Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones

Modelo de métricas para apoyar la evaluación de la calidad de los requisitos de software

Mayra Alejandra Castillo Motta
Rubén Darío Dorado Córdoba
Director: Ph.D. MS.c. César Jesús Pardo Calvache

Contenido

1 Introducción

La ingeniería de requisitos es un proceso fundamental en el ciclo de vida de los proyectos de desarrollo de software [1]. Su objetivo es definir de manera precisa las necesidades del negocio y traducirlas en tareas que puedan ser implementadas durante el desarrollo de la solución [2]. Sin embargo, los artefactos resultantes pueden contener defectos de calidad que afectan la comprensión y ejecución del proyecto, que pueden causar problemas como sobrecostos y reprocesos [3]. Para mitigar estos problemas, es crucial contar con herramientas y estrategias que detecten y aborden estos defectos desde el inicio.

Un enfoque para detectar de manera temprana posibles defectos de calidad en los requisitos de software, son los Requirements Smells, en español conocido como olores de requisitos [1]. Un olor en un requisito se define como un indicador de una violación de calidad, que puede conducir a un defecto, con una ubicación concreta y un mecanismo de detección concreto [3]. La norma ISO/IEC/IEEE 29148 [4] propone un conjunto de malos olores asociados a los requisitos de software: (i) lenguaje subjetivo, (ii) adverbios y adjetivos ambiguos, (iii) lagunas, (iv) términos abiertos no verificables, (v) superlativos, (vi) frases comparativas, (vii) declaraciones negativas, (viii) pronombres vagos y (ix) referencias incompletas [3].

Como alcance de proyecto, se seleccionaron cuatro de los olores definidos en la ISO/IEC/IEEE 29148 [4]: (i) lenguaje subjetivo, (ii) adverbios y adjetivos ambiguos, (iii) frases comparativas y (iv) superlativos. Además, se consideraron evaluar seis características de calidad de los requisitos de software: (i) no ambiguo, (ii) consistente, (iii) completo, (iv) verificable, (v) singular y (vi) necesario. Estas características fueron adaptadas a su forma negativa como defectos de calidad. A partir de estos defectos, se definió un conjunto de riesgos que podrían presentarse durante la especificación de requisitos y generar inconvenientes en etapas posteriores del proceso de desarrollo de software.

El modelo de métricas propuesto se diseñó según el enfoque Goal-Question-Metric – GQM, que en español es conocido como Objetivo-Pregunta-Métrica [5]. Se definió un conjunto objetivos de medida enfocados a reducir cada uno de los riesgos propuestos. Posteriormente, se definió un listado de preguntas y métricas para cada objetivo, que buscan medir de forma cuantitativa un conjunto de caracteristicas de calidad relevantes en los requisitos de software.

2 Marco teórico

A continuación, se presenta la definición de conceptos considerados clave para la comprensión de la propuesta presentada en este documento.

2.1 Requisito

Condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado [6].

2.2 Olor de requisito

Un olor de requisito es un indicador de una violación de calidad, que puede conducir a un defecto, con una ubicación concreta y un mecanismo de detección concreto [3]. Nascimento et al. [7] define los olores de requisitos como signos de imprecisión o ambigüedad en la declaración de requisitos, causando altos costos de refactorización.

2.3 ISO/IEC/IEE 29148

Este estándar especifica los procesos requeridos e implementados en las actividades de ingeniería que dan como resultado requisitos para sistemas y productos de software (incluidos los servicios) a lo largo del ciclo de vida de un proyecto [4]. Con base en los criterios de lenguaje natural de la norma ISO/IEC/IEEE 29148, se define el siguiente conjunto de malos olores: (i) lenguaje subjetivo, (ii) adverbios y adjetivos ambiguos, (iii) lagunas, (iv) términos abiertos no verificables, (v) superlativos, (vi) frases comparativas, (vii) declaraciones negativas, (viii) pronombres vagos y (ix) referencias incompletas [3].

2.4 Métrica

Definición operativa que describe en términos muy específicos, un atributo de un producto o un proyecto, y la manera en que el proceso de control de calidad lo medirá. Las métricas de calidad se emplean en los procesos de aseguramiento de la calidad y de control de calidad [8].

2.5 Goal Question Metric (GQM)

El enfoque Goal-Question-Metric – GQM, que en español es conocido como Objetivo-Pregunta-Métrica, propuesto por Basili et al. [5], Caldiera et al. [5] y Rombach et al. [5], está orientado a la definición de una métrica que mida cierto objetivo a través de preguntas. El resultado de la aplicación del enfoque GQM es la especificación de un sistema de medición dirigido a un conjunto particular de problemas y reglas para la interpretación de los datos de medición. El modelo de medición resultante tiene tres niveles: (i) Conceptual (Objetivo), donde se establece un objetivo para cada elemento de medición, este objetivo puede estar relacionado con la mejora de algún aspecto del software o del proceso de desarrollo; (ii) Operacional (Pregunta), donde se formula un conjunto de preguntas de manera objetiva o subjetiva a partir de los objetivos establecidos en el nivel conceptual, estas preguntas buscan detallar los objetivos y ayudar a entender cómo se pueden alcanzar; y (iii) Cuantitativo (Métrica), donde se asocian datos que permiten responder de manera cuantitativa a cada pregunta formulada en el nivel operacional, estos datos se recogen y analizan para proporcionar una medida del cumplimiento del objetivo

3 Modelo de métricas propuesto

A continuación, se describe de manera detallada el modelo de métricas propuesto, que tiene como objetivo proporcionar una estructura sistemática y eficiente para evaluar la calidad de los requisitos de software a partir de la detección de malos olores. Su implementación tiene como meta facilitar la identificación temprana del subconjunto de olores tomado de la norma ISO/IEC/IEE [4]: (i) lenguaje subjetivo, (ii) adverbios y adjetivos ambiguos, (iii) frases comparativas y (iv) superlativos, que permita la comprensión y la medición de la calidad de los requisitos, contribuyendo así a la corrección de los requisitos y optimización del ciclo de vida del desarrollo de software.

3.1 Vista general

3.1.1 Aportes

El modelo de métricas ofrece una metodología sistemática y eficaz para evaluar la calidad de los requisitos de software desde una perspectiva preventiva, permitiendo la identificación temprana

de malos olores, considearos como indicadores de defectos de calidad. Mediante la aplicación de este modelo, las organizaciones podrán realizar una evaluación cuantitativa y objetiva de los requisitos de software, facilitando la detección y corrección de malos olores en las primeras fases del ciclo de vida del desarrollo de software, reduciendo los costos, tiempos y riesgos del proyecto. Asimismo, el modelo de métricas facilitará la comprensión y la medición de la calidad de los requisitos de software, al proporcionar una estructura clara para formular preguntas específicas y seleccionar las métricas adecuadas, según el olor detectado en los requisitos.

3.1.2 Alcance

El alcance del modelo de métricas se centra en los requisitos de software expresados en lenguaje natural, independientemente de la técnica de especificación que se utilice, ya sea documentos, casos de uso, historias de usuario u otras. El modelo de métricas se fundamenta en el enfoque Goal-Question-Metric – GQM, que ofrece una metodología para definir objetivos, preguntas y métricas orientados a evaluar la calidad de los requisitos. El modelo propuesto se puede aplicar tanto en la fase de análisis como en la fase de validación de los requisitos, esto con el fin de verificar y mejorar su calidad en términos de: (i) no ambiguo, (ii) consistente, (iii) completo, (iv) verificable, (v) singular y (vi) necesario. Estas características de calidad fueron seleccionadas de la norma ISO/IEC/IEE 29148 [4], al verse influenciadas directamente por la presencia de los olores que forman parte del subconjunto seleccionado

3.1.3 Criterios empleados

Los criterios considerados para el diseño del modelo de métricas fueron:

- Generar métricas específicas, fáciles de entender y de utilizar por parte de las empresas de desarrollo de software, que permitan evaluar la calidad de los requisitos de sus proyectos.
- Seleccionar características de calidad representativas para los requisitos de software, que son de vital importancia para el éxito o fracaso de los proyectos de software.
- El modelo de métricas debe ser flexible y adaptable, permitiendo ajustarse a las necesidades específicas de cada empresa y proyecto, teniendo en cuenta que cada requisito y contexto de software puede tener particularidades únicas.

Fundamentar las métricas en metodologías y estándares internacionales como el enfoque GQM, para obtener un modelo de métricas sólido y confiable que pueda ser adaptado a la industria del software

3.2 Método de trabajo

Se utilizó como base el enfoque GQM [5] para la definición del modelo de métricas propuesto, considerando especialmente aquellos elementos más relevantes que ofrecen datos confiables. Este enfoque tiene como finalidad desarrollar métricas coherentes que posibiliten una medición clara del impacto de los riesgos que se abordan en el proyecto.

3.3 Definición del modelo de métricas

3.3.1 Mecanismo de definición de las métricas

A partir de una búsqueda en la literatura de los olores de requisitos, fueron seleccionados cuatro olores como alcance de este modelo: (i) lenguaje subjetivo, (ii) adverbios y adjetivos ambiguos,

(iii) frases comparativas y (iv) superlativos. Estos cuatro olores fueron tomados de la norma ISO/IEC/IEE 29148 [4], de esta misma norma también fueron seleccionadas seis características de calidad a considerar de los requisitos de software, entre ellas: (i) no ambiguo, (ii) consistente, (iii) completo, (iv) verificable, (v) singular y (vi) necesario. Estas características fueron adaptadas a su forma negativa como defectos de calidad, como se muestra en la Tabla 1. Dichos defectos de calidad fueron seleccionados como alcance de nuestro modelo porque consideramos que son los que tienen una relación directa con los cuatro olores seleccionados. En el modelo propuesto se busca ampliar y adaptar el conocimiento en este campo, como mejora de la calidad de los requisitos de software escritos en lenguaje español. En la Figura 1 se muestra un resumen de los elementos que conforman y se relacionan con el modelo de métricas propuesto.

