosa.
- Drupal. (09 de Noviembre de 2021). *Documentation*. Obtenido de <https://www.drupal.org/docs>
- Escobar Terán, C., Calderón Serrano, J., Cevallos Romero, E., & Escobar Terán, H. (05 de Julio de 2017). Epistemología de las Ciencias Computacionales en Ingeniería. *Revista Didasc@lia*, VIII(3), 213-224.
- Farinha, C., & Mira da Silva, M. (23 de OCTubre de 2011). Web-Based Focus Groups for Requirements Elicitation. *ICSEA 2011 : The Sixth International Conference on Software Engineering Advances*, 504-509. Obtenido de http://www.thinkmind.org/download.php?articleid=icsea_2011_22_10_10070
- Fatima, T., & Mahmood, W. (2019). *Requirement Engineering in Agile* (Vol. 9). Karachi, Pakistan: I.J. Education and Management Engineering. doi:DOI: 10.5815/ijeme.2019.04.03
- Flório, C., Lencastre, M., Pimentel, J., & Araujo, J. (Octubre de 2019). iStar-p: A Modelling Language for Requirements Prioritization. *International Conference on Conceptual Modeling*, 11788, 540-548. doi:https://doi.org/10.1007/978-3-030-33223-5_44
- FSF. (17 de Noviembre de 2019). *The Free Software Definition*. Recuperado el 17 de Noviembre de 2019, de <https://www.gnu.org/philosophy/free-sw.html>
- Galinier, S., & Laporte, C. (1-3 de Octubre de 2018). Connecting Business Development and Systems Engineering with ISO/IEC 29110 Standard in Small and Medium

- Enterprises of France. *2018 IEEE International Systems Engineering Symposium (ISSE)*, 1-7. doi:<https://doi.org/10.1109/SysEng.2018.8544393>
- Gambo, I., Ikono, R., Achimugu, P., & Iroju, O. (Abril de 2018). A Ranking Model for Software Requirements Prioritization during Requirements Engineering: a case study. *International Journal of Computer Science and Information Security (IJCSIS)*, 16(4), 255-268.
- Giraldo, F., Chicaiza, Á., España, S., & Pastor, Ó. (15 de Marzo de 2021). Empirical validation of a quality framework for evaluating modelling languages in MDE environments. *Software Quality Journal*, 275–307. doi:<https://doi.org/10.1007/s11219-021-09554-1>
- Gómez Checa, J. (2018). *Entorno de Soporte a la Mejora y Certificación de Procesos Software basado en estándares ISO/IEC 33000 e ISO/IEC 12207*. Escuela Superior De Informática. Castilla-La Mancha: Universidad De Castilla-La Mancha.
- Google. (20 de Diciembre de 2019). *Blogger Features*. Recuperado el 29 de Marzo de 2016, de <https://support.google.com/blogger/?hl=es#topic=3339243>
- Heath, F., & Gupta, T. (2021). *The Professional Scrum Master (PSM I) Guide* (1ra. ed.). (G. Gala, Ed.) Birmingham: Packt Publishing Ltd.
- Heikkil'a, V. T., Paasivaara, M., Lasssenius, C., Damian, D., & Engblom, C. (10 de Enero de 2017). Managing the requirements flow from strategy to release in large-scale agile development: a case study at Ericsson. *Empirical Software Engineering*, 1-45. doi:10.1007/s10664-016-9491-z
- Hernández Sampieri, R., & Mendoza Torres, C. (2018). *Metodología De La Investigación: Las rutas cuantitativa, cualitativa y mixta* (Primera Edición ed.). México D.F., México: McGraw-Hill/Interamericana Editores, S.A. de C.V.
- Hossain, S. S., Bhuiyan, T., Jubayer, S. A., Rahman, S., Islam, S., & Rawshan, L. (29 de Junio de 2019). Customer Feedback Prioritization. *Customer Feedback Prioritization Technique: A Case Study on Lean Startup*, 11623, 70-81. doi:https://doi.org/10.1007/978-3-030-24308-1_6
- Hudaib, A., Masadeh, R., Haj Qasem, M., & Alzaqebah, A. (16 de Enero de 2018). Requirements Prioritization Techniques Comparison. *Canadian Center of Science and Education*, 12(2), 62-80. doi:<https://doi.org/10.5539/mas.v12n2p62>
- Hujainah, F., Binti Abu Bakar, R., & Abdullateef Abdulgabbler, M. (2019). StakeQP: A semi-automated stakeholder quantification and prioritisation technique for requirement selection in software system projects. *Decision Support Systems*, 121, 94-108. doi:<https://doi.org/10.1016/j.dss.2019.04.009>

- Hujainah, F., Binti Abu Bakar, R., Nasser, A., Al-haimi, B., & Zamli, K. (2021). SRPTackle: A semi-automated requirements prioritisation technique for scalable requirements of software system projects. *Information and Software Technology*, 131. doi:<https://doi.org/10.1016/j.infsof.2020.106501>
- Ibriwesh, I., Ho, S.-B., Chai, I., & Tan, C.-H. (Noviembre de 2019). Prioritizing solution-oriented software requirements using the multiple perspective prioritization technique algorithm: An empirical investigation. *SAGE Journals*, 1, 12. doi:DOI: 10.1177/1063293X18808559
- IEEE. (22 de noviembre de 2019). *IEEE Taxonomy*. Obtenido de https://www.ieee.org/documents/taxonomy_v101.pdf
- Inacal. (2016). *Ingeniería de software y sistemas. Procesos del ciclo de vida del software*. Dirección de Normalización. Lima: Inacal.
- ISO. (2016). *ISO 15489-1 - Information and documentation - Records management*. Ginebra, Suiza: International Organization for Standardization.
- Jacobson, I., Lawson, H., Ng, P.-W., McMahon, P., & Goedicke, M. (2019). *The Essentials of Modern Software Engineering*. (M. Tamer Ozsu, Ed.) ACM Books series. doi:<https://doi.org/10.1145/3277669>
- Jarzębowicz, A., & Sitko, N. (02 de Octubre de 2020). Agile Requirements Prioritization in Practice: Results of an Industrial Survey. *24th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, 176, 3446-3455. doi:<https://doi.org/10.1016/j.procs.2020.09.052>
- Karen Tia, T., Nuryasin, I., & Maskur. (Enero de 2019). Simulation Model for Rational Unified Process (RUP) Software Development Life Cycle. *Jurnal SISTEMASI*, 8(1), 176-184. doi:<https://doi.org/10.32520/stmsi.v8i1.420>
- Khush Bakht, I., Inayat, I., & Allah Bukhsh, F. (28-30 de Agosto de 2019). Non-functional Requirements Prioritization: A Systematic Literature Review. *2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 379-386. doi:<https://doi.org/10.1109/SEAA.2019.00064>
- Kothawar, S., & Gupta Vajrapu, R. (2018). *Software Requirements Prioritization Practices in Software Start-ups: A Qualitative research based on Start-ups in India*. Blekinge Institute of Technology, Department of Software Engineering. Karlskrona: Faculty of Computing.
- Kravchenko, T., & Bruskin, S. (30 de Junio de 2017). Prioritization of requirements for effective support of the communication process with customers of a commercial bank. *Business Informatics*, 7-16. doi:<http://dx.doi.org/10.17323/1998-0663.2017.2.7.16>.

