EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Doctorado

Uso de Reglas de Dominio para Evaluar Recursos de Servicios Web de Aprendizaje

Presentada por

M. S. I. José Antonio Sandoval Acosta

Como requisito para la obtención del grado de

Doctor en Ciencias de la Computación

Directora de tesis

Dra. Olivia Graciela Fragoso Díaz

Comité Revisor

Dr. René Santaolaya Salgado
Dr. Noé Alejandro Castro Sánchez
Dr. Juan Carlos Rojas Pérez
Dr. Francisco Javier Álvarez Rodríguez

Cuernavaca, Morelos, México. Agosto de 2023.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

TECNOLÓGICO
NACIONAL DE MÉXICO

cenidet®
Centro Nacional de Investigación
y Desarrollo Tecnológico

Carta de Aceptación del Trabajo Doctoral

EDUCACIÓN | TECNOLOGICO NACIONAL DE MEXICO

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

SEP TecNM CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
RECIBIDO
03 AGO 2023
SUBDIRECCIÓN ACADÉMICA

ESC\FORDOC09

Cuernavaca, Morelos, 02/agosto/2023

ASUNTO: ACEPTACIÓN DEL TRABAJO DE TESIS DOCTORAL

MARÍA YASMÍN HERNÁNDEZ PÉREZ
JEFA DEL DEPARTAMENTO DE CIENCIAS COMPUTACIONALES
PRESENTE

Los abajo firmantes, miembros del Comité Tutorial de la Tesis Doctoral del alumno **JOSÉ ANTONIO SANDOVAL ACOSTA** manifiestan que después de haber revisado su trabajo de tesis doctoral titulado **"USO DE REGLAS DE DOMINIO PARA EVALUAR RECURSOS DE SERVICIOS WEB DE APRENDIZAJE"**, realizado bajo la dirección de OLIVIA GRACIELA FRAGOSO DÍAZ, el trabajo se ACEPTA para proceder a su impresión.

ATENTAMENTE
"Excelencia en Educación Tecnológica®
"Educación Tecnológica al Servicio de México"

OLIVIA GRACIELA FRAGOSO DÍAZ
CENIDET

NOÉ ALEJANDRO CASTRO SÁNCHEZ
CENIDET

RENÉ SANTAOLAYA SALGADO
CENIDET

JUAN CARLOS ROJAS PÉREZ
CENIDET

FRANCISCO JAVIER ÁLVAREZ RODRÍGUEZ
UNIV. AUTÓNOMA DE AGUASCALIENTES

C.c.p.: María Elena Gómez Torres / Jefa del Depto. de Servicios Escolares
Carlos Manuel Astorga Zaragoza / Subdirector Académico
Expediente

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
03 AGO 2023
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO

cenidet

Interior Internado Palmira S/N, Col. Palmira, C. P. 62490, Cuernavaca, Morelos
Tel. 01 (777) 3627770, ext. 3202, e-mail: dcc@cenidet.tecnm.mx | cenidet.tecnm.mx

2023 Francisco VILLA

Carta de Autorización de Impresión de Tesis Doctoral

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación y Desarrollo Tecnológico
Subdirección Académica

Cuernavaca, Mor., 03/agosto/2023
No. De Oficio: SAC/128/2023
Asunto: Autorización de impresión de tesis

JOSÉ ANTONIO SANDOVAL ACOSTA
CANDIDATO AL GRADO DE DOCTOR EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"USO DE REGLAS DE DOMINIO PARA EVALUAR RECURSOS DE SERVICIOS WEB DE APRENDIZAJE"**, ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Conocimiento y tecnología al servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/lmz

cenidet

Interior Internado Palmira S/N, Col. Palmira, C. P. 62490, Cuernavaca, Morelos
Tel. 01 (777) 3627770, ext. 4104, e-mail: acad_cenidet@tecnm.mx tecnm.mx | cenidet.tecnm.mx

2023 Francisco VILLA

Agradecimientos

"*El significado de la vida es descubrir tu don,*
el propósito de la vida es regalarlo"

Pablo Picasso
(1881 - 1973)

Agradezco profundamente a mi directora de tesis la Dra. Olivia Graciela Fragoso Díaz por su apoyo y dedicación, por convertirse en una guía en mi formación y en una amistad sincera para toda la vida. De igual forma, agradezco a mi comité revisor, conformado por el Dr. René Santaolaya Salgado, el Dr. Juan Carlos Rojas Pérez, el Dr. Noé Alejandro Castro Sánchez y el Dr. Francisco Javier Álvarez Rodríguez por su invaluable apoyo y guía.

De lgual forma, agradezco al Tecnológico Nacional de México (TecNM), especialmente a sus campus Cenidet e ITS de Guasave por las facilidades recibidas; asimismo, a la Secretaría de Educación Pública (SEP) y al Consejo Nacional de Humanidades, Ciencias y Tecnología (CONAHCYT) por el apoyo y beca recibidos.

Especialmente, agradezco a mi familia por su apoyo y espera para que pudiera concluir mi anhelo y mi meta. Finalmente, a mis compañeras y compañeros estudiantes, tanto de doctorado como de maestría, por su apoyo, amistad y compañerismo.

Dedicatoria

"*A veces sentimos que lo que hacemos es tan solo una gota en el mar,*
pero el mar sería menos si le faltara una gota"

Madre Teresa de Calcuta
(1910 - 1997)

Quiero dedicar muy especialmente este trabajo a mi madre Doña Plácida, ya que estos 4 años debió esperarme lejos para concluir mi formación.

También, a mi padre Don Antonio, a mi hermano César y a mi gran amigo Iván, que en nuestro viaje por las estrellas ya se han marchado, adelantándose a los que aún permanecemos aquí esforzándonos por ser mejores cada día.

A mis amigos Alberto, Andrés, Jesús Manuel, Mario, Miguel y Jesús Rosario, que han permanecido siempre cerca, me han aconsejado y me han apoyado con su valiosa amistad.

A mis hermanas y hermanos que durante este tiempo permanecieron al lado de mi madre para que ella pudiera tener una vida tranquila en mi ausencia.

Finalmente, a todas aquellas personas que se han visto involucradas en este esfuerzo y me han apoyado incondicionalmente.

Resumen

"*Cuánto más leas, más cosas sabrás.*
Cuántas más cosas aprendas, a más lugares viajarás"
Dr. Seuss
(1904-1991)

El problema que aborda este trabajo de investigación nace en el uso masivo de recurso de aprendizaje obtenidos de Internet se ha vuelto una actividad cotidiana en la educación, en todos los niveles. Tanto docentes como estudiantes hacen búsquedas de recursos relacionados con las asignaturas y temas que tienen asignados en ingeniería de software con la idea de tener una mejor comprensión de dichos temas, particularmente del diagrama de clases. Muchas veces no existe un parámetro que indique la calidad del recurso y como ésta fue medida, ni el objetivo real para el que fue creado.

Estas cuestiones en una institución que pone al servicio de sus estudiantes recursos tecnológicos para realizar las búsquedas y obtención de información electrónica pueden influir en la calidad del aprendizaje de sus estudiantes; sin embargo, no se encontró un método o modelo que pueda usarse para medir la calidad de los recursos de aprendizaje o que haya sido creado para satisfacer tal necesidad.

Para dar solución a la problemática identificada, se requirió realizar un trabajo de investigación para identificar aquellos atributos que podrían presentar defectos en el diagrama de clases, afectando la calidad del mismo y su comprensión. Se diseñaron de reglas de dominio y la propuesta de un modelo de calidad que atiende todos esos atributos identificados. Adicionalmente, se desarrolló una herramienta de software que realiza la implementación de las reglas de dominio y del modelo de calidad, evaluando recurso de aprendizaje contenidos en archivos formato PPTX. Por otro lado, se trabajó en una metodología que consiste en dos etapas principales, en las que se definió la solución del problema planteado. Asimismo, se diseñó un conjunto de casos de prueba para validar cuantitativamente el modelo de solución presentado mediante un corrimiento de pruebas con 115 diagramas de clases seleccionados.

Abstract

"An investment in knowledge pays the best interest"

Benjamin Franklin
(1706-1790)

The massive use of learning resources obtained from the Internet has become a daily activity in education, at all levels, both teachers and students search for resources related to the subjects and topics on software engineering assigned to them with the idea of having a better understanding. of these subjects and with this achieve a better learning. Many times, there is no parameter in the description of the found resource that indicates the actual objective for which it was created, or that indicates its quality and how it was measured.

These issues in an institution that puts technological resources at the service of its teachers and students to carry out searches and obtain electronic information can influence the quality of student learning; however, no method or model was found which has this quality or it has been created to satisfy such an objective.

In order to solve the identified problem, it was required to carry out a research work to identify those aspects and characteristics that could present defects in the class diagram, emerging its quality and its understanding. Domain rules and the proposal of a quality model that addresses all those identified attributes are designed. Additionally, a software tool was implemented that performs the application of domain rules and the quality model, evaluating learning resources contained in PPTX format files. On the other hand, we worked on a methodology consisting of two main stages, in which the solution to the problem was defined. Likewise, a set of test cases was concluded to quantitatively validate the solution model presented through a test run with 115 selected class diagrams.

Índice

Lista de Figuras

Lista de Tablas

"Todo aquello que el hombre
ignora no existe para él.
Por eso, el universo de cada uno se
resume al tamaño de su saber"

Albert Einstein
(1879 - 1955)

Capítulo 1 Introducción

CAPÍTULO 1. INTRODUCCIÓN

1.1. Introducción

El aprendizaje electrónico o *e-learning*, se considera como el medio que puede resolver necesidades de educación y capacitación en diferentes ámbitos. En las universidades se utiliza cada vez más debido a que ofrece facilidades para entregar cursos a estudiantes que no necesariamente tienen una condición de lugar y tiempo. Es decir, no viven cerca de su lugar de estudio, además, tienen compromisos de trabajo que no les facilitan el asistir a cursos presenciales; este es un escenario muy común actualmente.

Los centros educativos como las universidades utilizan recursos de aprendizaje (RA) de diversos tipos y formatos, siendo aquellos en formato electrónico en los que se enfoca este trabajo. Los RA pueden ser: videos, presentaciones, esquemas, diagramas, imágenes, textos, audios, libros electrónicos, bases de datos, guías, por mencionar algunos [1]. Estos RA pueden ser muy diversos y se pueden obtener de diversos lugares en Internet o en repositorios institucionales. Este tipo de RA son utilizados en educación a distancia de forma masiva.

Un problema recurrente es que los RA no necesariamente cumplen el objetivo de aprendizaje para el que son usados y esto tiene su origen en diversas fuentes, una de las más importantes es la calidad de los RA que se emplean.

1.2. Antecedentes

El diseño de los recursos de aprendizaje ha sido, y sigue siendo, tema de investigación y desarrollo. A través del uso de la tecnología informática, y más aún, desde el advenimiento de Internet, la proliferación de materiales educativos en formato digital ha ido creciendo año tras año. Los materiales, así como los entornos digitales donde estos son producidos propician un nuevo modo de saber y de producir saber [2].

Los principales antecedentes de esta tesis son los trabajos: [3], [4], [5], [6] y [7]. Los cuales, están orientados a la clasificación de los Servicios Web de Aprendizaje (SWA); la medición de la calidad de los Recursos de Aprendizaje (RA) desde un enfoque de fondo; y, a la medición de la calidad de los SWA.

1.2.1. Servicios Web de Aprendizaje (SWA)

En [3], el autor documenta la definición, clasificación, características, limitaciones y forma de construcción de los OA. Muestra también, qué elementos conforman un OA y la unión de dichos elementos a través de metadatos. El autor menciona que el uso de los OA en la enseñanza acelera el proceso de aprendizaje, pero su construcción requiere un poco más de trabajo ya que está directamente enlazado con procesos cognitivos y de aprendizaje complejos. Con este trabajo se inicia la línea de investigación para proponer alternativas a los OA. Asimismo, se propone tratar a cada elemento que conforma a un OA como un SW, para potencializar las características de los OA, facilitando su reúso y proporcionando mayor apertura a la distribución del contenido educativo a través de la Web y de esta manera ampliar el alcance de uso (o de utilidad) de los OA.

1.2.2. Definición de Elementos del WSDL para Servicios Web de Aprendizaje

En [4], el autor define las capas que conforman un WSDL (*Web Services Description Language*). También, documenta el protocolo SOAP (*Simple Object Accesss Protocol*), el cual, es un protocolo que define las reglas para el intercambio de información estructurada entre aplicaciones.

En este trabajo son analizados los elementos que conforman los archivos de descripción WSDL. Con el fin de mejorar la descripción de los SWA, se proponen también una serie de nuevos elementos a agregar al archivo de descripción WSDL, el cual, se encuentra en formato XML.

1.2.3. Esquema de Clasificación de Servicios Web de Aprendizaje

En [5], el autor aborda el problema de la escasa precisión en la recuperación de los SWA. Para solucionar el problema propone un esquema de clasificación de SWA por dominio de aplicación a través de descriptores, en un sistema que permita recuperarlos con mayor precisión.

1.2.4. Marco Orientado a Objetos para Medir la Calidad en Servicios Web de Aprendizaje

En [6], el autor realiza una evaluación de la calidad para SWA, para contribuir en la toma de decisiones en actividades de clasificación, selección y utilización de los mismos. El autor define un modelo de calidad para los SWA y el componente de software desarrollado cuya

arquitectura es un marco orientado a objetos. El modelo tiene la característica de extensibilidad, ya que ofrece facilidad de incorporar nuevos atributos en los niveles jerárquicos.

1.2.5. Integración de Recursos de Aprendizaje en MOODLE con base en el Modelo de Servicios Web

En [7], la autora realiza la integración de los SWA a la plataforma de MOODLE para el consumo y presentación de los SWA de tal manera que formen parte de un curso específico. También, identifica 4 tipos de SWA; los cuatro tipos de SWA pueden ser integrados para llegar a proporcionar la misma funcionalidad de un objeto de aprendizaje y ser integrados y administrados en diversas plataformas educativas.

La Figura 1 muestra una línea del tiempo sobre los antecedentes del problema de investigación.

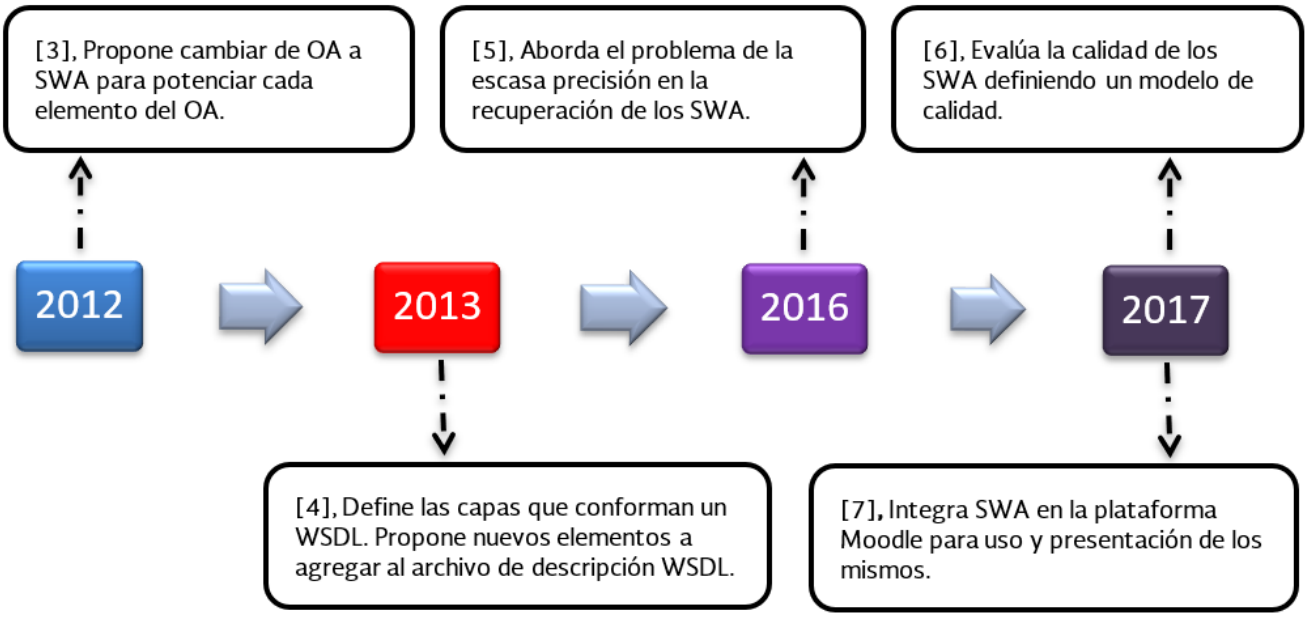


Figura 1. Línea del Tiempo de Antecedentes de Investigación

En los trabajos relacionados que se analizaron se propone el cambio de los OA al modelo de SWA; también, se desarrollaron modelos para clasificar los SWA y medir su calidad; asimismo, para medir la calidad de los RA desde un enfoque de análisis de forma y no de fondo.

1.3. Definición del Problema

Actualmente existe una gran cantidad de Recursos de Aprendizaje de Ingeniería de Software disponibles en Internet empleados en procesos de formación de recursos humanos, tanto en las universidades como en el lugar del trabajo. Sin embargo, no existe garantía de que esos recursos puedan cumplir los requerimientos para los que se utilizan. Uno de los problemas es que los Recursos de Aprendizaje de Ingeniería de Software que contienen Diagramas de Clases UML no pueden ser seleccionados objetivamente para fines de aprendizaje debido a que no se tiene información sobre su calidad y son usados de manera indiscriminada, lo cual, el uso de esos recursos puede prevenir que se logre el objetivo que atiende.

1.4. Objetivos

1.4.1. Objetivo General

- Establecer un mecanismo con base en reglas de dominio que permita evaluar la calidad de los Recursos de Aprendizaje para el dominio de Ingeniería de Software (IS) en los Diagramas de Clases (DC) UML.

1.4.2. Objetivos Específicos

1. Definir las reglas de dominio para los tipos de RA que se van a atender.
2. Identificar los defectos que pueden tener los RA, para evaluarlos con las reglas de dominio que se definieron.
3. Identificar aquellos atributos que se deben considerar en una evaluación.

1.5. Justificación del Estudio

El proceso de aprendizaje electrónico de Ingeniería de Software para el tema de Diagramas de Clase UML se ve fuertemente afectado debido a la existencia de gran cantidad de recursos de aprendizaje de IS en diversos formatos digitales, los cuales, son accedidos desde distintas plataformas en las que no es fácil identificar la calidad de dichos recursos, por lo que el resultado esperado no siempre se logra al nivel deseado.

1.6. Alcances y Limitaciones

1.6.1. Alcances

- El tipo de recursos de aprendizaje que se evaluarán son recursos para el área de IS, en particular para diagramas de clases UML.
- Las reglas de dominio están diseñadas para el dominio de los diagramas de clases del tipo de recursos a evaluar.
- Se define el indicador o modelo de calidad para medir el grado de correctitud.
- Se generan valores de calidad de los RA.

1.6.2. Limitaciones

- No se identifica cuál es el mejor valor de calidad para los RA.
- No se evalúan atributos que no están definidos en las reglas de dominio.
- No se evalúa la calidad pedagógica ni atributos pedagógicos de los RA.
- Los RA deben estar integrados en un archivo formato PPTX.
- No se evalúa la calidad de los SWA, ni la calidad externa de los RA.
- No se evalúa código fuente, por lo que no realiza ingeniería inversa, ni se evalúan atributos basadas en código fuente.

1.7. Estructura del Documento

El presente documento tiene la siguiente estructura. En el Capítulo I se hace una descripción del problema de investigación y se establecen los objetivos de este trabajo, así como sus alcances y limitaciones. En el Capítulo II se hace la presentación del Marco Conceptual y Teórico de la investigación, incluyendo la revisión trabajos relacionados. En el Capítulo III se presenta la metodología de desarrollo del proyecto de investigación, en la que se realiza la definición de la solución a los objetivos planteados en el proyecto. En el Capítulo IV se realiza la descripción de la herramienta de software desarrollada para la realización de pruebas y validación de resultados. En el Capítulo V se presentan los resultados del proyecto de investigación. Por otro lado, en el Capítulo VI se presentan las conclusiones del presente trabajo y los productos logrados en el proyecto doctoral, así como los trabajos futuros propuestos. Finalmente, el Capítulo VII tiene un conjunto de anexos que facilitan la contextualización y entendimiento de los hallazgos y resultados presentados en esta tesis.

"Largo es el camino de la enseñanza
por medio de teorías;
breve y eficaz por medio de ejemplos"

Lucio Anneo Séneca
(4 a. de C. – 65)

Capítulo 2
Marco Conceptual y Trabajos Relacionados

CAPÍTULO 2. MARCO CONCEPTUAL Y TRABAJOS RELACIONADOS

En este capítulo se definen una serie de conceptos que dan sustento a los términos mencionados en el desarrollo de este proyecto doctoral. Adicionalmente, se presenta la sección que contiene 78 trabajos relacionados que contextualizan la investigación.

2.1. Marco Conceptual

2.1.1. Modelo

En [8], se define a un modelo como: una representación abstracta y simplificada de un objeto o fenómeno del mundo real. Asimismo, se establece que las características típicas de un modelo son:

- Sólo describen los aspectos del objeto o fenómeno que se consideran relevantes para la comprensión y uso predefinido del modelo.
- Encapsulan la experiencia y permiten una representación explicita de la experiencia.
- Pueden ser creados para diferentes propósitos como planificación, control o predicción.
- Tienen su propio ciclo de vida, es decir, pueden especificarse, construirse, implementarse, analizarse, utilizarse, evaluarse, evolucionan o se rechazan.

Por otra parte, en [9] los autores definen un modelo como: Una abstracción del sistema, especificando el sistema modelado desde un cierto punto de vista y un determinado nivel de abstracción. Un punto de vista es, por ejemplo, una vista de especificación o una vista de diseño del sistema.

2.1.2. Servicios Web (SW)

Gran variedad de definiciones acerca de los SW es dada por diferentes líderes del sector, grupos de investigación y consorcios de SW.

El Consorcio W3C [10] define a un SW como: Un sistema de software diseñado para soportar la interoperabilidad de la interacción máquina - maquina a través de una red. Tiene una interfaz descrita en un formato procesable por una máquina (específicamente WSDL). Otros sistemas interactúan con el SW de una manera prescrita por su descripción

utilizando mensajes SOAP, normalmente transmitidos utilizando HTTP con una serialización XML en conjunto con otras normas relacionadas con la Web.

Un Servicio Web (SW) es un término genérico para una función de software interoperable de máquina a máquina que se aloja en una ubicación direccionable de la red.

Por su parte, definición proporcionada por IBM en [11] indica que un SW tiene una interfaz, que oculta los detalles de la implementación para que pueda usarse independientemente de la plataforma de hardware o software en la que se implementa, e independientemente del lenguaje de programación en el que está escrito. Esta independencia fomenta que las aplicaciones basadas en SW estén acopladas de manera flexible, orientadas a componentes, implementaciones de tecnología cruzada. Los SW se pueden utilizar solos o con otros SW para llevar a cabo una agregación compleja o una transacción comercial.

Mientras que Microsoft [12] define los SW como sigue: Los SW extienden la infraestructura de la World Wide Web (WWW) para proporcionar los medios para que el software se conecte a otras aplicaciones de software. Las aplicaciones acceden a los SW a través de protocolos Web ubicuos y formatos de datos como HTTP, XML y SOAP, sin necesidad de preocuparse por cómo se implementa cada SW. Los SW combinan los mejores aspectos del desarrollo basado en componentes y la Web, y son una piedra angular del modelo de programación Microsoft .NET.

2.1.3. Aprendizaje Electrónico (E-Learning)

En [13], los autores definen al *e-Learning* como: La convergencia de Internet y el aprendizaje, o el aprendizaje habilitado para Internet; el uso de las tecnologías Web para crear, fomentar, entregar y facilitar el aprendizaje, en cualquier momento y en cualquier lugar. En [14] los autores definen al *e-Learning* como: El uso de las Tecnologías de la Información y Comunicación (TIC) para mejorar y/o apoyar el aprendizaje.

2.1.4. Capacitación Electrónica (E-Training)

En [15], el autor menciona que el e-training se enfoca en ayudar a los empleados a desarrollar las habilidades que necesitan para realizar tareas específicas. Esto incluye proporcionar información sobres ciertos procesos y operaciones diarias. El *e-Training* hace todo esto en un ambiente en línea. En esencia, el *e-Training* puede preparar a los empleados para las situaciones previstas, mientras que el *e-Learning* prepara a los empleados para lo inesperado.

2.1.5. Arquitectura Orientada a Servicios (SOA)

SOA es una arquitectura para la construcción de aplicaciones empresariales como un conjunto de componentes de caja negra ligeramente acoplados y orquestados para proporcionar un nivel de servicio bien definido uniendo procesos empresariales [16].

2.1.6. Sistema Gestor de Aprendizaje (LMS)

Un LMS es definido como un software diseñado para entregar, rastrear y administrar la formación y la educación. Un paquete de alta solución que permite la entrega y administración de contenido y recursos a todos los estudiantes y empleados [17].

2.1.7. Servicio Web de Aprendizaje (SWA)

En [7], la autora define los SWA como un componente de software, cuya funcionalidad principal es entregar contenido educativo, que además se pueden describir, publicar, localizar e invocar gracias a la adopción de protocolos y estándares abiertos como SOAP (Simple Object Access Language), JSON (Javascript Object Notation), REST (Representational State Transfer), y HTTP (Hypertext Transfer Protocol). Asimismo, se definen cuatro principales tipos de SWA: SWA de objetivo, SWA de contenido, SWA de actividad y SWA de evaluación los cuales corresponden a los elementos pedagógicos que deben estar contenidos en un objeto de aprendizaje.

2.1.8. Calidad en el Software

Para ISO [18], la Calidad en el Software se define como el grado en que el producto de software satisface las necesidades declaradas e implícitas cuando se utiliza bajo las condiciones especificadas.

2.1.9. Atributo de Calidad

Para [19] un atributo de calidad se define como las características que tiene un objeto físico o abstracto y que afectan su funcionamiento, además de considerar que los atributos de calidad juegan un rol importante cuando se diseñan sistemas de software.

2.1.10. Métrica de Calidad

En [20], el autor define la métrica de calidad como aquello que nos permite medir atributos y entenderlos. Las métricas permiten tomar decisiones ilustradas, saber si nuestras

decisiones fueron las correctas, al evaluar las consecuencias de esas decisiones. Las métricas son siempre racionales.

2.1.11. Regla de Dominio

Según [21], una regla de dominio se define como una condición que se utiliza para validar, corregir y normalizar valores de un dominio. Una regla de dominio debe cumplirse en todo el dominio para que los valores de dominio se consideren precisos y compatibles con los requisitos. Las reglas de dominio pueden incluir reglas de validación que se utilizan para validar valores de dominio, pero no para corregir datos en los proyectos de calidad de datos. Para [22], las reglas se definen como declaraciones estructuradas de alto nivel que limitan, controlan e influyen en la lógica de un proceso. Según [23], las reglas deben formalizarse para facilitar su uso. Para [23], una regla de dominio podría clasificarse de la siguiente manera:

1. Reglas de integridad. Identifican restricciones o afirmaciones que deben cumplirse.
2. Reglas de derivación. Se refieren a una o más condiciones y una o varias conclusiones.
3. Reglas de reacción. Consisten en eventos que ocurrirán, condiciones a satisfacer y acciones a ejecutar.
4. Reglas de producción. Se refieren a una o más condiciones y una o más acciones producidas, las reglas de producción pueden implementar reglas de reacción.
5. Reglas de transformación. Controlan los cambios en el estado de un sistema.
6. Regla de asignación deóntica. Esta regla restringe las acciones en términos de obligaciones, permisos y prohibiciones.

Para propósitos de este trabajo, la definición de [22] y la clasificación proporcionada por [23] son las que tienen mayor relación con esta tesis.