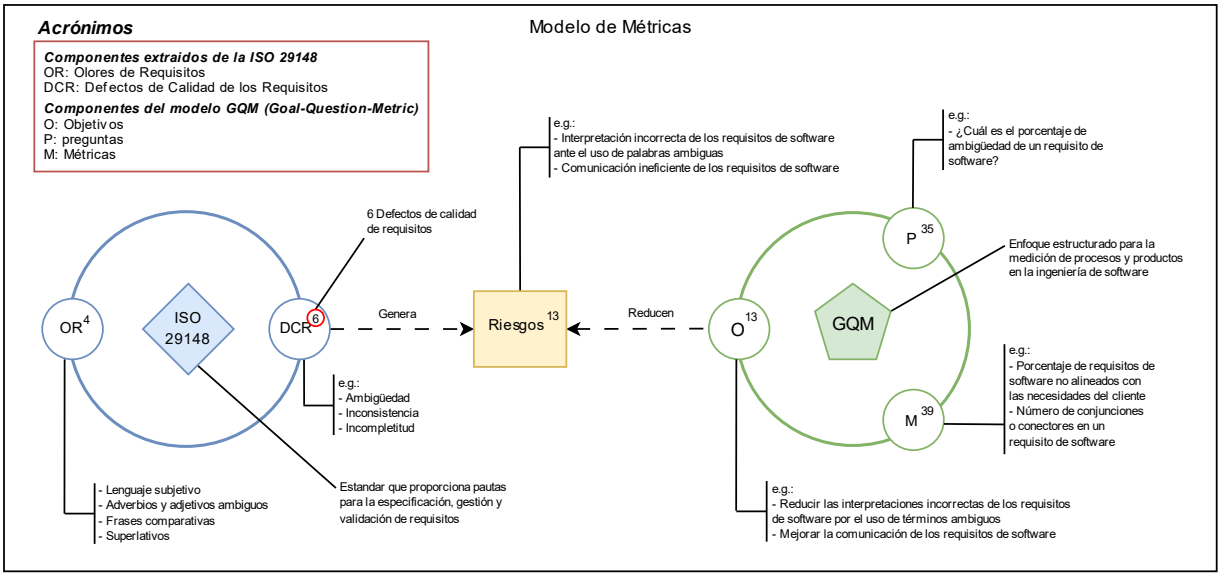


Figura 1. Estructura del modelo de métricas propuesto.

Tabla 1. Defectos de calidad a medir en el modelo de métricas.

#	Nombre	Adaptado de	Descripción
1	Ambigüedad	ISO/IEC/IEE 29148 [4]	El requisito puede interpretarse de más de una manera. El requisito no se establece de forma sencilla y es difícil de entender.
2	Inconsistencia	ISO/IEC/IEE 29148 [4]	El requisito tiene conflictos con otros requisitos.
3	Incompletitud	ISO/IEC/IEE 29148 [4]	Requisitos que necesitan ampliación porque no son medibles y describen de forma incompleta las capacidades y características necesarias para satisfacer las necesidades de las partes interesadas.
4	No verificabilidad	ISO/IEC/IEE 29148 [4]	El requisito no tiene los medios suficientes para demostrar que el sistema satisface el requerimiento solicitado. Es difícil recopilar pruebas que demuestren que el sistema puede satisfacer el requisito especificado.
5	Compuesto	ISO/IEC/IEE 29148 [4]	Requisitos que están compuestos por más de una necesidad, y que podría dividirse.
6	Innecesario	ISO/IEC/IEE 29148 [4]	El requisito no define una capacidad, característica, restricción y/o factor de calidad esencial, al ser eliminado, ser sustituido por otras capacidades del producto o proceso.

La medición de calidad de este modelo está basada en los defectos de calidad descritos en la Tabla 1, aclarando que al evaluar este subconjunto de defectos, buscamos tener un alcance mayor a los cuatro olores seleccionados de la literatura, aunque son el foco principal del modelo. A partir de los defectos de calidad seleccionados, se definieron los riesgos presentados en la Tabla 2, riesgos que posteriormente fueron usado como base para la definición de los objetivos del modelo.

Tabla 2. Conjunto de riesgos.

#	Defecto de calidad	Riesgo	
		Id.	**Nombre**
1	Ambigüedad	R1	Interpretación incorrecta de los requisitos de software ante el uso de palabras ambiguas.
		R2	interpretación incorrecta de los requisitos de software por los diferentes stakeholders, por la falta de uso de terminología estándar.
2	Inconsistencia	R3	Los requisitos de software no se alinean con las expectativas o necesidades del cliente.
		R4	Uso de términos opuestos que generan confusión en el ciclo de desarrollo del software.
		R5	Comunicación ineficiente de los requisitos de software.
3	Incompletitud	R6	Dificultad para entender los detalles de un requisito por el uso de palabras ambiguas.
		R7	Desafíos técnicos no previstos por el uso de palabras que no permiten dimensionar retos o limitaciones técnicas.
4	No verificabilidad	R8	Incapacidad para determinar si el software cumple con el requerimiento esperado por el cliente, al no redactar de forma precisa las funcionalidades del software.
		R9	Incremento en los errores o defectos del software, al no poder diseñar o ejecutar pruebas por la falta de información clara y precisa en los requisitos de software.
5	Compuesto	R10	Incremento en la estimación de los requisitos de software, debido a la redacción de requisitos de software demasiado grandes.
6	Innecesario	R11	Especificación de requisitos de software innecesarios dentro del proyecto que aumentan costos y tiempo de desarrollo.

Acrónimos utilizados: Identificador (Id.).

3.3.2 Objetivos (Goals)

En la Tabla 3 se presentan los objetivos de medida asociados a cada uno de los riesgos presentados en la Tabla 2.

Tabla 3. Objetivos asociados a los riesgos.

Id.	Objetivo	Riesgo
O1	Reducir las interpretaciones incorrectas de los requisitos de software por el uso de términos ambiguos.	R1
O2	Aumentar el uso de terminología estándar que pueda ser interpretada de manera correcta por los diferentes stakeholders de un proyecto.	R2
O3	Reducir la cantidad de requisitos de software que no reflejen las necesidades y/o expectativas del cliente.	R3
O4	Reducir el uso de términos o conceptos opuestos en los requisitos de software.	R4
O5	Mejorar la comunicación de los requisitos de software entre los miembros del equipo.	R5
O6	Mejorar el nivel de detalle de los requisitos de software al detectar las palabras que pueden llevar a omitir de detalles importantes.	R6
O7	Reducir la cantidad de retos técnicos no previstos causados por el uso de palabras que no aportan información completa.	R7

O8	Reducir la imprecisión en la redacción de los requisitos de software para mejorar el cumplimiento de las expectativas del cliente.	R8
O9	Mejorar la claridad de los requisitos de software para facilitar el diseño y ejecución de pruebas, reduciendo así los errores o defectos del software.	R9
O10	Reducir el tiempo de estimación de los requisitos de software al identificar cuáles son demasiado grandes.	R10
O11	Reducir la cantidad de requisitos de software innecesarios que no contribuyen valor al producto final.	R11

Acrónimos utilizados: Identificador (Id.).

3.3.3 Preguntas (Questions)

En la Tabla 4 se presentan las preguntas asociadas a los objetivos de medida de la Tabla 3.

Tabla 4. Preguntas asociadas a los objetivos de medida.

Id.	Preguntas	Objetivo
P1	¿Cuántos términos que componen un requisito de software son ambiguos?	O1
P2	¿Cuál es el porcentaje de ambigüedad de un requisito de software?	
P3	¿Cuál es el porcentaje de ambigüedad de un conjunto de requisitos de software?	
P4	¿Cuál es porcentaje de heterogeneidad de los requisitos de software por el uso de sinónimos?	O2
P5	¿Cuántos requisitos de software no están alineados con las necesidades del cliente?	O3
P6	¿Cuál es el porcentaje de requisitos de software que no están alineados con las necesidades del cliente?	
P7	¿Cuál es el porcentaje de requisitos de software que presentan malos olores que podrían generar expectativas irreales sobre las necesidades del cliente?	
P8	¿Cuántos términos opuestos se presentan en un requisito de software?	O4
P9	¿Cuál es la frecuencia de uso de términos opuestos en un requisito de software?	
P10	¿Cuál es el tiempo total dedicado a la comprensión de los requisitos de software?	
P11	¿Cuál es el tiempo promedio dedicado a la comprensión de requisitos de software?	
P12	¿Cuánto esfuerzo se estimó en la comunicación de los requisitos de software?	O5
P13	¿Cuánto esfuerzo se realizó en la comunicación de los requisitos de software?	
P14	¿Cuál es la relación entre el esfuerzo realizado en la comunicación de los requisitos de software respecto al esfuerzo estimado?	
P15	¿Cuánto tiempo total se ha dedicado en la aclaración de requisitos de software?	
P16	¿Cuál es el tiempo promedio dedicado en la aclaración de requisitos de software?	
P17	¿Cuántos términos pueden generar omisión de detalles importantes en un requisito de software?	O6
P18	¿Cuál es el porcentaje de términos que pueden generar omisión de detalles importantes en un requisito de software?	
P19	¿Los requisitos de software fueron revisados por todos los stakeholders?	
P20	¿Cuál es el porcentaje de requisitos de software que fueron revisados por todos los stakeholders?	
P21	¿Cuántos términos de un requisito de software pueden enmascarar desafíos técnicos?	O7
P22	¿Los requisitos de software fueron revisados por desarrolladores con experiencia?	
P23	¿Cuál es el porcentaje de requisitos de software que fueron revisados por desarrolladores con	

	experiencia?	
P24	¿Cuántos términos imprecisos se identifican en un requisito de software?	O8
P25	¿Cuál es el porcentaje de imprecisión de un requisito de software?	
P26	¿Cuántas pruebas de software no se pueden diseñar debido a la falta de claridad o ausencia de los criterios de aceptación o escenarios en los requisitos de software?	O9
P27	¿Cuántas semanas fueron estimadas para implementar una historia de usuario?	O10
P28	¿Cuántos términos tiene un requisito de software?	
P29	¿Cuántas conjunciones o conectores contiene un requisito de software?	
P30	¿Cuántos de los requisitos de software definidos son innecesarios para las funcionalidades del producto?	O11
P31	¿Cuál es el porcentaje de los requisitos de software innecesarios?	