- Kyosev, T. (15 de Agosto de 2014). Comparing Requirements Prioritization Methods in Industry: A study of the Effectiveness of the Ranking Method, the Binary Search Tree Method and the Wiegers Matrix. 93. Postbus, Utrecht, Países Bajos: Universiteit Utrecht. Obtenido de <http://dspace.library.uu.nl/bitstream/handle/1874/298562/Thesis.pdf>
- Laplante, P. (2018). *Requirements Engineering for Software and Systems* (3era. ed.). Boca Raton, California, EE.UU.: CRC Press. Obtenido de <https://lccn.loc.gov/2017016218>
- Laporte, C., Tremblay, N., Menaceur, J., Poliquin, D., & Houde, R. (29 de Mayo de 2017). Systems Engineering and Management Processes for Small Organizations with ISO/IEC 29110: An implementation in a Small Public Transportation Company. *2017 Annual IEEE International Systems Conference (SysCon)*, 1-8. doi:<https://doi.org/10.1109/SYSCON.2017.7934718>
- Lavado, L. (2020). *Epistemología e Investigación*. (L. Lavado Mallqui, Ed.) Lima, Chosica, Perú: Fondo Editorial Universidad Nacional de Educación Enrique Guzmán y Valle.
- Leshob, A., Hadaya, P., & Renard, L. (28 de Noviembre de 2019). Software Requirements Prioritization with the Goal-Oriented Requirement Language. *International Conference on e-Business Engineering*, 41, 187-198. doi:https://doi.org/10.1007/978-3-030-34986-8_13
- Lorentz, J. (17 de Mayo de 2009). The Effect of Requirements Prioritization on Avionics System Conceptual Design. Washington, D.C, USA. Obtenido de http://www.researchgate.net/publication/36711646_The_effect_of_requirements_prioritization_on_avionics_system_conceptual_design
- Lozano Alarcón, C., & Montenegro Risco, D. (Noviembre de 2020). *Método para la Gestión del Desarrollo del Software, basada en la NTP ISO/IEC 12207:2006 en la Municipalidad Provincial de Chiclayo*. Tesis de titulación, Universidad Nacional Pedro Ruiz Gallo, Facultad de Ingeniería Civil, de Sistemas y Arquitectura, Lambayeque.
- Luna Aguinaga, A., & Córdova Llacsahuache, L. (2019). *Implementación de un Portal E-commerce Usando NTP-ISO/IEC 12207 Bajo el Proceso de la Metodología Ágil Scrum para la Empresa Mafargia Import E.I.R.L. en la Banda de Shilcayo – San Martín – Perú*. Facultad de Ingeniería y Arquitectura. Tarapoto: Universidad Peruana Unión.
- Lynn Cooke, J. (2016). *PRINCE2 Agile™: An Implementation Pocket Guide*. Cambridgeshire: Axelos.

- Mani Subedi, I. (2019). *Software inspection tool for defects classification on requirement documents using collaborative approach*. Dakota del Norte, Estados Unidos: Escuela para graduados de la Universidad de Dakota del Norte. Obtenido de <https://library.ndsu.edu/ir/bitstream/handle/10365/31342/Software%20Inspection%20Tool%20for%20Defects%20Classification%20on%20Requirement%20Documents%20Using%20Collaborative%20Approach.pdf?sequence=1&isAllowed=y>
- Martín Jiménez, L. (2018). *Filosofía de la Técnica y de la Tecnología* (1era Edición ed.). Oviedo, España: Pentalfa Ediciones.
- Maximini, D. (2018). *The Scrum Culture, Introducing Agile Methods in Organizations* (2da. ed.). Cham, Switzerland: Springer. doi:<https://doi.org/10.1007/978-3-319-73842-0>
- Mayekar, D. (2017). *Decoupling Drupal: A Decoupled Design Approach for Web Applications*. Mumbai, India: Apress. doi:<https://doi.org/10.1007/978-1-4842-3321-4>
- McZara, J., Sarkani, S., Holzer, T., & Eveleigh, T. (09 de Setiembre de 2014). Software requirements prioritization and selection using linguistic tools and constraint solvers—a controlled experiment. *Empirical Software Engineering*. doi:10.1007/s10664-014-9334-8
- Melegati, J., Goldman, A., Kon, F., & Wang, X. (10 de Febrero de 2019). A model of requirements engineering in software startups. *Information and Software Technology*, 109, 92-107. doi:<https://doi.org/10.1016/j.infsof.2019.02.001>
- Mousaei, M., & Javdani Gandomani, T. (2018). A New Project Risk Management Model based on Scrum Framework and Prince2 Methodology. *(IJACSA) International Journal of Advanced Computer Science and Applications*, 9(4), 442-449. doi:<http://dx.doi.org/10.11591/eei.v9i3.1905>
- Mowad, R., Abdelazim Muhammad, K., & Elfakharany, E. (2020). A Supporting Tool for Requirements Prioritization Process in Agile Software Development. *Future Computing and Informatics Journal*, 5(2). Obtenido de <https://digitalcommons.aaru.edu.jo/fcij/vol5/iss1/2/>
- Mubarok, A., & Tjahjadi, D. (Diciembre de 2019). Pemodelan Arsitektur Enterprise Sistem Informasi Customer Relationship Management Menggunakan Enterprise Unified Process. *{ILKOM Jurnal Ilmiah*, 11, 231-240. doi:<http://dx.doi.org/10.33096/ilkom.v11i3.478.231-240>
- Muhammad, Y., Mustapha, A., & Ibrahim, N. (21 de Noviembre de 2018). An Approach for Managing Large-Sized Software Requirements during Prioritization. 2018

- IEEE Conference on Open Systems (ICOS)*, 98-103.
doi:<https://doi.org/10.1109/ICOS.2018.8632806>
- Muñoz, M., Mejia, J., & Laporte, C. (10-12 de Octubre de 2018). Implementación del Estándar ISO/IEC 29110 en Centros de Desarrollo de Software de Universidades Mexicanas: Experiencia del Estado de Zacatecas. *XI Seminario Internacional de Ciencias de la Computación*, 43-54. doi:<https://doi.org/10.17013/risti.n.pi-pf>
- Muñoz, M., Peralta, M., & Laporte, C. (23-25 de Octubre de 2019). Análisis de las debilidades que presentan las Entidades Muy Pequeñas al implementar el estándar ISO/IEC 29110: Una comparativa entre estado del arte y el estado de la práctica. *XII Seminario Internacional de Ciencias de la Computación*, 85-96. doi:<http://dx.doi.org/10.17013/risti.34.85-96>
- Mustafa, B., & Zainuddin, A. (27,28 de Agosto de 2014). An Experimental Design to Compare Software Requirements Prioritization Techniques. *Computational Science and Technology (ICCST)*, 1-5.
doi:<http://dx.doi.org/10.1109/ICCST.2014.7045010>
- Muzamal Liaqat, R., Azam, F., Adeel Ahmed, M., & Mehboob, B. (Octubre de 2016). A Majority Voting Goal Based Technique for Requirement Prioritization. *2016 22nd International Conference on Automation and Computing (ICAC)*, 435-439. doi:<https://doi.org/10.1109/IConAC.2016.7604958>
- Nimesh Ratnayake, R. (2015). *WordPress Web Application Development* (Second Edition ed.). (R. Raman, Ed.) Birmingham, Reino Unido: Packt Publishing Ltd.
- Nivela Cornejo, M., Echeverría Desiderio, S., & Sánchez Villarroel de Gallardo, M. (Julio de 2021). Los CMS en el Desarrollo de Estilos de Aprendizaje. *Revista Publicando*, 8(31), 177-194. doi:<https://doi.org/10.51528/rp.vol8.id2243>
- Noor Bor Hazlini, N., Zulzalil, H., Hassan, S., & Mohd Ali, N. (Noviembre de 2019). Requirements Prioritization Techniques Focusing on Agile Software Development A Systematic Literature Review. *International Journal of Scientific & Technology Research*, 8(11), 2118-2125. Obtenido de <https://www.ijstr.org/paper-references.php?ref=IJSTR-1119-25544>
- Okesola, J., Adebisi, M., Okokpujie, K., Odepitan, D., Goddy-Worlu, R., Iheanetu, O., . . . Adebisi, A. (2019). *A systematic review of requirement engineering practices in agile model* (Vol. 01). Nigeria: International Journal of Mechanical Engineering and Technology (IJMET). Obtenido de <http://www.iaeme.com/ijmet/issues.asp?JType=IJMET&VType=10&IType=02>
- Pastrana, M., Ordóñez, H., & Cobos, C. (20 de Noviembre de 2019). ISO 29110 en Colombia: de la teoría a la práctica. *Revista Guillermo De Ockham*, 17(2), 71-80. doi:<https://doi.org/10.21500/22563202.4299>