2.1.12. Recurso de Aprendizaje (RA)

En [24], el autor documenta que los recursos de aprendizaje (RA) son elementos que se pueden utilizar en el proceso de aprendizaje electrónico tales como documentos de texto,

presentaciones, archivos en formato PDF, archivos multimedia (imágenes, audios, videos), páginas Web, entre otros.

2.1.13. Unified Modeling Language (UML)

El autor de [25] define el UML como un lenguaje estándar que sirve para escribir los planos del software, puede utilizarse para visualizar, especificar, construir y documentar todos los artefactos que componen un sistema con gran cantidad de software. El estándar UML puede usarse para modelar desde sistemas de información hasta aplicaciones distribuidas basadas en Web, pasando por sistemas empotrados de tiempo real. UML es solamente un lenguaje por lo que es sólo una parte de un método de desarrollo de software, es independiente del proceso, para que sea optimo debe usarse en un proceso dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

2.1.14. Diagrama de Clases UML

Un diagrama de clases es un modelo que representa un conjunto de clases, interfaces, atributos, métodos, colaboraciones y sus relaciones. Al igual que otros diagramas los diagramas de clases pueden contener notas y restricciones. También, pueden contener paquetes o subsistemas, los cuales su usan para agrupar los elementos de un modelo en partes más grandes. A veces se colocan instancias en los diagramas de clases, cuando se quiera mostrar el tipo (posiblemente dinámico) de una instancia [25].

2.2. Revisión de Trabajos Relacionados

Se realizó una revisión de trabajos relacionados basándose en el protocolo propuesto por Kitchenham en [26]. En este protocolo se establecen 3 etapas principales. En la Figura 2 se muestran las etapas del diagrama del protocolo utilizado.

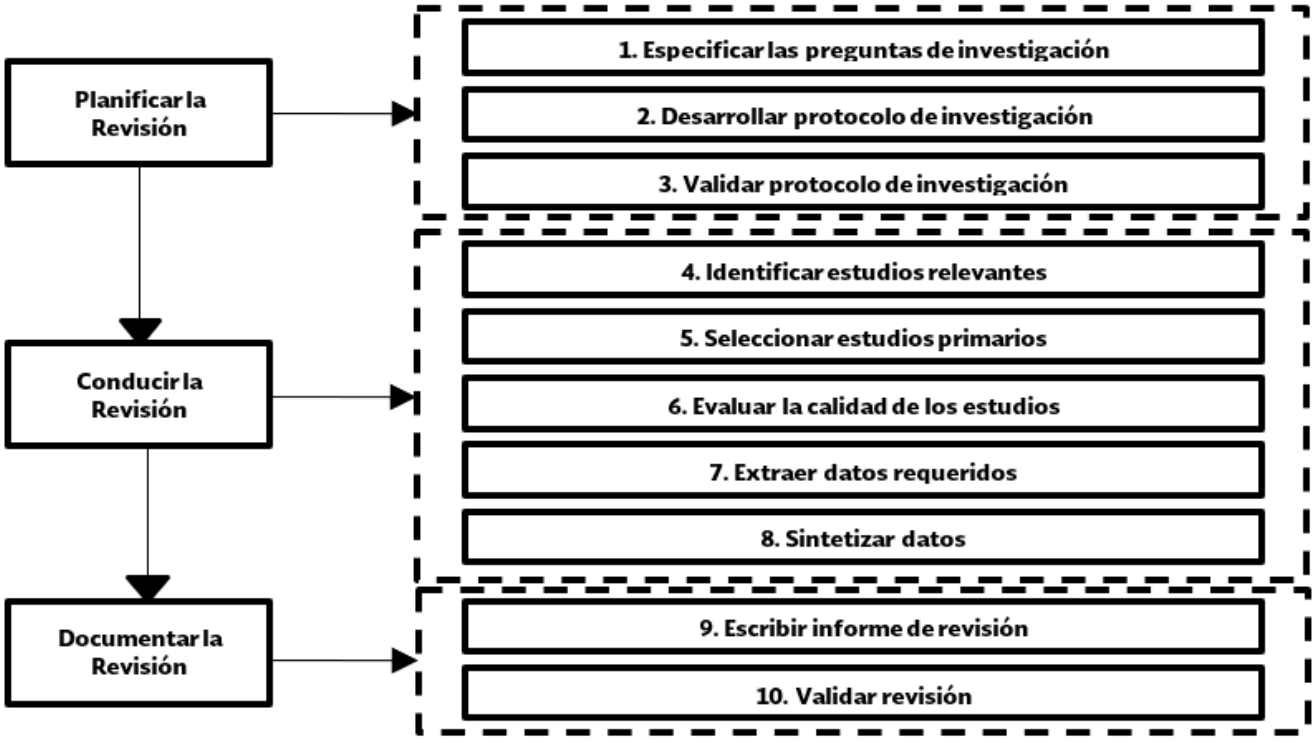


Figura 2. Diagrama de las etapas del protocolo de Kitchenham [26].

2.2.1. Pregunta de Investigación

Para realizar la búsqueda de trabajos relacionados se buscó una pregunta de investigación que ayude a contextualizar los requerimientos actuales en los recursos de aprendizaje electrónicos para IS. La pregunta de investigación es:

P1: ¿Cuáles son los enfoques actuales que analizan calidad en Diagramas de Clase UML?

2.2.2. Fuentes Bibliográficas Consultadas

La búsqueda de las publicaciones se realizó en 4 distintas bases de datos de fuentes bibliográficas, las cuales se describen en la Tabla 1.

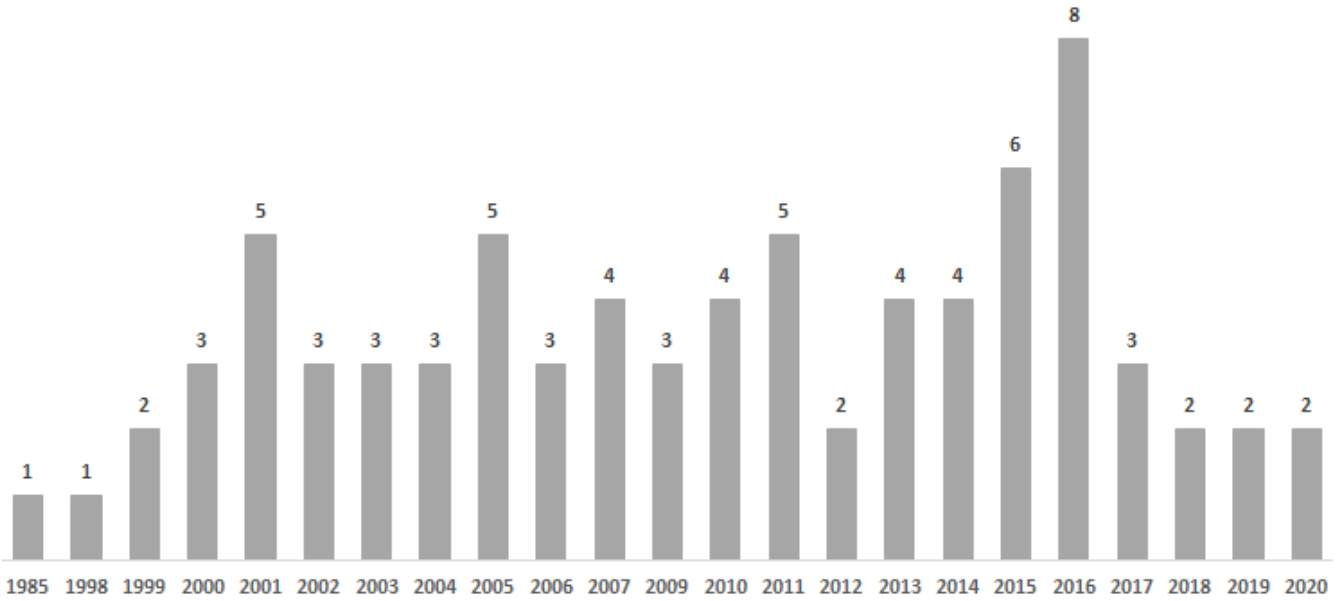
Tabla 1. Fuentes Bibliográficas

No.	Base de Datos	Cantidad
1	IEEE	29
2	Google Scholar	36
3	ACM	10
4	Springer	3
	Total de fuentes bibliográficas	78

2.2.3. Clasificación de Publicaciones por Año de Publicación

Las 78 publicaciones consideradas para el estudio de trabajos relacionados fueron clasificadas de acuerdo a su año de publicación, sin tener una limitante sobre el año en que haya publicado la misma. Esto, debido a que se considera que existen trabajos valiosos que pueden aportar al logro de los objetivos de este trabajo independientemente de su antigüedad. La Figura 3 muestra la Gráfica de Clasificación de Publicaciones por Año.

Figura 3. Gráfica de Clasificación de Publicaciones por Año



2.2.4. Enfoques Abordados en las Publicaciones

Respondiendo la pregunta de investigación, se identificaron varios enfoques en las publicaciones seleccionadas, con el fin de clasificar dichas publicaciones y tener una mejor comprensión de su aportación y uso en este proyecto. Cabe mencionar que algunos

trabajos abordan más de un enfoque. Los enfoques encontrados se describen en la Tabla 2. Asimismo, la Figura 4 muestra la Gráfica de Distribución de Publicaciones por Enfoque.

Tabla 2. Enfoques que Abordan los Trabajos Relacionados

No.	Enfoque	Cantidad
1	Análisis de Métricas de Calidad.	21
2	Herramientas para la Medición de Calidad en Diagramas UML.	13
3	Grado de Entendimiento del Diagrama UML.	22
4	Estética del Diagrama UML.	9
5	Herramientas para la Medición de Expresiones y Emociones en e-Learning.	13

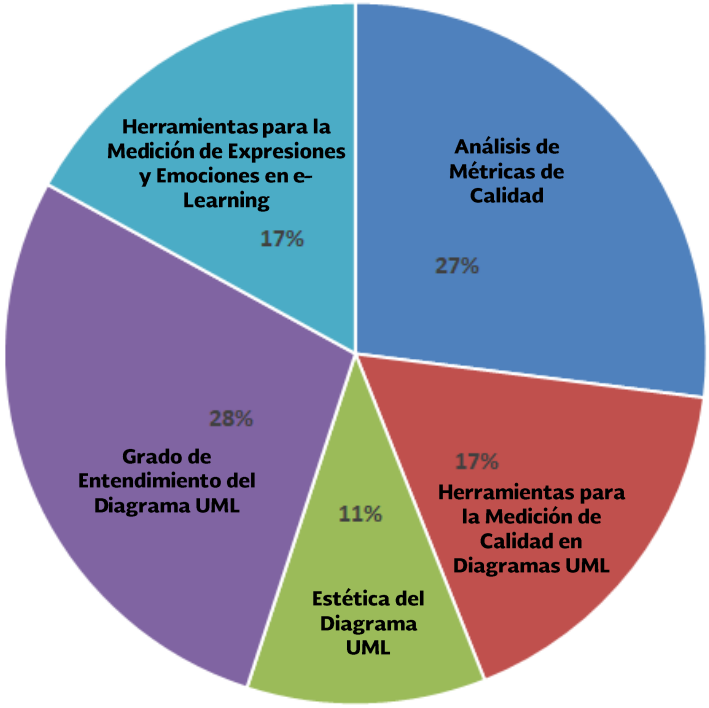


Figura 4. Gráfica de Distribución de Publicaciones por Enfoque

2.2.5. Descripción de los Trabajos Relacionados

Los enfoques de trabajos relacionados que se muestran en la Tabla 2 son listados en un orden acorde a dicha tabla, con un total de 78 trabajos incluidos. Los enfoques que se encontraron son: Análisis de Métricas de Calidad, Herramientas para la Medición de Calidad en Diagramas UML, Grado de Entendimiento del Diagrama UML, Estética del

Diagrama UML y Herramientas para la Medición de Expresiones y Emociones en E-Learning.

2.2.5.1. Análisis de Métricas de Calidad

En el primer rubro que se encontró, se detectó una gran cantidad de trabajos que abordan las mediciones mediante la definición de métricas de calidad. Se tomaron en cuenta aquellos trabajos que realizan mediciones a los diagramas, excluyendo los que definen o analizan métricas basadas en código fuente, por lo que podemos considerar que el análisis realizado en estos trabajos relacionados es un análisis estático y de arriba hacia abajo.

En [27], los autores analizan los *Open Educational Resources* (OER) que benefician a la Ingeniería de Software. En este trabajo se mencionan tanto los OER físicos como los electrónicos. Su enfoque es principalmente en los recursos electrónicos, como lo son, vídeos, MOOC y juegos multimedia. Su principal búsqueda es para identificar recursos que puedan ayudar a mejorar el aprendizaje de ingeniería de software estableciendo criterios para encontrar recursos relacionados con: requerimientos de software, gestión de proyectos, pruebas y refactorización. En [28], los autores presentan un modelo enfocado en cubrir lagunas que existen en varios puntos del diseño orientado a objetos (DOO). El primero es el punto de vista de los interesados, es decir, las necesidades y requisitos reales. El segundo es que los desarrolladores se concentran más en los requisitos funcionales y dejan de lado los requisitos no funcionales. El tercero es la medición de los productos diseñados. El cuarto aborda la complejidad que aporta el diagrama de clases al DOO. Considerando 4 aspectos básicos del DOO, asociación, agregación, generalización y composición. Por otra parte, en [29], los autores presentan un enfoque en el acoplamiento entre objetos, presentando el análisis de dos métricas ya diseñadas previamente, que son: Coupling Between Objects (CBO) y NAS (Number of Associations) que mide el número de asociaciones entre una clase y sus pares en el Object Model Diagram (OMT). En uno de sus planteamientos mencionan que a medida que el acoplamiento entre clases se incrementa, el entendimiento de una clase se reduce. Mientras que en [30], los autores realizan un trabajo sobre el uso de métricas dinámicas para la evaluación del diseño orientado a objetos. El análisis se basa en EOC (export object coupling), OQFS (Object Request for Service), IOC (import object coupling), OPFS (Object Response for Service). Los autores mencionan que las métricas de calidad deben relacionarse con aspectos externos como lo son mantenibilidad, reusabilidad, entendimiento y susceptibilidad a errores. Además, el autor menciona que las métricas propuestas pueden ser usadas en las primeras etapas del DOO. En [31], los autores se enfocan en el diagrama de clases. La

complejidad estructural es un atributo de calidad interno, que los autores piensan podría estar estrechamente relacionado con uno de los atributos de calidad externos más críticos, como la mantenibilidad del diagrama de clases. En este trabajo se presentan dos experimentos llevados a cabo para corroborar si esas métricas están cerradas a la mantenibilidad del diagrama de clases que, por lo tanto, podrían usarse como indicadores de mantenimiento temprano. El objetivo de [32], es mostrar que, en las etapas iniciales del diseño, las métricas que miden atributos internos, como estructuras complejidad y tamaño de los diagramas de clases UML, se puede utilizar como Indicadores de mantenibilidad del diagrama de clases. Por otro lado, en [33], los autores proponen la aplicación del algoritmo HITS en el DOO para evaluar la calidad de un modelo, representado en un diagrama de clases o cualquier otro tipo de diagrama. Se propone modificar el algoritmo para tener en cuenta el número de mensajes discretos intercambiados entre clases, es posible identificar clases que implican un mal diseño. Mientras que los autores de [34] analizan y comparan algunas métricas típicas para diagramas de clases UML de diferentes autores, diferentes puntos de vista, diferentes tipos de relaciones, diferentes tipos de valores métricos, complejidad y validaciones. Por su parte los autores de [35] tienen como objetivo presentar una descripción general de trabajos relacionados sobre métricas aplicables a la medición de atributos de calidad para diagramas de clases, diagramas de casos de uso, diagramas de diagramas de estado y expresiones OCL. Los autores de [36] presentan un estudio del estado del arte sobre medidas del Paradigma Orientado a Objetos (POO) que se pueden aplicar para medir los atributos de calidad internos de los diagramas de clases. Por otra parte, en [37], los autores presentan un ejemplo de integración de métricas en el proceso de desarrollo orientado a objetos. Al abordar las primeras etapas del diseño, como, el diagrama de clases, para anticipar defectos o errores a nivel de diseño, lo que permite una reducción de los costos y problemas posteriores. En [38], los autores comparan las ventajas y desventajas de métricas típicas de complejidad de diagramas de clases como lo son: comprensibilidad, analizabilidad y mantenibilidad, desde el punto de vista de la ingeniería de software experimental reciente. Mientras que en [39], los autores hacen una serie de definiciones que son útiles para identificar los elementos que conforman de manera formal el diagrama de clases UML. Mientras que en [40], los autores realizan un análisis de calidad de los Open Educational Resources (OER) que se encuentran en Internet y pueden ser accedidos libremente, así como herramientas de software para su uso y búsqueda. Por otro lado, los autores de [41] presentan una serie de métricas para evaluar la complejidad del diagrama de clases UML, enfocándose principalmente en los distintos tipos de relación que estos tienen. También, se realizan mediciones sobre número de atributos, número de

operaciones, número de clases, entre otros. De igual forma, los autores de [42], realizan una investigación que está basada en reglas de abstracción ya existentes por separado y que pretenden ser unidas para crear un conjunto más completo para refinar los diagramas de clase desde los niveles bajos a niveles más altos de abstracción, proponiendo agrupar clases lógicamente por contexto de las mismas, entre otras técnicas. Los autores de [43] proponen una serie de métricas con el fin de clarificar y medir características del diseño orientado a objetos como lo son: polimorfismo, encapsulación, herencia, ocultación de información, y paso de mensajes. Por otro lado, en [44], los autores se enfocan en problemas de modelado de reglas de dominio basadas en UML. Definen varios tipos de reglas que resultan útiles para esta investigación. Además, plantean la relación entre los modelos de clases UML y el vocabulario OWL. También, se plantea la especificación de reglas en diagramas de clases con ayuda de OCL. Mientras que en [45], los autores presentan un enfoque basado en reglas, que a su vez se basa en el modelo evento-condición-acción (ECA), y apoyado por un lenguaje basado en reglas de proceso de negocio (RbBPDL). Esto permite convertir un proceso en un conjunto de reglas que se utiliza para comprobar qué tan flexible es un proceso. En este trabajo se hace una definición de distintos tipos de reglas que pueden ser aplicados a diferentes áreas, incluyendo la IS. Por otro lado, en [46], los autores presentan una herramienta para evaluar recursos multimedia de aprendizaje. En esta herramienta es posible evaluar dimensiones como: calidad del contenido, motivación, diseño de presentación, usabilidad, accesibilidad, reusabilidad y apego a los estándares establecidos. Finalmente, en [47] los autores tienen como objetivo establecer un modelo para evaluar y medir el impacto de los recursos de aprendizaje mejorados con tecnología, con el fin de tener un mayor uso en el aprendizaje en comparación con los programas donde se utilizan. El modelo evalúa niveles como: el Grado de Satisfacción del Usuario y el Grado de Aprendizaje Alcanzado.

2.2.5.2. Herramientas para la Medición de Calidad en Diagramas UML

En este rubro se encontraron herramientas que miden la calidad de los diagramas desde distintos enfoques, en muchos casos la medición se hace mediante el análisis de los metadatos del archivo que contiene el diagrama, o mediante un archivo XML. Para este rubro se consideraron algunos trabajos que realizan análisis de abajo hacia arriba debido a la relevancia de sus resultados.

En [48], los autores presentan la descripción de una serie de reglas y herramientas para IS, particularmente para el diagrama de clases representado en UML. Su enfoque principal en el diagrama de clases, y menciona el fuerte enfoque que tienen las herramientas CASE

hacia la implementación, no hacia los diagramas visuales. Enumera una serie de reglas respecto a la disposición de los diagramas de clase y sobre su dibujado, la estética de los mismos, así como las relaciones espaciales. Se revisan diferentes herramientas entre ellas librerías de Java y gráficas. Mientras que en [49], los autores presentan una serie de métricas del diagrama de clases UML. Su enfoque principal es hacia el desarrollo de software, menciona como uno de los atributos a evaluar el grado de entendimiento del diagrama. Las gráficas presentadas aportan muy buen entendimiento del proceso de recolección y proceso de la información de las métricas evaluadas sin hacer mención de las técnicas utilizadas para recolectar y procesar la información. Por otro lado, en [50], los autores presentan un estudio respecto a cómo predecir fallas en el software por medio de la complejidad de diseño de diagramas de clase UML. Se enfoca principalmente en los fallos de software ya funcionando intentando aplicar regresión para predecir posibles fallas a futuro por medio del diagrama. Por otra parte, en [51], los autores muestran una herramienta llamada TUPUX que al parecer ya cuenta con publicaciones previas. Se menciona una serie de reglas referentes a la asociación, agregación, generalización, y relaciones de asociación entre clases. Se realiza una comparación entre el resultado de estudiantes graduados y no graduados para determinar si aplican correctamente las reglas establecidas, arrojando como resultado que los estudiantes identifican incorrectamente relaciones entre clases resultando en un cálculo incorrecto de los puntos de función. En [52], los autores mencionan una metodología de medición y modelación de software del "*Ocean Parameters Measurement System*". El método presentado abarca todo el proceso de desarrollo iniciando con la etapa de diseño; sin embargo, para la etapa de diseño, se enfoca principalmente en el diagrama de casos de uso, mientras que el diagrama de clases es escasamente mencionado. En [53], los autores presentan un análisis entre dos diagramas del UML, diagrama de clases y diagrama de secuencia, haciendo una comparación en las inconsistencias encontradas al momento de pasar el contenido del primer diagrama para formar el segundo. La idea principal es desarrollar un algoritmo y la herramienta correspondiente. Para ello, se establecen ciertas reglas que deben contener los diagramas tanto de clase como de secuencia los cuales están generados con *Enterprise Architect* y son exportados a XML para su posterior análisis, el cual, se realiza por medio de una herramienta en C#.Net. Mientras que en [54], los autores presentan una nueva herramienta llamada UXSOM basada en Java en formato XMI, UXF y XML, para el análisis de métricas en el diagrama de clases. Asimismo, realiza un análisis comparativo de las capacidades de dicha herramienta respecto a otras ya existentes en el mercado, mencionando que atributos del diagrama de clases no son cubiertos por las otras herramientas. UXSOM clasifica los paquetes, operaciones, atributos y relaciones,

encontradas en el archivo XML que se obtuvo desde cualquiera de las herramientas con las que hace comparación. Por otro lado, en [55], los autores presentan un conjunto de métricas basado en el diagrama de clases UML para evaluar objetivamente diagramas de clases UML. Se utiliza el lenguaje Java para evaluar la complejidad de los diagramas de clases UML en varios aspectos, y los verifica con un conjunto de reglas. Mientras que en [56], los autores describen una herramienta que automatiza el cálculo de los datos importantes y métricas que son aplicables a los diagramas de clases UML. La herramienta recopila información analizando el formato XMI del diagrama de clases para producir metadatos, y después, utiliza estos datos para calcular las métricas. El documento también describe algunas áreas en las que se pueden aplicar métricas de diseño orientado a objetos para mejorar calidad del software. Y, para finalizar este enfoque, en [57] los autores tienen como objetivo demostrar la posibilidad de utilizar técnicas de inteligencia artificial (IA) para mejorar la creación de Recursos de Aprendizaje (RA) a través de sus metadatos. En este trabajo se encontraron algunos aprendizajes clave como: 1) la IA puede categorizar los recursos de aprendizaje basados en texto, 2) la clasificación automatizada basada en la IA puede proporcionar un medio para ayudar a los usuarios a realizar la clasificación, pero por ahora no es factible reemplazar la participación humana.

2.2.5.2.1. Herramientas para la Medición de Calidad en Diagramas UML que realizan el Análisis de Abajo Hacia Arriba

En [58], los autores presentan una forma de medición calidad del DOO que requiere el análisis desde el código fuente. Presenta una herramienta llamada MUSE que está diseñada en Perl, con la cual se hace la evaluación de las reglas y se presentan los resultados, el artículo no menciona cuales son las 39 reglas que se evalúan. Mientras que en [59] está aún inconcluso y los autores presentan un avance del mismo. Su enfoque es sobre una serie de métricas en el diseño orientado a objetos analizando varios sistemas de software. Sin embargo, su enfoque es sobre el software y luego revisión de los diagramas. No está enfocado en revisión de los diagramas en la etapa inicial de diseño, sino que requiere de un análisis desde el código fuente. En [60], los autores presentan una herramienta diseñada para analizar las métricas del Diagrama de Clases UML. La herramienta está diseñada en C-Sharp (C#) y utiliza analizadores como Gold y CM. La herramienta está basada en código fuente ya que hace una revisión del mismo lo convierte a archivo XML y por medio de otras herramientas analizadoras genera las métricas.

2.2.5.3. Grado de Entendimiento del Diagrama UML

Este es el rubro donde se encontró mayor cantidad de trabajos relacionados, se trata de un área importante debido a la dificultad detectada para la comprensión de los diagramas en la etapa de diseño de software, esto lo plasman los propios autores en sus hallazgos. Los trabajos presentados en esta sección son todos con enfoque de análisis estático.