Acrónimos utilizados: Identificador (Id.).

3.3.4 Métricas (Metrics)

En esta sección se presenta la definición e interpretación de las métricas propuestas para cada uno de los objetivos del modelo.

3.3.4.1 Definición de métricas base y derivadas

La definición de las métricas de modelo propuesto está orientada a alcanzar los objetivos y los riesgos definidos en las secciones anteriores. Asimismo, mediante el enfoque GQM, se establecieron las preguntas que responden a las métricas propuestas que pueden ser consultadas en la Tabla 5, la cual organiza la información de la siguiente manera: en la columna: ¿Qué mide?, se establece el atributo que mide, en la columna: ¿Cómo lo mide?, se indica la métrica, su descripción, el tipo de métrica y la escala de cada métrica [9]. El modelo de métricas está conformado por un total de 34 métricas bases y derivadas; a partir de estas, se definieron 16 métricas de tipo indicador qué por medio del uso de fórmulas, permiten medir la calidad de los requisitos de software. Las métricas de tipo indicador están distribuidas entre los seis (6) defectos de calidad de la siguiente manera: (i) ambigüedad con 2 métricas, (ii) inconsistencia con 6 métricas, (iii) incompletitud con 3 métricas, (iv) no verificabilidad con 2 métricas, (v) compuesto con 2 métricas e (vi) innecesario con 1 métrica.

Por otra parte, en cada métrica se propone un criterio de decisión, el cual ayuda a evaluar los resultados obtenidos y provee al usuario final del modelo indicadores de calidad que le permiten tomar decisiones respecto a los requisitos que está evaluando o para futuros requisitos, mejorando el proceso de especificación o evitando repetir errores.

Tabla 5. Conjunto de métricas bases y derivadas propuestas.

#	¿Qué mide?	¿Cómo lo mide?			
		Métrica	**Descripción**	**Tipo**	**Escala**
1	Porcentaje	PA	Porcentaje de ambigüedad de un requisito de software.	Indicador	Porcentaje
2	Cantidad	NTA	Número de términos ambiguos de un requisito de software.	Base	Términos
3	Cantidad	NTT	Número total de términos de un requisito de software.	Base	Términos

4	Porcentaje	PGA	Porcentaje global de ambigüedad de un conjunto de requisitos de software.	Indicador	Porcentaje
5	Cantidad	NTR	Número total de requisitos de software evaluados.	Base	Requisitos
6	Porcentaje	PHRS	Porcentaje de heterogeneidad entre los requisitos de software evaluados por el uso de sinónimos.	Indicador	Porcentaje
7	Cantidad	NRSOR	Número de requisitos de software que contienen términos que son sinónimos de términos de otros requisitos.	Base	Requisitos
8	Porcentaje	PRNA	Porcentaje de requisitos de software no alineados con las necesidades del cliente.	Indicador	Porcentaje
9	Cantidad	NRNA	Número de requisitos de software no alineados con las necesidades del cliente.	Base	Requisitos
10	Porcentaje	PRCEI	Porcentaje de requisitos de software que crean expectativas irreales de las necesidades del cliente.	Indicador	Porcentaje
11	Cantidad	NRCEI	Número de requisitos de software que crean expectativas irreales de las necesidades del cliente.	Base	Requisitos
12	Frecuencia	FTOR	Frecuencia de uso de términos opuestos en un requisito de software.	Indicador	Razón
13	Cantidad	NTOR	Número de términos opuestos de un requisito de software.	Base	Términos
14	Promedio	PTCR	Promedio de tiempo de comprensión de requisitos de software.	Indicador	Horas/Requisito
15	Tiempo	TTCR	Tiempo total de comprensión de los requisitos de software.	Base	Horas
16	Razón	RECR	Relación del esfuerzo realizado en la comunicación de los requisitos de software respecto al esfuerzo estimado.	Indicador	Razón
17	Esfuerzo	ERCR	Esfuerzo realizado en la comunicación de los requisitos de software.	Base	Horas/Persona
18	Esfuerzo	EECR	Esfuerzo estimado en la comunicación de los requisitos de software.	Base	Horas/Persona
19	Promedio	PTAR	Promedio de tiempo de aclaración de requisitos de software.	Indicador	Horas/Requisito
20	Tiempo	TTAR	Tiempo total de aclaración de requisitos de software.	Base	Horas
21	Porcentaje	PRRS	Porcentaje de requisitos de software revisados por todos los stakeholders.	Indicador	Porcentaje
22	Cantidad	NRRS	Número de requisitos de software revisados por todos los stakeholders.	Base	Requisitos
23	Cantidad	NTEDT	Numero términos que pueden esconder desafíos técnicos.	Indicador	Términos
24	Afirmación	TEDT	Indica si un término en específico puede esconder desafíos técnicos.	Base	Si/No
25	Porcentaje	PRRDE	Porcentaje de requisitos de software revisados por desarrolladores con experiencia.	Indicador	Porcentaje
26	Cantidad	NRRDE	Número de requisitos de software revisados por desarrolladores con experiencia.	Base	Requisitos
27	Porcentaje	PI	Porcentaje de imprecisión de un requisito de software.	Indicador	Porcentaje
28	Cantidad	NTI	Número de términos imprecisos en un requisito de software.	Base	Términos
29	Cantidad	NPND	Número de pruebas de software que no se pueden diseñar.	Indicador	Pruebas
30	Cantidad	NSEHU	Número de semanas estimadas para implementar una historia de usuario.	Indicador	Semanas
31	Cantidad	NCCR	Número de conjunciones o conectores en un requisito de software.	Indicador	Términos

32	Afirmación	CC	Indica si un término en específico es un conector o una conjunción.	Base	Si/No
33	Porcentaje	PRI	Porcentaje de requisitos de software innecesarios.	Indicador	Porcentaje
34	Cantidad	NRI	Número de requisitos de software innecesarios.	Base	Requisitos

3.3.4.2 Métricas para medir la ambigüedad de los requisitos

Para medir la ambigüedad de los requisitos de software se han definido dos (3) métricas. Inicialmente, la métrica PA (Porcentaje de ambigüedad de un requisito de software) es descrita en la Tabla 6, y permite medir el grado de ambigüedad de un requisito de software, a partir de la cantidad de términos ambiguos presentes en el requisito.

Tabla 6. Métrica de porcentaje de ambigüedad.

<table>
<tr><th colspan="4">Porcentaje de ambigüedad de un requisito de software</th></tr>
<tr><td>Acrónimo</td><td colspan="3">PA</td></tr>
<tr><td>Propósito</td><td colspan="3">Medir el porcentaje de ambigüedad de un requisito de software, a partir de la cantidad de términos ambiguos presentes en el requisito.</td></tr>
<tr><td>Unidad</td><td colspan="3">Porcentaje</td></tr>
<tr><td>Escala</td><td colspan="3">[0, 100]</td></tr>
<tr><td>Ecuación</td><td colspan="3">$$PA = \frac{NTA}{NTT} * 100$$</td></tr>
<tr><td>Variables</td><td colspan="3">• PA: Porcentaje de ambigüedad de un requisito de software.
• NTA: Número de términos ambiguos de un requisito de software.
• NTT: Número total de términos de un requisito de software.</td></tr>
<tr><td rowspan="5">Criterio de decisión</td><td colspan="3">La interpretación del resultado del porcentaje de ambigüedad (PA) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Sin ambigüedad</td><td>El requisito está definido de manera clara, sin lugar a interpretaciones diversas.</td><td>PA = 0</td></tr>
<tr><td>Ambigüedad leve</td><td>La mayoría de los términos son claros y se puede entender la intención general del requisito de software.</td><td>0% < PA ≤ 2%</td></tr>
<tr><td>Ambigüedad regular</td><td>El requisito de software presenta una cantidad considerable de términos que no son claros, lo que puede llevar a interpretaciones diferentes.</td><td>2% < PA ≤ 10%</td></tr>
<tr><td>Ambigüedad alta</td><td>La descripción del requisito de software es confusa y no es clara, lo que dificulta entender lo que se espera del software.</td><td>10% < PA ≤ 100%</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">Se tiene el siguiente ejemplo de requisito de software: “Como usuario de la aplicación deseo pedir domicilios que lleguen a mi dirección en un tiempo mínimo para poder cenar a tiempo”, podemos analizar:
• Se encuentra el término “mínimo”, considerado un olor de requisito relacionado con la ambigüedad.
• El requisito de ejemplo consta de 22 términos.
Aplicando la fórmula tenemos:
$$NTA = 1\ término$$
$$NTT = 22\ términos$$
Conociendo esto y aplicando la fórmula podemos determinar:
$$PA = \frac{1}{22} * 100 = \ 4.54\ \%$$
El resultado es 4.54%, que se cataloga como ambigüedad regular.</td></tr>
</table>

Por otro lado, la métrica PHRS (Porcentaje de heterogeneidad de los requisitos de software por el uso de sinónimos) es descrita en la Tabla 7, y permite determinar el grado de heterogeneidad que presenta un conjunto de requisitos de software. Esta métrica proporciona una medida cuantitativa para identificar y abordar posibles ambigüedades derivadas del uso de sinónimos en los requisitos, contribuyendo así a mejorar la calidad y la comprensión del conjunto de requisitos de software.

Tabla 7. Métrica de porcentaje de heterogeneidad.