- Pressman, R. S., & Maxim, B. (2019). *Software Engineering: A Practitioner's Approach* (Ninth Edition ed.). Penn Plaza, New York, EE.UU.: McGraw-Hill Education.
- Quispe Chucya, C. (2021). *La norma ISO 15489 como instrumento para la normalización de la gestión documental*. Facultad de Letras y Ciencias Humanas. Lima: Universidad Nacional Mayor de San Marcos.
- Rahim, M., Chowdhury, A., & Das, S. (21-23 de Diciembre de 2017). RIZE: A Proposed Requirements Prioritization Technique for Agile Development. *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, 634-637.
doi:<https://doi.org/10.1109/R10-HTC.2017.8289039>
- Rashdan, A. (2021). *Requirement prioritization in Software Engineering: A Systematic Literature Review on Techniques and Methods*. Linnaeus University, Department of computer science and media technology. Växjö: Faculty of Technology.
- Rida Asghar, A., Tabassum, A., Nazir Bhatti, S., & Ali Shah, S. (2017). The Impact of Analytical Assessment of Requirements Prioritization Models: An Empirical Study. (*IJACSA*) *International Journal of Advanced Computer Science and Applications*, 8(2), 303-313.
doi:<https://dx.doi.org/10.14569/IJACSA.2017.080240>
- Saghir, S., & Mustafa, T. (30 de Noviembre de 2016). Requirements Prioritization Techniques for Global Software Engineering. *Journal of Information Communication Technologies and Robotic Applications*, 7, 23-33.
doi:<https://doi.org/10.22555/pjets.v6i1.1143>
- Saleh, S., Maruful Huq, S., & Rahman, M. (7-9 de Febrero de 2019). Comparative Study within Scrum, Kanban, XP Focused on Their Practices. *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*.
doi:<https://doi.org/10.1109/ECACE.2019.8679334>
- Sánchez Medero, G., & Pastor Albaladejo, G. (Marzo de 2020). The use of a wiki to boost open and collaborative learning in a Spanish university. *Knowledge Management & E-Learning*, 12(1), 1-17.
doi:<https://doi.org/10.34105/j.kmel.2020.12.001>
- Sapunkov, A., & Afanasieva, T. (15 de Marzo de 2019). Software for Automation of User Requirements. *Proceedings of the 2019 2nd International Conference on Geoinformatics and Data Analysis*, 1-5.
doi:<https://doi.org/10.1145/3318236.3318251>
- Scrum Alliance. (09 de Junio de 2015). *What Is Scrum?* Recuperado el 14 de Junio de 2015, de http://www.scrumalliance.org/pages/what_is_scrum

- Sevilla, G., Zapata, S., Torres, E., & Collazos, C. (3-5 de Mayo de 2014). Using Wikis as Collaborative Strategy to Support Software Requirements Elicitation. *Computing Colombian Conference (9CCC)*, 54-61.
doi:<http://dx.doi.org/10.1109/ColumbianCC.2014.6955339>
- Seyff, N., Todoran, I., Caluser, K., & Singer, L. (24 de Abril de 2015). Using popular social network sites to support requirements elicitation, prioritization and negotiation. *Journal of Internet Services and Applications*.
doi:<http://dx.doi.org/10.1186/s13174-015-0021-9>
- Shao, F., Peng, R., Lai, H., & Wang, B. (Abril de 2017). DRank: A Semi-automated Requirements Prioritization Method based on Preferences and Dependencies. *Journal of Systems and Software*, 126(ISSN:0164-1212), 141-156.
doi:<https://doi.org/10.1016/j.jss.2016.09.043>
- Sher, F., Jawawi, D. N., Mohamad, R., & Muhammad, I. (2014). *Requirements Prioritization Techniques and Different Aspects for Prioritization*. Langkawi, Malaysian: IEEE. doi:<http://dx.doi.org/10.1109/MySec.2014.6985985>
- Sher, F., Jawawi, D., & Mohammad, R. (31 de Julio de 2019). Requirements prioritization aspects quantification for value-based software developments. *Journal of Theoretical and Applied Information Technology*, 97(14), 3969-3979.
- Sipos, D. (2020). *Drupal 9 Module Development* (3era. ed.). Birmingham, England: Packt Publishing Ltd.
- So, P. (2018). *Decoupled Drupal in Practice: Architect and Implement Decoupled Drupal*. New York, EE.UU.: Apress. doi:<https://doi.org/10.1007/978-1-4842-4072-4>
- Sommerville, I. (2016). *Software Engineering* (Tenth edition ed.). (M. Horton, Ed.) Edinburgh Gate, Harlow, England: Pearson Education Limited.
- Sommerville, I. (2021). *Engineering Software Products, An Introduction to Modern Software Engineering* (1era. ed.). Londres, Inglaterra: Pearson Education Limited.
- Song, W. (Febrero de 2017). Requirement management for product-service systems: Status review and future trends. *Computers in Industry*, 85, 11-22.
doi:<http://dx.doi.org/10.1016/j.compind.2016.11.005>
- Sparx Systems. (10 de Noviembre de 2021). *Requirements Diagram*. Recuperado el 10 de Noviembre de 2021, de https://sparxsystems.com/enterprise_architect_user_guide/15.2/model_domains/requirements_diagram2.html

- Squarespace. (20 de Diciembre de 2019). *About Our Company - Squarespace*. Recuperado el 20 de Diciembre de 2019, de <http://www.squarespace.com/about/company>
- Stallman, R. (2020). La definición de Software libre. *Massachusetts Institute of Technology*, 151-154. Obtenido de <https://stallman.org/gpg.html>
- Suárez Ardura, M. (2017). Notas a propósito de la distinción entre gnoseología y epistemología. *El Basilisco*(48), 89-92.
- Summers, B. (2021). *Effective Methods for Software Engineering*. Florida, EEUU: CRC Press.
- The Joomla Project Team. (18 de Diciembre de 2019). *Joomla*. Recuperado el 18 de Diciembre de 2019, de <http://www.joomla.org/about-joomla.html>
- Tomlinson, T. (2017). *Enterprise Drupal 8 Development: For Advanced Projects and Large Development Teams*. (T. Green, Ed.) Oregon, EE.UU.: Apress. doi:<https://doi.org/10.1007/978-1-4842-0253-1>
- Tong, Z., Zhuang, Q., Guo, Q., & Ma, P. (21 de Junio de 2014). Research on Technologies of Software Requirements Prioritization. (Y. Yuan, X. Wu, & Y. Lu, Edits.) *Communications in Computer and Information Science*(ISSN: 1865-0929), 9-21. doi:[10.1007/978-3-662-43908-1_2](https://doi.org/10.1007/978-3-662-43908-1_2)
- Tufail, H., Qasim, I., Faisal Masood, M., Tanvir, S., & Haider Butt, W. (16 de Mayo de 2019). Towards the selection of Optimum Requirements Prioritization Technique: A Comparative Analysis. *The 5th International Conference on Information Management*, 227-231. doi:<https://doi.org/10.1109/INFOMAN.2019.8714709>
- Tufail, H., Qasim, I., Faisal Masood, M., Tanvir, S., & Haider Butt, W. (16 de Mayo de 2019). Towards the selection of Optimum Requirements Prioritization Technique: A Comparative Analysis. *The 5th International Conference on Information Management*, 227-231. doi:<https://doi.org/10.1109/INFOMAN.2019.8714709>
- Turner, M. (2015). *Microsoft® Solutions Framework Essentials: Building Successful Technology Solutions* (Primera Edición ed.). Redmond, Washington, USA: Microsoft Press.
- Udjaja, Y. (29 de Agosto de 2018). Ekspanpixel Bladsy Stranica: Performance Efficiency Improvement of Making Front-End Website Using Computer Aided Software Engineering Tool. *Procedia Computer Science*, 135, 292-301. doi:<https://doi.org/10.1016/j.procs.2018.08.177>