En [61], los autores presentan un estudio enfocado al entendimiento del diagrama UML principalmente desde un punto de vista semántico. Realizan revisiones a través de una herramienta llamada *SD Metrics* y por medio de un diccionario ya existente que fue creado por expertos, la valoración va de 0 a 1 siendo cero una valoración pobre y 1 una valoración excelente. Se aborda desde un punto de vista de entendimiento del contenido del diagrama y no de la codificación. No se verifica la legibilidad del diagrama ni el grado de exactitud de las relaciones. Por otro lado, en [62], los autores realizan una revisión del estado del arte, en la cual, muestran un estudio muy detallado identificando todos los pasos seguidos en el mismo. Se enfocan principalmente en la comprensión del modelo de proceso y abordan diferentes tipos de diagramas y aspectos de UML incluyendo el diagrama de clases. Mientras que en el estudio [63], los autores mencionan al inicio la dificultad que existe para aprender, comprender y utilizar UML sobre todo para aprendices y usuarios inexpertos, además de ser caro y no existe una relación clara de costo-beneficio, por lo que el estudio realmente se basa en el beneficio que aporta el uso de UML y la justificación de los costos implicados en el desarrollo de software. Por otro lado, en [64], los autores recomiendan minimizar la cantidad de "esquinas en escuadra" en las relaciones, debido a que es más fácil seguir líneas rectas que seguir líneas que van cambiando de dirección. Por otra parte, en [65], los autores mencionan en el título al diagrama de casos de uso, sin embargo, en el texto del artículo hay referencias a varios tipos de diagramas, entre ellos el de clases UML. En [66], los autores describen un algoritmo evolutivo que diseña diagramas de clases UML. Desarrollan el diseño mutando las posiciones de los símbolos de clase, relaciones de herencia y asociaciones. El proceso está controlado por una función de que se calcula a partir de varias métricas conocidas y algunas nuevas métricas de diseño. En [67], los autores presentan un metamodelo común y unificado impulsado por una ontología de tal manera que unifica varios tipos de diagramas, entre ellos el diagrama de clases UML. En [68], los autores presentan un estudio sobre 5 tipos de notación UML de diagramas de clases, en el que se pretende averiguar cuál es elegido mayormente por los usuarios en función de su mejor comprensión al momento de modelar. Mientras que en [69], los autores buscan realizar

un refinamiento del diagrama de clases concreto (el diagrama completo) a un diagrama de clases abstracto, no se busca que el diagrama agregue abstracción, sino que se pueda realizar una abstracción del diagrama completo por lo que las pruebas y comparaciones son entre dos diagramas el abstracto y el concreto. En el trabajo se aborda tanto el diagrama de clases como el diagrama de casos de uso. Por otro lado, en [70] los autores realizan un estudio sobre la refactorización del diagrama de clases considerando el código "*smell*" y refactorizándolo. Por su parte, los autores de [71] analizan atributos de calidad de los diagramas UML que pueden dificultar su comprensión, los atributos analizados son: tamaño del diagrama UML y calidad del diagrama UML. Se realizan mediciones sobre varios tipos de diagramas UML. Mientras que, en [72], los autores realizan un estudio a través de una herramienta de software que utiliza redes neuronales para detectar los elementos de un diagrama de clases. En el estudio no se desarrolla ningún tipo de regla de dominio, sino que se busca realizar el trabajo como apoyo para docentes y estudiantes. Se busca en el futuro que la herramienta pueda interpretar distintos tipos de diagramas. En [73], los autores abordan el tema de la abstracción como elemento de gran relevancia en el diseño orientado a objetos, ya que permite crear diseños conceptuales sin introducirse a modelos con demasiados detalles y creando diagramas más sencillos de interpretar. Por otro lado, en el estudio [74], los autores presentan un experimento realizado con estudiantes avanzados de IS, presentándoles diagramas de clases para ser evaluados por al menos 3 personas cada uno con la intensión de evaluar la comprensibilidad de los diagramas, analizando una serie de características del diagrama como lo son: facilidad para leerlo, completitud, información extra incluida, complejidad, correctitud, *layout*, nombres de las clases, atributos y operaciones, número de las clases, número de atributos, número de operaciones, entre otros. En [75], los autores realizan una investigación sobre el uso del diagrama de clases UML en el desarrollo de software, argumentando que se trata del principal diagrama utilizado para ello. Realizan cuestionarios con un centenar de practicantes de software realizando preguntas como: cual es el tipo de diagrama que más utilizan; si el diagrama de clases mejora su entendimiento del negocio donde desarrollan software; si el diagrama de clases es utilizado como "medio de comunicación" en el equipo; si el diagrama de clases les da una idea clara respecto a las necesidades de mantenimiento del software; entre otras. Mientras que, en [76] los autores presentan una investigación acerca del efecto que tiene un buen *layout* en el entendimiento del diagrama de clases, asumiendo que el diagrama es una de las principales fuentes de información para los desarrolladores al momento de interpretar los requisitos y desarrollar el software. Por otra parte, en [77] los autores recomiendan minimizar la cantidad de cruces de líneas ya que entre mayor cantidad de

cruces más difícil es para el ojo humano detectar que clases están conectadas entre sí dificultándose la interpretación de los DC. Mientras que, en [78] los autores tienen como objetivo encontrar un modelo que pueda predecir la comprensibilidad y modificabilidad del diagrama de clases UML. En [79] los autores presentan un algoritmo que calcula las diferencias de dos modelos dados como archivos XMI. La salida también es un archivo XMI, contiene el modelo unificado de los dos modelos originales con información adicional sobre diferencias. El cálculo en sí mismo es configurable para capturar la semántica de un modelo real o parte del modelo en el algoritmo. Por su parte, los autores de [80] mencionan que la falta de uniformidad y la gran cantidad de defectos contenidos en los modelos UML dan como resultado una falta de comunicación entre diferentes actores en el proceso de desarrollo de software. Para prevenir estos problemas, en este trabajo se proponen convenciones de modelado, análogo a las convenciones de codificación para programación. Mientras que en [81], los autores presentan una herramienta llamada *Metric View Evolution* que tiene como objetivo la gestión de la calidad de los modelos orientados a objetos durante el desarrollo. Para finalizar este rubro, en [82], los autores realizan un análisis formal de los diagramas de casos de uso. Se propone un modelo formal de casos de uso y su construcción para las relaciones típicas entre casos de uso. Dos métodos de análisis formal y se presenta la verificación. El trabajo sirve como base para analizar y establecer posibles modelos relativos a otros tipos de diagramas como el diagrama de clases.

2.2.5.4. Estética del Diagrama de Clases UML

Este rubro es en el que se encontró menor cantidad de trabajos relacionados. El objetivo de la mayoría de los autores es demostrar que la estética del diagrama es un atributo que influye en la comprensión del mismo, el análisis realizado es en todos los casos estático.

En [83], los autores presentan una serie de lineamientos respecto a la estética del diagrama UML. Se enfocan principalmente en mejorar la calidad de los diagramas a través del cumplimiento de reglas establecidas, así como el tiempo de trabajo, no se aborda codificación. Las características que pretende regular en el diagrama son: diseño, estructura, diagramación y semántica. Se aborda el tema de la correctitud del diagrama, pero no se aborda el tema de la comprensión del mismo por parte de los lectores. Mientras que en [84], los autores presentan un enfoque en el que el tamaño del diagrama UML es lo fundamental respecto su entendimiento. La hipótesis planteada refiere que entre mayor es el tamaño del diagrama el sujeto que lo modela tiene un desempeño más pobre. En contraste, un diagrama pequeño es más fácil de usar e interpretar independientemente de

su calidad, ya que es pequeño. Los autores mencionan 4 principios que gobiernan un diagrama UML, y que son los que pueden ser evaluados: 1) principios de diseño gráfico y visualización; 2) minimización de cruces, curvas y tamaño de líneas; 3) notaciones como orden de los elementos visuales muy enfocado en el flujo visual, minimización del desorden visual; y 4) la parte práctica, uso de colores, tamaños y posiciones que sirvan de guía a los lectores. Por su parte, los autores de [85] presentan un análisis de la herramienta VIATRA, que sirve para verificar, validar y mejorar la calidad de las herramientas diseñadas con base en la notación UML, y que puede revisar de forma automática los requerimientos de consistencia y completitud. En [86], los autores investigan el efecto estético de los diagramas de clases UML y los diagramas de colaboración. En este trabajo, se busca identificar qué características prefieren los usuarios. También, evalúan los efectos de minimizar, por ejemplo: dobleces o esquinas cuadradas, y cruces de líneas. Mientras que en [87], los autores mencionan algunas reglas o "leyes" sobre cómo puede ser un diagrama de clases más entendible para el lector. En [88], los autores investigan cómo la forma del contorno de los diagramas influye en los juicios estéticos sobre dos dimensiones del atractivo estético: belleza e interés. Se emplean diferentes esquemas para el enlace de nodo en diagramas y los comparan con formas rellenas uniformemente, variando dos variables importantes: complejidad y curvatura. En sus experimentos concluyen que los diagramas con contornos más curvos se percibieron como más hermosos, mientras que los diagramas con contornos más complejos se consideraron más interesantes. El estudio busca mejorar la visualización estética de los diagramas. Por otro lado, en [89], los autores reportan un experimento cuyo objetivo es identificar una lista ordenada de las características estéticas preferidas por los sujetos cuando se incorpora en diagramas de colaboración y clases UML. Dicha lista podría indicar a los diseñadores de interfaces de las herramientas CASE. así como, la forma de diseñar sus diagramas para obtener la mejor respuesta de los usuarios. El enfoque principal del estudio tuvo como objetivo identificar lo que es "estéticamente agradable" en los diagramas. Mientras que en [90], los autores hacen un estudio experimental sobre el tamaño del diagrama y cómo influye en el entendimiento, y como afecta el modelado. Asimismo, analizan atributos como nombres de clases, elementos conectores, estructura de las clases, y el anidado dentro de las clases. Y, para finalizar este rubro, en [91], los autores realizan un estudio enfocado principalmente en diagramas de clase en el que analizan aspectos estéticos relacionados con el diseño del mismo que pueden afectar su entendimiento.

2.2.5.5. Herramientas para la Medición de Expresiones y Emociones en E-Learning

En este rubro el objetivo encontrado en los trabajos relacionados es una medición de las emociones que tienen los aprendices de IS al momento de realizar trabajo con diagramas que presentan defectos, o bien, que están correctos, realizando un análisis de dichas emociones mediante sensores y herramientas de software. En este rubro se consideró tanto el análisis estático como el análisis dinámico, así como el análisis de abajo hacia arriba.

En [92], los autores realizaron una revisión profunda y sistemática de artículos relacionados con métricas y calidad en el diseño de diagramas UML. Mencionan 5 preguntas de investigación las cuales al final son respondidas con base a los análisis de literatura relacionada. El trabajo principalmente se enfoca en los diagramas UML y en procesos de desarrollo formal de software de gran tamaño. Mientras que en [93], se presenta un enfoque principalmente en el DOO ya que proporciona métricas sobre la profundidad del árbol de herencia y el número de subclases de una clase en particular. Asimismo, el número de subclases dependientes de una clase tomando en cuenta únicamente las dependencias directas y excluyendo las clases o dependencias heredadas. Por otro lado, en el estudio realizado en [94] los autores tienen como objetivo mejorar la calidad de los diagramas de clase UML a través del uso de Algoritmos Genéticos. Los autores utilizan una métrica de acoplamiento entre objetos para medir el grado de dependencia de los componentes del sistema. El trabajo se enfoca principalmente en los diagramas UML clasificando las clases de objetos y estableciendo sus relaciones por medio de algoritmos genéticos. Como resultado, los autores demuestran que los algoritmos genéticos tienen potencial para evaluar el diagrama de clases UML. Por su parte, el autor de [95] presenta un estudio en el que se analizan básicamente 5 métricas de calidad en el DOO, principalmente dirigidos a evaluar aspectos de herencia, con el fin de medir la complejidad del software, principalmente se enfoca al software, y menciona que se utilizaron estudiantes que ya están avanzados en POO para realizar trabajo de mantenimiento de software. El propio autor menciona que la mayor limitante del estudio son los datos usados para validar las métricas. En [96], los autores presentan un trabajo en el cual se analizan modelos de calidad, se revisa cuáles son los aspectos evaluados en los modelos y las métricas correspondientes a esos aspectos. Se analiza también los lenguajes de modelado, identificando una serie de características en éstos, como lo son: *Domain appropriateness, Participant language knowledge appropriateness, Knowledge*

externalizability appropriateness, Comprehensibility appropriateness, Technical actor interpretation appropriatenes. En este trabajo se toma en cuenta también la calidad de las herramientas de modelado. Por otra parte, en [97], los autores presentan un análisis de 22 diagramas UML, en el cual, se miden 11 atributos o métricas de calidad descritos en los siguientes párrafos. Se realiza un estudio en el que se presenta la estadística correspondiente. El trabajo se enfoca principalmente en el tiempo de mantenimiento por lo que las mediciones y estadísticas presentadas se concentran en ese rubro principalmente. Los autores de [98] presentan un estudio del estado del arte respecto a las métricas aplicadas para medir la calidad del diagrama de clases. También, son propuestas nuevas métricas en el mismo documento. En las métricas propuestas en algunos casos se muestra la fórmula para obtenerla. Sin embargo, no se muestra la comprobación con base en casos reales analizados. El enfoque del estudio es la mantenibilidad. El principal logro del estudio es la identificación y propuesta de métricas de calidad que pueden ser aplicadas en las primeras etapas del DOO, enfocándose fuertemente en los diagramas UML. Mientras que en [99], los autores proponen una serie de métricas basadas en CMMI, tomando en cuenta 3 modelos de calidad: sintaxis, semántica y estética. Si bien en el trabajo se mencionan 13 diagramas de UML, el principal que se aborda es el diagrama de clases. Establecen una conexión entre los 3 modelos de calidad considerando que un posible error en uno de ellos probaría errores e interpretaciones incorrectas de los otros dos. La revisión de los elementos de los diagramas se enfoca principalmente en la correctitud del mismo, integridad, consistencia y simetría, considerando los 3 modelos de calidad. En [100], los autores abordan el problema de medir la calidad de los diseños orientados a objetos utilizando métricas dinámicas. Se presenta un conjunto de métricas para medir la calidad de diseños en una fase inicial de desarrollo. El conjunto consta de métricas para la complejidad dinámica y el acoplamiento de objetos basado en escenarios de ejecución.

2.2.5.5.1 Herramientas para la Medición de Expresiones y Emociones en E-Learning que Consideran un Análisis Dinámico y de Abajo Hacia Arriba.

Por su parte, los autores de [101] hacen una revisión de varios grupos de métricas propuestas por otros autores. El estudio tiene como objetivo proponer algoritmos para hacer el cálculo de las métricas basándose en el mapeo de los metadatos del diagrama de clases en notación UML. Mientras que en [102], los autores realizan una propuesta de "estrategia de detección" con una propuesta de 15 métricas, las cuales están clasificadas entre 4 categorías: *Class* (*GodClass, DataClass, ShotgunSurgery, RefusedBequest,*

ISPViolation); *Method* (*GodMethod, FutureEnvy, TemporaryField*); *Subsystem* (*GodPackage, MisplacedClass*); *Micro-Design* (*LackOfBridge, LackOfStrategy, LackOfState, LackOfSingleton, LackOfFacade*). Estas impactan directamente en uno o varios de los criterios de buen diseño orientado a objetos que son: *Low coupling* (COUPL), *high cohesion* (COHES), *manageable complexity* (COMPLX) y *proper data abstraction* (ENCAPS). Por otro lado, en [103], los autores hacen una revisión de métricas ya estandarizadas y propone el uso de métricas de red, basándose en un análisis de código y realizando ingeniería a la inversa para obtener el diagrama de clases por medio de una herramienta de software que también ya existe, por lo que realiza un análisis de un grupo de herramientas asignándoles una ponderación de desempeño. El trabajo se enfoca principalmente en la ingeniería a la inversa. Finalizando este rubro, en [104], los autores realizan un análisis de métricas para el desarrollo de diagramas de clases. En este trabajo, se realiza la revisión de un conjunto de métricas por pares ya existentes y sus relaciones encontradas en el diagrama de clases, aunque la revisión se valida con base en código fuente.

Debido a sus características. este trabajo de tesis puede ubicarse dentro de los rubros de: Análisis de Métricas de Calidad; Herramientas para la Medición de Calidad en Diagramas UML y Grado de Entendimiento del Diagrama UM, además de tratarse de un trabajo que realiza análisis estático de los diagramas de clases. En las Tablas 3, 4, 5, 6 y 7 se muestran por separado los trabajos relacionados por enfoque, año de publicación y la base de datos de donde se obtuvieron. El Anexo 2 muestra la tabla completa de trabajos relacionados.

En el enfoque de Análisis de Métricas de Calidad se consideran un total de 21 trabajos relacionados que se muestran en la Tabla 3. En el análisis de dichos trabajos se pudo constatar que no existen publicaciones que aborden temas con un enfoque como el que se dió en este trabajo de investigación. Además, se pudo identificar que las métricas de calidad evaluadas están enfocadas en atributos de calidad correspondientes a la etapa de diseño de software.

Tabla 3. Trabajos Relacionados con Enfoque en Análisis de Métricas de Calidad

No.	AÑO	PUBLICACIÓN
1	2016	IEEE
2	2016	Google Scholar
3	1998	Google Scholar
4	1999	Google Scholar
5	2001	Google Scholar

6	2003	IEEE
7	2003	IEEE
8	2004	Google Scholar
9	2004	Google Scholar
10	2005	IEEE
11	2005	IEEE
12	2010	IEEE
13	2014	Google Scholar
14	2014	Google Scholar
15	2000	ACM
16	2002	ACM
17	2001	Google Scholar
18	2005	Springer
19	2001	Google Scholar
20	2007	Google Scholar
21	2016	Google Scholar

El enfoque en Herramientas para la Medición de Calidad en Diagramas UML cuenta con un total de 13 trabajos analizados, los cuales se muestran en la Tabla 4. Con la revisión realizada a este enfoque se pudo corroborar que a este momento no existen publicaciones en las que se realicen mediciones de atributos de calidad como los que se analizan en este trabajo.

Tabla 4. Trabajos Relacionados con Enfoque en Herramientas para la Medición de Calidad en Diagramas UML

No.	AÑO	PUBLICACIÓN
22	2003	Google Scholar
23	2015	IEEE
24	2013	IEEE
25	2015	Google Scholar
26	2013	IEEE
27	2013	IEEE
28	2016	IEEE
29	2018	IEEE
30	2011	Google Scholar
31	2015	Google Scholar
32	2009	IEEE
33	2009	IEEE
34	2010	Google Scholar

El enfoque en Grado de Entendimiento del Diagrama UML cuenta con un total de 22 trabajos analizados, que se enumeran en la Tabla 5. Mediante el análisis de estos trabajos se pudo verificar esta tendencia de investigación.

Tabla 5. Trabajos Relacionados con Enfoque en Grado de Entendimiento del Diagrama UML

No.	AÑO	PUBLICACIÓN
35	2011	IEEE
36	2017	Google Scholar
37	2014	ACM
38	1981	IEEE
39	2019	Google Scholar
40	2006	ACM
41	2015	ACM
42	2001	ACM
43	2017	IEEE
44	2000	ACM
45	2016	Google Scholar
46	2020	IEEE
47	2007	Google Scholar
48	2015	Google Scholar
49	2019	Google Scholar
50	2011	Google Scholar
51	1985	IEEE
52	2004	IEEE
53	2005	Google Scholar
54	2006	Google Scholar
55	2007	IEEE
56	2010	Google Scholar

El enfoque en Estética del Diagrama UML cuenta con un total de 9 trabajos revisados que se listan en la Tabla 6. Este análisis permitió identificar la importancia de este enfoque, asimismo, se pudo observar que la estética es uno de los atributos que cuenta con menos trabajos relacionados para propósitos de esta investigación.

Tabla 6. Trabajos Relacionados con Enfoque en Estética del Diagrama UML

No.	AÑO	PUBLICACIÓN
57	2015	ACM
58	2017	Springer
59	2002	Google Scholar
60	2002	Google Scholar
61	2006	IEEE
62	2018	Google Scholar
63	2001	Springer
64	2016	IEEE
65	2001	ACM

El enfoque en Herramientas para la Medición de Expresiones y Emociones en E-Learning cuenta con un total de 13 trabajos revisados, los trabajos encontrados se listan en la Tabla 7. Este análisis permitió corroborar las herramientas que en distintos trabajos de investigación utilizan para hacer sus mediciones, y la diferencia que existe con las propuestas que se realizan en esta tesis.

Tabla 7. Trabajos Relacionados con Enfoque en Herramientas para la Medición de Expresiones y Emociones en E-Learning

No.	AÑO	PUBLICACIÓN
66	2011	Google Scholar
67	2010	IEEE
68	2016	Google Scholar
69	2007	IEEE
70	2007	IEEE
71	2016	Google Scholar
72	2000	Google Scholar
73	2012	ACM
74	2018	Google Scholar
75	2005	IEEE
76	204	ACM
77	1999	Google Scholar
78	2013	IEEE

2.3. Discusión

El uso de los RA electrónicos actualmente en la educación está ampliamente difundido. Las tecnologías actuales y la masificación del e-learning requieren que los RA que son utilizados en particular los que son de acceso abierto y de forma masiva sean evaluados, con el fin de que aquellos usuarios que los necesitan y buscan cotidianamente tengan información precisa sobre su calidad y usabilidad.

Las propuestas de los autores son muy importantes, ya que permiten tener una perspectiva muy amplia de lo que es necesario evaluar en los RA, particularmente, los que atienden el dominio de IS. Los grupos que se identificaron por enfoque tienen la diferencia que analizan atributos de calidad, lo cual es una ventaja para propósitos de este trabajo, ya que se identifican de primera instancia una serie de atributos que pueden ser medidos. Asimismo, la definición de reglas de dominio que se verá en capítulos posteriores se ve fuertemente apoyada con la información obtenida en estos trabajos relacionados.

Cabe mencionar que el análisis de trabajos relacionados sirvió también para corroborar que no existen actualmente investigaciones que evalúen la calidad de los recursos de aprendizaje para IS con un enfoque de reglas de dominio como se hace en este trabajo de tesis.

Finalmente, el análisis de trabajos relacionados que se realizó ayudó a verificar la originalidad del proyecto de investigación, ya que no se encontró un trabajo que realice el mismo tipo de aplicación de reglas de dominio para IS.

"El cambio es el resultado de cualquier aprendizaje verdadero"

Leo Buscaglia
(1924 – 1998)

Capítulo 3
Metodología de Solución

CAPÍTULO 3. METODOLOGÍA DE SOLUCIÓN

En este capítulo se describe la metodología utilizada para la solución del problema que ya se ha descrito. Así mismo, se describe una serie de reglas de dominio y se define el modelo de calidad para evaluar los recursos.

Para realizar la solución se trabajó con una metodología que consiste en dos etapas, las cuales, se muestran en la Figura 5.

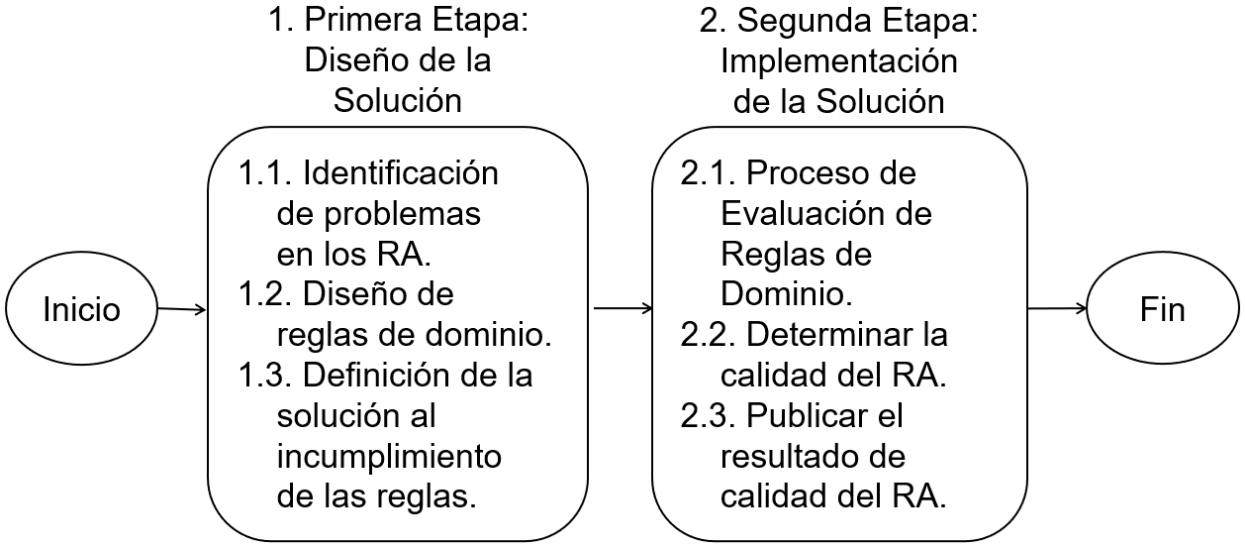


Figura 5. Etapas de la Metodología de Solución

3.1. Primera Etapa: Diseño de la Solución.

La primera etapa se refiere al diseño de la solución plasmado en la Figura 5. Esta actividad consistió en el análisis de Recursos de Aprendizaje que contienen diagramas de clases, con el fin de identificar problemas que puedan afectar la calidad de los mismos; se realizó el diseño de las reglas de dominio que aporten solución a dichos defectos; y, dichas reglas se implementaron en una herramienta de software. El diseño de la solución consta de 3 puntos principales.

3.1.1. Identificación de problemas en los RA

A partir del análisis que se realizó de los trabajos relacionados, se analizaron 211 RA que contienen diagramas de clases, en los cuales, se encontró una serie de características que representan elementos de calidad en los diagramas, por lo tanto, es necesario poder medirlas. Las características identificadas se muestran en la Tabla 8.

Tabla 8. Características de Calidad del Diagrama de Clases

No.	Característica
1	Cantidad de interfaces.
2	Cantidad de clases estáticas.
3	Cantidad de clases hoja.
4	Cantidad de clases hijas por superclase.
5	Profundidad del árbol de herencia.
6	Cantidad de jerarquías de herencia.
7	Cantidad de atributos.
8	Cantidad de operaciones.
9	Cantidad de operaciones públicas.
10	Cantidad de atributos heredados.
11	Cantidad de elementos que dependen de una misma clase.
12	Cantidad de relaciones en el diagrama.
13	Cantidad de relaciones por clase.
14	Cantidad de operaciones heredadas.
15	Cantidad de atributos repetidos entre clases asociadas.
16	Cantidad de operaciones repetidas entre clases asociadas.
17	Cantidad de atributos repetidos entre clases NO relacionadas.
18	Cantidad de operaciones repetidas entre clases NO relacionadas.
19	Cantidad de atributos repetidos entre clases coasociadas (asociadas con una tercera clase).
20	Cantidad de operaciones repetidas entre clases coasociadas (asociadas con una tercera clase).

3.1.2. Diseño de Reglas de Dominio

Durante el desarrollo del proyecto se identificaron y definieron reglas de dominio en la sección 3.1.2.2, para diagramas de clases UML, y se realizó una búsqueda exhaustiva de recursos; específicamente recursos en los que pudieran presentarse los defectos mencionados en la Tabla 8.

3.1.2.1. Definición de Relaciones, Clases y Sus Elementos

La definición de las reglas de dominio se realiza a partir de los hallazgos en los diagramas de clases analizados en la primera etapa del proyecto, en la que se analizaron 145 diagramas de clases. Para esto, primero es necesario determinar los elementos que componen el diagrama de clases UML. Considerando la definición de [105], las clases y las

relaciones se puede definir como un conjunto de elementos tales que: las reglas son expresadas de manera formal utilizando teoría de conjuntos y se plantean las fórmulas para asignar un valor cuantitativo a la regla, esto se visualiza con el valor resultante del conjunto que afecta la fórmula.

Una clase es un conjunto de atributos y métodos, que puede definirse de la siguiente forma:

C = {a1,..., am, m1,..., mn} donde $C \subseteq A \cup M$
A = {a | ∀ a A(a)} donde a es un atributo de clase.
M = {m | ∀ m (Me(me) ∧ Mi(mi) ∨ Ma(ma))} donde:
Me: es el conjunto de métodos pertenecientes a una clase;
me: es un método que pertenece a Me.
Ma(ma) es el conjunto de métodos abstractos;
ma: es un método abstracto que pertenece a Ma.
Mi(mi) es el conjunto de métodos implementados;
mi: es un método implementado que pertenece a Mi.

Los diagramas de clases tienen relaciones. Una relación representa una conexión entre conjuntos.

$R = \{(x,y) \mid (x, y) \in C \times C\}$;
Una relación puede definirse de la siguiente forma:
Sea α una relación entre dos conjuntos:
$\alpha = \exists x \exists y\ R_1(x, y)$ where $R_1(x, y)$ = "*x* is connected with *y*".
Una relación de herencia puede definirse de la siguiente forma:
Sea β una relación de herencia entre dos conjuntos:
$\beta = \exists x \exists y R_2(x, y)$ where $R_2(x, y)$ = "*x* inherits from *y*".
Una relación de agregación puede definirse de la siguiente forma:
Sea γ una relación de agregación entre dos conjuntos:
$\gamma = \exists x \exists y R_3(x, y)$ where $R_3(x, y)$ = "*x* is part of *y*"
Una relación de composición puede definirse de la siguiente forma:
Sea δ una relación de composición entre dos conjuntos:
$\delta = \exists x \exists y R_4(x, y)$ where $R_4(x, y)$ = "*x* exists if and only if *y* exists".

3.1.2.2. Formalización de Reglas de Dominio

Se realizó la formalización de las reglas de dominio considerando las características de los DC, lo establecido por la notación UML para los DC, y los trabajos relacionados. La formalización contribuyó a establecer como puede medirse si las reglas se cumplen o no, de tal manera que a cada regla definida corresponde una forma de asignarle un valor cuantitativo. En esta sección se describen 56 reglas de dominio en total. La Tabla 9 contiene un listado general de las 56 reglas de dominio, mientras que la definición completa de cada una de ellas se encuentra en el Anexo 1.