<table>
<tr><th colspan="4">Porcentaje de heterogeneidad de los requisitos de software por el uso de sinónimos</th></tr>
<tr><td>Acrónimo</td><td colspan="3">PHRS</td></tr>
<tr><td>Propósito</td><td colspan="3">Medir el grado de claridad y uniformidad en la terminología utilizada en un conjunto de requisitos de software.</td></tr>
<tr><td>Unidad</td><td colspan="3">Porcentaje</td></tr>
<tr><td>Escala</td><td colspan="3">[0, 100]</td></tr>
<tr><td>Ecuación</td><td colspan="3">$$PHRS = \frac{NRSOR}{NTR} * 100$$</td></tr>
<tr><td>Variables</td><td colspan="3">• PHRS: Porcentaje de heterogeneidad entre los requisitos de software evaluados por el uso de sinónimos.
• NRSOR: Número de requisitos de software que contienen términos que son sinónimos de términos de otros requisitos.
• NTR: Número total de requisitos de software evaluados.</td></tr>
<tr><td rowspan="5">Criterio de decisión</td><td colspan="3">La interpretación del resultado del porcentaje de heterogeneidad (PHRS) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Sin heterogeneidad</td><td>Todos los requisitos de software utilizan términos que no tienen sinónimos en otros requisitos; es decir, existe claridad y uniformidad en la terminología de los requisitos.</td><td>PHRS = 0</td></tr>
<tr><td>Heterogeneidad leve</td><td>Se presentan pocos términos con sinónimos, pero la mayoría de los requisitos son comprendidos de manera correcta.</td><td>0% < PHRS ≤ 20%</td></tr>
<tr><td>Heterogeneidad regular</td><td>Se presenta una cantidad considerable de términos que tienen sinónimos en otros requisitos, generando diversas interpretaciones.</td><td>20% < PHRS ≤ 40%</td></tr>
<tr><td>Heterogeneidad alta</td><td>No existe claridad y uniformidad en la terminología de los requisitos.</td><td>40% < PHRS ≤ 100%</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">Teniendo en cuenta las siguientes historias de usuario:
• HU001: Como usuario del banco deseo poder realizar transacciones a otras cuentas desde la aplicación para evitar la necesidad de ir a una sucursal física.
• HU002: Como empleado del banco deseo poder ver el valor de las transferencias enviadas y recibidas por una persona para conocer su flujo de caja.
• HU003: Como usuario del banco deseo poder ver el saldo de mi cuenta desde la aplicación móvil para conocer mejor mi estado financiero.
Se identifica que dos de tres historias de usuario, HU001 y HU002 contienen dos palabras que son relacionadas: “transacciones” y “transferencias”.
Aplicando la fórmula tenemos:
$$NRSOR = 2\ requisitos$$
$$NTR = 3\ requisitos$$
Conociendo esto y aplicando la fórmula podemos determinar:
$$PHRS = \frac{2}{3} * 100 = 66.66\ \%$$
El resultado es 66.66%, que se cataloga como grado de heterogeneidad alto.</td></tr>
</table>

3.3.4.3 Métricas para medir la inconsistencia de los requisitos

Para medir la inconsistencia de los requisitos de software se han definido seis (6) métricas. Inicialmente, la métrica PRNA (Porcentaje de requisitos de software no alineados con las necesidades del cliente) es descrita en la Tabla 8, y permite conocer el nivel de discrepancia entre los requisitos y las necesidades del cliente.

Tabla 8. Métrica de porcentaje de requisitos no alineados con las necesidades del cliente.

<table>
<tr><th colspan="4">Porcentaje de requisitos de software no alineados con las necesidades del cliente</th></tr>
<tr><td>Acrónimo</td><td colspan="3">PRNA</td></tr>
<tr><td>Propósito</td><td colspan="3">Evaluar la conformidad de los requisitos de software con las necesidades y expectativas del cliente.</td></tr>
<tr><td>Unidad</td><td colspan="3">Porcentaje</td></tr>
<tr><td>Escala</td><td colspan="3">[0, 100]</td></tr>
<tr><td>Ecuación</td><td colspan="3">$$PRNA = \frac{NRNA}{NTR} * 100$$</td></tr>
<tr><td>Variables</td><td colspan="3">• PRNA: Porcentaje de requisitos de software no alineados con las necesidades del cliente.
• NRNA: Número de requisitos de software no alineados con las necesidades del cliente.
• NTR: Número total de requisitos de software evaluados.</td></tr>
<tr><td rowspan="5">Criterio de decisión</td><td colspan="3">La interpretación del resultado del porcentaje de requisitos no alineados (PRNA) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Alineación excelente</td><td>Todos los requisitos de software se ajustan a las necesidades y expectativas del cliente.</td><td>PRNA = 0%</td></tr>
<tr><td>Alineación aceptable</td><td>Se presentan pocos requisitos que requieren ajustes para mejorar la concordancia con las expectativas del cliente.</td><td>0% < PRNA ≤ 20%</td></tr>
<tr><td>Alineación insuficiente</td><td>Se presenta una cantidad considerable de requisitos que deben ser modificados para satisfacer las expectativas del cliente.</td><td>20% < PRNA ≤ 50%</td></tr>
<tr><td>Alineación deficiente</td><td>La mayoría o todos requisitos carecen de concordancia con las necesidades y expectativas del cliente.</td><td>50% < PRNA ≤ 100%</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">En un nuevo proyecto de software, se han definido un total de 15 requisitos de software. Sin embargo, luego de una revisión con el cliente, se identificó que 8 de estos requisitos no reflejaban sus necesidades (del cliente).
Aplicando la fórmula tenemos:
$$NRNA = 8\ requisitos$$
$$NTR = 15\ requisitos$$
Conociendo esto y aplicando la fórmula podemos determinar:
$$PRNA = \frac{8}{15} * 100 = \ 53.33\ \%$$
El resultado es 53.33%, que se cataloga como grado de alineación deficiente, es decir que la mayoría de los requisitos de software definidos no reflejan las necesidades y expectativas del cliente.</td></tr>
</table>

Por otro lado, la métrica PRCEI (Porcentaje de requisitos de software que crean expectativas irreales de las necesidades del cliente) es descrita en la Tabla 9, y permite determinar el porcentaje de requisitos que distorsionan las expectativas del cliente, a partir de la presencia de términos que se consideran malos olores: (i) lenguaje subjetivo, (ii) adverbios y adjetivos ambiguos, (iii) comparativos o (iv) superlativos.

Tabla 9. Métrica de porcentaje de requisitos que crean expectativas irreales de las necesidades del cliente.

<table>
<tr><th colspan="4">Porcentaje de requisitos de software que crean expectativas irreales de las necesidades del cliente</th></tr>
<tr><td>Acrónimo</td><td colspan="3">PRCEI</td></tr>
<tr><td>Propósito</td><td colspan="3">Medir el porcentaje de requisitos que distorsionan las necesidades del cliente, a partir de la cantidad de términos que se consideran malos olores.</td></tr>
<tr><td>Unidad</td><td colspan="3">Porcentaje</td></tr>
<tr><td>Escala</td><td colspan="3">[0, 100]</td></tr>
<tr><td>Ecuación</td><td colspan="3">$$PRCEI = \frac{NRCEI}{NTR} * 100$$</td></tr>
<tr><td>Variables</td><td colspan="3">• PRCEI: Porcentaje de requisitos de software que crean expectativas irreales de las necesidades del cliente.
• NRCEI: Número de requisitos de software que crean expectativas irreales de las necesidades del cliente.
• NTR: Número total de requisitos de software evaluados.</td></tr>
<tr><td>Criterio de decisión</td><td colspan="3">La interpretación del resultado del porcentaje de requisitos que crean expectativas irreales (PRCEI) está dada por:
<table>
<tr><th>Grado</th><th>Descripción</th><th>Rango</th></tr>
<tr><td>Sin expectativas irreales</td><td>Los requisitos están definidos de manera precisa y sin elementos que puedan distorsionar las expectativas del cliente.</td><td>PRCEI = 0%</td></tr>
<tr><td>Expectativas irreales leve</td><td>Se presentan pocos requisitos que contienen malos olores que distorsionan las expectativas del cliente.</td><td>0% < PRCEI ≤ 20%</td></tr>
<tr><td>Expectativas irreales regulares</td><td>Se presenta una cantidad considerable de requisitos que contienen malos olores que distorsionan las expectativas del cliente.</td><td>20% < PRCEI ≤ 50%</td></tr>
<tr><td>Expectativas irreales altas</td><td>La mayoría o todos los requisitos presentan malos olores que distorsionan las expectativas del cliente.</td><td>50% < PRCEI ≤ 100%</td></tr>
</table></td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">En un nuevo proyecto de software, se han definido 15 requisitos de software. Sin embargo, luego de una revisión, se identificó que 4 de estos requisitos contenían malos olores que generaban expectativas irreales sobre las necesidades del cliente.

Aplicando la fórmula tenemos:
$$NRCEI = 4\ requisitos$$
$$NTR = 15\ requisitos$$
Conociendo esto y aplicando la fórmula podemos determinar:
$$PRCEI = \frac{4}{15} * 100 = \ 26.66\ \%$$
El resultado es 26.66%, que se cataloga como grado expectativas irreales altas, es decir que varios requisitos de software que presentan malos olores pueden crear expectativas irreales sobre las necesidades reales del cliente.</td></tr>
</table>

La métrica FTOR (Frecuencia de uso de términos opuestos en un requisito de software) es descrita en la Tabla 10, y permite determinar la proporción de términos opuestos respecto al total de términos en un requisito. Este enfoque cuantitativo permite identificar y cuantificar la presencia de complejidades lingüísticas que podrían afectar la claridad y la calidad de los requisitos de software.

Tabla 10. Métrica de frecuencia de uso de términos opuestos.