- UNMSM. (8 de Enero de 2014). *Programas y Lineas de investigacion en la Universidad Nacional Mayor de San Marcos*. Obtenido de http://vrinvestigacion.unmsm.edu.pe/images/vri/documentos/Programas_Lineas_inv_2014.pdf
- Valdivia Caballero, J. (2021). Modelo de procesos para el desarrollo del front-end de aplicaciones web. *Facultad de Ingeniería de Sistemas e Informática, Unidad de Posgrado - Universidad Nacional Mayor de San Marcos*, 169. Lima, Perú.
- Valencia, F., & Arcos, H. (2018). Unified Process Applied to the Implementation of the Finite Difference Time Domain Method in Python through Object Oriented Programming. (O. N. CENACE, Ed.) *Revista técnica energía*, 14(1). doi:<http://dx.doi.org/10.37116/revistaenergia.v14.n1.2018.168>
- Vigneshwar Alavandhar, J., & Nikiforova, O. (Enero de 2017). Several Ideas on Integration of SCRUM Practices within Microsoft Solutions Framework. *Applied Computer Systems*, 71-80. doi:<https://doi.org/10.1515/acss-2017-0010>
- Wohlin, C., Runeson, P., Host, M., Ohlsson, M., Regnell, B., & Wesslen, A. (2012). *Experimentation in Software Engineering* (Primera Edición ed., Vol. I). Berlin, Heidelberg, Alemania: Springer. doi:10.1007/978-3-642-29044-2
- WordPress Foundation. (18 de Diciembre de 2019). *WordPress*. Recuperado el 18 de Diciembre de 2019, de <https://wordpress.org/>
- WordPress Foundation. (8 de Noviembre de 2021). *Plugin API*. Recuperado el 8 de Noviembre de 2021, de http://codex.wordpress.org/Plugin_API
- WordPress Foundation. (08 de Noviembre de 2021). *Plugin Directory*. Recuperado el 28 de Julio de 2021, de <https://wordpress.org/plugins/>
- WordPress Foundation. (8 de Noviembre de 2021). *Plugins*. Recuperado el 8 de Noviembre de 2021, de <https://wordpress.org/support/article/plugins/>
- Yaseen, M., Ibrahim, N., & Mustapha, A. (2019). *Requirements Prioritization and using Iteration Model for Successful Implementation of Requirements* (Vol. 10). Batu Pahat, Johor, Malasia: International Journal of Advanced Computer Science and Applications (IJACSA).
- Yaseen, M., Mustapha, A., Ur Rahman, A., Khan, S., & Kamal, W. (Marzo-Abril de 2019). Importance of Requirements Prioritization in Parallel Developing Software Projects. *International Journal of Computer Science Engineering (IJCSE)*, 9(2), 171-179.
- Yuan, J., & Hengyuan, Z. (2019). *The Challenge for Practitioners to Adopt Requirement Prioritization Techniques in Practice*. Faculty of Computing. Karlskrona: Blekinge Institute of Technology.

Zahoor, F., & Sarwar Bajwa, I. (26-27 de Agosto de 2014). Automatic Extraction of Catchphrases from Software License Agreement. *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, 189-194. doi:<http://dx.doi.org/10.1109/IHMSC.2014.148>

ÍNDICE DE PALABRAS

A

Análisis de requisitos	40
Árbol de búsqueda binario	94

C

<i>Cloud</i>	103
CMMI.....	44
CMS	16, 72, 108, 123, 154
Código Abierto	107

D

Drupal.....	111, 156
-------------	----------

E

Elicitación de requisitos	105
Enterprise Architect	103
Epistemología	26
ERS..... Véase Especificación de requisitos de software	
Especificación de Requisitos	40
Especificación de requisitos de software	43
EUP	48

F

Fases de la gestión de requisitos	40
Foros	105
FOSS . Véase <i>Software Libre</i> y de <i>Código Abierto</i> , Véase <i>Software de Código Abierto</i>	
<i>Freeware</i>	70

G

Gestión de Requisitos	39
-----------------------------	----

H

Herramientas de Software	102
Hipótesis	73

I

Ingeniería de Requisitos	37
ISO/IEC 15504	45
ISO/IEC 29110	55

J

Joomla	111, 156
--------------	----------

K

<i>Kanban</i>	58
---------------------	----

L

Licencia de Software	68
----------------------------	----

M

<i>MoProSoft</i>	57
MoReq.....	121, 122
<i>MSF</i>	50

N

NTP-ISO/IEC 12207	54
-------------------------	----

O

<i>Open Source</i>	67
<i>OpenUP</i>	49

P

Plugin.....	112, 163
<i>Prince2</i>	59
Priorización de requisitos	16, 62, 66
Proceso analítico jerárquico	90
Programación Extrema	51

R

Rational RequisitePro	103
Requisitos del sistema	34
Requisitos del usuario	34
Requisitos funcionales	35
Requisitos No funcionales	35
<i>RUP</i>	47

S

Scrum	52
Sistema de Apoyo a Grupos	81
Software	37
Software colaborativo.....	60
Software Comercial	70
Software de Código Abierto	67
Software Libre	69
Software Libre y de Código Abierto	67
Software Privado.....	70
Software Propietario.....	69
Stakeholder	36, 37, 93, 103, 109

U

UML	102
<i>UP46</i>	

V

VAVéase Voto Acumulativo	
Validación de requisitos.....	41
Variable independiente	74
Variables	74
Voto Acumulativo	92, 109, 122

W

Wiki	105
WordPress.....	112, 122, 156, 160, 163

X

XP Véase Programación Extrema

ANEXOS

ANEXO A – Selección de la Técnica de Priorización

Existen numerosas técnicas de priorización de requisitos, tal como se demostró en el estado del arte. Sin embargo, la selección de la técnica correcta que permita la implementación de un modelo viable y funcional requiere un proceso de análisis.

La Priorización de Requisitos es un proceso de toma de decisiones complejo (Hujainah, Binti Abu Bakar, Nasser, Al-haimi, & Zamli, 2021).

Tal como afirmaron (Sher F. , Jawawi, Mohamad, & Muhammad, 2014), es muy difícil justificar la selección de cualquier técnica como la mejor. Ninguna técnica se puede considerar superior a otra en todas las situaciones (Tufail, Qasim, Faisal Masood, Tanvir, & Haider Butt, 2019).

Tal como se sustenta en el Anexo - B «Selección de la Herramienta CMS», se seleccionó a *WordPress* para la implementación de la priorización de requisitos. La implementación de la técnica de priorización se realizará a través de los componentes llamados *plugins*.

Se evalúa cada una de las técnicas de priorización preseleccionadas y se determina cuál es la adecuada para ser implementada con los *plugins* y formar parte del modelo. Si hubiese más de una técnica, se elegirá a la más adecuada, previo análisis y evaluación.

De las técnicas de priorización preseleccionadas se evalúa una a una y se determina cuál es la adecuada para ser implementada con los *plugins* y formar parte del modelo. Si hubiese más de una técnica se elegirá a la más adecuada, previo análisis y

evaluación.

Se hizo una evaluación de la técnica a utilizaren la investigación entre las siguientes:

- Técnica de Asignación numérica (AN): Esta técnica asigna un valor a cada requisito, el cual indica la importancia percibida por el *stakeholder*. Se requeriría de un *plugin* que permita definir categorías tales como: alto, medio y bajo u otra escala, por ejemplo, de cinco. Sin embargo, no existen *plugins* que permitan asignar estos valores para que el usuario los seleccione.
- Proceso Analítico Jerárquico (PAJ): Esta técnica consiste en la selección por pares. El analista que realiza la comparación tiene que decidir qué requisito es más importante, y en qué medida mediante una escala del 1 al 9. Luego, se aplica el algoritmo para obtener la lista priorizada. Tras evaluar los *plugins*, se determina que no existen componentes que permitan la comparación por pares ni su respectiva asignación de un valor escalar.
- Priorización de Valor Orientado (PVO): Esta es una técnica que permite visualizar a los *stakeholders* los requisitos para una adecuada toma de decisiones. Para realizar el proceso de visualización entre los *plugins* disponibles para *WordPress*, no existe alguno que cuente con las características requeridas, al tratarse de una matriz de doble entrada para los datos y los requisitos con respecto a su valor para la organización.
- Voto Acumulativo (VA): Esta técnica permite a los *stakeholders* realizar una priorización de los requisitos, e indicar cuál es su nivel de preferencia. Cada *stakeholder* recibe una cantidad igual de puntos, que los distribuye de acuerdo con su criterio. Este proceso de votación es similar a los usados en las distintas redes sociales, sitios de videos, publicación de noticias y blogs. Existen dos tipos de votación: uno que le asigna un voto positivo o negativo a un post, y otro que le asigna una cantidad de puntos (estrellas) también al post. Es este último tipo de calificación es el que permitiría asignar los votos, para usar la técnica de

priorización de requisitos. Si un *stakeholder* decide asignarle un voto, le asigna una estrella; si decide asignarle cinco, le asigna cinco estrellas. Si para el *stakeholder* no tiene ningún nivel de priorización, no le asigna ninguna estrella. Existen numerosos *plugins* que permiten calificar los *posts* (en los cuales se ingresarán los requisitos) mediante votos o asignación de estrellas.

- Árbol de Búsqueda Binaria (ABB): La técnica de búsqueda binaria implica clasificar a cada requisito, tratándolo como nodo y etiquetándolo. Luego de una búsqueda en el conjunto de *plugins* disponibles, se determina que no existen aquellos que permitan realizar estas tareas de selección y etiquetado de nodos; por lo tanto, no es posible utilizar esta técnica.
- Planificación del Juego (PJ): Esta técnica consiste en reunir o agrupar los requisitos en tres grupos. Es posible asignar a un grupo un requisito, utilizando los *plugins* de votación. Sin embargo, la técnica no indica qué hacer en caso un *stakeholder* no seleccione alguna de estas agrupaciones, por lo cual no es posible implementar esta técnica con algún *plugin* existente.