Tabla 9. Listado General de Reglas de Dominio

No.	Regla
1.	Definición de elementos en el DC.
2.	Definición de clases en el DC.
3.	Definición de interfaces en el DC.
4.	Definición de clases estáticas en el DC.
5.	Definición de clases sin relación con otras clases en el DC.
6.	Definición de clases abstractas en el DC.
7.	Definición de clases abstractas que no son heredadas sobre una subclase.
8.	Definición de clases que contienen operaciones abstractas pero la clase no se definió como abstracta.
9.	Definición de clases concretas con relación distinta a herencia con una abstracción y que contienen operaciones o atributos de la abstracción.
10.	Definición de clases abstractas que son subclases una clase concreta.
11.	Definición de clases con herencia múltiple en el DC.
12.	Definición de relaciones en el DC.
13.	Definición de relaciones de asociación en el DC.
14.	Definición de relaciones de asociación directa en el DC.
15.	Definición de relaciones de agregación en el DC.
16.	Definición de relaciones de Composición en el DC.
17.	Definición de relaciones de dependencia en el DC.
18.	Definición de relaciones de realización en el DC.
19.	Definición de relación de Generalización en el DC.
20.	Definición de relaciones reflexivas en el DC.
21.	Definición de relaciones incompletas en el DC.
22.	Definición de relaciones no reconocidas por el estándar UML en el DC.
23.	Definición de relaciones que repiten las mismas clases en sus extremos (no reflexivas) en el DC.
24.	Definición de interfaces con relación distinta a realización con una clase concreta.
25.	Definición de atributos en el DC.
26.	Definición de operaciones en el DC.

27.	Definición de operaciones abstractas en el DC.
28.	Definición de atributos y operaciones estáticos en el DC.
29.	Definición de clases cuyos nombres no están en letra bold en el DC.
30.	Definición de atributos u operaciones cuyos nombres están texto bold en el DC.
31.	Definición de operaciones no implementadas entre una interfaz y una clase.
32.	Definición de Operaciones que no se declararon públicas en interfaces.
33.	Definición de Atributos contenidos en interfaces en el DC.
34.	Definición de operaciones repetidos entre Subclase y Superclase.
35.	Definición de atributos repetidos entre Subclase y Superclase.
36.	Definición de operaciones repetidas entre clases asociadas en el DC.
37.	Definición de atributos repetidos entre clases asociadas en el DC.
38.	Definición de operaciones repetidas entre clases no relacionadas.
39.	Definición de atributos repetidos entre clases no relacionadas.
40.	Definición de operaciones repetidas entre clases coasociadas.
41.	Definición de atributos repetidos entre clases coasociadas.
42.	Definición de operaciones repetidas entre 2 o más clases que se ubican en distinto árbol de herencia.
43.	Definición de párrafos de texto con disposición distinta a horizontal.
44.	Definición de nombres de clase con alineación distinta a centro.
45.	Definición de atributos con alineación distinta a izquierda en el DC.
46.	Definición de operaciones con alineación distinta a izquierda en el DC.
47.	Definición de párrafos de texto con tamaño de letra pequeño en el DC.
48.	Definición de párrafos de texto con tamaño de letra grande en el DC.
49.	Definición de párrafos de texto con color de letra distinto al recomendado.
50.	Definición de clases con color de fondo distinto al recomendado.
51.	Definición de figuras con color de borde distinto al recomendado.
52.	Definición de figuras con bordes con grosor distinto al recomendado.
53.	Definición de relaciones con línea punteada pero distintas a DASH.
54.	Definición de cruces entre líneas que representan relaciones en el DC.
55.	Definición de líneas que representan relaciones y que tienen más de dos dobleces (excepto reflexivas).
56.	Definición de operaciones abstractas no implementadas en el DC.

3.1.3. Definición de la Solución de Incumplimiento a las Reglas de Dominio en los RA

La solución desarrollada consiste en la definición de un modelo de calidad con base en las características documentadas. Una vez identificadas las características, se diseñó una herramienta de software que permite procesar diagramas de clases UML e implementar las reglas de dominio en los mismos, teniendo como resultado un reporte que contiene los defectos detectados y las reglas de dominio aplicadas.

3.1.3.1. Modelo de Calidad para Evaluar los Diagramas de Clases UML

La razón para crear un modelo de calidad es poder medir de manera cuantificable los atributos de calidad de los DC. Los autores en [106] consideran la medición, como una teoría auxiliar que establece la relación entre indicadores y constructos, partiendo de la base, que la medición tiene la misma importancia para la investigación científica que la teoría fundamental o sustantiva, mediante la cual se busca explicar los fenómenos, por medio de la especificación de las relaciones de unos conceptos, con otros. La representación gráfica del modelo de calidad creado se puede ver en la Figura 6.

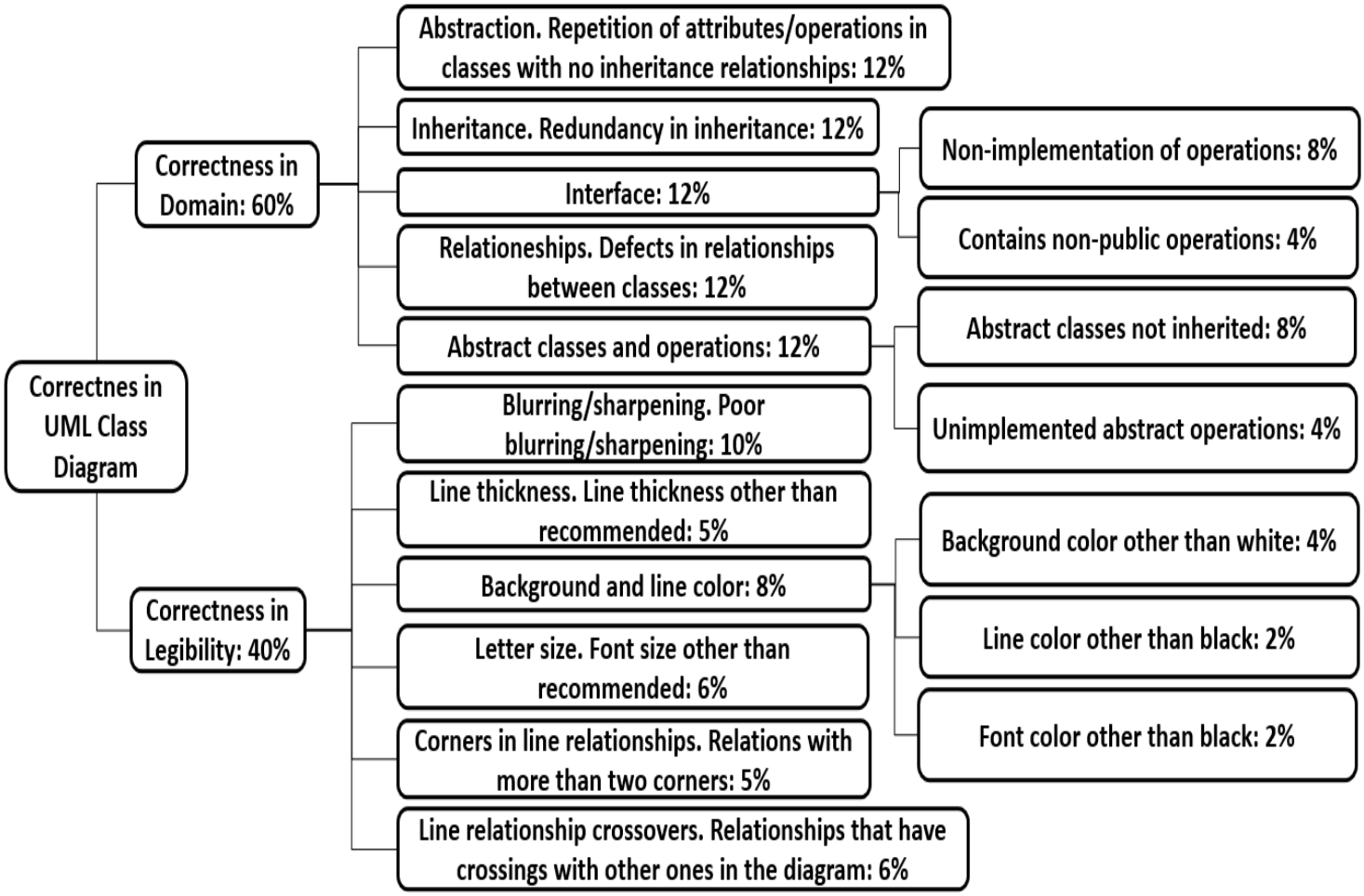


Figura 6. Diagrama del Modelo de Calidad en el Diagrama de Clases UML

La confiabilidad según [106] es la propiedad que busca que cualquier procedimiento de medición genere los mismos resultados en eventos repetidos. De igual forma indican que es imposible que un proceso de medición esté libre de errores, por lo que se busca alcanzar

que en los distintos procesos de medición exista consistencia de la misma. Asimismo, mencionan que la validez de las mediciones está presente cuando el proceso medición mide lo que realmente se pretende medir con el mismo. La validez de la medición está relacionada con la selección de los elementos que se pretende medir, es decir, si los elementos seleccionados son parte del dominio específico, para esto, el dominio debe estar bien definido.

La principal finalidad de un modelo de calidad es especificar y evaluar el cumplimiento de criterios del producto, para lo cual se aplican medidas internas y/o medidas externas [107]. Para poder emitir un valor de calidad se diseñó el modelo que se muestra en la Figura 6. El atributo principal del modelo de calidad se nombró "correctitud".

Para [108] la correctitud de un diagrama se puede ver desde dos puntos diferentes de vista. Desde un punto de vista interno, hay algunas propiedades típicas que pueden probarse automáticamente para determinar este tipo de corrección, como, la satisfactibilidad del diagrama, la validez de sus clases y asociaciones, la no redundancia de sus restricciones o la capacidad de ejecución de sus operaciones. Por otro lado, desde un punto de vista externo, la correctitud se refiere a la precisión de un diagrama con respecto a los requisitos del usuario. Asimismo, [109] menciona que la correctitud se logra mediante la adopción de un particular lenguaje y siguiendo todas sus reglas de inferencia asociadas. Esta última definición es la que más se apega al trabajo de este proyecto.

Para propósitos de este trabajo, la correctitud de los diagramas de clases se forma por dos factores principales, que son: Correctitud en dominio UML y Correctitud en Legibilidad. Dichos factores fueron subdivididos y se les asignaron porcentajes de importancia como se presentan en la Figura 6. Estos valores, se asignaron como valores iniciales debido a que en la revisión de trabajos relacionados durante el desarrollo del proyecto no se encontró ningún trabajo que sugiriera valores de importancia de cada atributo. Los valores iniciales podrán cambiar dependiendo del contexto de uso o cuando expertos realicen su propia valoración.

3.1.3.2. Correctitud en Dominio UML

Este rubro se refiere a la correctitud en la aplicación de las reglas propias en el diseño del diagrama de clases UML y representa el 60% del valor de calidad asignado en la medición a un DC. La Tabla 10 muestra la medición, que se divide de la siguiente forma:

Tabla 10. Rubros del Modelo de Calidad para Correctitud en Dominio

Parámetro de calidad	Descripción
Abstracción 12%	La abstracción tiene varios rubros que son revisados por medio de las reglas de dominio diseñadas. Muchas veces cuando varias clases que no están relacionadas entre sí o tienen algún tipo de relación que no es herencia repiten entre ellas operaciones y atributos, cuando esto ocurre las reglas de dominio diseñadas incluyen la recomendación de generar una nueva clase o abstracción que contenga esas operaciones y atributos repetidos y puedan ser heredados a las subclases correspondientes [110]. Se consideran los atributos que indican que posiblemente se requiere una abstracción nueva para eliminar repeticiones de campos y operaciones entre clases que no tienen relación entre ellas y que no son clase padre-hija. • Cantidad de clases abstractas que son subclases una clase concreta. • Cantidad de operaciones abstractas en el diagrama. • Cantidad de atributos repetidos entre clases no relacionadas. • Cantidad de operaciones repetidas entre clases no relacionadas. • Cantidad de atributos repetidos entre clases “Hermanas o asociadas”. • Cantidad de operaciones repetidas entre clases “Hermanas o asociadas”. • Cantidad de clases nuevas que requieren se reubique una relación.
Herencia 12%	Analizando las reglas respecto a la herencia en UML se diseñaron las reglas de dominio respecto a las operaciones y clases cuando son heredadas desde una superclase. En este caso cuando son heredados no deben repetirse en las subclases ya que es redundante, además de que puede causar confusión el hecho de que en algunos casos son representados y en otros no. Por lo que las reglas de dominio recomiendan que una vez que las operaciones y atributos se representaron en la superclase ya no deben ser representados en las subclases ya que la relación de herencia hereda todas esas características [110]. Se consideran todas aquellas características que indican que existen defectos en las relaciones como por ejemplo la redundancia de operaciones y atributos que ya fueron heredados. • Cantidad de clases que tienen más de una superclase relacionada. • Cantidad de jerarquías de herencia. Cuando es mayor a 2 se considera fuera del valor de calidad. • Profundidad del árbol de herencia. Cuando es mayor a 4 se considera fuera de valor de calidad. • Cantidad de operaciones repetidas entre 2 o más clases que se ubican en distinto árbol de herencia. • Cantidad de atributos repetidos entre superclase y subclase. • Cantidad de operaciones repetidas entre superclase y subclase.
Relaciones 12%	En algunos casos se detectó que la representación de las relaciones tiene defectos, ya que por ejemplo utilizan un solo tipo de línea para todas como si se tratara en todos los casos de asociaciones, cuando en realidad algunas de esas relaciones del diagrama pueden ser herencia u otro tipo de relación distinta a la asociación. Para este caso se diseñaron reglas de dominio que buscan detectar inconsistencias en el tipo de relación que hay entre dos clases y los atributos y operaciones que tienen definidos [111].

	Se consideran aquellas características que indican que la relación establecida entre dos clases podría ser equivocada. • Cantidad de clases que contienen operaciones abstractas pero la clase no se definió como abstracta. • Cantidad de clases concretas con relación distinta a herencia y que contiene métodos o atributos de la abstracción. • Cantidad de atributos repetidos entre clases asociadas. • Cantidad de operaciones repetidas entre clases asociadas. • Cantidad de atributos repetidos entre clases coasociadas. • Cantidad de operaciones repetidas entre clases coasociadas. • Cantidad de conexiones duplicadas entre dos clases. • Cantidad de relaciones no reconocidas por el estándar UML.
Interfaz 12%	Las interfaces son un caso muy particular, ya que sus operaciones deben ser implementadas por las clases que deben aplicar el comportamiento que la interfaz especifica. Las reglas de dominio respecto a interfaces se diseñaron para: • Verificar si las operaciones fueron implementadas. • Si dichas operaciones tienen tipo de acceso público. • Si la interfaz contiene atributos, entre otras situaciones que pueden generar confusión al momento de hacer uso de una interfaz. • También, cuando existen operaciones repetidas entre clases que no son hermanas y que además no forman parte del mismo árbol de herencia, es posible recomendar la creación de una interfaz para aplicar el comportamiento [111] [112]. Se consideran todas aquellas características de interfaces como la implementación de sus operaciones en otra clase. • Cantidad de operaciones no implementadas entre la interfaz y la clase concreta. • Cantidad de operaciones en las interfaces que no son públicas. • Cantidad de interfaces que contienen atributos. • Cantidad de atributos repetidos entre clases que forman parte de árboles de herencia distintos.
Clases y operaciones abstractas 12%	Las clases abstractas también deben implementar sus operaciones abstractas en otra clase, además las clases abstractas al contener comportamiento que debe ser implementado no pueden ser instanciadas, por lo que las reglas de dominio se diseñaron para verificar precisamente esas características: • Que la clase tenga una relación de herencia para que se pueda implementar el comportamiento. • Que las subclases que heredan de esa clase abstracta implementen las operaciones que se definieron como abstractas (para este caso no se considera redundancia). • Que las clases abstractas no sean clase hoja, esto, debido a que una clase abstracta no puede ser instanciada, por lo que requiere que exista una relación de herencia con al menos una subclase que si pueda ser instanciada. Se consideran aquellas características de las clases abstractas debido a que estas no se pueden instanciar [110] por lo que no heredar sus propiedades en otra clase sería un defecto.

	• Cantidad de clases abstractas que no se heredaron sobre al menos una subclase. • Cantidad de operaciones no implementadas entre la abstracción y la subclase.

3.1.3.3. Correctitud en Legibilidad

Este rubro se refiere a que tan legible es el diagrama al momento de analizarlo visualmente, ya que un diagrama que no puede ser interpretado por el ojo humano es generalmente descartado de forma inmediata, las reglas de dominio se diseñaron para verificar los rubros mostrados en la Tabla 11.

La legibilidad para [113] es un conjunto de características que facilitan o dificultan la comprensión eficaz de un documento. La legibilidad está ligada a la comprensión, pero no son lo mismo, se divide en 6 tipos, que son:

1) Legibilidad Física. Es la posibilidad sensorial que tiene el usuario de dominar el contenido de un documento.
2) Legibilidad Lingüística. Es la relación entre los aspectos gramaticales y morfológicos contenidos en el texto de un documento, y que impactan al momento que el usuario interpreta el contenido del documento.
3) Legibilidad Psicológica. Examina si el contenido del documento despierta o no el interés del usuario o si provoca su rechazo.
4) Legibilidad Conceptual. Examina si el contenido puede ser interpretado por el usuario (dificultad conceptual) y si el contenido cumple alguna función dentro del contexto cultural del usuario (significatividad).
5) Legibilidad Estructural. Examina si la organización del contenido del documento permite su lectura o la hace difícil o imposible, y
6) Legibilidad Pragmática. Examina si una vez que el usuario termino la lectura del documento, este tendrá una respuesta global deseable.

Tabla 11. Rubros del Modelo de Calidad para Correctitud en Legibilidad

Parámetro de calidad	Descripción
Difuminación/ nitidez 10%	Para la nitidez del diagrama se consideró el valor más alto de la Correctitud en Legibilidad, esto, debido a que un diagrama que no tiene nitidez no puede ser interpretado por el ojo humano, por lo que difícilmente podría ser comprendido en su totalidad; cuando un diagrama tiene cierta nitidez, causa problema para darle seguimiento con el ojo humano e interpretarlo correctamente; mientras que un

	diagrama que tiene la nitidez correcta puede ser interpretado más fácilmente por el ojo humano [84]. Cuando un diagrama presenta baja nitidez se debe poner el parámetro en 1 para indicar que existe ese defecto. • Nivel de difuminación: 0 = sin difuminación; 1 = presenta difuminación; se considera en valor de calidad.
Colores 8%	El uso de colores en los diagramas no está recomendado por el estándar UML, el cual recomienda en su lugar representar las los diagramas con fondo blanco, las clases y relaciones en negro y los nombres, atributos y operaciones con texto plano [112]. Se considera la recomendación del UML de que los diagramas deben representarse con fondo blanco, letra y bordes en color negro. • Color de relleno de la diapositiva, *blue*. • Color de relleno de la diapositiva, *red*. • Color de relleno de la diapositiva, *green*. • Color de relleno en figura distinto al Predefinido. • Color de línea en figura distinto a los recomendados. • Cantidad de figuras con línea transparente. • Color de línea en conector distinto a los recomendados. • Cantidad de *MEMBERS* (atributo u operación) con color de texto distinto al predefinido. Los cruces de las líneas que representan relaciones suelen ser difíciles de dar seguimiento por el ojo humano sobre todo cuando los cruces son repetitivos [84] [86] [114].
Cruces de relaciones 6%	Se contabilizan los cruces de líneas que representan relaciones entre clases mismo [48], [77], [84], [86], [111]. • Cantidad de cruces de conectores. • Cantidad de cruces de figuras en general.
Tamaño de letra 6%	El propio estándar UML recomienda el uso de texto plano, eso implica que el texto pueda ser leído e interpretado por el ojo humano de forma normal si otra ayuda, el tamaño de letra por tanto debe estar ajustado a que el ojo humano pueda reconocer dicho texto de forma fácil, por tanto, el tamaño de letra no debe ser pequeño y tampoco demasiado grande [112]. Se considera el tamaño de letra ya sea muy pequeño y muy grande. • Cantidad de clases con texto con tamaño de fuente menor a la recomendada. • Cantidad de clases con texto con tamaño de fuente mayor a la recomendada.
Dobleces o esquinas en relaciones 5%	Los dobleces en los bordes de las relaciones tienen el problema que deben ser seguidos por el ojo humano para verificar donde termina cada relación, el exceso de dobleces en una misma relación dificulta ese seguimiento y puede dificultar la interpretación del diagrama [115]. Se contabilizan los dobleces en líneas que representan relaciones exceptuando aquellas que son reflexivas. Cualquier línea que tenga más de dos dobleces es considerada en el conteo. • Cantidad de conexiones con más de dos dobleces o esquinas en escuadra sin contar relaciones reflexivas.

Grosor Bordes 5%	El grosor de los bordes de líneas y clases pude provocar que se vea más oscuro el diagrama afectando la legibilidad, por lo que los bordes gruesos no son recomendados [84]. El grosor de bordes oscurece el diagrama lo cual viene mencionado en revisión de literatura. • Cantidad de conectores que tienen un *height* o un *width* más pequeño que el configurado. • Cantidad de bordes de figuras con grosor distinto al recomendado.

3.2. Segunda Etapa: Implementación de la Solución

La segunda etapa de la metodología de solución corresponde la Implementación de Solución. En esta etapa se realizó la implementación de la solución por medio de una herramienta de software, la cual, se encarga de:

- Analizar el diagrama de clases contenido en un RA en formato PPTX mediante la lectura de sus metadatos.
- Procesar las reglas de dominio previamente diseñadas.
- Determinar el valor de calidad del RA.
- Publicar finalmente el resultado que refleja la correctitud del RA.

La Correctitud del DC bajo el modelo propuesto se calcula bajo la siguiente fórmula:

Correctitud = [Correctitud en Dominio] + [Correctitud en Legibilidad]

3.2.1. Proceso de Evaluación de Reglas de Dominio

El proceso de evaluación de los DC se hace a partir de las reglas de dominio especificadas en la sección 3.1.2.2 se realiza mediante de una herramienta de software que se desarrolló para la interpretación de DC contenidos en archivos en formato PPTX. Se realizó un desarrollo basado en el Paradigma Orientado a Objetos utilizando el lenguaje Java, con la librería Apache-POI de uso libre, así como el manejador de base de datos MySQL. El diseño de esta herramienta se explica a detalle en el Capítulo IV de este documento.

3.2.1.1. Diagrama de Proceso de Reglas de Dominio

La Figura 7 muestra el diagrama del proceso de lectura del archivo PPTX. El proceso de la herramienta consiste en 9 pasos.

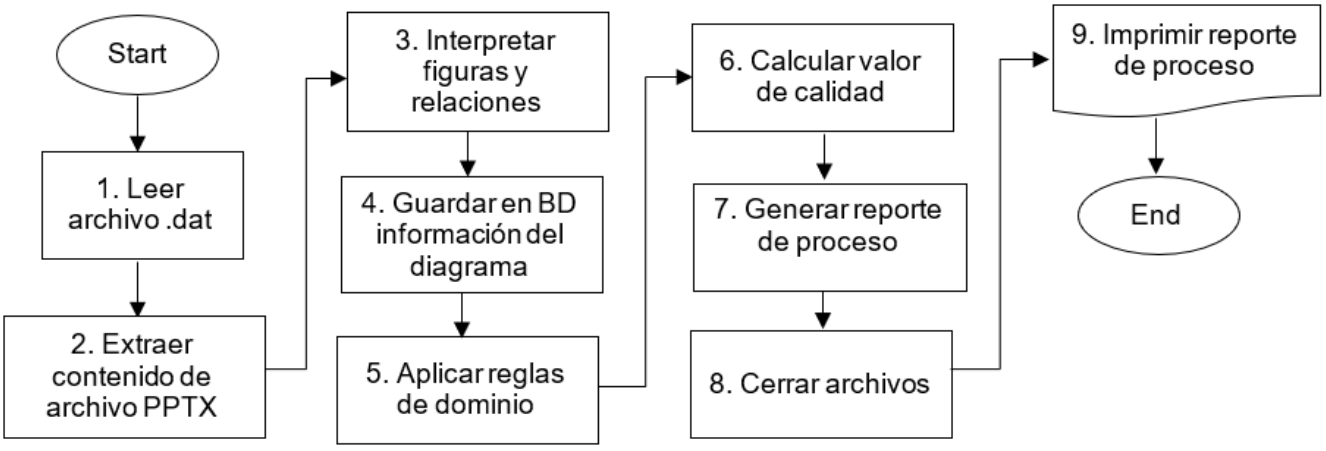


Figura 7. Diagrama de Flujo del Proceso de Lectura de Archivo PPTX

1. Abrir y leer el archivo de configuración de la herramienta que contiene entre otros datos: nombre del MySQL server, usuario y contraseña; ruta y nombre del archivo PPTX a leer; nombre del archivo de reporte general; y otras configuraciones.

2. Al iniciar la lectura del archivo PPTX, se consultan sus metadatos y se extrae todo su contenido de figuras interpretables por Apache POI, todos estos datos extraídos se almacenan en la base de datos en MySQL previo a su interpretación.

3. Se interpreta y caracteriza el contenido del archivo guardado en la base de datos como lo son: Clases, atributos, operaciones; conexiones y relaciones de tipo herencia, asociación, agregación, composición, realización; entre otros, además de formar el DC en de forma lógica con todas las características detectadas.

4. Toda la información generada en el paso anterior se guarda en la BD. Una vez que se interpretó todo el contenido del archivo PPTX, a partir de este punto el archivo PPTX ya no es leído más y todo el proceso restante se realiza con la información contenida en la base de datos.

5. Se procesan y aplican las reglas de dominio, esto genera la mayor parte del reporte de proceso de diagramas de clases UML, cuyos resultados se guardan en la BD. Con base en los atributos de calidad identificados en los trabajos relacionados y en el modelo de calidad mostrado en la sección 3.1.3.1, se definieron una serie de reglas de dominio que fueron incluidas en la herramienta de software, en la mayoría de sus casos, estás reglas tienen como objetivo servir como base de la evaluación del RA que contiene el diagrama de clases UML.

6. Se Realiza el cálculo del valor de calidad del diagrama con base en el modelo propuesto.
7. Se genera el reporte del proceso, el cual, se guarda también en un archivo de texto.
8. Se cierran los archivos y la conexión con la base de datos.
9. Se imprime el reporte del proceso para ser analizado por el usuario.

3.2.2. Determinar la Calidad del RA

Una vez que las reglas de dominio se aplicaron es posible determinar el valor de calidad del RA. Esto se hace mediante los valores guardados en la base de datos, estos son valores que pueden ser personalizados, es decir, el modelo de calidad puede ser calibrado, y es posible agregar nuevos rubros para ser evaluados, o bien, eliminar rubros que en determinado momento pueden ser considerados innecesarios. También, es posible asignar valores distintos a porcentajes de los rubros del modelo de calidad diseñado.

3.2.3. Publicar el Resultado de Calidad del RA

Como paso final del proceso, se publica un reporte que contiene los resultados de la aplicación de las reglas de dominio.

3.3. Discusión

Con base en el estudio realizado se diseñaron las reglas de dominio necesarias para medir la calidad de los RA que contienen diagramas de clases. También, se diseñó un modelo de calidad. Estos son aspectos de gran importancia, ya que se cuenta con un modelo que sirve como referencia para la evaluación de RA que se encuentran en distintos lugares de Internet, los cuales, son descargados y utilizados asumiendo que son correctos al 100%, lo cual podría no ser así.

Tanto el modelo de calidad como las reglas de dominio tienen la ventaja de proporcionar a los usuarios y aprendices de IS una herramienta importante en sus actividades. Asimismo, el modelo creado puede crecer en cantidad de reglas de dominio, por lo que no se trata de un modelo rígido o estático, sino un modelo dinámico que puede adaptarse a nuevas necesidades del dominio.

"Nada en la vida es para ser temido,
es sólo para ser comprendido.
Ahora es el momento de entender más,
de modo que podamos temer menos"

Marie Curie
(1867 – 1934)

Capítulo 4
Implementación de la Solución

CAPÍTULO 4. IMPLEMENTACIÓN DE LA SOLUCIÓN

En ese capítulo se describen la implementación del modelo de calidad descrito en la Figura 6, y el diseño como el diseño de la herramienta de software, que se creó con el objetivo de hacer la aplicación de las reglas de dominio desarrolladas y descritas en la sección 3.1.2.2.