Frecuencia de uso de términos opuestos en un requisito de software	
Acrónimo	FTOR
Propósito	Evaluar la presencia de términos opuestos en los requisitos de software.
Unidad	Frecuencia
Escala	[0, 1]

<table>
<tr><td>Ecuación</td><td colspan="3">$$FTOR = \frac{NTOR}{NTT}$$</td></tr>
<tr><td>Variables</td><td colspan="3">• FTOR: Frecuencia de uso de términos opuestos en un requisito de software.
• NTOR: Número de términos opuestos de un requisito de software.
• NTT: Número total de términos de un requisito de software.</td></tr>
<tr><td rowspan="5">Criterio de decisión</td><td colspan="3">La interpretación del resultado de la frecuencia de uso de términos opuestos (FTOR) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Sin uso de términos opuestos</td><td>El requisito no presenta términos opuestos en su descripción. Por lo tanto, la consistencia del requisito se excelente.</td><td>FTOR = 0</td></tr>
<tr><td>Uso de términos opuestos leve</td><td>El requisito presenta pocos términos opuestos en su descripción. Por lo tanto, la consistencia del requisito es aceptable.</td><td>0 < FTOR ≤ 0.02</td></tr>
<tr><td>Uso de términos opuestos regular</td><td>El requisito presenta una cantidad considerable de términos opuestos en su descripción. Por lo tanto, la consistencia del requisito es cuestionable.</td><td>0.02 < FTOR ≤ 0.15</td></tr>
<tr><td></td><td>Uso de términos opuestos alto</td><td>La mayoría de los términos presentes en el requisito son opuestos. Por lo tanto, la consistencia del requisito es inaceptable.</td><td>0.15 < FTOR ≤ 1</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">Teniendo en cuenta el siguiente requisito de ejemplo: “El software debe ser fácil de usar para principiantes, pero también debe proporcionar funcionalidades avanzadas para usuarios expertos”, podemos analizar:
• Se encuentran los términos “principiantes” y “usuarios expertos”, considerados como términos opuestos entre sí.
• El requisito de ejemplo consta de 18 términos.
Aplicando la fórmula tenemos:
$$NTOR = 2\ términos$$
$$NTT = 18\ términos$$
Conociendo esto y aplicando la fórmula podemos determinar:
$$FTOR = \frac{2}{18} = 0.11$$
El resultado es 0.11, que se cataloga como grado de uso de términos opuestos regular. Esto significa que el requisito de software utiliza términos opuestos de manera moderada.</td></tr>
</table>

Por otro lado, la métrica PTCR (Promedio de tiempo de comprensión de requisitos de software) es descrita en la Tabla 11, y permite evaluar la eficiencia en la comprensión de los requisitos. Al calcular el tiempo promedio dedicado a la interpretación de los requisitos, la métrica proporciona una medida cuantitativa para identificar y abordar posibles complicaciones derivadas del uso de términos opuestos, contribuyendo a mejorar la eficacia y la claridad en la comunicación de los requisitos de software.

Tabla 11. Métrica de promedio de tiempo de comprensión de requisitos.

<table>
<tr><th colspan="2">Promedio de tiempo de comprensión de requisitos de software</th></tr>
<tr><td>Acrónimo</td><td>PTCR</td></tr>
<tr><td>Propósito</td><td>Evaluar la eficiencia en la comprensión de los requisitos.</td></tr>
<tr><td>Unidad</td><td>Promedio</td></tr>
<tr><td>Escala</td><td>(0, ∞)</td></tr>
<tr><td>Ecuación</td><td>$$PTCR = \frac{TTCR}{NTR}$$</td></tr>
</table>

<table>
<tr><td>Variables</td><td>

PTCR: Promedio de tiempo de comprensión de requisitos de software.
TTCR: Tiempo total de comprensión de los requisitos de software (en horas).
NTR: Número total de requisitos de software evaluados.

</td></tr>
<tr><td>Criterio de decisión</td><td>La interpretación del resultado del promedio de tiempo de comprensión de requisitos (PTCR) está dada por:
<table>
<tr><th>Grado</th><th>Descripción</th><th>Rango</th></tr>
<tr><td>Tiempo de comprensión excelente</td><td>Existe una eficiente y rápida comprensión de los requisitos, donde no se impone demoras significativas en el proceso de interpretación.</td><td>0 < PTCR ≤ 0.5</td></tr>
<tr><td>Tiempo de comprensión aceptable</td><td>Se requiere un tiempo considerable para comprender los requisitos, donde el esfuerzo dedicado a la comprensión se considera aceptable.</td><td>0.5 < PTCR ≤ 1</td></tr>
<tr><td>Tiempo de comprensión deficiente</td><td>Existe una deficiencia en la comprensión de los requisitos, donde el proceso de interpretación se ve afectado negativamente, requiriendo un tiempo excesivo para aclaraciones.</td><td>PTCR > 1</td></tr>
</table>
</td></tr>
<tr><td>Ejemplo de uso</td><td>En un nuevo proyecto de software, se han definido un total de 10 requisitos de software. Adicional, se indicó que se dedicaron 12 horas para la aclaración de dudas y comprensión de estos 15 requisitos definidos.
Aplicando la fórmula tenemos:
$$TTCR = 12\ horas$$
$$NTR = 10\ requisitos$$
Conociendo esto y aplicando la fórmula podemos determinar:
$$PTCR = \frac{12}{10} = 1.2\ horas/requisito$$
El resultado es 1.2, que se cataloga como grado de tiempo de comprensión deficiente, es decir que se presenta un exceso de tiempo en la compresión de los requisitos debido al uso de términos opuestos.</td></tr>
</table>

La métrica RECR (Relación entre el esfuerzo realizado en la comunicación de los requisitos de software respecto al esfuerzo estimado) es descrita en la Tabla 12, y permite evaluar la eficiencia en la comunicación de requisitos; calculando la proporción entre el esfuerzo realmente dedicado y el esfuerzo estimado inicialmente, esta métrica proporciona una medida cuantitativa para identificar discrepancias y evaluar el desempeño en la gestión del esfuerzo dedicado a la comunicación de requisitos.

Tabla 12. Métrica de relación entre el esfuerzo realizado y estimado en la comunicación de los requisitos.

<table>
<tr><th colspan="2">Relación entre el esfuerzo realizado en la comunicación de los requisitos de software respecto al esfuerzo estimado</th></tr>
<tr><td>Acrónimo</td><td>RECR</td></tr>
<tr><td>Propósito</td><td>Evaluar la eficiencia en la planificación y ejecución de la comunicación de requisitos.</td></tr>
<tr><td>Unidad</td><td>Razón</td></tr>
<tr><td>Escala</td><td>(0, ∞)</td></tr>
<tr><td>Ecuación</td><td>$$RECR = \frac{ERCR}{EECR}$$</td></tr>
<tr><td>Variables</td><td>

RECR: Relación del esfuerzo realizado en la comunicación de los requisitos de software respecto al esfuerzo estimado.
ERCR: Esfuerzo realizado en la comunicación de los requisitos de software.
EECR: Esfuerzo estimado en la comunicación de los requisitos de software.

</td></tr>
<tr><td>Criterio de decisión</td><td>La interpretación del resultado de la relación del esfuerzo realizado en la comunicación de los requisitos respecto al esfuerzo estimado (RECR) está dada por:</td></tr>
</table>

<table>
<tr><td></td><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td></td><td>Esfuerzo de comunicación de requisitos excelente</td><td>El esfuerzo real dedicado a la comunicación de los requisitos fue menor que el esfuerzo estimado inicialmente; indicando una eficiencia en la comunicación, donde se logró transmitir los requisitos de manera efectiva con un menor tiempo del previsto.</td><td>RECR < 1</td></tr>
<tr><td></td><td>Esfuerzo de comunicación de requisitos aceptable</td><td>El esfuerzo real dedicado a la comunicación de los requisitos coincide con el esfuerzo estimado inicialmente; indicando una precisión en la planificación y comunicación de los requisitos, donde se utilizó el tiempo previsto.</td><td>RECR = 1</td></tr>
<tr><td></td><td>Esfuerzo de comunicación de requisitos deficiente</td><td>El esfuerzo real dedicado a la comunicación de los requisitos fue mayor que el esfuerzo estimado inicialmente; indicando una deficiencia en la comunicación, donde no se logró transmitir los requisitos de manera efectiva en el tiempo previsto.</td><td>RECR > 1</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">En un nuevo proyecto de software, se estimó para la comunicación de los requisitos de software un esfuerzo de 4 horas/persona. Al finalizar con la tarea de comunicación, se evidenció que el esfuerzo realizado fue de 6 horas/persona.
Aplicando la fórmula tenemos:
$$ERCR = 6\ horas/persona$$
$$EECR = 4\ horas/persona$$
Conociendo esto y aplicando la fórmula podemos determinar que:
$$RECR = \frac{6}{4} = 1.5$$
El resultado es 1.5, que indica que se realizó más esfuerzo del estimado en la comunicación de los requisitos de software.</td></tr>
</table>

Por otro lado, la métrica PTAR (Promedio de tiempo dedicado a la aclaración de requisitos de software) es descrita en la Tabla 13, y permite determinar evaluar la eficiencia en el proceso de aclaración de requisitos. Al calcular el tiempo promedio dedicado a la resolución de dudas, esta métrica proporciona una medida cuantitativa para identificar y abordar posibles complicaciones en la comprensión de los requisitos de software, contribuyendo a mejorar la eficacia y la claridad en la fase de aclaración.

Tabla 13. Métrica de promedio de tiempo dedicado a la aclaración de requisitos.