ANEXO B – Selección de la herramienta CMS

La propuesta de la presente Tesis es trabajar con Sistemas de Gestión de Contenidos (CMS).

En la actualidad se han identificado más de 36,554 tecnologías de Internet que incluyen el análisis, la publicidad, *hosting*, CMS y muchos más (BuiltWith® Pty Ltd, 2019).

Los quince (15) CMS más difundidos a nivel global a diciembre de 2019, según (BuiltWith® Pty Ltd, 2019). Los tres mayores CMS son: *WordPress* (1° puesto – 53.14%), *Joomla* (2° puesto – 8.08%) y *Drupal* (7° puesto – 3.25%), que representan el 61.81% del total de CMS instaladas. *MediaWiki* representa el 0.38% de las plataformas instaladas. Se muestran los quince (15) CMS más difundidos a nivel global, según (BuiltWith® Pty Ltd, 2019, pág. CMS technologies Web Usage Statistics). *Wix* (3° puesto) es un editor para páginas web online, no soporta plugins, así que no cumple con una de las características requeridas. *Squarespace*⁷ (4° puesto) es un CMS que se provee como un SaaS⁸, por tanto, de pago, por ello tampoco cumple con el tipo de licencia requerido por el modelo. *CPanel*⁹ (5° puesto) es un panel de administración de servidores y diseño de páginas web, sin embargo, no permite ampliarle funcionalidades a través de componentes o plugins. No hay tecnología de foros o *wikis* entre los 10 mayores utilizados.

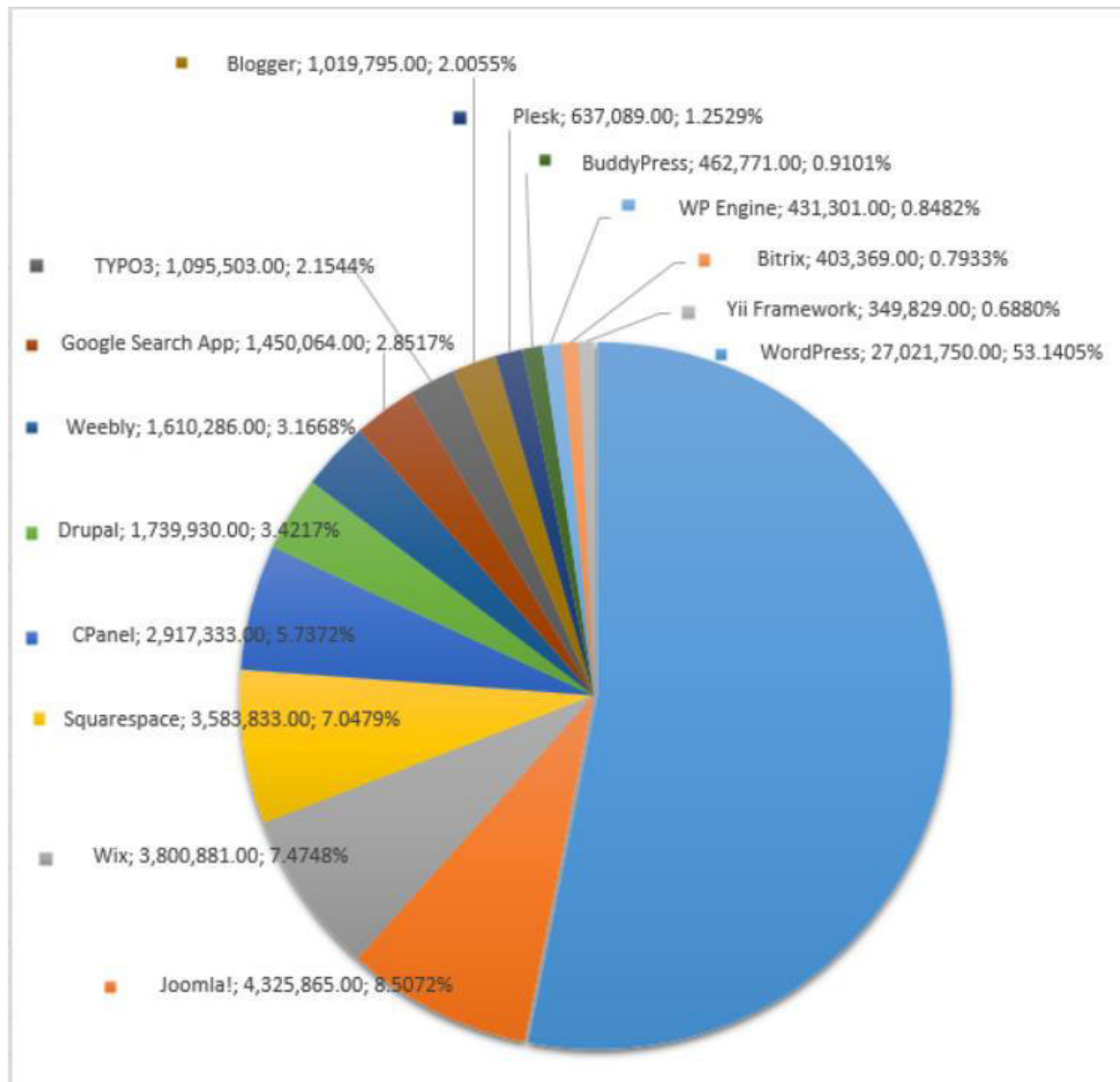
⁷ *Squarespace* proporciona herramientas y servicios creativos para ayudar a cualquier persona construir y gestionar su marca en línea (Squarespace, 2019).

⁸ *Software as a service* (SaaS): Software como servicio.

⁹ Blogger es un servicio que permite la publicación de blogs, alojado en los servidores de Google, se requiere una cuenta de *Gmail*. Incluye múltiples opciones y servicios que facilitan las labores de publicación de entradas, sin tener conocimientos de *HTML*, *JavaScript* ni *CSS*; sino a través del uso de formularios (Google, 2019).

Figura 28

Lista de los quince softwares colaborativos más usados el 2019



Nota. Elaboración propia a partir de los datos de **(BuiltWith® Pty Ltd, 2019)**.

Se requiere elegir la más adecuada herramienta entre las CMS más difundidas y gratuitas.

La lista final de herramientas candidatas incluye a:

1. *WordPress*.
2. *Joomla*.
3. *Drupal*.

WordPress es un *CMS* enfocado a la creación de cualquier tipo de sitio *web*, aunque ha alcanzado una gran relevancia usado para la creación de blogs. Las características incluyen una arquitectura de *plugins* y un sistema de plantillas (WordPress Foundation, 2019). Los *plugins* son pequeños componentes que agregan características específicas.

Joomla es un *CMS* que permite organizar y facilitar la creación de documentos y otros contenidos de un modo cooperativo, es una solución de código abierto y está disponible libremente para cualquiera que desee utilizarlo (The Joomla Project Team, 2019).

Drupal es un *CMS* modular y muy configurable, desarrollado y mantenido por una activa comunidad de usuarios. Destaca por la calidad de su código y de las páginas generadas, el respeto de los estándares de la *web*, y un énfasis especial en la usabilidad y consistencia de todo el sistema (Buytaert, 2015). Se distribuye bajo la licencia *GNU General Public License (GPL)*. Permite el uso de temas y la ampliación de funcionalidades mediante módulos (Drupal, 2021).

Los tres son gratuitos y de código abierto (*open-source*), lo cual significa que se tiene acceso al código fuente en el que están programados, y por tanto se pueden modificar, lo cual es una ventaja respecto a *CMS* propietarios, como *Blogger* o *Sitefinity*.

Para elegir la herramienta *CMS* más adecuada, se definió un conjunto de parámetros (descritos en la Tabla 7), que incluyen criterios y objetivos fácilmente verificables respecto a las características de los *CMS*. Para cada criterio de evaluación se halla el dato. La herramienta que obtenga el primer lugar recibe tres (03) puntos, la segunda dos (02) y la tercera un punto (01). Si tienen el mismo valor, recibirán el mismo puntaje.