4.1. Requisitos Funcionales y de Instalación de la Herramienta de Software

La herramienta de software se diseñó con el fin de evaluar un RA que contiene uno o varios diagramas de clases UML, dicho recurso debe estar construido en formato PPTX. Se asume que cada archivo contiene una diapositiva con un solo diagrama de clases; sin embargo, si el archivo PPTX contiene más de una diapositiva, o una diapositiva contiene más de un diagrama de clases, el archivo será evaluado creando una nueva entidad por cada diapositiva.

4.1.1. Requisitos Funcionales

Los requerimientos funcionales según [116] son aquellos que describen las capacidades de la herramienta de software, las acciones y respuestas que el sistema realiza, así como los datos que administrará la misma. Los requisitos funcionales para la herramienta de software son los siguientes:

- Realizar el inicio de sesión de usuarios al sistema.
- Realizar configuración de nuevos usuarios al sistema.
- Realizar apertura de recursos para su cálculo de calidad.
- Caracterizar el contenido de un recurso de acuerdo a los DC UML mediante los metadatos que contiene el archivo del recurso.
- Calcular valores de calidad de acuerdo a las reglas de dominio del modelo implementado.
- Guardar toda la información de la caracterización de los recursos
- Guardar los cálculos realizados en la BD.
- Desplegar reportes en pantalla con los valores de calidad calculados previamente en los recursos.

- Generar reporte de valores de calidad calculados previamente en los recursos en archivo formato CSV.
- Descargar en el dispositivo del usuario el archivo formato CSV generado.
- Cerrar la sesión activa del usuario.

4.1.2. Requisitos de Instalación

La herramienta de software funciona en plataforma Web y requiere la instalación de las siguientes tecnologías de software:

- Lenguaje de programación JAVA 8 o una versión más reciente.
- Herramienta APACHE POI 4.1.2 o más reciente, siendo está una API de JAVA para trabajar con documentos Microsoft Office.
- Servidor Apache Tomcat 7.0.32 o superior, o un servidor compatible.
- Manejador de bases de datos MySQL Standard Edition.

4.2. Diseño de la Herramienta de Software

4.2.1. Diagrama de Casos de Uso

Con el fin de identificar las formas de uso del y como se interactúa con la misma se desarrolló el de Diagrama de Casos de Uso correspondiente. La Figura 8 muestra el diagrama de casos de uso de la herramienta, el usuario es el encargado de realizar las siguientes acciones:

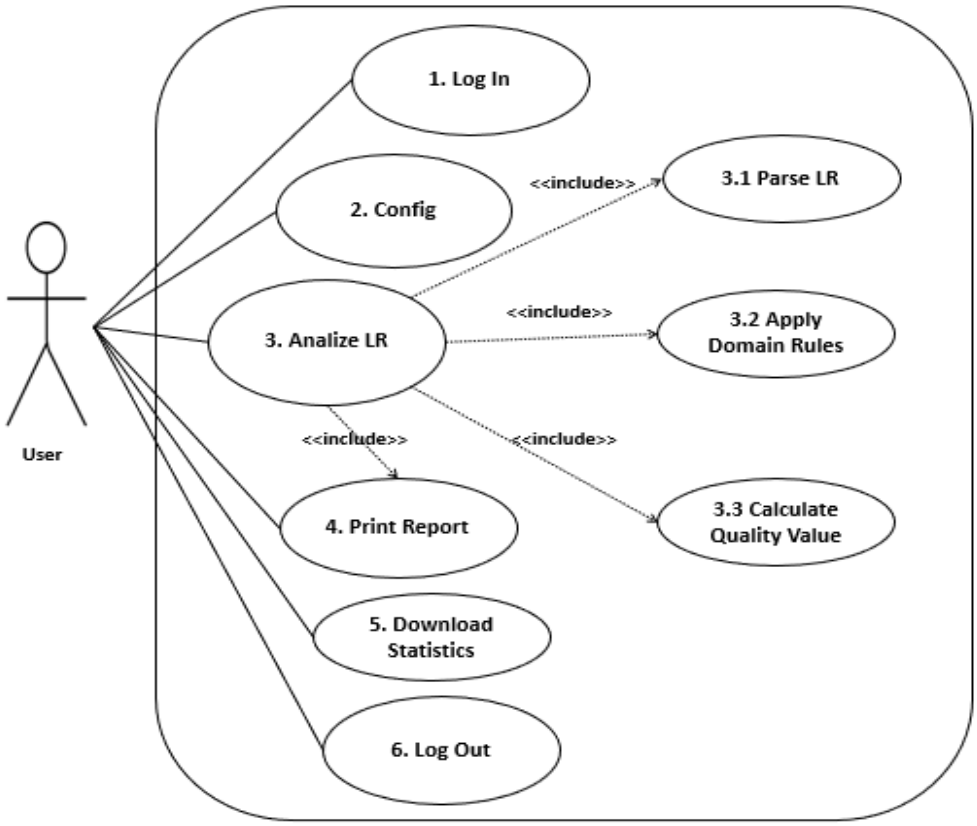


Figura 8. Diagrama de Casos de Uso de la Herramienta de Software

4.2.1.1. Especificaciones de Casos de Uso

Tabla 12. Especificación del caso de uso Log In

ID	**1**	**Nombre**	**Log In**
Elaborado por		JASA.	
Actores		USER, Admin.	
Objetivo		Permitir al usuario iniciar una nueva sesión activa en el sistema.	
Precondiciones		- El usuario debe estar registrado en el sistema.	
Postcondiciones		- El usuario podrá iniciar una sesión activa en el sistema.	
Flujo de eventos			
Acción del actor		**Respuesta del Sistema**	
1. El usuario ingresa contraseña y usuario del sistema		2. Validar datos. 3. Permitir ingreso al sistema. 4. Mostrar la interfaz correspondiente.	

Tabla 13. Especificación del caso de uso Config

ID	**2**	**Nombre**	**Config**
Elaborado por		JASA.	
Actores		Admin.	
Objetivo		Permitir al usuario configurar nuevos accesos al sistema	
Precondiciones		- El usuario debe haber iniciado una sesión en el sistema previamente.	
Postcondiciones		- El usuario podrá crear nuevos accesos al sistema.	
Flujo de eventos			
Acción del actor		**Respuesta del Sistema**	
1. Capturar información de nuevo usuario.		2. Validar datos. 3. Guardar información en la base de datos. 4. Mostrar pantalla de confirmación de usuario creado.	

Tabla 14. Especificación del caso de uso Analize LR

ID	**3**	**Nombre**	**AnalizeLR**
Elaborado por		JASA.	
Actores		USER, Admin.	
Objetivo		Permitir al usuario analizar un recurso	
Precondiciones		- El usuario debe estar registrado en el sistema. - El usuario debe haber iniciado una sesión en el sistema previamente. - El recurso debe estar en el sitio configurado para análisis	
Postcondiciones		- El usuario podrá analizar el recurso	
Flujo de eventos			
Acción del actor		**Respuesta del Sistema**	
1. Seleccionar recurso a analizar		2. Validar existencia del recurso. 3. Abrir el recurso 4. Extraer contenido del recurso 5. Procesar el contenido y guardarlo en la BD 6. Realizar cálculos de calidad del recurso utilizando las reglas implementadas. 7. Informar al usuario del fin del proceso mostrando el mensaje correspondiente.	

Tabla 15. Especificación del caso de uso Print Report

ID	**4**	**Nombre**	**Print Report**
Elaborado por		JASA.	
Actores		USER, Admin.	
Objetivo		Permitir al usuario imprimir un reporte de recursos procesados	
Precondiciones		- El usuario debe estar registrado en el sistema. - El usuario debe haber iniciado una sesión en el sistema previamente.	
Postcondiciones		- El usuario podrá iniciar el reporte de recursos procesados	
Flujo de eventos			
Acción del actor		**Respuesta del Sistema**	
1. Seleccionar tipo de reporte que se desea (individual o completo)		2. Generar nuevo reporte 3. Mostrar reporte en pantalla.	

Tabla 16. Especificación del caso de uso Download Statistics

ID	**5**	**Nombre**	**Download Statistics**
Elaborado por		JASA.	
Actores		USER, Admin.	
Objetivo		Perite al usuario iniciar el análisis de un recurso	
Precondiciones		- El usuario debe estar registrado en el sistema. - El usuario debe haber iniciado una sesión en el sistema previamente.	
Postcondiciones		- El usuario podrá analizar el recurso	
Flujo de eventos			
Acción del actor		**Respuesta del Sistema**	
1. Seleccionar tipo de reporte que se desea (individual o completo)		2. Generar nuevo reporte completo 3. Descargar el nuevo reporte en el dispositivo del usuario. 4. Mostrar mensaje de conclusión en pantalla.	

Tabla 17. Especificación del caso de uso Log Out

ID	**6**	**Nombre**	**Log Out**
Elaborado por		JASA.	
Actores		USER, Admin.	
Objetivo		Permitir al usuario que inició una sesión cerrar la sesión activa en el sistema.	
Precondiciones		- El usuario debe estar registrado en el sistema. - El usuario debe haber iniciado una sesión en el sistema previamente.	
Postcondiciones		- El usuario podrá cerrar la sesión activa en el sistema.	
Flujo de eventos			
Acción del actor		**Respuesta del Sistema**	
1. Click en el enlace para cerrar sesión.		2. Cerrar sesión activa.	

4.2.2. Diagrama de Clases

El diagrama de clases tiene como propósito describir los componentes de un sistema que típicamente en el modelado llamamos como clases, estas, a su vez se integran por atributos y operaciones, asimismo, las clases de un diagrama se conectan mediante distintos tipos de relaciones, todo en su conjunto describe el funcionamiento conceptual de un diseño orientado a objetos. La Figura 9 muestra el diagrama de clases de la herramienta de software y después de la figura se describen brevemente las clases.

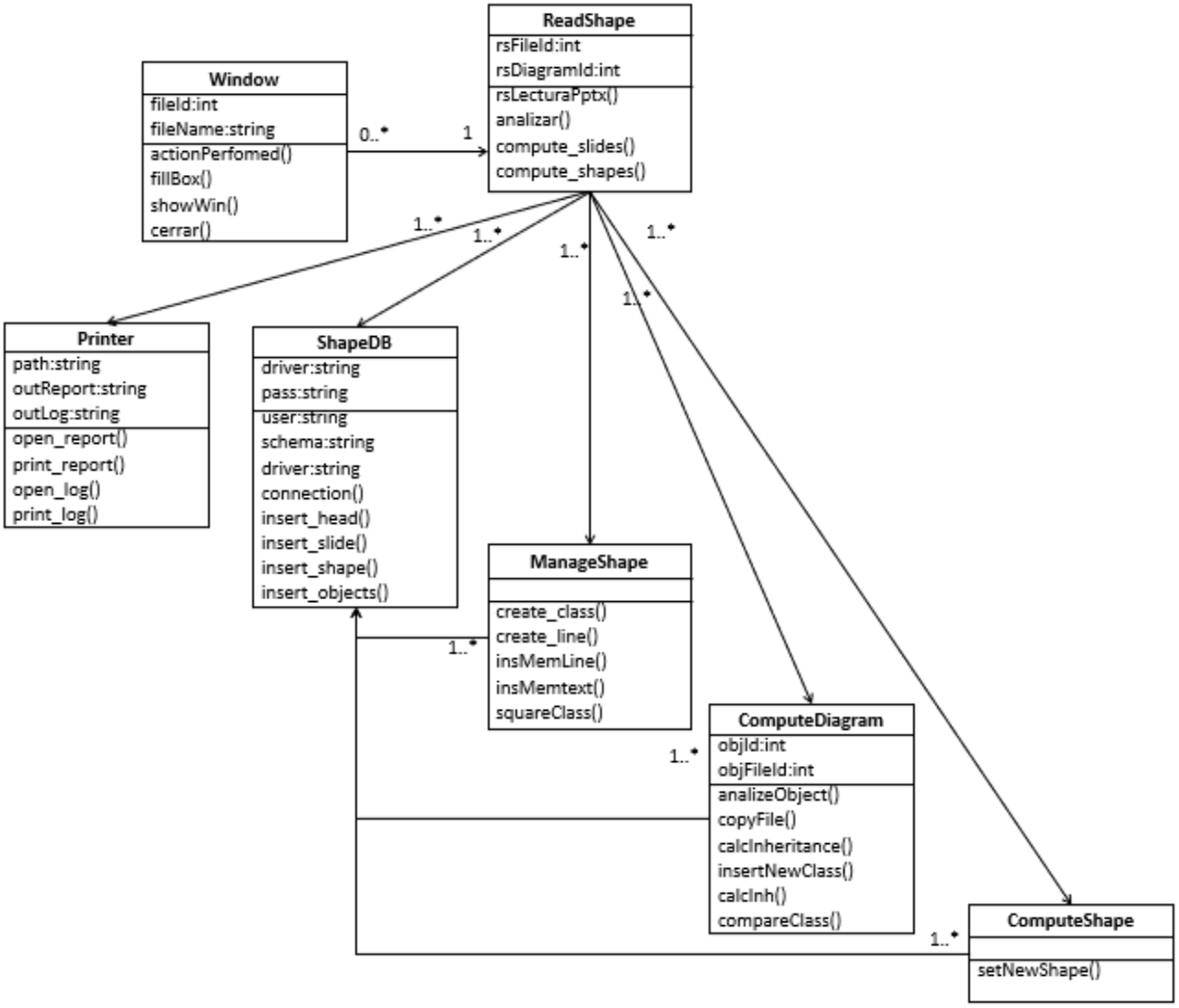


Figura 9. Diagrama de Clases de la Herramienta de Software

- La clase *Window* es la clase cliente y desde donde el usuario iniciará el trabajo para lanzar el proceso de lectura del archivo PPTX y de aplicación de las reglas de dominio, esta clase se relaciona mediante una asociación con la clase *ReadShape*.

- Una vez lanzado el proceso la clase *ReadShape* es la encargada de orquestar el proceso de interpretación de las figuras contenidas en el diagrama a través de las clases: *Printer, ShapeDB, ManageShape*, *MComputeDiagram y ComputeShape*, con las que tiene una relación de asociación.

- la clase *ShapeDB* realiza de la carga de la información a la BD. Asimismo, recibe información a guardar desde las clases: *ReadShape*, *ComputeShape*, *ManageShape*, *Printer*, y *ComputeDiagram*, con las que tiene una relación de asociación.

- La clase *ComputeDiagram* se encarga de realizar los cálculos de la información que debe generarse respecto al contenido de la diapositiva del recurso que es analizado, enviando esa información a la clase *ShapeDB* para que sea guardada en la BD.

- La clase *ComputeShape* se encarga de realizar las conexiones con los SW que calculan los valores del modelo de calidad, enviando esa información a la clase *ShapeDB* para que sea guardada en la BD.

- Finalmente, la clase *Printer* se encarga de crear un reporte de la aplicación en archivo de texto de las reglas de dominio para el usuario de la herramienta.

El diagrama de clases muestra solo algunos de los métodos contenidos debido al tamaño de las mismas.

4.3. Diagrama de Despliegue UML de la Herramienta

El diagrama de despliegue tiene como objetivo mostrar la forma en que los componentes físicos de una solución están relacionados entre sí, asimismo, que componentes de software integran la solución. En la Figura 10 se muestra el Diagrama de Despliegue de la herramienta desarrollada para implementar el modelo.

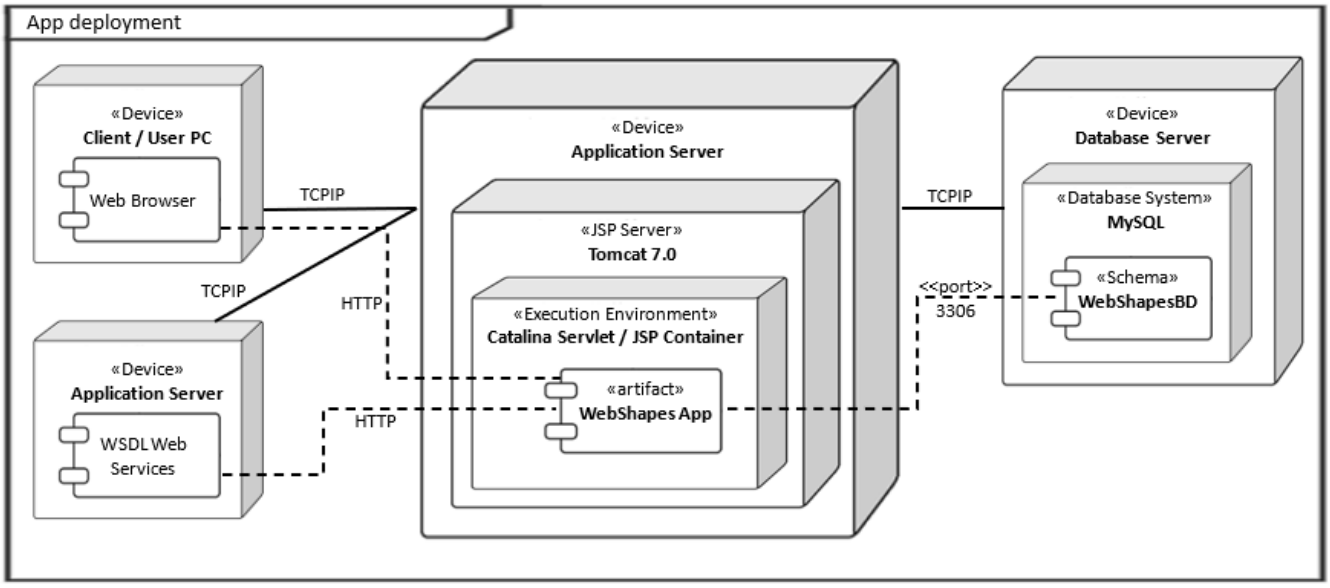


Figura 10. Diagrama de Despliegue UML de la Herramienta de Software

La Tabla 18 muestra la descripción de los componentes del diagrama de despliegue.

Tabla 18. Descripción los Componentes del Diagrama de Despliegue

Componente – Nodo	Descripción del Componente
Client / User PC	El usuario se conecta a la herramienta de software mediante un dispositivo conectado a internet y vía web browser. Este nodo se conecta con el Servidor de aplicaciones mediante de TCPIP. Contiene el artefacto Web Browser, el cual puede ser una versión compatible con la PC del usuario, y se comunica vía HTTP con el artefacto WebShapes App.
Application Server (local network)	El componente Web Server es quien contiene los elementos del servidor de aplicaciones Apache Tomcat, asimismo, almacena también los elementos de Apache server, que son parte requerida de la herramienta de software. Contiene a su vez el artefacto WebShapes App, el cual tiene comunicación con WebShapes DB mediante HTTP.
Database Server	El componente Database Server recibe la conexión desde el Web Server mediante el puerto 3306 en la red local y contiene el esquema de base de datos requerido para la herramienta solución de software.
Application Server (web services)	El nodo Application Server (web services) puede ser externo a la red local o ser desplegado dentro de la misma opcionalmente; cuando es externo se conecta mediante el InteApplicationrnet Server y recibe peticiones del Web Server vía HTTP, este contiene los servicios web que se requieren para aplicar las reglas de dominio diseñadas.

4.4. Diseño de la Base de Datos de la Herramienta de Software

La base de datos está compuesta por una serie de tablas que guardan la información de los objetos que contiene el diagrama, así como sus características. También, contiene una tabla de encabezado por archivo y una tabla de encabezado por diagrama asumiendo que cada diapositiva del archivo PPTX contiene un solo diagrama de clases UML. La Tabla 19 describe el esquema de la base de datos y en la Figura 11 se muestra el esquema de la Base de Datos.

Tabla 19. Esquema de la Base de Datos

Tabla	Descripción
AdminShape	Contiene los usuarios del sistema, así como parámetros para la administración del mismo.
FileHead	Contiene la información referente al archivo que se procesó. Tiene el propósito de guardar una línea de encabezado por archivo, su relación principal es con la tabla *Slide*, que es una relación de uno a muchos, ya que un registro en a tabla *FileHead* puede tener varios registros en la tabla *Slide*. Además, tiene una relación uno a uno con la tabla *Umlreport*, ya que por cada archivo se genera un reporte de proceso, que se guarda dicha tabla.
Slide	Contiene la información procesada y concentrada de cada diapositiva que contiene el archivo PPTX. Aunque se asume que cada archivo contiene una diapositiva y por ende un diagrama de clases, la herramienta está preparada para procesar todas las diapositivas que el archivo PPTX contenga; por lo que se genera un registro por cada diapositiva que contenga el archivo. Tiene una relación de uno a muchos con la tabla *Umlobjects*, ya que por cada registro pueden generarse un número indeterminado de registros en *Umlobjects*, dichos registros representan figuras, relaciones y características contenidas en un recurso de una diapositiva.
UmlObjects	Contiene los objetos que conforman el diagrama de clases por diapositiva del archivo PPTX. Estos objetos pueden ser: clases, relaciones, comentarios, líneas, etcétera. Tiene una relación de uno a muchos con la tabla *Umlobjmembers*, esto debido a que cada registro en *umlobjects* que representa un objeto en el recurso, puede tener relacionado varios registros de características de dicho objeto, dichas características se conservan en los registros guardados en *Umlobjmembers*.
UmlObjMembers	Contiene las características de cada objeto, como lo son atributos y operaciones, texto, entre otras características del recurso analizado.
UmlRules	Contiene las definiciones de las reglas de dominio que pueden ser aplicadas, por lo que tiene tantos registros como reglas hay definidas en el modelo de calidad. Tiene una relación de uno a muchos con *umlqualityvalue*, por lo que un registro de *umlrules* puede estar relacionado con varios elementos *umlqualityvalue*.
UmlMetrics	Esta tabla guarda las métricas que se utilizan para realizar las mediciones de las reglas de dominio
UmlQualityValue	Contiene los valores en porcentajes que se aplican con base en las reglas de dominio.
UmlReport	Contiene la información resultante de la aplicación de las reglas de dominio y se utiliza para generar el reporte final del proceso.

En la Tabla 18 se explican los tipos de relaciones que hay entre las distintas tablas del esquema de base de datos. Asimismo, la Figura 11 muestra el diagrama entidad-relación del diseño de base de datos de la herramienta de software.

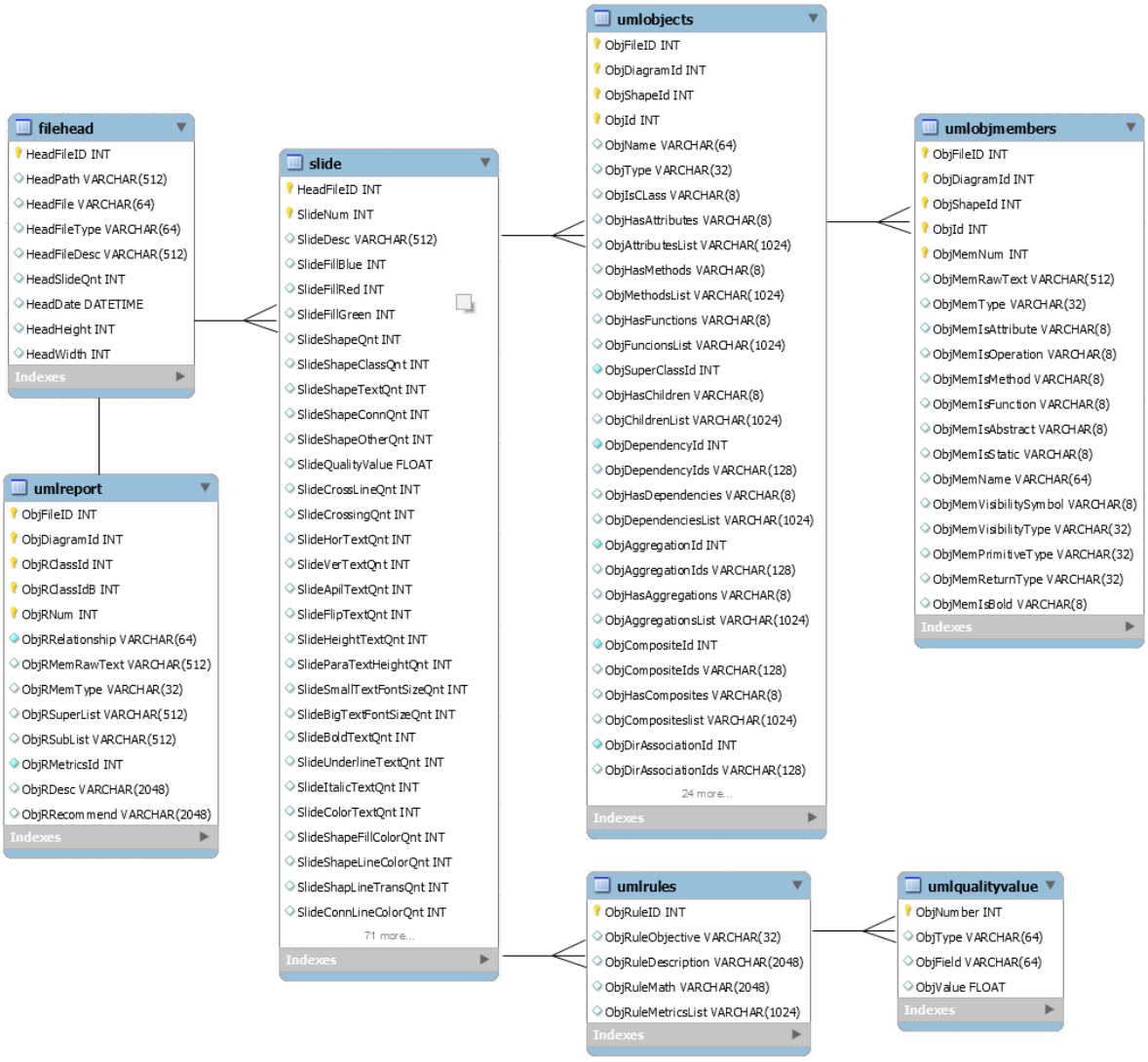


Figura 11. Diagrama Entidad-Relación de la BD de la Herramienta de Software

4.5. Casos de Prueba para la Aplicación de las Reglas de Dominio

Los casos de prueba se diseñaron considerando aquellos defectos más recurrentes encontrados en el análisis de literatura y que castigan la calidad de los RA.

4.5.1. Caso de prueba 1 "Hospital"

Caso de prueba del Diagrama de Clases UML "Hospital" La Figura 12 muestra el caso del diagrama nombrado "Hospital".

La parte del diagrama que será analizada en este caso es la referente a la estética del mismo. A simple vista pueden observarse cruces de líneas que representan relaciones, así como colores en el fondo de las clases y en bordes de líneas, que van contra lo recomendado por [112]. Asimismo, en el diagrama puede observarse la redundancia de operaciones en distintas clases, sin que exista una abstracción o interfaz que implemente dicho comportamiento como lo menciona [110].

Figura 12. Diagrama de Clases "Hospital"

4.5.1.1. Especificación Formal del Ejemplo

Las clases "TEST-REPORT" y "PATIENT" a simple vista tienen al menos un atributo en común, por lo que se tomaron como ejemplo para el análisis de este caso. La Figura 13 muestra el diagrama de Venn del subsistema de prueba.