<table>
<tr><th colspan="4">Promedio de tiempo dedicado a la aclaración de requisitos de software</th></tr>
<tr><td>Acrónimo</td><td colspan="3">PTAR</td></tr>
<tr><td>Propósito</td><td colspan="3">Evaluar la eficiencia en el proceso de aclaración de requisitos.</td></tr>
<tr><td>Unidad</td><td colspan="3">Promedio</td></tr>
<tr><td>Escala</td><td colspan="3">(0, ∞)</td></tr>
<tr><td>Ecuación</td><td colspan="3">$$PTAR = \frac{TTAR}{NTR}$$</td></tr>
<tr><td>Variables</td><td colspan="3">• PTAR: Promedio de tiempo de aclaración de requisitos de software.
• TTAR: Tiempo total de aclaración de requisitos de software (en horas).
• NTR: Número total de requisitos de software evaluados.</td></tr>
<tr><td>Criterio de decisión</td><td colspan="3">La interpretación del resultado del promedio de tiempo de aclaración de requisitos (PTAR) está dada por:
<table>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Tiempo de aclaración de requisitos excelente</td><td>Se presenta eficiencia en el proceso de aclaración de requisitos, donde se logra resolver dudas y confusiones de manera rápida y efectiva, minimizando el tiempo invertido en este aspecto.</td><td>0 < PTAR ≤ 0.5</td></tr>
<tr><td>Tiempo de aclaración de</td><td>Se presenta un tiempo considerable para la aclaración de requisitos, permitiendo resolver dudas de manera adecuada</td><td>0.5 < PTAR ≤ 1</td></tr>
</table></td></tr>
</table>

	requisitos aceptable	sin incurrir en excesos de tiempo o recursos.	
	Tiempo de aclaración de requisitos deficiente	Se presenta un tiempo excesivo para abordar y resolver dudas de los requisitos, recurriendo a excesos de tiempo y recursos.	PTAR > 1
Ejemplo de uso	En un nuevo proyecto de software, se han definido un total de 20 requisitos de software. Sin embargo, fue necesario realizar algunas reuniones de aclaración de requisitos, dedicando un total de 12 horas para la aclaración de dudas presentadas. Aplicando la fórmula tenemos: $$TTAR = 12\ horas$$ $$NTR = 20\ requisitos$$ Conociendo esto y aplicando la fórmula podemos determinar que: $$PTAR = \frac{12}{20} = 0.6\ horas/requisito$$ El resultado es 0.6, que se cataloga como un tiempo de aclaración de requisitos aceptable.		

3.3.4.4 Métricas para medir la incompletitud de los requisitos

Para medir la incompletitud de los requisitos de software se han definido tres (3) métricas. Inicialmente, la métrica PRRS (Porcentaje de requisitos de software que fueron revisados por todos los stakeholders) es descrita en la Tabla 14, y permite evaluar el nivel de revisión y validación de los requisitos por parte de todos los interesados clave. Al calcular el porcentaje de requisitos que ha sido revisado y avalado por los stakeholders, la métrica busca proporcionar una medida cuantitativa que refleje la cobertura y calidad de la validación de los requisitos. Este enfoque contribuye a detectar posibles falencias de manera temprana, garantizando que la especificación de los requisitos refleje de manera precisa y completa lo que se espera del software.

Tabla 14. Métrica de porcentaje de requisitos que fueron revisados por los stakeholders.

Porcentaje de requisitos de software que fueron revisados por todos los stakeholders			
Acrónimo	PRRS		
Propósito	Evaluar el nivel de revisión y validación de los requisitos por parte de todos los interesados clave.		
Unidad	Porcentaje		
Escala	[0, 100]		
Ecuación	$$PRRS = \frac{NRRS}{NTR} * 100$$		
Variables	• **PRRS**: Porcentaje de requisitos de software revisados por todos los stakeholders. • **NRRS**: Número de requisitos de software revisados por todos los stakeholders. • **NTR**: Número total de requisitos de software evaluados.		
Criterio de decisión	La interpretación del resultado del porcentaje de requisitos que fueron revisados por los stakeholders (PRRS) está dada por:		
	Grado	**Descripción**	**Rango**
	Revisión completa	Indica que todos los requisitos han sido revisados por todos los stakeholders.	PRRS = 100
	Revisión alta	Indica que la mayoría de los requisitos han sido revisados por todos los stakeholders.	80% ≤ PRRS < 100%
	Revisión media	Indica que pocos requisitos han sido revisados por todos los stakeholders.	40% ≤ PRRS < 80%
	Revisión baja	Indica que ninguno o la minoría de los requisitos han sido revisados por todos los stakeholders.	0% ≤ PRRS < 40%

Ejemplo de uso	En una revisión de 6 casos de uso, 3 de ellos fueron revisados por los stakeholders involucrados. Aplicando la fórmula tenemos: $$NRRS = 3\ requisitos$$ $$NTR = 6\ requisitos$$ Conociendo esto y aplicando la fórmula podemos determinar: $$PRRS = \frac{3}{6} * 100 = 50\ \%$$ El resultado es 50%, que se cataloga como porcentaje de revisión medio. Esto significa que la mitad de los requisitos no fueron revisados por todos los stakeholders, y queda una deuda de revisión, con un esfuerzo similar al ya realizado, para finalizar la revisión.

La métrica NTEDT (Número de términos que pueden enmascarar desafíos técnicos en un requisito de software) es descrita en la Tabla 15, y permite evaluar la cantidad términos ambiguos que pueden esconder desafíos técnicos que pueden afectar la estimación o viabilidad de un requisito. Al proporcionar un resultado numérico, NTEDT permite a los equipos de desarrollo y de gestión de proyectos anticipar y abordar posibles desafíos técnicos durante la fase de especificación de requisitos, contribuyendo así a mejorar la calidad y la viabilidad de los requisitos.

Tabla 15. Métrica del número de términos que pueden enmascarar desafíos técnicos en un requisito.

<table>
<tr><th colspan="4">Número de términos que pueden enmascarar desafíos técnicos en un requisito de software</th></tr>
<tr><td>Acrónimo</td><td colspan="3">NTEDT</td></tr>
<tr><td>Propósito</td><td colspan="3">Evaluar de manera cuantitativa la presencia de términos que podrían ocultar desafíos técnicos en un requisito de software.</td></tr>
<tr><td>Unidad</td><td colspan="3">Cantidad</td></tr>
<tr><td>Escala</td><td colspan="3">[0, 1]</td></tr>
<tr><td>Ecuación</td><td colspan="3">$$NTEDT = \sum_{i=1}^{NTT} TEDTi$$</td></tr>
<tr><td>Variables</td><td colspan="3">• NTEDT: Numero términos que pueden esconder desafíos técnicos.
• TEDT: Indica si un término en específico puede esconder desafíos técnicos.
• NTT: Número total de términos de un requisito de software.</td></tr>
<tr><td rowspan="4">Criterio de decisión</td><td colspan="3">La interpretación del resultado del número de términos que pueden enmascarar desafíos técnicos en un requisito (NTEDT) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Riesgo bajo</td><td>No se presentan términos que podrían ocultar desafíos técnicos.</td><td>NTEDT = 0</td></tr>
<tr><td>Riesgo medio</td><td>Se presentan pocos términos que podrían ocultar desafíos técnicos.</td><td>1 ≤ NTEDT < 2</td></tr>
<tr><td></td><td>Riesgo alto</td><td>Se presenta una cantidad considerable de términos que podrían ocultar desafíos técnicos.</td><td>2 ≤ NTEDT</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">Teniendo en cuenta el siguiente requisito de ejemplo: “El sistema de vuelo autónomo del dron de transmisión televisiva debe seguir al balón en un partido de futbol de forma automática y veloz, para poder grabar de primera mano las acciones más emocionantes de un partido”, se puede notar la presencia de 2 términos de 37 que pueden ocultar desafíos técnicos:
• El término número 22 del requisito, “automática”, puede abarcar distintos desafíos difíciles de abordar, por ejemplo, el dron debería saber en qué zonas no puede moverse para no interferir con el partido o para no chocar, el dron podría confundir balones de repuesto que están fuera de la cancha con el balón en juego, entre otros.
• El término número 24 del requisito, “veloz”, es ambiguo y no da información de la velocidad que el dron debe ser capaz de alcanzar, por ejemplo, para seguir la velocidad que puede alcanzar un</td></tr>
</table>

	balón golpeado por un jugador profesional. Si no se aclaran estos puntos desde el momento de la especificación de requisitos y, además; existieran limitaciones en las tecnologías vigentes de drones para alcanzar ciertas velocidades, el requisito podría ser inviable o tener sobrecostos. Aplicando la fórmula tenemos: $TEDT_{22} = 1\ (si)$ $TEDT_{24} = 1\ (si)$ $TEDT_i = 0, i \neq 22 \wedge i \neq 24$ $NTT = 37\ términos$ Conociendo esto y aplicando la fórmula podemos determinar: $NTEDT = 2\ términos$ El resultado es 2, que se cataloga como riesgo alto de ocultar desafíos técnicos.

Por otro lado, la métrica PRRDE (Porcentaje de requisitos de software que fueron revisados por desarrolladores con experiencia) es descrita en la Tabla 16, y permite evaluar la calidad, precisión y completitud de los requisitos mediante la participación de desarrolladores con experiencia en las revisiones de los requisitos. Estas revisiones sirven para que personas experimentadas puedan detectar olores, también conocidos como indicadores de mala calidad en los requisitos, y de esta forma, se puedan anticipar desafíos técnicos no previstos que puedan afectar la estimación y las consecuencias que esto conlleva, como retrasos o sobrecostos.

Tabla 16. Métrica de porcentaje de requisitos que fueron revisados por desarrolladores con experiencia.