Tabla 8*Comparación de las herramientas CMS candidatas*

	Herramienta	WordPress		Joomla		Drupal	
Nº	Resultados	Dato	Puntaje	Dato	Puntaje	Dato	Puntaje
1.	Popularidad <i>CMS open source</i>	1er. lugar	3	3er. lugar	2	4to. lugar	1
2.	<i>Websites</i> activos	>30 millones	3	>2,1 millones	2	>1,6 millones	1
3.	Resultados en <i>Google Trends</i> ¹⁰	78	3	54	2	73	1
4.	Temas gratuitos ¹¹	+8,900	3	+900	1	+1,800	2
5.	<i>Plugins</i> gratuitos ¹²	+59,000	3	+8,700	1	+29,000	2
6.	Tiempo de instalación ¹³	5 min.	3	10 min.	2	10 min.	2
7.	Dificultad para programar <i>plugins</i> ¹⁴	Baja	3	Media	2	Alta	1
8.	Tiempo de respuesta	2.3	3	4.2	2	4.2	2
9.	Tiempo de respuesta con cache	2.1	3	3.1	2	3.1	2
10.	Nivel de seguridad <i>web</i>	Muy bueno	3	Bueno	2	Bueno	2
	Puntaje		30		18		16

Elaboración propia.

¹⁰ Se refiere a las tendencias de búsqueda en el motor *Google*.¹¹ Cuantas plantillas de diseño gratuitas existen disponibles para su descarga.¹² Cuantos *plugins* gratuitos existen disponibles para su descarga.¹³ Un menor tiempo de instalación se considera un mejor atributo.¹⁴ Que tan amigable o complejo es agregarle funcionalidades adicionales a través de *plugins*.

Los criterios para seleccionar la herramienta incluyeron:

1. Número total de descargas: Indica cuantas veces en sus diferentes versiones ha sido descargado el *software*. Un mayor número de descargas es mejor.
2. La cantidad de *websites* activos: Debido a que la propuesta pretende implementar la funcionalidad en una herramienta de amplia difusión y que exista una menor curva de aprendizaje. Un mayor número de *websites* activos es mejor.
3. *Google Trends*: Indica las tendencias en los resultados de búsqueda de *Google Search* (Alphabet , 2021).
4. Temas gratuitos: Es la cantidad de presentaciones o interfaces de usuario que puede utilizar el *software*. Un mayor número de temas gratuitos es mejor.
5. *Plugins* gratuitos: Es la cantidad de extensiones o módulos que se pueden instalar en el *CMS* y que amplían sus funcionalidades. Un mayor número de *plugins* gratuitos es mejor.
6. Tiempo de instalación: Un mayor tiempo de instalación implica que se requiere de un mayor número de pasos, cuenta con más opciones de configuración y es más complejo de instalar. Un menor tiempo es mejor.
7. Dificultad para implementar *plugins*: Porque se requieren algunas funcionalidades no disponibles en los *plugins* existentes. Una menor dificultad como indicador es mejor.
8. Tiempo de respuesta¹⁵: Cuánto tiempo demora en cargar la página en el navegador del cliente. Un menor tiempo de respuesta es mejor.
9. Tiempo de respuesta con *cache*¹⁶: Cuánto tiempo demora en cargar la página en el navegador del cliente, si se utiliza alguna técnica que utilice la *cache* que gestionan los navegadores. Un menor tiempo de respuesta es mejor.
10. Nivel de seguridad *web*: Los nuevos sistemas se están volviendo cada vez más

¹⁵ En su tesis de investigación (Alphabet , 2021) utiliza el criterio de evaluación Tiempo de respuesta.

¹⁶ *Ibíd.*

complejos, requieren interdisciplinariedad y mejor seguridad (Galinier & Laporte, 2018). Un mayor nivel de seguridad es mejor.

El puntaje máximo posible de obtener es de treinta (30) puntos. Los resultados obtenidos fueron, para *WordPress* treinta (30) puntos, *Joomla* obtuvo dieciocho (18) puntos y *Drupal* dieciséis (16).

Como resultado de la evaluación, se eligió a *WordPress* como herramienta para la implementación de la priorización de requisitos.

ANEXO C – Selección del Plugin para la Votación de los Requisitos

El proceso de votación sobre la lista de requisitos en el gestor de contenidos *WordPress* requiere utilizar un *plugin*. Se excluyó de la evaluación a aquellos plugins que requerían editar el código fuente de *WordPress*, tuviesen una arquitectura muy compleja, o no tuviesen una instalación automática y aquellos que fallaron durante la evaluación.

Para elegir el *plugin* adecuado se propuso una lista de tres plugins candidatos, a los cuales se les evaluó bajo una lista de diez criterios:

1. El total de instalaciones activas, que informará sobre en cuántos sitios *web* se está utilizando el *plugin*, será un indicador del funcionamiento del *plugin* en entornos de producción.
2. El rating en *WordPress* son los votos que le asignan los usuarios al *plugin*, de acuerdo al desempeño que ha tenido.
3. El total de descargas indica cuál ha sido el uso del *plugin*, en sus distintas versiones a lo largo del tiempo.
4. El soporte proporcionado es el porcentaje de consultas que realizan los usuarios que absuelve el desarrollador.
5. Se requiere que el *plugin* tenga una licencia compatible con el proyecto, por lo cual se evalúa la facilidad de instalación indica el nivel de dificultad que tiene la instalación del *plugin*.
6. Facilidad de instalación: Si el proceso de instalación es complejo puede inducir al usuario a errores, y no aprovechar las características del plugin o incluso impedir su instalación.
7. La facilidad de configuración indica cuánto esfuerzo se requiere para tener operativo el *plugin*. Un *plugin* complejo es difícil de configurar y puede inducir a error o frustración al usuario.

8. Una buena arquitectura indica cuán viable es implementar nuevas funcionalidades utilizando el *plugin*. Se incluye aspectos de diseño de las tablas. Una mala arquitectura haría inviable la implementación del modelo propuesto.
9. Los requisitos adicionales son aquellos que el desarrollador del *plugin* exige para permitir utilizar su componente en el *website*.
10. Es importante verificar si el *plugin* es compatible con la versión vigente e instalada en el sitio *web*, porque con cada versión de *WordPress* surgen nuevas funcionalidades y ocurren cambios en su estructura, que podrían hacer incompatible el *plugin* con esa versión en particular de *WordPress*.

El puntaje máximo posible de obtener es 30 puntos.

Los resultados obtenidos de la evaluación fueron: para *Universal Star Rating* 21 puntos, *Multi Rating* obtuvo 19 puntos y *Simple Rating* 25.

Como resultado de la evaluación, se eligió a *Simple Rating* como *plugin* para la implementación de la priorización de requisitos.

Tabla 9

Comparación de los plugins de WordPress para implementar el Voto Acumulativo

Herramienta	Universal Star Rating		Multi Rating		Simple Rating	
Resultados	Dato	Puntaje	Dato	Puntaje	Dato	Puntaje
Total de instalaciones activas ¹⁷	+5,000	3	+3,000	2	+1,000	1
Rating en <i>WordPress</i>	4.6/5.00	3	4.5/5.00	2	4.5/5.00	2
Total de descargas	+46,000	2	+56,000	2	+11,000	1
Soporte proporcionado ¹⁸	33% de respuestas	1	80% de respuestas	2	100% de respuestas	3
Licencia compatible con el proyecto	Si	3	Si	3	Si	3
Facilidad de instalación	Excelente	3	Excelente	3	Excelente	3
Facilidad de configuración ¹⁹	Media	2	Baja	1	Excelente	3
Arquitectura del plugin²⁰	Complejo	1	Complejo	1	Sencillo	3
Tiene requisitos adicionales ²¹	No	3	No	3	No	3
Nivel de compatibilidad con todas las versiones de <i>WordPress</i>²²	Bajo	1	Bajo	1	Alto	3
Puntaje		22		20		25

Elaboración propia.

¹⁷ Cuantas instalaciones se encuentran en uso en la actualidad.

¹⁸ Se refiere al porcentaje de consultas del público que absuelve el desarrollador del *plugin*.

¹⁹ Hace referencia a cuan sencillo es configurar el *plugin* para poder trabajar con él.

²⁰ Una arquitectura sencilla facilita el acoplamiento de nuevos módulos y funcionalidades, una arquitectura compleja la dificulta.

²¹ Algunos requisitos adicionales son por ejemplo requerir conexión a internet, registrarse en la página del desarrollador u otros.

²² Debido a las actualizaciones que tiene *WordPress*, es posible que haya incompatibilidades que impida que el *plugin* funcione en ciertas versiones del CMS.