Sean TEST-REPORT y PATIENT clases del ejemplo;
Sea U = {X | X es un atributo o método de las clases ejemplo};
Siendo U el universo de atributos y métodos de la Figura 12;
Entonces, U = {patno, patname, dis_symptom, test, dis_name, symtoms, gender, address, date, register, login(), view(), specifysymtoms(), paybills(), view_disease()};
Donde PATIENT tiene una relación de asociación TEST-REPORT;
Donde PATIENT = {patno, patname, symtoms, gender, address, date, register, login(), view(), specifysymtoms(), paybills(), view_disease()} es un conjunto finito de atributos o métodos del ejemplo;
Donde TEST-REPORT = {patno, patname, dis_symptom, test, dis_name, view(),} es un conjunto finito de atributos o métodos del ejemplo;
Sea PATIENT ∩ TEST-REPORT = {Ø};
Sea PATIENT ∩ TEST-REPORT = {patno, patname, view()};

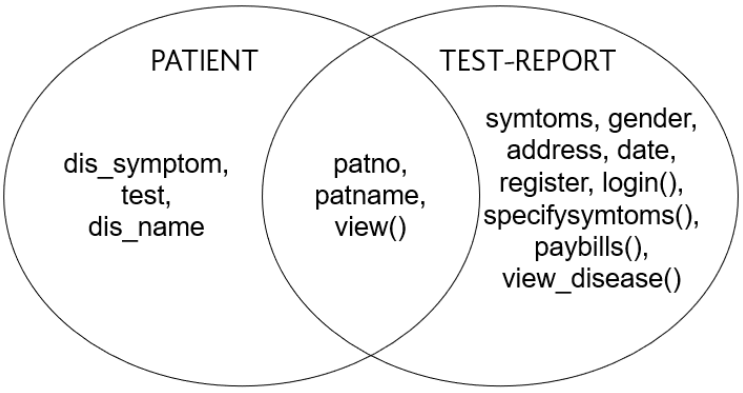


Figura 13. Diagrama de Venn del Caso de Prueba

Operación "view()": Al realizar una comparación entre las clases asociadas "PATIENT" y "TEST-REPORT", se puede observar una repetición de al menos una de sus operaciones. La relación entre ambas clases es de asociación por lo que no se asume que exista polimorfismo. Al aplicar la regla correspondiente se emite la recomendación de crear una

abstracción que solucione dicho defecto, o bien, revisar el tipo de relación que existe entre ambas clases.

Respuesta de la herramienta de software:

Superclass Id: [109]; name: [PATIENT]:
Relationship Type: ASSOCIATION
List [ASSOCIATION] List:
Member Rawtext [view()] Type [OPERATION]
Including Class List:
Detail: Repeated class member between class [PATIENT] and associated class [TEST-REPORT]
Rule: [51] [Cantidad de operaciones repetidas entre clases asociadas];
Recommendation: [51] [Abstract class may be created to apply inheritance, or wrong relationship];
Superclass Id: [109]; name: [PATIENT]:
Relationship Type: ASSOCIATION
List [ASSOCIATION] List:
Member Rawtext [patname:String] Type [ATTRIBUTE]
Including Class List:
Detail: Repeated class member between class [PATIENT] and associated class [TEST-REPORT]
Rule: [52] [Cantidad de atributos repetidos entre clases asociadas];
Recommendation: [52] [Abstract class may be created to apply inheritance, or wrong relationship];
Superclass Id: [109]; name: [PATIENT]:
Relationship Type: ASSOCIATION
List [ASSOCIATION] List:
Member Rawtext [patno:String] Type [ATTRIBUTE]
Including Class List:
Detail: Repeated class member between class [PATIENT] and associated class [TEST-REPORT]
Rule: [52] [Cantidad de atributos repetidos entre clases asociadas];
Recommendation: [52] [Abstract class may be created to apply inheritance, or wrong relationship];

De igual forma, la aplicación de reglas de dominio pudo detectar el cruce de líneas de relaciones en el diagrama, con un total de 5.

Diagram Cross line connectors, Qnty: 5

4.5.2. Caso de Prueba 2 “Blast”

Revisión del Diagrama de Clases UML “Blast”. La Figura 14 muestra el caso de prueba del diagrama llamado “Blast”. En este diagrama se pueden ver varias situaciones en su representación, por ejemplo: se encierran en rojo 2 de ellas.

- La primera, si bien no es un error, puede ser producto de una confusión por parte del diseñador, ya que la clase Blast tiene 3 clases padre en su representación, la herencia múltiple es aceptada por muchos lenguajes; sin embargo, es posible que la dirección de las flechas que indican herencia se invirtió por equivocación, por lo que debe darse una alerta para su revisión.
- La segunda, la clase MultipleAligent se encuentra alejada del resto y su relación de agregación no hace contacto con ninguna otra, por lo que el usuario debe asumir a cuál unirla, existiendo la posibilidad de que ocurra una asignación errónea.
- Además de los dos puntos que se encontraron en la revisión visual rápida, es posible que el diagrama contenga otros defectos que la herramienta de software puede detectar con base en el análisis de los metadatos del archivo y la aplicación de las reglas de dominio.

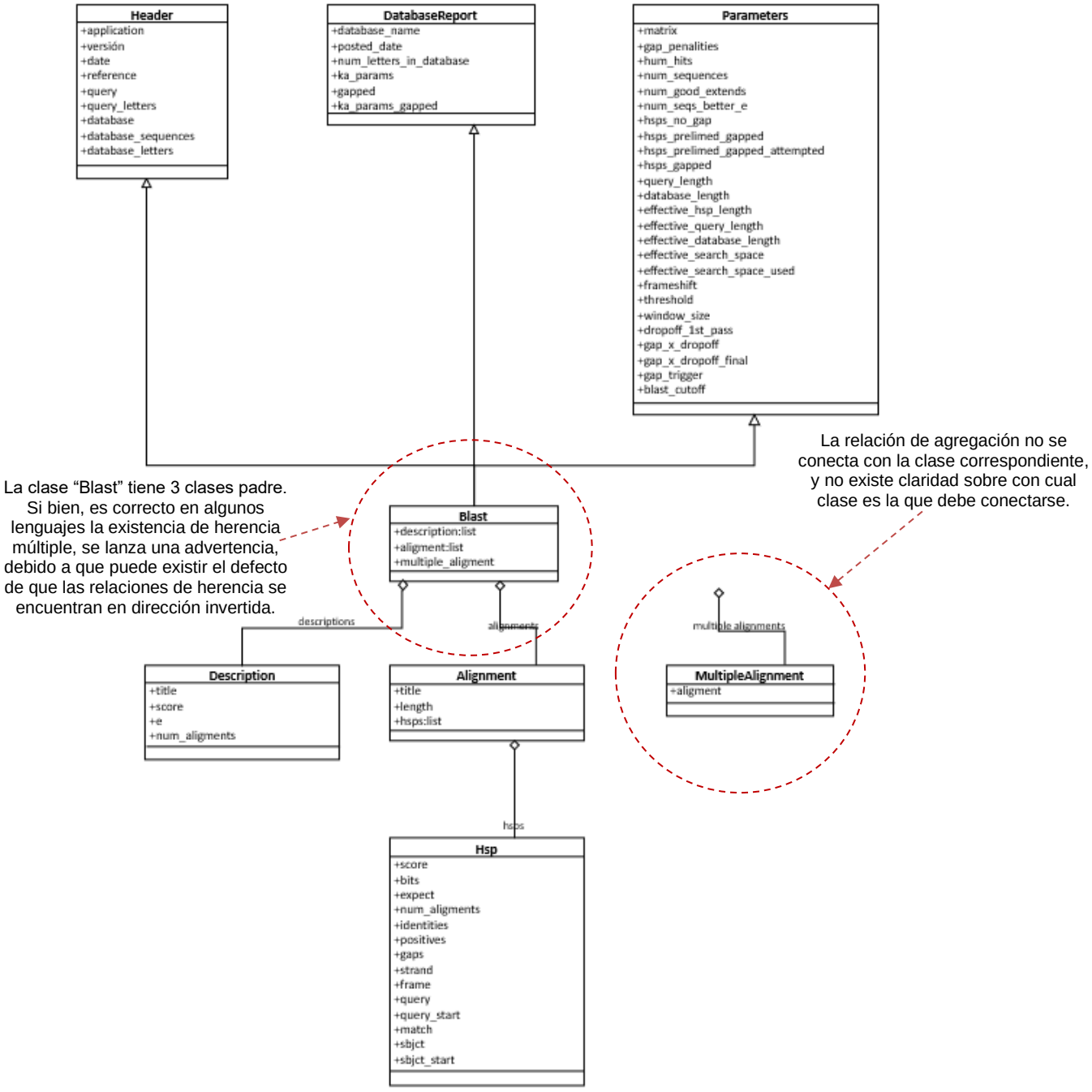


Figura 14. Segmento a Analizar del Diagrama de Clases “Blast”

Al hacer la evaluación del diagrama, la herramienta de software detecta los defectos mencionados, presentando en el reporte de usuario lo siguiente:

1. Se detecta que la clase "MultipleAlignment" no tiene una relación completa hacia al menos otra clase del diagrama.

Respuesta de la herramienta de software:

Superclass Id: [55]; name: [MultipleAlignment]:
Relationship Type: CLASS
List [CLASS] List:
Member Rawtext [] Type [CLASS]
Including Class List:
Detail: Isolated class (no relationships detected)
Rule: [4] [Cantidad de clases sin relación con otras clases en el diagrama];
Recommendation: [4] [Class must have relationship with another diagram class];

2. Se detecta que existen varias jerarquías de herencia y no solo una por lo que se plasma en el informe para el usuario. Asimismo, se detecta que una clase tiene múltiples superclases; lo cual, también es presentado en el reporte de usuario.

Respuesta de la herramienta de software:

Diagram Class Inheritance Hierarchy Qnty: 3
Diagram Class Depth of Inheritance Tree: 2
Isolated class qnty: 1
Diagram class whith multi-inheritance, Qnty: 1
Superclass Id: [52]; name: [Blast]:
Relationship Type: INHERITANCE
List [INHERITANCE] List:
Member Rawtext [] Type []
Including Class List: Class Id [45] Name [DatabaseReport]; Class Id [68] Name [Header]; Class Id [71] Name [Parameters];
Detail: Multi-inheritance found
Rule: [54] [Cantidad de clases que tienen más de una clase padre];
Recommendation: [54] [Multi-inheritance detected, or relationship error];

4.6. Discusión

La herramienta de software desarrollada es un instrumento importante para la aplicación del modelo de calidad creado y explicado en el capítulo 3. Esta herramienta tiene como función evaluar los RA descargados y proporcionar al usuario una referencia respecto la calidad de los mismos.

Es importante mencionar que la herramienta puede analizar los RA diseñados en formato PPTX. Se asume por default que cada RA contiene una sola diapositiva; sin embargo, si un RA contiene más de una diapositiva, todas serán procesadas sin problema, cada diapositiva en este caso es considerada un RA por separado.

Ventajas de la herramienta de software creada:

- La herramienta de software realiza su análisis del contenido de los RA mediante los metadatos de los archivos PPTX, por lo que su funcionamiento es independiente de los formatos proporcionados por herramientas de diseño comerciales que existen en el mercado.
- La herramienta de software al estar diseñada para agregar nuevas reglas de dominio tiene capacidad de crecimiento sin tener que realizar cambios significativos en su código original al hacer uso de un conjunto de servicios web independientes a la herramienta.
- El modelo de calidad creado también tiene capacidad de crecimiento y modificación, y al aplicarse los cambios en la herramienta las modificaciones en su código fuente son mínimas o nulas, ya que dichos cambios se realizarán principalmente en los registros de configuración contenidos en la base de datos.
- Es importante realizar evaluaciones de los RA de manera rápida para tener un punto de referencia de los recursos que se han seleccionado para fines de aprendizaje, por lo que la implementación del modelo de calidad en un software facilita dicha acción.

"La ciencia más útil es aquella cuyo fruto es el más comunicable"

Leonardo Da Vinci
(1452– 1519)

Capítulo 5 Resultados

CAPÍTULO 5. RESULTADOS

En este capítulo se describen los resultados obtenidos de este trabajo de investigación, así como los resultados de las pruebas realizadas a 115 RA en formato PPTX y que contienen DC, los cuales fueron obtenidos de distintas fuentes en Internet con el fin de realizar una revisión tanto de las reglas de dominio que fueron diseñadas como del modelo de calidad implementado.

5.1. Implementación de las Reglas de Dominio

Como parte de los resultados de este proyecto se implementaron una serie de Reglas de Dominio previamente definidas. Las reglas se dividieron en varios tipos que las identifican respecto a su uso en el dominio, estos tipos son: Medición, Abstracción, Correctitud, Estética, Interfaz, Herencia, Asociación y Legibilidad. Se identifican por la forma de evaluarlas, Medición significa que la regla se aplica y se puede medir de manera directa. En las otras se requiere identificar primero como se cumple con un atributo del dominio y luego se cuantifica, por lo que no se miden directamente. La Tabla 20 muestra la implementación de las reglas de dominio en la herramienta de software.

Tabla 20. Reglas de Dominio

No.	Regla de Dominio	Tipo
1	**Cantidad de clases** A mayor cantidad de clases es mayor la cantidad de relaciones y la complejidad del DC.	Medición
2	**Cantidad de interfaces** Las operaciones contenidas en interfaces deben ser implementadas en una relación de realización por lo que las clases no contienen operaciones puede generar confusión en su implementación.	Medición
3	**Cantidad de clases Estáticas** Una clase estática no puede ser instanciada por lo que representar una en el DC UML incrementa su complejidad y puede dificultar su comprensión. La clase estática se identifica con el nombre subrayado [112].	Medición
4	**Cantidad de clases sin relación con otras clases en el diagrama** Las clases que no presentan ninguna relación con otras clases en el diagrama pueden ser confusas y dificultar la comprensión del DC.	Medición
5	**Cantidad de clases abstractas** Las clases abstractas requieren ser generalizadas ya sea en otra clase abstracta o en una clase concreta, a mayor cantidad de clases abstractas mayor herencia y complejidad en el DC [110].	Medición
6	**Cantidad de clases abstractas que no son heredadas sobre una subclase**	Correctitud en dominio

	Las clases abstractas no pueden ser instanciadas por lo que requieren ser heredadas sobre otra clase abstracta o una clase concreta para llegar a ser utilizadas [110]. Mejor valor: 0	
7	**Cantidad de Jerarquías de generalización** A mayor cantidad de jerarquías de generalización mayor herencia y mayor complejidad en el DC. Mejor valor: <=2	Correctitud en dominio
8	**Profundidad del árbol de herencia** A mayor profundidad en el árbol de herencia mayor cantidad de clases, atributos y operaciones que son heredados, lo cual, aumenta la complejidad del DC. Mejor valor: <=4	Correctitud en dominio
9	**Cantidad de clases que contienen operaciones abstractas pero la clase no se definió como abstracta** Cuando una clase contiene operaciones abstractas, dicha clase se considera como abstracta también poniendo letra en tipo itálica el nombre de la clase ya que la clase no puede ser instanciada [110]. Mejor valor: 0	Correctitud en dominio
10	**Cantidad de clases concretas con relación distinta a herencia con una abstracción y que contienen operaciones o atributos de la abstracción** Las subclases heredan íntegramente atributos y operaciones, por lo que a menos que se trate de operaciones abstractas no deben ser repetidas en la subclase [110]. Mejor valor: 0	Correctitud en dominio
11	**Cantidad de clases abstractas que son subclases una clase concreta** Una clase concreta es una clase hoja por lo que no debería ser clase padre de una clase abstracta ya que al invertir el funcionamiento natural de las abstracciones puede generarse confusión. Las clases hoja no pueden ser extendidas [110]. Mejor valor: 0	Correctitud en dominio
12	**Cantidad de asociaciones** A mayor cantidad de relaciones de asociación mayor complejidad en el DC.	Medición
13	**Cantidad de asociaciones directas** A mayor cantidad de relaciones de asociación directas mayor complejidad en el DC.	Medición
14	**Cantidad de agregaciones** A mayor cantidad de relaciones de agregación mayor complejidad en el DC.	Medición
15	**Cantidad de dependencias** A mayor cantidad de relaciones de dependencia mayor complejidad en el DC.	Medición
16	**Cantidad de realizaciones** A mayor cantidad de relaciones de realización mayor complejidad en el DC.	Medición
17	**Cantidad de Generalizaciones**	Medición

	A mayor cantidad de relaciones de Generalización mayor es mayor la profundidad del árbol de herencia, y posiblemente mayor la cantidad de jerarquía de generalización. Implica mayor complejidad en el DC.	
18	**Cantidad de relaciones incompletas** La presencia de relaciones que no tienen conexión con una clase en uno de sus extremos o en ambos puede provocar mayor dificultar para comprender el DC. Mejor valor: 0	Correctitud en dominio
19	**Cantidad de interfaces con relación distinta a realización** Si en la clase concreta o abstracta se implementan las operaciones de la interfaz entonces debería ser una relación de realización [111]. Mejor valor: 0	Correctitud en dominio
20	**Cantidad de atributos en el diagrama** A mayor cantidad de atributos mayor complejidad en el DC.	Medición
21	**Cantidad de operaciones en el diagrama** A mayor cantidad de operaciones mayor complejidad en el DC.	Medición
22	**Cantidad de atributos y operaciones estáticos** La comprensión de los atributos y operaciones estáticos requiere de mayor conocimiento del paradigma orientado a objetos, por lo que su uso implica tener estudiantes más avanzados en dicho paradigma. Los atributos y operaciones estáticos se identifican con el nombre subrayado [112].	Medición
23	**Cantidad de clases cuyos nombres no están en letra bold** El tipo de letra para los nombres de clase debe ser *bold* [111]. Mejor valor: 0	Correctitud en legibilidad
24	**Cantidad de atributos u operaciones cuyos nombres están con letra *bold*** Los nombres de atributos y operaciones son en texto plano [112]. Mejor valor: 0	Correctitud en legibilidad
25	**Cantidad de operaciones no implementadas entre clase e interfaz** Aplica cuando la relación entre ambas es de realización [111]. Las relaciones distintas a realización con una interfaz no son recomendadas, esto, debido a que las interfaces no pueden ser instanciadas [111]. Mejor valor: 0	Correctitud en dominio
26	**Cantidad de Operaciones que no se declararon públicas en interfaces** Todas las operaciones de una interfaz deben ser públicas [112]. Mejor valor: 0	Correctitud en dominio
27	**Cantidad de Atributos contenidos en interfaces** Las interfaces no deberían contener atributos [111]. Mejor valor: 0	Correctitud en dominio
28	**Cantidad de operaciones repetidos entre Subclase y Superclase**	Correctitud en dominio

	Los miembros de la superclase que no son abstractos son heredados sin cambio, por lo que se considerarían repetidos. Los miembros abstractos de la superclase deben ser implementados por la clase hija, por lo que no se consideran repetidos [117]. Mejor valor: 0	
	Cantidad de atributos repetidos entre Subclase y Superclase	Correctitud en dominio
29	Los atributos son heredados sin cambio por lo que no deberían repetirse [117]. Mejor valor: 0	
	Cantidad de operaciones repetidas entre clases asociadas con una misma clase (clases hermanas)	Correctitud en dominio
30	Cuando varias clases que son socias de una misma clase, pero no son socias entre ellas mismas contienen operaciones en común, es posible que se requiera una abstracción para solucionar la repetición de estos miembros, o quizás, la relación de asociación podría ser más bien de herencia. Mejor valor: 0	
	Cantidad de atributos repetidos entre clases asociadas con una misma clase (clases hermanas)	Correctitud en dominio
31	Cuando varias clases que son socias de una misma clase, pero no son socias entre ellas mismas contienen atributos en común, es posible que se requiera una abstracción para solucionar la repetición de estos miembros, o quizás, la relación de asociación podría ser más bien de herencia. Mejor valor: 0	
	Cantidad de párrafos de texto con disposición distinta a horizontal	Correctitud en legibilidad
32	La forma natural del texto en las clases es horizontal por lo que la disposición distinta puede entorpecer su lectura y comprensión. La recomendación de [112], es que el texto sea plano [118]. Mejor valor: 0	
33	**Cantidad de nombres de clase o estereotipo con alineación distinta a centro** Los nombres de clase y estereotipos deben tener alineación al centro [112]. Mejor valor: 0	Correctitud en legibilidad
	Cantidad de atributos con alineación distinta a izquierda	Correctitud en legibilidad
34	Los atributos se deben alinear a la izquierda por regla [112]. Mejor valor: 0	
	Cantidad de operaciones con alineación distinta a izquierda	Correctitud en legibilidad
35	Las operaciones se deben alinear a la izquierda por regla [112]. Mejor valor: 0	
	Cantidad de párrafos de texto con tamaño de letra pequeño	Correctitud en legibilidad
36	La recomendación de [112], es que el texto sea plano. Los atributos y operaciones con letra muy pequeña suelen ser confusos y difíciles de interpretar, en este caso se apega a la regla del texto plano mencionada con anterioridad. Mejor valor: 0	

37	**Cantidad de párrafos de texto con tamaño de letra grande** La recomendación de [112], es que el texto sea plano, por lo que los tamaños grandes de letra no son recomendables y pueden afectar la distribución del DC, en este caso se apega a la regla del texto plano mencionada con anterioridad. Mejor valor: 0	Correctitud en legibilidad
38	**Cantidad de líneas de texto con color de letra distinto al recomendado** Los nombres de atributos y operaciones son en texto plano [112]. Bajo la definición de texto plano proporcionada por OMG el texto plano es en color negro. Mejor Valor: 0	Correctitud en legibilidad
39	**Cantidad de clases con color de fondo distinto al recomendado** Las clases y objetos del diagrama son en formato plano [112]. Bajo la definición de texto plano proporcionada por OMG el fondo plano es en color blanco. Mejor Valor: 0	Correctitud en legibilidad
40	**Cantidad de figuras con color de borde distinto al recomendado** Las clases y objetos del diagrama son en formato plano [112]. Bajo la definición proporcionada por OMG las líneas se deben representar en color negro. Mejor Valor: 0	Correctitud en legibilidad
41	**Cantidad de figuras con bordes con grosor distinto al recomendado** Las figuras con borde muy delgado o muy grueso pueden ser difícil de interpretar. El grosor del borde debe ser configurado en la herramienta de software. Mejor valor: 1	Correctitud en legibilidad
42	**Cantidad de conexiones línea punteada pero distintas a DASH** La forma recomendada para línea punteada es DASH [112]. Debiendo omitir el uso de otros formatos de línea punteada que incluyen puntos o círculos, por ejemplo. Mejor valor: 0	Correctitud en legibilidad
43	**Cantidad de cruces entre líneas que representan relaciones** Los cruces entre líneas de relaciones en el diagrama pueden ser confusos y dificultar la navegación y comprensión del mismo [48], [77], [84], [86], [111]. Mejor valor: 0	Correctitud en dominio
44	**Cantidad de figuras que representan relaciones que tienen más de dos dobleces (excepto reflexivas)** Las relaciones que se forman por varias líneas y que siguen un camino largo entre una clase y otra suelen ser confusas, si contienen más de dos arcos, esto puede incrementar su complejidad. Las relaciones reflexivas son una	Correctitud en dominio

	excepción ya que apuntan a una misma clase por lo que se requiere por lo general de 2 a 3 arcos. Mejor valor: 0	
	Cantidad de relaciones reflexivas	Medición
45	Las asociaciones reflexivas requieren mayor conocimiento del paradigma orientado a objetos por lo que se incrementa la complejidad del DC.	
	Cantidad de operaciones abstractas no implementadas en el diagrama	Medición
46	Las operaciones abstractas deben ser implementadas en una subclase. A mayor cantidad de operaciones abstractas mayor complejidad en el DC. Mejor valor: 0	
	Cantidad de operaciones repetidas entre clases no relacionadas	Correctitud en dominio
47	Cuando varias clases que no tienen ninguna relación entre ellas mismas contienen operaciones en común, es posible que se requiera una abstracción para solucionar la repetición de estos miembros. Mejor valor: 0	
	Cantidad de atributos repetidos entre clases no relacionadas	Correctitud en dominio
48	Cuando varias clases que no tienen ninguna relación entre ellas mismas contienen atributos en común, es posible que se requiera una abstracción para solucionar la repetición de estos miembros. Mejor valor: 0	
	Cantidad de operaciones repetidas entre clases coasociadas	Correctitud en dominio
49	Dos clases asociadas con una misma tercera clase repiten operaciones y dichas operaciones pueden o no encontrarse en la tercera clase. Mejor valor: 0	
	Cantidad de atributos repetidos entre clases coasociadas	Correctitud en dominio
50	Dos clases asociadas con una misma tercera clase repiten atributos y dichos atributos pueden o no encontrarse en la tercera clase. Mejor valor: 0	
	Cantidad de operaciones repetidas entre clases asociadas	Correctitud en dominio
51	Dos clases que tienen conexión por medio de una relación de asociación repiten operaciones. Mejor valor: 0	
	Cantidad de atributos repetidos entre clases asociadas	Correctitud en dominio
52	Dos clases que tienen conexión por medio de una relación de asociación repiten atributos. Mejor valor: 0	
	Cantidad de conexiones que tienen las mismas clases en sus extremos	Correctitud en dominio
53	Dos clases no deberían tener más de una relación del mismo tipo entre ellas. Mejor valor: 1	
54	**Cantidad de clases con herencia múltiple**	Correctitud en dominio

	Las clases no deberían tener múltiples padres, ya que esto puede alterar la profundidad del árbol de herencia. Se lanza alerta ya que se trata de una característica que es configurable, debido a que diversos lenguajes permiten la herencia múltiple.	
	Cantidad de operaciones repetidas entre 2 o más clases que se ubican en distinto árbol de herencia	Correctitud en dominio
55	Una operación que se repite entre elementos de dos o más arboles de herencia distintos puede significar la necesidad de crear una interfaz. Mejor valor: 0	
	Cuando es creada una nueva abstracción, y sus subclases tienen un mismo tipo de relación con otra clase o interfaz	Correctitud en dominio
56	Se deben eliminar todas las relaciones repetidas de las subclases con una misma clase/interfaz y crear una sola relación entre la nueva abstracción y la clase relacionada. Mejor Valor: 0	

Como resultado de las pruebas realizadas con una serie de 115 DC obtenidos desde distintas fuentes de Internet, se consiguió información relevante respecto a la calidad de los mismos.

5.1.1. Resultados Agrupados Respecto a los Recursos de Aprendizaje que contienen DC

Los resultados se agruparon en varios rubros, siendo el primero de ellos el que contempla los resultados sobre los 115 DC analizados.

En la Tabla 21 se pueden observar los resultados generales resultantes del proceso respecto a la aplicación de las reglas de dominio, siendo la presencia de colores fondo y líneas con 51 y 38 recursos presentándolas respectivamente; la difuminación de imágenes con 23 recursos afectados; y clases aisladas con 11 recursos afectados; los defectos con mayor incidencia en los diagramas de clases.

Tabla 21. Tabla de Frecuencias

Atributo de Calidad	Frecuencia
LRs with blurring.	23
LRs with background colors.	51
LRs with colored lines.	38
LRs with isolated classes.	11
LRs with figures out of frame.	7
LRs with incomplete relationships.	9
LRs with crossing lines.	40
LRs with relationships with more than two corners.	9
LRs with abstract classes not inherited.	6
LRs with operations not implemented from abstraction.	2

En la Figura 15 se muestra la gráfica de defectos respecto a las pruebas realizadas sobre 115 DC seleccionados desde diversas fuentes de Internet.

Los resultados de las pruebas se dividieron en grupos: primero, aquellos procedentes de la Correctitud en Dominio; y, segundo: aquellos procedentes de la Correctitud en Legibilidad. Asimismo, también hay grupos que hacen referencia a los elementos que conforman las clases y otros objetos del propio DC.