<table>
<tr><th colspan="4">Porcentaje de requisitos de software que fueron revisados por desarrolladores con experiencia</th></tr>
<tr><td>Acrónimo</td><td colspan="3">PRRDE</td></tr>
<tr><td>Propósito</td><td colspan="3">Evaluar la calidad, precisión y completitud de los requisitos mediante la participación de desarrolladores con experiencia en las revisiones de los requisitos.</td></tr>
<tr><td>Unidad</td><td colspan="3">Porcentaje</td></tr>
<tr><td>Escala</td><td colspan="3">[0, 100]</td></tr>
<tr><td>Ecuación</td><td colspan="3">$PRRDE = \frac{NRRDE}{NTR} * 100$</td></tr>
<tr><td>Variables</td><td colspan="3">• PRRDE: Porcentaje de requisitos de software revisados por desarrolladores con experiencia.
• NRRDE: Número de requisitos de software revisados por desarrolladores con experiencia.
• NTR: Número total de requisitos de software evaluados.</td></tr>
<tr><td rowspan="6">Criterio de decisión</td><td colspan="3">La interpretación del resultado de la frecuencia de uso de términos opuestos (PRRDE) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Revisión completa</td><td>Indica que todos los requisitos han sido revisados por desarrolladores con experiencia.</td><td>PRRDE = 100</td></tr>
<tr><td>Revisión alta</td><td>Indica que la mayoría de los requisitos han sido revisados por desarrolladores con experiencia.</td><td>80% ≤ PRRDE < 100%</td></tr>
<tr><td>Revisión media</td><td>Indica que pocos requisitos han sido revisados por desarrolladores con experiencia.</td><td>40% ≤ PRRDE < 80%</td></tr>
<tr><td>Revisión baja</td><td>Indica que ninguno o la minoría de los requisitos han sido revisados por desarrolladores con experiencia.</td><td>0% ≤ PRRDE < 40%</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">En una revisión de 10 historias de usuario, 9 de ellas fueron revisadas por desarrolladores experimentados.
Aplicando la fórmula tenemos:
$NRRDE = 9\ requisitos$
$NTR = 10\ requisitos$
Conociendo esto y aplicando la fórmula podemos determinar:</td></tr>
</table>

	$$PRRDE = \frac{9}{10} * 100 = 90\ \%$$ El resultado es 90%, que se cataloga como porcentaje de revisión alto**¡Error! No se encuentra el origen de la referencia.**. Esto significa que la mayoría de los requisitos fueron revisados por desarrolladores con experiencia, y solo queda un porcentaje menor de requisitos pendiente a realizar su respectiva revisión por desarrolladores con experiencia.

3.3.4.5 Métricas para medir la no verificabilidad de los requisitos

Para medir la no verificabilidad de los requisitos de software se han definido dos (2) métricas. Inicialmente, la métrica PI (Porcentaje de imprecisión de un requisito de software) es descrita en la Tabla 17, y permite medir el grado de imprecisión de un requisito de software, a partir de la cantidad de términos imprecisos presentes en el requisito.

Tabla 17. Métrica de porcentaje de imprecisión de un requisito.

<table>
<tr><th colspan="4">Porcentaje de imprecisión de un requisito de software</th></tr>
<tr><td>Acrónimo</td><td colspan="3">PI</td></tr>
<tr><td>Propósito</td><td colspan="3">Medir el porcentaje de imprecisión de un requisito de software, a partir de la cantidad de términos imprecisos presentes en el requisito.</td></tr>
<tr><td>Unidad</td><td colspan="3">Porcentaje</td></tr>
<tr><td>Escala</td><td colspan="3">[0, 100]</td></tr>
<tr><td>Ecuación</td><td colspan="3">$$PI = \frac{NTI}{NTT} * 100$$</td></tr>
<tr><td>Variables</td><td colspan="3">• PI: Porcentaje de imprecisión de un requisito de software.
• NTI: Número de términos imprecisos en un requisito de software.
• NTT: Número total de términos de un requisito de software.</td></tr>
<tr><td rowspan="5">Criterio de decisión</td><td colspan="3">La interpretación del resultado del porcentaje de imprecisión (PI) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Sin imprecisión</td><td>El requisito está definido de manera precisa, proporcionando una base sólida para su verificación.</td><td>PI ≤ 0</td></tr>
<tr><td>Imprecisión leve</td><td>El requisito es lo suficientemente preciso y no impide realizar su respectiva verificación.</td><td>0% < PI ≤ 5%</td></tr>
<tr><td>Imprecisión regular</td><td>El requisito de software presenta una cantidad considerable de términos imprecisos, lo que puede llevar a impedir la verificación del requisito.</td><td>5% < PI ≤ 15%</td></tr>
<tr><td>Imprecisión alta</td><td>La descripción del requisito de software es completamente imprecisa, lo que dificulta entender el requisito y así mismo, realizar su verificación.</td><td>15% < PI ≤ 100%</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">Teniendo en cuenta el siguiente requisito de ejemplo: “El software debe ser rápido y fácil de usar”, podemos analizar que:
• Se encuentran los términos “rápido” y “fácil” de usar, considerado un olor de requisito relacionado con la imprecisión.
• El requisito de ejemplo consta de 9 términos.
Aplicando la fórmula tenemos:
$$NTI = 2\ términos$$
$$NTT = 9\ términos$$
Conociendo esto y aplicando la fórmula podemos determinar:
$$PI = \frac{2}{9} * 100 = \ 22.22\ \%$$
El resultado es 22.22%, que se cataloga como imprecisión alta.</td></tr>
</table>

Por otro lado, la métrica NPND (Indicador de pruebas que no se pueden diseñar) es descrita en la Tabla 18, y permite evaluar el número de pruebas de software que no se pueden diseñar debido a la falta de claridad, ausencia de los criterios de aceptación o escenarios en los requisitos de software.

Tabla 18. Métrica de indicador de pruebas que no se pueden diseñar.

<table>
<tr><th colspan="4">Indicador de pruebas que no se pueden diseñar</th></tr>
<tr><td>Acrónimo</td><td colspan="3">NPND</td></tr>
<tr><td>Propósito</td><td colspan="3">Evaluar la claridad y la suficiencia de los requisitos de software para el diseño de pruebas correspondientes.</td></tr>
<tr><td>Unidad</td><td colspan="3">Cantidad</td></tr>
<tr><td>Escala</td><td colspan="3">[0, ∞)</td></tr>
<tr><td>Ecuación</td><td colspan="3">$NPND = Número\ de\ pruebas\ que\ no\ se\ pueden\ diseñar$</td></tr>
<tr><td>Variables</td><td colspan="3">• NPND: Número de pruebas de software que no se pueden diseñar.</td></tr>
<tr><td rowspan="5">Criterio de decisión</td><td colspan="3">La interpretación del resultado del número de pruebas que no se pueden diseñar (NPND) está dada por:</td></tr>
<tr><td>Grado</td><td>Descripción</td><td>Rango</td></tr>
<tr><td>Excelente</td><td>Es posible diseñar todas las pruebas para la verificación de los requisitos definidos.</td><td>NPND = 0</td></tr>
<tr><td>Bueno</td><td>No se pueden diseñar pocas pruebas para la verificación de los requisitos definidos.</td><td>0 < NPND ≤ 5</td></tr>
<tr><td>Regular</td><td>No se pueden diseñar una cantidad considerable de pruebas para la verificación de los requisitos definidos.</td><td>5 < NPND ≤ 15</td></tr>
<tr><td>Malo</td><td>No se pueden diseñar una cantidad alta de pruebas para la verificación de los requisitos definidos.</td><td>15 < NPND</td></tr>
<tr><td>Ejemplo de uso</td><td colspan="3">Un conjunto de requisitos de software tiene 20 pruebas que no se pueden diseñar debido a la falta de claridad o ausencia de los criterios de aceptación o escenarios.
Aplicando la fórmula tenemos que:
$NPND = 20\ pruebas$
El resultado es 20, que se cataloga como Malo. Esto indica que muchos requisitos de software no son claros y no se pueden diseñar pruebas para ellos.</td></tr>
</table>

3.3.4.6 Métricas para medir los requisitos compuestos

Para medir los requisitos de software compuestos se han definido dos (2) métricas. Inicialmente, la métrica NSEHU (Número de semanas estimadas para finalizar una historia de usuario) es descrita en la Tabla 19, y permite conocer de manera cuantitativa la concisión de las historias de usuario en términos de tiempo estimado para su implementación.

Tabla 19. Métrica de indicador de semanas estimadas para finalizar una historia de usuario.

Indicador de semanas estimadas para finalizar una historia de usuario	
Acrónimo	NSEHU
Propósito	Proporcionar una evaluación cuantitativa del tiempo estimado en semanas para la implementación de historias de usuario.
Unidad	Cantidad
Escala	[0, 100]
Ecuación	$NSEHU = Número\ de\ semanas\ estimadas\ para\ implementar\ una\ historia\ de\ usuario$
Variables	• **NSEHU**: Número de semanas estimadas para implementar una historia de usuario.
Criterio de	La interpretación del número de semanas estimadas para finalizar una historia de usuario (NSEHU)

<table>
<tr><td>decisión</td><td>está dada por:
<table>
<tr><th>Grado</th><th>Descripción</th><th>Rango</th></tr>
<tr><td>Requisito conciso</td><td>Indica que la historia de usuario es manejable y puede completarse en una iteración o ciclo de desarrollo relativamente corto.</td><td>0 < NSEHU ≤ 1,5</td></tr>
<tr><td>Concisión aceptable</td><td>Aunque la historia podría ser más concisa, la estimación sigue siendo razonable y permite una implementación en un plazo aceptable.</td><td>1,5 ≤ NSEHU < 2,5</td></tr>
<tr><td>Concisión baja</td><td>Sugiere que la historia de usuario es extensa y puede beneficiarse de una subdivisión en historias más pequeñas y manejables.</td><td>2,5 ≤ NSEHU</td></tr>
</table></td></tr>
<tr><td>Ejemplo de uso</td><td>Una historia de usuario obtuvo una estimación de 10 días para su implementación, tomando en cuenta 5 días laborales por semana, tenemos que la historia de usuario fue estimada para finalizar en 2 semanas.
Aplicando la fórmula tenemos:
$$NSER = 2\ semanas$$
El resultado es 2, que se cataloga como concisión aceptable. Esto significa que la historia de usuario no es concisa y podría dividirse en historias de usuario con estimaciones menores, aunque no es un caso tan notorio.</td></tr>
</table>

Por otro lado, la métrica NCCR (Número de conjunciones o conectores en un requisito de software) es descrita en la Tabla 20, y permite evaluar la concisión de los requisitos de software al medir el número de conjunciones o conectores presentes en su descripción. Este valor nos da un indicio de que el requisito está abarcando demasiadas funcionalidades o necesidades, y puede ser dividido.