ANEXO D – Implementación del Plugin para el Cálculo del Voto Acumulativo

Para realizar el cálculo de los puntos asignados por los votantes y hallar su equivalente en votos, se requiere la implementación del algoritmo descrito en el capítulo referente al aporte. Para ello, se desarrolla un *plugin* de *WordPress* que realice el cálculo y muestre los resultados.

Para desarrollar el *plugin* se requiere comprender cómo trabaja *WordPress*, cuál es la arquitectura de un *plugin*, y cómo se le brinda información a *WordPress* sobre las funcionalidades que se va a agregar y hacer que se integre en la herramienta *CMS*.

La estructura del *plugin* requiere un archivo *PHP* principal. En la parte superior del archivo se ubica el encabezado del *plugin*, que contiene la información estándar del mismo. Este encabezado permite a *WordPress* identificar el *plugin*, y agregarlo al formulario de gestión de *plugins* para que el administrador pueda ser activado, cargado, y que pueda ejecutar sus funciones. Sin la cabecera, el *plugin* no se activará y nunca se ejecutará. La información proporcionada incluye el nombre del *plugin*, la *URL* del *plugin*, la descripción, el número de la versión, el nombre del autor, y la página *web* del autor.

Figura 29

Información estándar del plugin WordPress

```
1 <?php
2
3 /**
4  * Plugin Name: Voto Acumulativo
5  * Plugin URI: http://itcvalue.com/
6  * Description: Permite elaborar el cálculo de los votos y obtener el listado de los requisitos priorizados.
7  * Version: 1.0
8  * Author: E. Daniel Bravo Loayza
9  * Author URI: http://itcvalue.com/
10 */
```

Elaboración propia.

Luego de la instalación se tiene en la página de *plugins* instalados las opciones

de acceso a los mismos. Desde allí se puede acceder a las opciones del *plugin* Voto Acumulativo.

Figura 30

Plugin instalado de Voto Acumulativo

☐	Voto Acumulativo Desactivar Editar	Permite elaborar el cálculo de los votos y obtener el listado de los requisitos priorizados. Versión 1.0 Por E. Daniel Bravo Loayza Visitar sitio web del plugin
☐	Plugin	Descripción

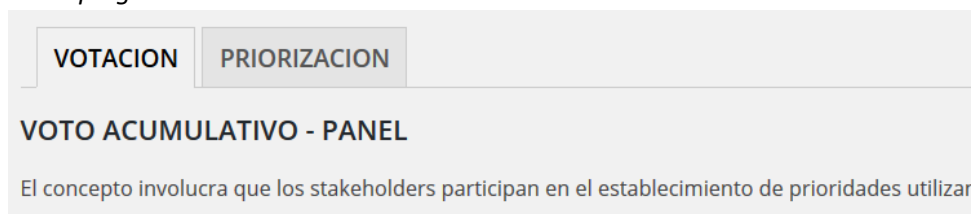
Elaboración propia.

Para acceder al espacio de trabajo del *plugin*, ir a Opciones > Voto Acumulativo.

El espacio de trabajo del *plugin* consta de dos pestañas, como se ve en la **Figura 32**, para la VOTACIÓN y la PRIORIZACIÓN.

Figura 31

Pestañas del *plugin* Voto Acumulativo



Elaboración propia.

La pestaña VOTACIÓN muestra los puntos que los votantes han asignado a cada uno de los requisitos. El *plugin* permite seleccionar a uno de los votantes para visualizar.

La pestaña PRIORIZACIÓN muestra el resultado final de la evaluación, aplicando el algoritmo que evalúa la asignación de los puntos de los votantes (véase la **Figura 32**). Se muestra el ID del requisito, el título del mismo y el puntaje en votos obtenidos. Los requisitos son ordenados por la cantidad de votos obtenidos, en orden descendente.

Figura 32

Pestaña Priorización del *plugin* Voto Acumulativo

VOTACION

PRIORIZACION

PRIORIZACION DE REQUISITOS - PANEL

Efectúa la priorización de los requisitos seleccionados mediante la evaluación de los votos sobre los requerimientos.

post_id	post_title	puntos_totales
46	Firmas electrónicas	243.75
44	Flujos de tareas	175.47
36	Presentación: Impresión	129.83
34	Presentación: Visualización de documentos de archivo	97.92
22	Captura de documentos de archivo	81.63
18	Autenticidad	63.07
16	Control de movimiento de documentos	50.95
12	Pistas de Auditoría	33.52
14	Copias de seguridad y recuperación	27.46
6	Manejo de metadatos	18.37

Elaboración propia.

En la Figura 33, se muestra el código fuente de la pestaña VOTACIÓN. En la Figura 34, se muestra el código fuente de la pestaña PRIORIZACIÓN.

El componente desarrollado Voto Acumulativo utiliza la licencia *GPL*.

Figura 33

Código fuente de la pestaña 'VOTACIÓN'

```
2      $dollars = 100;
3      $user_id = "";
4      $sQuery = "";
5
6      // Verifica a sido seleccionado un usuario, caso contrario muestra los votos de todos los usuarios.
7      if(isset($_GET['userid'])) {
8          if($_GET['userid'] != "-1") {
9              $user_id = " where user_id='".$_GET["userid"]."';
10             $sQuery = "SELECT DISTINCT user_id FROM `wp_spr_votes`".$user_id;
11         } else {
12             $sQuery = "SELECT DISTINCT user_id FROM `wp_spr_votes`";
13         }
14     } else {
15         $sQuery = "SELECT DISTINCT user_id FROM `wp_spr_votes`";
16     }
17     // Ejecuta el query
18     $result = $wpdb->get_results($sQuery);
19
20     echo "<div style='position: relative; min-width: 255px; border: 1px solid #E5E5E5; box-shadow: 0px
21     1px 1px rgba(0, 0, 0, 0.04); background: none repeat scroll 0% 0% #FFF;'>";
22     echo "<table style='border-collapse: collapse; border: 1px solid white; width: 400px;' border='1'>";
23     echo
24     "<tr><td><strong>post_id</strong></td><td><strong>user_id</strong></td><td><strong>points</strong></td><td><strong>votos</strong></td></tr>";
25
26     // Recorrer cada voto de los usuarios
27     foreach($result as $row) {
28         $user_id = $row->user_id;
29
30         $votos = array();
31         $total = 0.0;
32         $k = 0.0;
33         $result2 = $wpdb->get_results("SELECT `post_id`, `user_id`, `points` FROM `wp_spr_votes` INNER
34         JOIN `wp_term_relationships` ON wp_term_relationships.object_id=wp_spr_votes.post_id WHERE
35         wp_term_relationships.term_taxonomy_id=2 and `user_id`='$user_id'");
36         foreach($result2 as $row2) {
37             $total = $total + $row2->points;
38             $votos[] = array($row2->post_id, $row2->points, "0");
39         }
40         $k = $dollars / $total;
41
42         // Para cada requisito le asigna los votos totales obtenidos
43         $result3 = $wpdb->get_results("SELECT wp_posts.ID, wp_posts.post_title FROM `wp_posts` INNER
44         JOIN `wp_term_relationships` ON wp_term_relationships.object_id=wp_posts.ID WHERE
45         wp_term_relationships.term_taxonomy_id=2 order by wp_posts.ID");
46         foreach($result3 as $row3) {
47             echo "<tr>";
48             echo "<td>".$row3->ID."</td>";
49             echo "<td>$user_id</td>";
50             $Existe = false;
51             foreach($votos as $voto) {
52                 if ($voto[0] == $row3->ID) {
53                     $voto[2] = $voto[1] * $k;
54                     echo "<td>$voto[1]</td>";
55                     echo "<td>".number_format($voto[2], 2)."</td>";
56                     $Existe = true;
57                 }
58             }
59             if (!$Existe) {
60                 echo "<td>0</td>";
61                 echo "<td>0.00</td>";
62             }
63             echo "</tr>";
64         }
65     }
66 }
```

Elaboración propia.