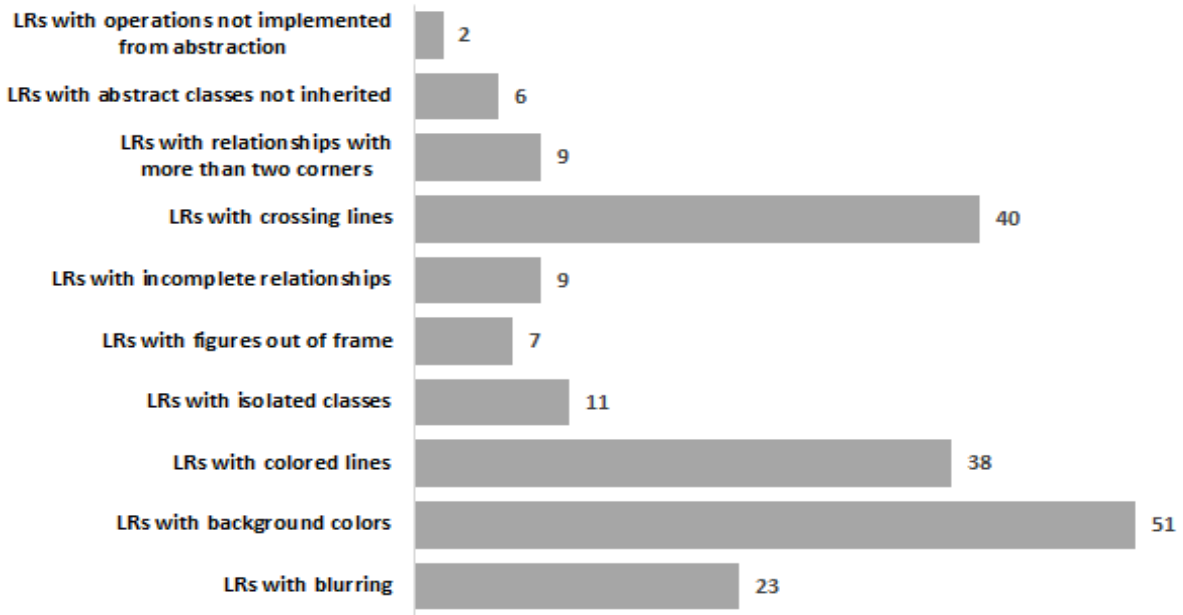


Figura 15. Gráfica: Porcentajes de Defectos Respecto al DC

5.1.2. Resultados en Legibilidad de los DC

En la Tabla 22 se muestran los porcentajes y cantidades encontrados respecto a la Legibilidad de los 115 DC utilizados en pruebas. Los defectos que tienen mayor presencia son: difuminación del DC con 23 ocurrencias; uso de colores de fondo en diagrama y clases con 51 ocurrencias, siendo este uno de los rubros con mayores incidencias; mientras que la cantidad de textos con un tamaño más grande al recomendado es solo una, siendo este rubro el que tiene menos incidencias.

Tabla 22. Tabla de Porcentajes de Legibilidad en DC

Atributo de Calidad	Frecuencia
LRs with blurring.	23
LRs with background colors.	51
LRs with colored lines.	38
LRs with color text.	14
LRs with text in big size.	1
LRs with wide width lines.	65

En la Figura 16 se muestra la gráfica correspondiente a los porcentajes de Correctitud en Legibilidad encontrados en los 115 DC analizados, siendo los colores de relleno en figuras y los cruces de líneas que representan relaciones los defectos que mayores porcentajes presentan.

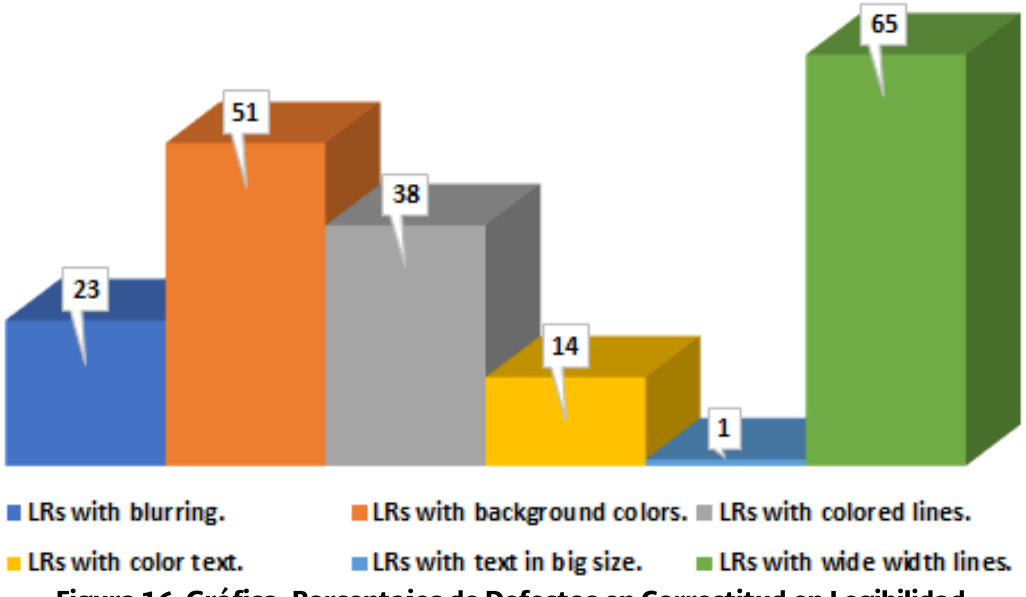


Figura 16. Gráfica: Porcentajes de Defectos en Correctitud en Legibilidad

5.1.3. Resultados Agrupados Respecto a las Clases Contenidas en los DC de los Recursos de Aprendizaje

En la Tabla 23 es posible ver los porcentajes de defectos encontrados en las clases contenidas en los 115 diagramas analizados. Un total de 924 clases forman parte de dichos diagramas. De las 924 clases contenidas en los DC un total de 575 presentaron colores de borde o línea, representando un 62.2% del total analizado, siendo este el defecto con mayor incidencia en respecto a esta tabla.

Tabla 23. Tabla de Porcentajes Dominio respecto a DC en sus Clases

Atributos de Calidad en Clases	Cantidad
Clases abstractas no heredadas.	7
Nombres de clase no centrado.	25
Nombres de clase sin letra Bold.	353
Clases con más de 1 conexión con otra clase.	33
Clases con color de relleno.	416
Clases con líneas con color.	575
Clases aisladas.	17

En la Figura 17 se muestra la gráfica correspondiente a los porcentajes de defectos encontrados respecto a las 924 clases que forman parte de los 115 DC analizados. En esta, es posible observar que la presencia de colores distintos al negro en relleno y bordes del DC son los defectos con mayor incidencia en los resultados de las pruebas realizadas.

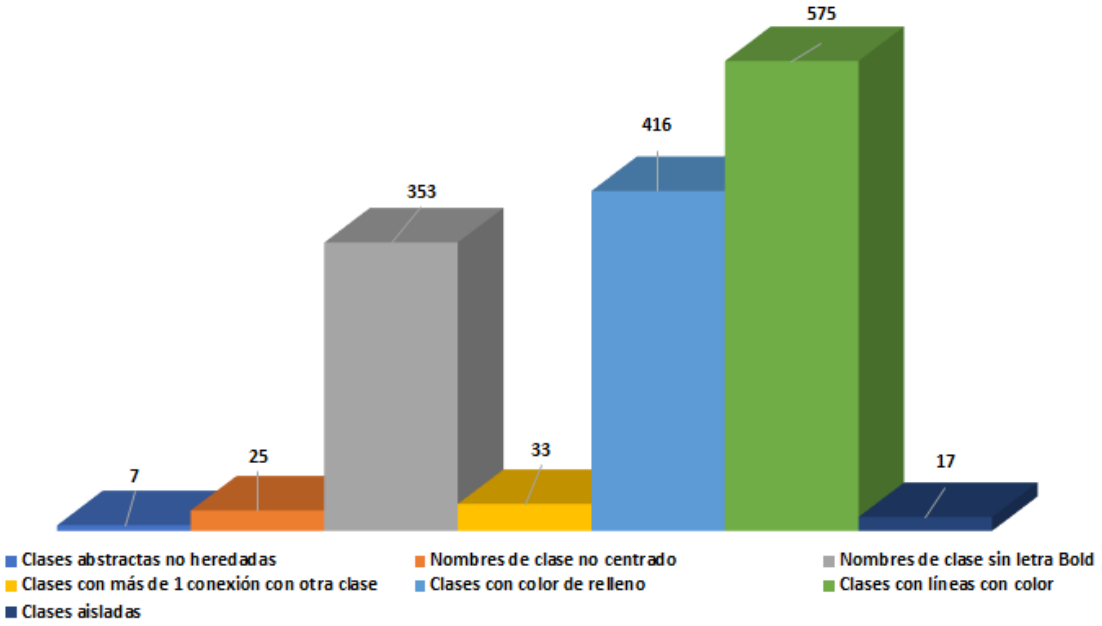


Figura 17. Gráfica: Porcentajes Dominio en DC en Clases

5.1.4. Resultados Agrupados Respecto los Atributos Contenidos en Clases de los DC

La Tabla 24 muestra la lista de rubros analizados para los atributos de clase. Con un total de 2,282 atributos distribuidos en 924 clases de 115 DC a los que se les realizaron pruebas. El defecto más presente en este rubro es la redundancia de atributos.

Tabla 24. Tabla de Porcentajes de Dominio en DC en Atributos

Atributos de Calidad en Atributos Contenidos en Clases	Cantidad
Atributos Repetido Clases Asociadas.	163
Atributos Repetido Clases Coasociadas.	119
Atributos Repetido Clases no relacionadas.	231
Atributos no alineados a la Izquierda.	83
Atributos En Interfaces.	13
Nombres de Atributos con Letra Bold.	56

En la Figura 18 se muestra la gráfica correspondiente a los porcentajes de Correctitud en de Atributos de Clase encontrados en los 115 DC analizados, distribuidos en 924 clases en total.

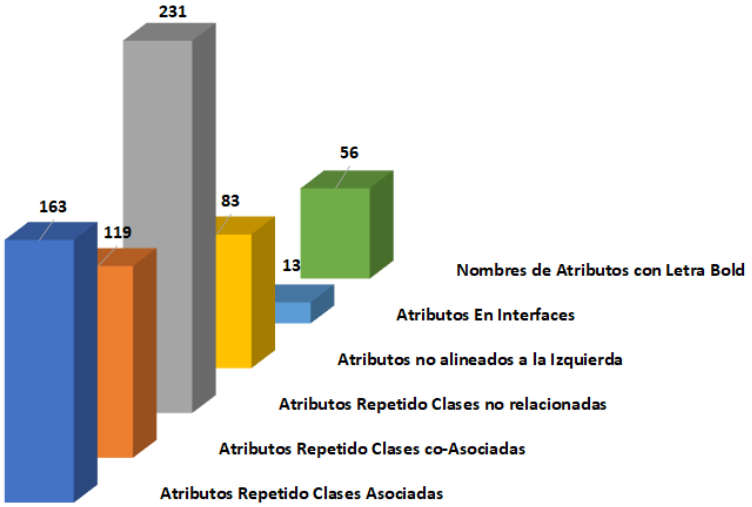


Figura 18. Gráfica: Porcentajes de Defectos en Correctitud de Atributos

5.1.5. Resultados Agrupados Respecto las Operaciones Contenidas en Clases de los DC

La Tabla 25 muestra los porcentajes de Dominio de DC en Operaciones de Clases. Con un total de 1,805 operaciones, distribuidas entre 924 clases, de un total de 115 diagramas estudiados. Los defectos que mayor incidencia presentan son los relacionados con la repetición de operaciones entre clases que tienen o no una relación de asociación. Asimismo, existe repetición entre clases que pertenecen a distintos árboles de herencia y no presentan ningún tipo de relación entre ellas.

Tabla 25. Tabla de Porcentajes de Dominio en DC en Operaciones

Atributos de Calidad en Operaciones Contenidas en Clases	Cantidad
Operaciones Repetidas entre Clases Asociadas.	168
Operaciones Repetidas entre Clases Coasociadas.	61
Operaciones Repetidas entre Clases no Relacionadas.	48
Operaciones no Implementadas desde Abstracción.	4
Operaciones no Alineadas a la Izquierda.	65
Nombres de Operaciones con Letra Bold.	20
Operaciones no Públicas en Interfaz.	16
Operaciones no Implementadas desde Interfaz.	56
Operaciones Repetidas entre Árboles de Herencia.	9

La Figura 19 muestra la Gráfica de Porcentajes de Defectos en Correctitud Operaciones de Clase, contenidas en las 924 clases analizadas.

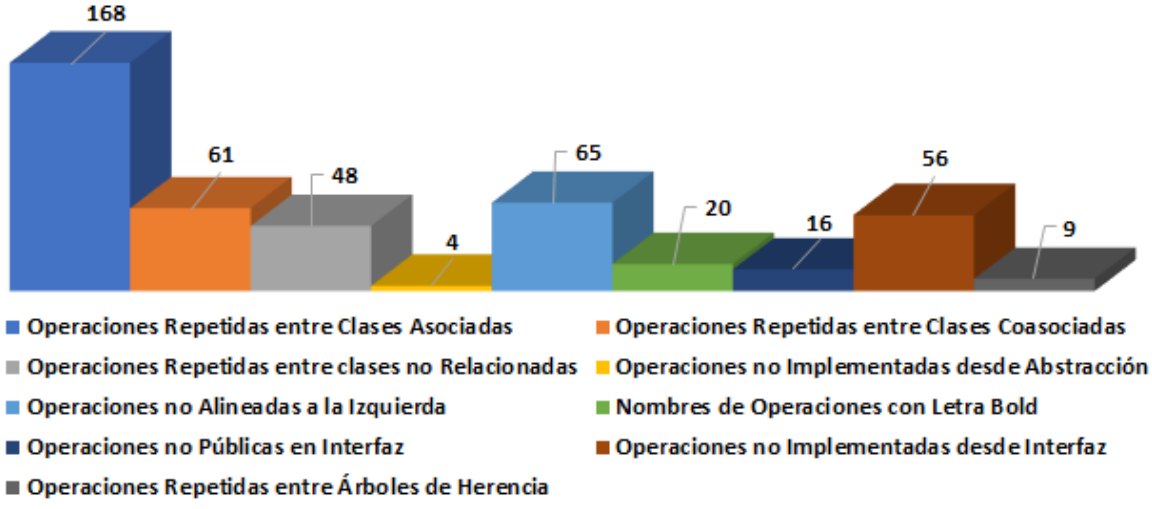


Figura 19. Gráfica: Porcentajes de Defectos en Correctitud Operaciones

5.1.6. Resultados Agrupados Respecto las Relaciones Contenidas en los DC

La Tabla 26 muestra el Porcentajes de Dominio en DC en las Relaciones entre clases en los DC. En esta tabla es posible apreciar que el defecto que tiene mayor cantidad de apariciones es el uso de colores de borde de las relaciones con 298 casos. Asimismo, el cruce de líneas que representan relaciones tiene un 20% de los 713 conectores presentes en los 115 DC analizados.

Tabla 26. Tabla de Porcentajes de Dominio en DC en Operaciones

Dominio en DC en Correctitud de Relaciones	Cantidad
Relaciones no identificadas.	2
Relaciones con más de 2 arcos o esquinas.	19
Relaciones con cruces de línea.	143
Relaciones Incompletas.	19
Relaciones con colores en bordes.	298

La Figura 20 muestra la Gráfica de Porcentajes de Defectos en Relaciones del DC. En esta, es posible ver los porcentajes en relación con el total de 1004 de relaciones contenidas en los 115 DC utilizados en pruebas.

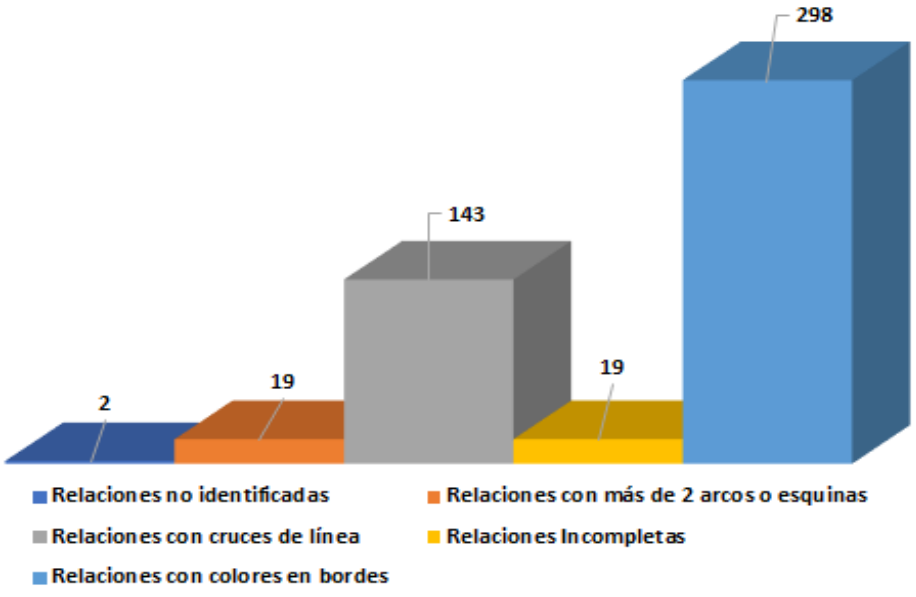


Figura 20. Gráfica: Porcentajes de Defectos en Relaciones del DC

5.2. Discusión

En este capítulo se muestran las funcionalidades de la herramienta desarrollada, así como la aplicación el modelo de calidad y reglas de dominio creadas. Con estos resultados se muestra la factibilidad de dichos instrumentos como un medio de evaluación de los RA descargados de Internet para propósitos de aprendizaje de IS.

En los diversos rubros analizados es posible ver que los defectos relacionados con la estética del diagrama tienen gran cantidad de incidencias, como lo son: difuminación, uso de colores en relleno y borde, y uso de tipos de letra muy pequeños. Asimismo, aquellos defectos relacionados con la abstracción tienen un alto porcentaje de incidencia, como lo es la repetición de atributos y operaciones entre clases relacionadas o no relacionadas.

Algunos problemas que se presentan al momento de evaluar los RA son: el gran tamaño que algunos de estos recursos tienen, lo que provoca mayor tiempo de análisis; gran cantidad de recursos no cuentan con una descripción clara y completa sobre su funcionalidad, lo que deriva en interpretaciones erróneas, y; muchos RA se hicieron con herramientas de diseño distintas, por lo que sus formatos no están estandarizados, lo que requiere ajustes en la herramienta de software creada con el fin de interpretar atributos muy particulares presentes en los RA.

Por lo tanto, podemos decir que, mediante las reglas de dominio diseñadas, el modelo de calidad creado y la herramienta de software desarrollada podemos:

- Tener una evaluación de los RA apegada a reglas de dominio.
- Contar con un modelo de calidad que soluciona un problema de la IS, que es, la evaluación de los RA que se utilizan masivamente con propósitos de aprendizaje.
- Disponibilidad de un proceso probado para la evaluación de los RA del dominio.

"Me parece haber sido sólo un niño
jugando en la orilla del mar,
divirtiéndose y buscando una piedra más
lisa o una concha más bonita de lo normal,
mientras el gran océano de la verdad
yacía ante mis ojos con todo por descubrir"

Isaac Newton
(1642 – 1727)

Capítulo 6 Conclusiones

CAPÍTULO 6. CONCLUSIONES

En este capítulo se describen los hallazgos más relevantes durante la investigación realizada y la integración de los productos académicos alcanzados en esta tesis. También se describen las aportaciones que este trabajo de investigación tiene. Finalmente, se describen los trabajos futuros recomendados para dar continuidad al trabajo realizado.

6.1. Conclusiones

Un aspecto relevante de este trabajo de investigación es la solución aportada a la problemática sobre la selección y búsqueda de RA de IS con fines de aprendizaje de diagramas de clases. La cantidad masiva de RA disponibles en Internet, sumado a la poca información disponible acerca de su calidad generan la necesidad de contar con un marco de referencia basado en reglas de dominio y un modelo de calidad.

Los objetivos plasmados en este trabajo de investigación se cumplieron en su totalidad y que se detallan a continuación:

- Respecto al objetivo general: se estableció un mecanismo basado en reglas de dominio, este objetivo se cumple con el diseño de tales reglas, así como de un modelo de calidad que se probó. Las reglas de dominio y el modelo de calidad son implementados mediante una herramienta de software que se diseñó con el propósito de realizar un análisis automatizado, proporcionando los valores de calidad de los RA de forma independiente o agrupada.
- Respecto al objetivo específico 1: se definieron un total de 56 reglas, enfocadas en medir la calidad de atributos que se detectaron durante la investigación realizada.
- Respecto al objetivo específico 2: se identificaron una serie de defectos contenidos en los RA y que se consideraron para evaluarse mediante las reglas de dominio.
- Respecto al objetivo específico 3: se logró al momento de realizar una evaluación, estos no necesariamente son considerados un defecto; sin embargo, al momento de agrupar resultados de las mediciones realizadas pudo identificarse que un defecto puede llevar a otro; de igual forma, el valor de un atributo puede influir en el valor o interpretación de otros.
- Además, los defectos encontrados en los RA son diversos, siendo algunos de los más comunes la falta de abstracción y la falta de legibilidad. Asimismo, encontraron defectos como: la existencia de relaciones no reconocidas por la notación UML; errores en el tipo o en la dirección de las relaciones; no implementación de

operaciones y clases abstractas; no implementación de operaciones en interfaces; uso de colores no recomendados por la notación UML; problemas de legibilidad del diagrama, tanto de imágenes excesivamente difuminadas, como uso de tipos de letra de tamaño muy pequeño; entre otros.

- Se encontró también, que los defectos contenidos en los DC con mayor incidencia tienen relación con la estética de los mismos, siendo la presencia de colores en fondo, letra y bordes y letra, lo cual afecta la legibilidad del DC. De igual forma, se identificó que, algunas veces un tipo de defecto puede llevar a otro tipo de defecto. Por ejemplo, un problema de aplicar erróneamente el concepto de abstracción puede llevar puede llevar a un problema de comprensibilidad, y así tener una secuencia de defectos.
- Por otro lado, en muchos de los casos analizados fue necesario realizar una interpretación minuciosa del diagrama de clases, esto, debido a que muchos RA no contenían anotaciones o descripciones asociadas que facilitaran su comprensión, o explicaran a grandes rasgos los requerimientos que se atendieron con el diseño de dichos diagramas de clases.
- Por otra parte, los RA analizados fueron desarrollados en distintas plataformas, lo que implica que algunos de estos RA contengan características particulares que solo se pueden utilizar o aplicar en dichas plataformas. Por lo que resulta importante que se busque la estandarización de los diseños mediante plataformas. Ya que la interpretación de puntos finos en los RA puede tener un costo de análisis y tiempo de ejecución muy alto.
- También, la herramienta de software creada para implementar el modelo de calidad tiene la ventaja de que puede crecer y ser mejorado, nuevas reglas de dominio pueden ser agregadas al modelo sin que exista la necesidad de modificar la herramienta de software ya que está preparada para ello.
- Finalmente, el modelo de calidad que se creó, las reglas de dominio diseñadas y la herramienta de software desarrollada para implementar la solución proporcionan un conjunto de herramientas que se probaron ampliamente, con las cuales, tenemos un punto de referencia respecto a la calidad de los recursos que contienen diagramas de clases y que son utilizados con propósitos de aprendizaje en IS, particularmente del DC.

6.2. Aportaciones

Este trabajo tiene varias aportaciones relevantes, que además pueden servir como punto de partida para futuras investigaciones, estas son:

- Un mecanismo de evaluación basado en un conjunto de reglas de dominio que han sido implementadas y fueron ampliamente probadas.
- Un modelo de calidad en el cual se encuentran implementadas las reglas de dominio que fueron diseñadas.
- Una herramienta de software automatizada, que puede extraer la información de los RA objetivo leyendo sus metadatos, además, esta herramienta tiene la capacidad de crecer y contar con mayor cantidad de reglas de dominio mediante configuraciones en la BD.
- Un conjunto de RA en formato PPTX ya probado y que puede servir para futuras investigaciones.

6.3. Trabajos Futuros

Como resultado del trabajo realizado se detectaron necesidades que requieren ser investigadas y desarrolladas a futuro, tales como:

- Aplicar el modelo de calidad en otros formatos de RA además del formato PPTX ya desarrollado, como lo es el formato PDF.
- Ampliar las posibilidades de análisis de RA incluyendo otras técnicas además de las ya usadas como lo es: 1) Lectura de archivos XMI para el análisis de los diagramas desarrollados mediante herramientas de diseño; y, 2) Inteligencia Artificial (IA) para analizar formatos como imágenes digitales y procesamiento de lenguaje natural para analizar los textos y descripciones que contengan los RA.
- Incluir en la herramienta de software desarrollada escenarios de fracaso y alternativos, esto puede ser mediante el uso de técnicas de IA.
- Incluir también otros tipos de diagramas UML que clarifiquen el desarrollo, como lo es Diagrama de Secuencia, Diagrama de Casos de Uso, y otros.
- Analizar las plataformas de diseño de RA enfocadas en diagramas de clases UML, con el fin de detectar características particulares de cada herramienta, que puedan ser de utilidad y por lo tanto consideradas dentro del análisis realizado para medir su calidad.

- Analizar una gama más amplia de atributos de calidad y por ende crear reglas adicionales a las que ya se presentaron en esta tesis.

6.4. Publicaciones

Para este propósito, se publicó el artículo titulado "*Systematic Review of Quality in Class Diagrams for Software Engineering Competencies*" a la revista "IEEE Revista Iberoamericana de Tecnologías del Aprendizaje (IEEE-RITA)", tratándose de una revista reconocida por el Journal Citation Reports (JCR).

- "Systematic Review of Quality in Class Diagrams for Software Engineering Competencies". (2022). IEEE-Revista Iberoamericana de Tecnologías del Aprendizaje (RITA). Vol. 17, Núm. 4. DOI: 10.1109/RITA.2022.3217169. Disponible en: https://ieeexplore.ieee.org/document/9930752.

6.5. Bibliografía

[1] A. Littlejohn, I. Falconer y L. Mcgill, "Characterising effective eLearning resources", *Computer & Education,* vol. 50, nº 3, pp. 757-771, 2008.

[2] J. Navarro y L. .Ramírez, Objetos de aprendizaje: formación de autores con el modelo redes de objetos, Guadalajara: Sistema de Universidad Virtual, 2005.

[3] P. Delgado, Servicios Web de Aprendizaje, Cuernavaca: Grupo de ingeniería de software, CENIDET, 2012.

[4] E. A. López, Definición de Elementos del WSDL para Servicios Web de Aprendizaje, Cuernavaca: Cenidet, 2013.

[5] P. Espinoza Pérez, Esquema de Clasificación de Servicios Web de Aprendizaje, Cuernavaca: CENIDET, 2016.

[6] H. Salinas, Marco Orientado a Objetos para Medir la Calidad en Servicios Web de Aprendizaje, Cuernavaca: Cenidet, 2017.

[7] B. Valenzuela, Integración de Recursos de Aprendizaje en Moodle con base en el Modelo de Servicios Web, vol. Tesis de Maestría, Cuernavaca: CENIDET, 2017.

[8] J. Münch, O. Armbrust, M. Kowalczyk y M. Soto, Software Process Definition and Management, 1 ed., Springer, 2012, p. 205.

[9] I. Jacobson, G. Booch y J. Rumbaugh, "El proceso unificado de desarrollo de software", p. 438, 2000.

[10] W3C, "W3C", W3C Working Group, 2004. [En línea]. Available: http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/. [Último acceso: Octubre 2018].

[11] IBM, "IBM", IBM Knlowledge Center , 2018. [En línea]. Available: https://www.ibm.com/support/knowledgecenter/en/SSGMCP_5.2.0/com.ibm.cics.ts.webservices.doc/concepts/dfhws_definition.html. [Último acceso: Octubre 2018].

[12] Microsoft, "MSDN", MSDN Library, 2018. [En línea]. Available: https://msdn.microsoft.com/en-us/library/ms950421.aspx. [Último acceso: Octubre 2018].

[13] Madjarov y O. Boucelma, "Data and Application Integration in Learning Content Management Systems: A Web Services Approach", pp. 272-286, 2006.

[14] Klašnja-Milićević, B. Vesin, M. Ivanović, Z. Budimac y L. Jain, "ELearning Systems: Intelligent Techniques for Personalization", *SpringerLink,* vol. 112, pp. 1868-4394, 2017.

[15] GetBridge, "GetBridge", Bridge, 2008. [En línea]. Available: https://www.getbridge.com/lc/articles/difference-e-learning-e-training. [Último acceso: Octubre 2008].

[16] J. Hurwitz, R. Bloor, M. Kaufman y F. Halper, Service Oriented Architecture for Dummies, Indianapolis: Wiley Publishing, Inc, 2009.