Tabla 20. Métrica de número de conjunciones o conectores en un requisito de software.

<table>
<tr><th colspan="2">Número de conjunciones o conectores en un requisito de software</th></tr>
<tr><td>Acrónimo</td><td>NCCR</td></tr>
<tr><td>Propósito</td><td>Evaluar la concisión de los requisitos de software al medir el número de conjunciones o conectores presentes en su descripción.</td></tr>
<tr><td>Unidad</td><td>Cantidad</td></tr>
<tr><td>Escala</td><td>[0, ∞)</td></tr>
<tr><td>Ecuación</td><td>$$NCCR = \sum_{i=1}^{NTT} CCi$$</td></tr>
<tr><td>Variables</td><td>• NCCR: Número de conjunciones o conectores en un requisito de software.
• CC: Indica si un término en específico es un conector o una conjunción.
• NTT: Número total de términos de un requisito de software.</td></tr>
<tr><td>Criterio de decisión</td><td>La interpretación del número de conjunciones o conectores en un requisito de software (NCCR) está dada por:
<table>
<tr><th>Grado</th><th>Descripción</th><th>Rango</th></tr>
<tr><td>Requisito conciso</td><td>El requisito no presenta ninguna conjunción o conector, lo que indica que está enfocado y aborda de manera clara y directa una única necesidad o funcionalidad.</td><td>NCCR = 0</td></tr>
<tr><td>Concisión aceptable</td><td>El requisito presenta una cantidad mínima de conjunciones o conectores. Aunque podría beneficiarse de cierta simplificación, aún es manejable y no se considera excesivamente extenso.</td><td>1 ≤ NCCR < 2</td></tr>
<tr><td>Concisión baja</td><td>El requisito presenta una cantidad alta de conjunciones o conectores, lo que indica que el requisito podría estar abarcando demasiadas funcionalidades o necesidades en un solo enunciado.</td><td>2 ≤ NCCR</td></tr>
</table></td></tr>
<tr><td>Ejemplo de uso</td><td>Teniendo en cuenta el siguiente requisito de ejemplo: “Como usuario del sitio web quiero que la página me permita iniciar sesión y registrarme, además de recuperar mi contraseña, para poder</td></tr>
</table>

	acceder al menú principal", podemos analizar: • La presencia de la conjunción "y", que corresponde al término número 14 de la historia de usuario, y el conector "además", que corresponde al término número 16 de la historia de usuario. • El requisito de ejemplo consta de 26 términos. Aplicando la fórmula tenemos: $CC_{14} = 1\ (si)$ $CC_{16} = 1\ (si)$ $CC_i = 0, i \neq 14 \wedge i \neq 16$ $NTT = 26\ términos$ Conociendo esto y aplicando la fórmula podemos determinar: $NCCR = 2$ El resultado es 2, que se cataloga como concisión baja. Esto significa que el requisito no es conciso y debería dividirse o replantearse para que abarque menos funcionalidades y obtenga una estimación menor.

3.3.4.7 Métricas para medir los requisitos innecesarios

Para medir los requisitos innecesarios se ha definido una (1) métrica. La métrica PRI (Porcentaje de los requisitos de software innecesarios) es descrita en la Tabla 21, y permite evaluar cuantitativamente el grado de requisitos que podrían considerarse innecesarios o poco relevantes para los objetivos del proyecto.

Tabla 21. Métrica de porcentaje de requisitos innecesarios.

Porcentaje de los requisitos de software innecesarios			
Acrónimo	PRI		
Propósito	Evaluar la presencia de requisitos innecesarios en un proyecto de software		
Unidad	Porcentaje		
Escala	[0, 100]		
Ecuación	$PRI = \frac{NRI}{NTR} * 100$		
Variables	• **PRI**: Porcentaje de requisitos de software innecesarios. • **NRI**: Número de requisitos de software innecesarios. • **NTR**: Número total de requisitos de software evaluados.		
Criterio de decisión	La interpretación del resultado del porcentaje de requisitos innecesarios (PRI) está dada por:		
	Grado	**Descripción**	**Rango**
	Sin requisitos innecesarios	No se presentan requisitos innecesarios dentro del proyecto, lo que indica que la especificación de requisitos está optimizada y enfocada en lo esencial.	PRI = 0
	Presencia de requisitos innecesarios leve	Se presenta una leve presencia de requisitos innecesarios, lo que indica que, aunque existe cierta optimización, se puede trabajar para eliminar o mejorar algunos requisitos adicionales que podrían considerarse menos críticos.	0% < PRI ≤ 5%
	Presencia de requisitos innecesarios regular	Se presenta una cantidad considerable de requisitos innecesarios que podrían ser revisados y potencialmente eliminados o refinados para mejorar la eficiencia y la relevancia del proyecto.	5% < PRI ≤ 20%
	Presencia de requisitos innecesarios alta	Se presenta una alta cantidad de requisitos innecesarios. En este nivel, se sugiere una revisión exhaustiva y una reevaluación de la especificación de requisitos para eliminar o mejorar aquellos que no aportan valor sustancial.	20% < PRI ≤ 100%
Ejemplo	Un proyecto de software tiene 50 requisitos en total. Después de una revisión detallada, se		

de uso	identificaron 15 requisitos como innecesarios. Aplicando la fórmula tenemos: $$NRI = 15\ requisitos$$ $$NTR = 50\ requisitos$$ Conociendo esto y aplicando la fórmula podemos determinar: $$PRI = \frac{15}{50} * 100 = \ 30\ \%$$ El resultado es 30%, que se cataloga como la presencia de requisitos innecesarios alta.

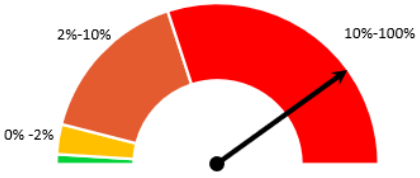
3.4 Instrumento de apoyo

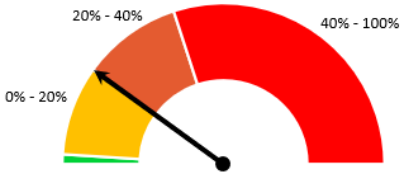
Se ha desarrollado un instrumento como herramienta de apoyo en Excel, el cual simplifica el uso y aplicación de las métricas. En las Figuras 2 y 3 se presenta una captura de pantalla de dicho instrumento para las Métricas 1 (PA) y 5 (FTOR). El instrumento completo puede ser descargado a través de la siguiente URL: https://bit.ly/3Tjnd26.

Figura 2. Captura de pantalla del instrumento para la Métrica 1 (PA) y la Métrica 2 (PHRS).

Porcentaje de ambigüedad (PA)		
Fórmula:		
PA=(NTA/NTT)*100		
Valor de NTA:		
3		
Valor de NTT:		
15		
Resultado:		
20,0%		
Aplica (Según entradas ingresadas):		
SI		
Rango de puntuación		**Nivel de cumplimiento**
0		Sin ambigüedad
0	2	Ambigüedad leve
2	10	Ambigüedad media
10	100	Ambigüedad alta

Porcentaje de heterogeneidad por el uso de sinónimos (PHRS)		
Fórmula:		
PHRS=(NRSOR/NTR)*100		
Valor de NRSOR:		
4		
Valor de NTR:		
20		
Resultado:		
20,0%		
Aplica (Según entradas ingresadas):		
SI		
Rango de puntuación		**Nivel de cumplimiento**
0		Sin heterogeneidad
0	20	Heterogeneidad leve
20	40	Heterogeneidad regular
40	100	Heterogeneidad alta





4 Referencias

[1] M. K. Habib, S. Wagner, and D. Graziotin, “Detecting Requirements Smells with Deep Learning: Experiences, Challenges and Future Work,” *IEEE International Conference on Requirements Engineering*, vol. 2021-September, pp. 153–156, Sep. 2021, doi: 10.1109/REW53955.2021.00027.

[2] “Ingeniería de requisitos.” Accessed: Sep. 21, 2022. [Online]. Available: https://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/ingenieria/ingenieria-requisitos

[3] H. Femmer, D. Méndez Fernández, S. Wagner, and S. Eder, “Rapid quality assurance with Requirements Smells,” *J Syst Softw*, vol. 123, pp. 190–213, 2016, doi: 10.1016/j.jss.2016.02.047.

[4] “ISO - ISO/IEC/IEEE 29148:2011 - Systems and software engineering — Life cycle processes — Requirements engineering.” Accessed: Feb. 11, 2023. [Online]. Available: https://www.iso.org/standard/45171.html

[5] V. R. Basili, G. Caldiera, and H. D. Rombach, “THE GOAL QUESTION METRIC APPROACH”.

[6] R. M. Agut, “Especificación de Requisitos Software según el estándar de IEEE 830,” 2000.

[7] R. Nascimento, E. Aranha, U. Kulesza, and M. Lucena, “Requirements Smells as indicators of poor quality in requirement specification: A systematic mapping of literature,” *21st Workshop em Engenharia de Requisitos, WER 2018*, Sep. 2018.

[8] C. M. F. MARTIN, “Guia Fundamentos para la Direccion de Proyectos 4ta Edicion,” *Guía de los Fundamentos para la Dirección de Proyectos*, pp. 343–346, 2013, Accessed: Mar. 03, 2024. [Online]. Available: https://www.academia.edu/18306947/Guia_Fundamentos_para_la_Direccion_de_Proyectos_4ta_Edicion

[9] D. Marcela, V. Bravo, P. César, and A. Collazos, “Modelo liviano de medidas para evaluar la mejora de procesos de desarrollo software MLM - PDS,” 2007, Accessed: Jan. 31, 2024. [Online]. Available: http://repositorio.unicauca.edu.co:8080/xmlui/handle/123456789/5970