Figura 34

Código fuente de la pestaña 'PRIORIZACIÓN'

```
2      $dollars = 100;
3      $requisitos_priorizados = array(); // Lista de requisitos priorizados, que se mostrará al final.
4      $requisitos_votos_totales = array();
5      $wp_spr_votes = array(); // Array que incluye cada uno de los votos
6      $result = $wpdb->get_results("SELECT DISTINCT user_id FROM `wp_spr_votes`");
7
8      // Recorre la votación efectuada para hallar los equivalentes de los votos de los puntos asignados.
9      foreach($result as $row) {
10         $user_id = $row->user_id;
11
12         $votos_user_id = array();
13         $total = 0.0;
14         $k = 0.0;
15         $result2 = $wpdb->get_results( "SELECT `post_id`, `user_id`, `points`, `post_title` FROM
16             `wp_spr_votes` LEFT JOIN `wp_posts` ON `post_id`='ID' where `user_id`='$user_id'");
17
18         foreach($result2 as $row2) {
19             $total = $total + $row2->points;
20             $votos_user_id[] = array($row2->post_id, $row2->points, $row2->post_title);
21         }
22         // Hallar la constante que distribuye los puntos asignados de los usuarios en votos.
23         $k = $dollars / $total;
24
25         // Guarda en el array de votación y su equivalente en votos de los puntos.
26         foreach($votos_user_id as $voto) {
27             $wp_spr_votes[] = array($voto[0], $voto[1], $voto[2], $voto[1] * $k);
28         }
29
30         // Para cada requisito se le asigna los votos totales obtenidos
31         $result3 = $wpdb->get_results("SELECT wp_posts.ID, wp_posts.post_title FROM `wp_posts` INNER JOIN
32             `wp_term_relationships` ON wp_term_relationships.object_id = wp_posts.ID WHERE
33             wp_term_relationships.term_taxonomy_id=2");
34         foreach($result3 as $row) {
35             $ID = $row->ID;
36             $puntos_totales = 0.0;
37
38             foreach($wp_spr_votes as $voto) {
39                 if ($voto[0] == $ID) {
40                     $votos_totales = $votos_totales + $voto[3];
41                 }
42             }
43
44             $requisitos_priorizados[] = array($row->ID, $row->post_title, $votos_totales);
45             $requisitos_votos_totales[] = $votos_totales;
46         }
47
48         // Ordenar los requisitos, en base a los votos totales obtenidos, en orden descendente.
49         array_multisort($requisitos_votos_totales, SORT_DESC, $requisitos_priorizados);
50
51         // Renderizar los resultados finales obtenidos
52         echo "<div style='position: relative; min-width: 255px; border: 1px solid #E5E5E5; box-shadow: 0px
53             1px 1px rgba(0, 0, 0, 0.04); background: none repeat scroll 0% 0% #FFF;'>";
54         echo "<table style='border-collapse: collapse; border: 1px solid white; width: 400px;' border='1'>";
55         echo
56             "<tr><td><strong>post_id</strong></td><td><strong>post_title</strong></td><td><strong>puntos_totales<
57             foreach($requisitos_priorizados as $requisito) {
58                 echo "<tr>";
59                 echo "<td>". $requisito[0]. "</td>";
60                 echo "<td>". $requisito[1]. "</td>";
61                 echo "<td>". number_format($requisito[2], 2). "</td>";
62                 echo "</tr>";
63             }
64         echo "</table>";
65         echo "</div>";
```

Elaboración propia.

ANEXO E – Listado de Requisitos Estándares para la Evaluación del Modelo

A continuación, se presenta una breve lista de diez requisitos extraídos del *MoReq*, utilizada para realizar la prueba de viabilidad del modelo de priorización de requisitos propuesto.

Se eligieron diez (10) requisitos porque es una cantidad que se puede expresar fácilmente, tanto en porcentaje como numéricamente.

Se buscó que la lista de requisitos propuestos para la evaluación sea una muestra representativa del *MoReq*, fácil de entender y de diferenciar los requisitos entre sí.

Los requisitos propuestos son los siguientes:

1. Firmas electrónicas:

El SGD debe ser capaz de conservar la información relativa a las firmas electrónicas, la encriptación y los datos de las correspondientes entidades de validación.

2. Flujos de tareas:

La función de flujo de tareas del SGD debe ofrecer flujos de tareas que son una serie de pasos, cada uno de los cuales sería, por ejemplo, el traslado de un documento o expediente de un participante a otro con vistas a la realización de la acción correspondiente.

3. Presentación: Impresión:

El SGD debe proporcionar al usuario maneras flexibles de imprimir los documentos de archivo y sus correspondientes metadatos, incluida la capacidad

de imprimir uno o varios documentos de archivo junto con los metadatos que determine el usuario.

4. Presentación: Visualización de documentos de archivo:

El SGD debe presentar los documentos de archivo que se hayan recuperado mediante una consulta.

El SGD debe ser capaz de presentar todos los tipos de documentos electrónicos de archivo que determine la organización, de tal modo que conserve la información de los documentos de archivo (por ejemplo, todas las características de la presentación visual y la estructura producidas por el paquete de aplicaciones que los generó) y que presente todos los componentes de un documento electrónico de archivo.

5. Captura de documentos de archivo:

El proceso de registro de documentos de archivo del SGD debe incluir los controles y la funcionalidad adecuados para:

- registrar y gestionar todos los documentos electrónicos de archivo, con independencia del método de codificación empleado y de otras características tecnológicas;
- garantizar que los documentos de archivo se asocian a uno o más expedientes;
- integrarse con el *software* de aplicaciones que genera los documentos de archivo;
- validar y controlar la entrada de metadatos en el SGD.

6. Autenticidad:

El SGD debe restringir el acceso a las funciones del sistema según el perfil del usuario y aplicando controles estrictos de administración del sistema. En caso de que sea posible y proceda, conviene que el SGD muestre una advertencia cada vez que se intente capturar un documento incompleto o incoherente de algún

modo que pueda repercutir en su futura autenticidad.

7. Control de movimiento de documentos:

El SGD debe contar con un instrumento de seguimiento que permita controlar y registrar información sobre la ubicación y movimientos de los expedientes, tanto tradicionales como electrónicos.

8. Copias de seguridad y recuperación:

El SGD debe estar dotado de procedimientos automáticos de copia de seguridad y restauración que permitan realizar copias periódicas de seguridad de todas las clases, los expedientes, los documentos de archivo, los metadatos y los atributos administrativos del repositorio del SGD o de algunos de ellos.

9. Pistas de auditoría:

El SGD debe mantener una pista de auditoría inalterable, capaz de capturar y almacenar de forma automática información sobre:

- todas las acciones relacionadas con los documentos de archivo electrónicos y los expedientes electrónicos;
- el usuario que inicia o realiza la acción;
- la fecha y la hora de la acción.

10. Manejo de metadatos:

El SGD debe soportar los metadatos de expedientes. Una vez se ha capturado un documento de archivo, el sistema debe permitir a los administradores las funcionalidades para añadir o modificar sus metadatos.

ANEXO F – Algoritmo de Implementación de la Técnica Voto Acumulativo - VA

Tal como se describió en el estado del arte, la técnica de Voto Acumulativo implica que los *stakeholders* participen en el establecimiento de prioridades utilizando un número de unidades imaginarias (100 puntos, 100 dólares, etc.), las cuales son distribuidas entre los requisitos a priorizar.

La técnica de voto acumulado, también es conocido como método de 100 dólares. En esta técnica, se dan 100 dólares a los *stakeholders* y ellos asignan un valor en dólares de los 100 asignados a cada requisito (Yaseen, Mustapha, Ur Rahman, Khan, & Kamal, 2019).

En la

Figura 35 se muestra el algoritmo básico de la técnica de Voto Acumulativo.

Figura 35

Algoritmo de la implementación del Voto Acumulativo

```
Leer lista_de_usuarios

para usuario de lista_de_usuarios hacer

    total_puntos = 0
    razon = 0

    Leer lista_de_requisitos

    para requisito de lista_de_requisitos
        total_puntos = total_puntos + requisito.puntos_asignados
        votos_del_usuario[] = array(requisito.id, requisito.puntos_asignados)
    fin-para

    razon = 100 / total_puntos

    para votos de votos_del_usuario hacer
        lista_de_votos = array(votos.id, votos.puntos * razón)
    fin-para

fin-para

// para cada requisito asignarle los votos totales obtenidos
Leer lista_de_requisitos

para requisito de lista_de_requisitos hacer

    voto_id = requisito.voto_id
    puntos_totales = 0

    para voto de lista_de_votos hacer
        si (lista_de_votos.id = voto_id) entonces
            puntos_totales = puntos_totales + lista_de_votos.puntos
        fin-si
    fin-para

    requisitos_priorizados[voto_id] = puntos_totales

fin-para

// Mostrar los resultados
para requisito de requisitos_priorizados hacer

    Escribir requisito.id + ": " + requisito.puntos_totales

fin-para
```

Elaboración propia.

Se deja esta página en blanco como fin del documento.