[17] N. Cavus y M. Alhih, "Learning Management Systems Use in Science Education", *Conference: 3rd Cyprus International Conference on Educational Research, CY-ICER 2014,* vol. 143, nº 1, pp. 517-520, 2014.

[18] ISO, Software engineering-Software product Quality Requirements and Evaluation (SQuaRE) Quality model, ISO Information Technology Standards, 2008.

[19] D. Ameller, M. Galster, P. Avgeriou y X. Franch, "A survey on quality attributes in service-based systems", *Software Quality Journal,* vol. 24, nº 2, pp. 271-299, 2015.

[20] Rex Black Consulting Services, "Metrics for Software Testing: Managing with Facts: Part 1: The Why and How of Metrics", pp. 1-11, 2010.

[21] Microsoft, "Cree una Regla de Dominio", 2017. [En línea]. Available: https://docs.microsoft.com/es-es/sql/data-quality-services/create-a-domain-rule?view=sql-server-2017. [Último acceso: 2018].

[22] The GUIDE Business Rules Project. Final Report, Defining Business Rules - What Are They Really?, B. R. Group, Ed., 2000, p. 77.

[23] G. Wagner, "Rule Modeling and Markup", *Springer-Verlag,* pp. 251-274, 2005.

[24] J. d. J. Gregorio, Sistema Constructor de Recursos Educativos como Servicios Web de Aprendizaje, Cuernavaca: Cenidet, 2017.

[25] R. ALARCÓN, DISEÑO ORIENTADO A OBJETOS CON UML, Grupo EIDOS, 2000.

[26] B. Kitchenham, "Procedures for Performing Systematic Reviews", *Empirical Software Engineering. Keele University Technical Report and NICTA Technical Report,* 2004.

[27] M. Villavicencio, V. Revelo y J. Pincay, "Towards the Evaluation of Open Educational Resources for Learning Software Engineering", *XLII Latin American Computing Conference,* pp. 1-9, 2016.

[28] S. Rajesh y A. Chandrasekar, "An Efficient Object-Oriented Design Model: By Measuring and Prioritizing the Design Metrics of UML Class Diagram with Preeminent Quality Attributes", *Indian Journal of Science and Technology,* vol. 9, nº 21, pp. 1-9, 2016.

[29] R. Harrison, S. Counsell y R. Nithi, "Coupling Metrics for Object-Oriented Design", *Software Metrics Symposium,* pp. 150-157, 1998.

[30] S. Yacoub, H. Ammar y T. Robinson, "Dynamic Metrics for Object Oriented Designs", *Proceedings Sixth International Software Metrics Symposium (Cat. No.PR00403),* pp. 50-61, 1999.

[31] M. Genero, M. Piattini y L. Jiménez, "Empirical Validation of Class Diagram Complexity Metrics", *International Symposium on Empirical Software Engineering (ISESE'02),* 2001.

[32] M. Genero, M. Piattini, E. Manso y G. Cantone, "Building UML Class Diagram Maintainability Prediction Models Based on Early Metrics", *Ninth International Software Metrics Symposium (METRICS'03),* pp. 1530-1435, 2003.

[33] A. Chatzigeorgiou, "Mathematical Assessment of Object-Oriented Design Quality", *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING,* vol. 29, nº 11, pp. 1050-1053, 2003.

[34] T. Yi, F. Wu y C. Gan, "A Comparison of Metrics for UML Class Diagrams", *Department of Computer Science & Engineering, Southeast University. ACM SIGSOFT Software Engineering Notes,,* vol. 29, nº 5, pp. 1-6, 2004.

[35] M. Genero, M. Piattini-Velthuis, J. Cruz-Lemus y L. Reynoso, "Metrics for UML Models", *UPGRADE, UML and Model Engineering,* 2004.

[36] M. Genero, M. Piattini y C. Calero, "A Survey of Metrics for UML Class Diagrams", *JOURNAL OF OBJECT TECHNOLOGY,* vol. 4, nº 9, pp. 59-92, 2005.

[37] B. Baldassari, C. Robach, L. Bouquet y J. Brosse, "Early metrics for Object Oriented Designs", *Testability Assessment, 2004. IWoTA 2004,* pp. 62-69, 2004.

[38] Y. Tong , "Comparison Research of Two Typical UML-Class-Diagram Metrics: Experimental Software Engineering", *International Conference on Computer Application and System Modeling (ICCASM),* vol. 12, pp. 12-90, 2010.

[39] R. Santaolaya, O. Fragoso-Diaz y S. Zamudio-Lopez, "Modelo formal para la reestructura de marcos orientados a objetos hacia arquitecturas modelo-vista-adaptador", *Ingeniería Investigación y Tecnología,* vol. XV, nº 2, pp. 187-198, 2014.

[40] A. Camilleri, U. Ehlers y J. Pawlowski, "State of the Art Review of Quality Issues related to Open Educational Resources (OER)", *Publications Office of the European Union. JRC Scientific and Policy Reports,* 2014.

[41] M. Genero, M. Manso, E. Piattini y F. Garcia, "Early Metrics for Object Oriented Information Systems", *6th International Conference on Object Oriented Information Systems,* pp. 414-425, 2000.

[42] A. Egyed, "Automated Abstraction of Class Diagram", *Transactions on Software Engineering and Methodology,* vol. 11, nº 4, pp. 449-491, 2002.

[43] D. Rodriguez y R. Harrison, "An Overview of Object-Oriented Design Metrics", *Grant Awarding Body,* 2001.

[44] G. Wagner, "Rule Modeling and Markup", *Springer-Verlag,* pp. 251-274, 2005.

[45] M. Boukhebouze, Y. Amghar, A.-N. Benharkat y Z. Maamar, "A rule-based approach to model and verify flexible business processes", *International Journal of Business Process Integration and Management,* pp. 287-307, 2011.

[46] T. L. Leacock y J. C. Nesbit, "A Framework for Evaluating the Quality of Multimedia Learning Resources.", *Educational Technology & Society,* vol. 10, nº 2, pp. 44-59, 2007.

[47] J. D. Pickering y V. C. T. Joynes, "A holistic model for evaluating the impact of individual technology-enhanced learning resources", *Medical Teacher,* vol. 38, nº 12, pp. 1242-1247, 2016.

[48] H. Eichelberger y J. Wolff, "UML Class Diagrams - State of the Art in Layout Techniques", *Proceedings of the 2nd International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT '03),* pp. 30-34, 2003.

[49] S. Rajesh y A. Chandrasekar, "Metrics measurement model: To measure the object-oriented design metrics", de *Seventh International Conference on Advanced Computing (ICoAC)*, 2015.

[50] A. Halim, "Predict fault-prone classes using the complexity of UML class diagram", *International Conference on Computer, Control, Informatics and Its Applications (IC3INA),* pp. 289-294, 2013.

[51] J. A. Pow-Sang, D. Villanueva, L. Flores y C. Rusu, "A Conversion Model and a Tool to Identify Function Point Logic Files Using UML Analysis Class Diagrams", *Joint Conference of the 23rd International Workshop on Software Measurement and the 8th International Conference on Software Process and Product Measurement,* 2013.

[52] H. Han, "A software modeling method of ocean parameters measurement system", *2016 IEEE/OES China Ocean Acoustics (COA),* pp. 9-11, 2016.

[53] E. Ekanayake y S. Kodituwakku, "Consistency Checking of UML Class and Sequence Diagrams", *International Conference on Ubi-Media Computing (UMEDIA),* pp. 98-103, 2015.

[54] M. Nuthakki, M. Mete, C. Varol y S. Suh, "UXSOM: UML Generated XML to Software Metrics", vol. 36, nº 3, pp. 1-6, 2011.

[55] H. XIAO , S. LI y B. WANG , "A tool for the application of software metrics to UML class diagram", *First International Workshop on Education Technology and Computer Science,* 2009.

[56] M. Girgis, T. Mahmoud y R. Nour, "UML Class Diagram Metrics Tool", *International Conference on Computer Engineering & Systems,* pp. 423-428, 2009.

[57] R. Leibbrandt, D. Yang, D. Pfitzner, D. Powers, P. Mitchell, S. Hayman y H. Eddy, "Smart Collections: Can Artificial Intelligence Tools and Techniques Assist with Discovering, Evaluating and Tagging Digital Learning Resources?", *12th Biennial School Library Association of Queensland, the 39th International Association of School Librarianship Annual Conference,* pp. 1-12, 2010.

[58] J. Braeuer, "Measuring Object-Oriented Design Principles", *30th IEEE/ACM International Conference on Automated Software Engineering,* pp. 882-885, 2015.

[59] V. Yadav y R. Singh, "Predicting Design Quality of Object-Oriented Software using UML diagrams", *3rd IEEE International Advance Computing Conference (IACC),* 2013.

[60] M. Turan, "Integrating Software Metrics with UML Class Diagrams", *Lecture Notes on Software Engineering,* vol. 3, nº 3, pp. 220-224, 2015.

[61] Y. Nakamura, K. Sakamoto, K. Inoue, H. Washizaki y Y. Fukazawa, "Evaluation of Understandability of UML Class Diagrams by Using Word Similarity", *Joint Conference of the 21st International Workshop on Software Measurement and the 6th International Conference on Software Process and Product Measurement,* pp. 178-187, 2011.

[62] A. Dikici y et al, "Factors Influencing the Understandability of Process Models: A Systematic Literature Review", *Information and Software Technology,* vol. 63, pp. 112-129, 2017.

[63] G. Scanniello, C. Gravino, C. Genero, J. Cruz-Lemus y G. Tortora, "On the Impact of UML Analysis Models on Source-Code Comprehensibility and Modifiability", *ACM Transactions on Software Engineering and Methodology (TOSEM),* vol. 23, nº 2, pp. 1-26, 2014.

[64] K. SUGIYAMA, S. TAGAWA y M. TODA, "Methods for Visual Understanding of Hierarchical System Structures", *IEEE Transactions on Systems, Man and Cybernetics,* vol. 11, nº 2, pp. 109-125, 1981.

[65] O. Filipova y O. Nikiforova, "Definition of the Criteria for Layout of the UML Use Case Diagrams", *Applied Computer Systems,* vol. 24, nº 1, pp. 75-81, 2019.

[66] J. Gudenberg, A. Niederle, M. Ebner y H. Eichelberger, "Evolutionary Layout of UML Class Diagrams", *Association for Computing Machinery,* pp. 163-164, 2006.

[67] P. Fillottrani y C. Keet, "An ontology-driven unifying metamodel of UML Class Diagrams, EER, and ORM2", *ata & Knowledge Engineering,* vol. 98, pp. 30-53, 2015.

[68] H. Purchase, L. Colpoys, M. McGill, D. Carrington y C. Britton, "UML class diagram syntax: an empirical study of comprehension", *Conferences in Research and Practice in Information Technology,* vol. 9, pp. 113-120, 2001.

[69] D. Feitelson y D. Tyszberowicz, "UML Diagram Refinement (focusing on class - and use case diagrams)", *IEEE/ACM 39th International Conference on Software Engineering,* pp. 735-745, 2017.

[70] S. Markovi´c y T. Baar, "Refactoring OCL Annotated UML Class Diagrams", *International Conference on Model Driven Engineering Languages and Systems,* pp. 25-47, 2005.

[71] S. Harald, "Diagram Size vs. Layout Flaws: Understanding Quality Factors of UML Diagrams", pp. 1-10, 2016.

[72] F. Huber y G. Hagel, "Towards detection and syntactical analysis in UML class diagrams for software engineering education", *IEEE Global Engineering Education Conference (EDUCON),* pp. 3-7, 2020.

[73] I. Pogarcic, M. Francic y V. Davidovik, "Abstraction as a Criterion for Defining Class Diagram and Object Diagrams", *Conference: Proceedings of the 10th International Conference on Information System Implementation and Modeling,* 2007.

[74] B. Karasneh, D. Stikkolorum y E. Larios, "Quality Assessment of UML Class Diagrams", *EduSymp@MoDELS (2015),* 2015.

[75] A. Hafeez, A. Ahmed, M. Furqan, W. Rehaman y I. Husain, "Importance and Impact of Class Diagram in Software development", *Indian Journal of Science and Technology,* vol. 12, nº 25, pp. 1-4, 2019.

[76] K. Jin-Man , K. Tae-Hee y L. Joa-Sang , "Effect of Layout and Complexity of Class Diagram on Model Comprehension", *Journal of Internet Computing and Services,* vol. 14, pp. 233-259, 2011.

[77] C. BATINI, L. FURLANI y E. NARDELLI, "What is a Good Diagram? A Pragmatic Approach", *IEEE Xplore,* 1985.

[78] M. Genero, M. Piatini y E. Manso, "Finding "Early" Indicators of UML Class Diagrams Understandability and Modifiability", *International Symposium on Empirical Software Engineering (ISESE'04),* pp. 207-216, 2004.

[79] U. Kelte, J. Wehren y J. Niere, "A Generic Difference Algorithm for UML Models.", *Software Engineering 2005,* pp. 105-116, 2005.

[80] C. Lange, B. DuBois, M. Chaudron y S. Demeyer, "Experimentally investigating the effectiveness and effort of modeling conventions for the UML", vol. 4199, 2006.

[81] C. Lange, M. Wijns y M. Chaudron, "UML-based Views for Monitoring Model Evolution and Quality", *11th European Conference on Software Maintenance and Reengineering (CSMR'07),* pp. 327-328, 2007.

[82] R. Klimek y P. Szwed, "Formal Analysis of Use Case Diagrams", *Computer Science,* vol. 11, pp. 30-059, 2010.

[83] H. Eichelberger y K. Schmid, "Guidelines on the aesthetic quality of UML class diagrams", *Information and Software Technology*, vol. 51, nº 12, pp. 1686-1698, 2009.

[84] H. Störrle, "On the Impact of Layout Quality to Understanding UML Diagrams: Size Matters", *Proceedings of 17th International Conference on Model Driven Engineering Languages and Systems*, pp. 518-534, 2017.

[85] G. Csertán, G. Huszerl, I. Huszerl, Z. Pap, A. Pataricza y D. Varró, "Visual Automated Transformations for Formal Verification and Validation of UML Models – VIATRA", pp. 267-270, 2002.

[86] H. Purchase, J. Allder y D. Carrington, "Graph Layout Aesthetics in UML Diagrams: User Preferences", *Journal of Graph Algorithms and Applications*, vol. 6, nº 3, pp. 255-279, 2002.

[87] D. Sun y W. Wong, "On Evaluating the Layout of UML Class Diagrams for Program Comprehension", *13th International Workshop on Program Comprehension*, vol. 14, pp. 233-259, 2006.

[88] C. Carbon, T. Mchedlidze, M. Raab y H. Wächter, "The Power of Shape: How Shape of Node-Link Diagrams Impacts Aesthetic Appreciation and Triggers Interest", *i-Perception*, vol. 9, nº 5, 2018.

[89] H. Purchase, J. Allder y D. Carrington, "User Preference of Graph Layout Aesthetics: A UML Study", *Springer-Verlag*, vol. 1984, pp. 5-18, 2001.

[90] H. Störrie, "Diagram Size vs. Layout Flaws: Understanding Quality Factors of UML Diagrams", vol. 10, pp. 1-10, 2016.

[91] C. Purchase, M. McGill, L. Colpoys y D. Carrington, "Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study", *Australian Symposium on Information Visualization. Conferences in Research and Practice in Information Technology*, vol. 9, pp. 129-137, 2001.

[92] M. Genero y et al, "A Systematic Literature Review on the Quality of UML Models", vol. 22, pp. 46-70, 2011.

[93] M. Marchesi, "OOA Metrics for the Unified Modeling Language", 2010.

[94] O. Deryugina, "Improving the structural quality of UML class diagrams with the genetic algorithm", *ITM Web of Conferences*, vol. 6, pp. 1-4, 2016.

[95] K. Breesam, "Metrics for Object-Oriented Design Focusing on Class Inheritance Metrics", *2nd International Conference on Dependability of Computer Systems (DepCoS-RELCOMEX'07)*, pp. 231-237, 2007.

[96] P. Mohagheghi y J. Mohagheghi, "Evaluating Quality in Model-Driven Engineering", *International Workshop on Modeling in Software Engineering (MISE'07: ICSE Workshop 2007),* pp. 6-6, 2007.

[97] B. Mathur y M. Kaushik, "Empirical Analysis of Metrics Using UML Class Diagram", *International Journal of Advanced Computer Science and Applications,* vol. 7, nº 5, pp. 32-37, 2016.

[98] M. Genero, M. Piattini y C. Calero, "Early measures for UML class diagrams", vol. 6, nº 4, pp. 489-505, 2000.

[99] M. Sharma y R. Vishwakarma, "CMMI Based Software Metrics to Evaluate OOAD", *Proceedings of the Second International Conference on Computational Science, Engineering and Information Technology,* pp. 19-25, 2012.

[100] S. Yacoub, H. Ammar y T. Robinson, "Designs Dynamic Metrics for Object Oriented Designs", vol. 5, pp. 50-61, 1999.

[101] O. Deryugina, "UML class diagram object-oriented metrics: algorithms of calculation", *7th Seminar on Industrial Control Systems: Analysis, Modeling and Computing (ICS 2018),* vol. 18, pp. 1-6, 2018.

[102] E. Marinescu, "Measurement and Quality in Object-Oriented Design", *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM'05),* pp. 701-704, 2005.

[103] F. Thung, D. Lo, M. Osman y M. Chaudron, "Condensing class diagrams by analyzing design and network metrics using optimistic classification", *Proceedings of the 22nd International Conference on Program Comprehension,* pp. 110-121, 2014.

[104] S. Sarica y T. Ovatman , "Software Design Metric Based Analysis of Dependency Patterns", *IEEE Xplore Digital Library. Second International Conference on Informatics & Applications (ICIA),* pp. 317-322, 2013.

[105] R. Santaolaya-Salgado, O. G. Fragoso-Díaz y S. A. Zamudio-López, "Formal Model for Restructuring of Object-Oriented Frameworks to Architecture Model-View-Adapter", *Ingeniería. Investigación y Tecnología,* pp. 187-198, 2014.

[106] J. Mendoza y J. B. Garza, "Measurement in the scientific research process: Content validity and reliability evaluation", *InnOvaciOnes de NegOciOs,* vol. 6, nº 1, pp. 17-32, 2009.

[107] N. BEVAN, "Los nuevos modelos de ISO para la calidad y la calidad en uso del software", de *alidad del producto y proceso software*, Ra-Ma, 2010, pp. 5-75.

[108] A. Queralt y E. Teniente, "Validation of UML Conceptual schemas with OCL Constraints and Operations", *Universitat Politècnica de Catalunya,* vol. 21, nº 2, pp. 1-41, 2009.

[109] M. Bacelar Valente, "The Correctness of Reasoning, Logical Models, and the Faithfulness Problem", vol. 26, nº 3, pp. 429-447, 2020.

[110] O. N. Fundation, UML Modeling Guidelines, Menlo Park, 2018.

[111] O. M. Group, Unified Modeling Language. Ver. 2.5.1., 2017.

[112] O. M. Group, Unified Modeling Language: Superstructure, 2005.

[113] F. Alliende, "Evaluación de la Legibilidad de los Materiales Escritos", *Lectura y Vida. Revista Latinoamericana de Lectura,* vol. 11, nº 2, 1990.

[114] C. BATINI, L. FURLANI y E. NARDELLI, "What is a Good Diagram? A Pragmatic Approach", *IEEE Computer Society and North-Holland,* pp. 312-319, 1985.

[115] K. SUGIYAMA, S. TAGAWA y M. TODA, "Methods for Visual Understanding of Hierarchical System Structures", *IEEE Transactions on Systems, Man and Cybernetics,* vol. 11, nº 2, pp. 109-125, 1981.

[116] S. W. Durand, Análisis y requerimientos de software, vol. 1, Huancayo: Universidad Continental, 2017, p. 128.

[117] M. Hailperin, B. Kaiser y K. Knight, Concrete Abstractions An Introduction to Computer Science Using Scheme, London: International Thomson Editores, 1999.

[118] A. Hunt y D. Thomas, Pragmatic Programmer. The: From Journeyman to Master, Addison Wesley, 1999.

"Si haces planes para un año, siembra arroz.
Si los haces para dos lustros, planta árboles.
Si los haces para toda la vida, educa a una persona".

Antiguo proverbio chino.

Capítulo 7 Anexos

CAPÍTULO 7. ANEXOS

7.1. Anexo 1: Definición de Reglas de Dominio

1. Definición de elementos en el DC

Sea *o* un elemento contenido en un conjunto de elementos de un DC;

Donde:

U = {X | X es un elemento de la notación UML contenido en un DC};
O es la cantidad de elementos contenidos en un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *O* el conjunto de elementos contenidos en un DC;
Sea *o* un elemento de *O*;
$O \neq \{\emptyset\}$;

Fórmula 1:

$$O = \sum_{i=1}^{n} o_i$$

2. Definición de clases en el DC

Sea *c* una clase contenida en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase contenida en un conjunto de elementos de un DC};
Sea *C* el conjunto de clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *c* un elemento de *C*;
$C \neq \{\emptyset\}$;

Fórmula 2:

$$C = \sum_{i=1}^{n} c_i$$

3. Definición de interfaces en el DC

Sea *ci* una clase de un conjunto de clases definidas como Interfaces en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase de un conjunto de clases definidas como Interfaces en un conjunto de elementos de un DC};
CI es el conjunto de clases definidas como interfaces contenidas en un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *A* el conjunto de clases de un DC;
Sea *a* un elemento de *A*;
Sea *CI* el subconjunto de clases definidas como interfaces y que forman parte de *A*;
Sea *ci* un elemento de *CI*;
CI ≠ {∅};

Fórmula 3:

$$CI = \sum_{i=1}^{n} ci_i$$

4. Definición de clases estáticas en el DC

Sea *ce* una clase definida como clase estática en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase definida como clase estática en un conjunto de elementos de un DC};
CE es el conjunto clases definidas como clases estáticas contenidas en un DC;

i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *A* el conjunto de clases de un DC;
Sea *a* un elemento de *A*;
Sea *CE* el conjunto de clases estáticas que forman parte de *A*;
Sea ce un elemento de *CE*;
CE ≠ {Ø};

Fórmula 4:

$$CE = \sum_{i=1}^{n} ce_i$$

5. Definición de clases sin relación con otras clases en el DC

Sea *csr* una clase sin relación representada con otras clases en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase sin relación representada con otras clases en un conjunto de elementos de un DC};
CSR es el conjunto de todas las clases sin relación representada con otras clases en el DC;
Sea *A* el conjunto de clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *CSR* el subconjunto clases en A que no tienen al menos una relación establecida con otra clase que forma parte de *A*;
Sea *csr* un elemento de *CSR*;
CSR ≠ {Ø};
|*CSR*| >= 1;
Valor de calidad: 0;
El conjunto *CSR* tiene al menos un elemento, por lo tanto: *A* tiene al menos una clase que no tienen al menos una relación establecida con otra clase que forma parte de *A*;

Fórmula 5:

$$CSR = \sum_{i=1}^{n} csr_i$$

6. Definición de clases abstractas en el DC

Sea *ca* una clase definida como clases abstractas en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase definida como clases abstractas en un conjunto de elementos de un DC};
CA es conjunto de clases definidas como clases abstractas contenidas en un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *A* el conjunto de clases de un DC;
Sea *a* un elemento de *A*;
Sea *CA* el subconjunto de clases abstractas que forma parte de *A*;
Sea *ca* un elemento de *CA*;
CA ≠ {Ø};

Fórmula 6:

$$CA = \sum_{i=1}^{n} ca_i$$

7. Definición de clases abstractas que no son heredadas sobre una subclase

Sea *canh* una clase definida como clase abstracta que no se heredan a alguna(s) subclase(s) de un conjunto de clases en un DC;

Donde:

U = {X | X es una clase definida como clase abstracta que no se heredó sobre una subclase de un conjunto de clases en un DC};

CANH es el conjunto de clases definidas como subclases abstractas que no es heredada al menos en una subclase de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *A* el conjunto de clases abstractas de un DC;
Sea M una relación de herencia con otra clase ya se abstracta o concreta que forman parte de *A*;
Sea *canh* un elemento de *CANH*;
CANH ≠ {Ø};
|*CANH*| > 0;
Valor de calidad: 0;
El conjunto *CANH* tiene al menos un elemento, por lo tanto: *A* tiene al menos una clase que no tienen al menos una relación establecida con otra clase que forma parte de *A*;

Fórmula 7:

$$CANH = \sum_{i=1}^{n} canh_i$$

8. Definición de clases que contienen operaciones abstractas pero la clase no se definió como abstracta

Sea *coac* una operación abstracta contenida en una clase que no se definió como clase abstracta en un conjunto de clases en un DC;

Donde:

U = {X | X es una operación abstracta contenida en una clase que no se definió como clase abstracta en un conjunto de clases en un DC};
COAC es la cantidad de clases concretas que contienen operaciones abstractas en un conjunto de elementos de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *A* una clase concreta en *C*;
Sea *Y* el conjunto de operaciones contenidas en *A*;
Sea *y* un elemento de *Y*;

Sea *COAC* un subconjunto de operaciones contenidas en *Y* que se definieron como operaciones abstractas en las clases que forman parte de A;
Sea *coac* un elemento de *COAC*;
COAC ≠ {Ø};
|*COAC*| > 0;
Valor de calidad: 0;
El conjunto *COAC* tiene al menos un elemento, por lo tanto: *C* tiene al menos una operación abstracta contenida en una clase que no se definió como clase abstracta;

Fórmula 8:

$$COAC = \sum_{i=1}^{n} coac_i$$

9. Definición de clases concretas con relación distinta a herencia con una abstracción y que contienen operaciones o atributos de la abstracción

Sea *crnha* una clase con relación distinta a herencia con una abstracción y que contiene operaciones o atributos de la abstracción en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase con relación distinta a herencia con una abstracción y que contiene operaciones o atributos de la abstracción en un conjunto de elementos de un DC};
CRNHA es el conjunto de clases con relación distinta a herencia con una abstracción y que contiene operaciones o atributos de la abstracción en un conjunto de clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *C* un conjunto de clases en un DC;
Sean *A* y *B* clases en *C*;
Sea *R* una relación que no es relación de herencia en el DC que conecta *A* con *B*;
Donde A está conectada con *B*;
Sea *Y* es el conjunto de operaciones o atributos contenidos en *A*;
Sea *y* un elemento de *Y*;
Sea *Z* el conjunto de operaciones o atributos contenidos en *B*;
Sea *z* un elemento de *Z*;

CRNHA = Y ∩ Z;
Sea *crnha* un elemento de en *CRNHA*;
CRNHA ≠ {Ø};
|*CRNHA*| > 0;
Valor de calidad: 0;
El conjunto *CRNHA* tiene al menos un elemento, por lo tanto: *C* tiene al menos una clase con relación distinta a herencia con una abstracción y que contiene operaciones o atributos de la abstracción;

Fórmula 9:

$$CRNHA = \sum_{i=1}^{n} crnha_i$$

10. Definición de clases abstractas que son subclases de una clase concreta

Sea *cascc* una clase definida como clase abstracta y que es subclase de una clase concreta en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase definida como clase abstracta y que es subclase de una clase concreta en un conjunto de elementos de un DC};
CASCC es el conjunto de clases definidas como clases abstractas y que son subclases de una clase concreta en un conjunto de clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *M* un conjunto de clases de un DC;
Sea *A* una clase concreta que forma parte de *M*;
Sea *B* una clase abstracta que forma parte de *M*;
Sea *R* una relación de herencia en el DC que conecta *A* con *B*;
Donde B hereda de *A*;
Sea *CASCC* es el conjunto de clases abstractas que forman parte de M y que heredan de una clase concreta;
Sea *cascc* un elemento de *CASCC*;
|*CASCC*| > 0;
Valor de calidad: 0;

El conjunto *CASCC* tiene al menos un elemento, por lo tanto: *M* tiene al menos una clase definida como clase abstracta y que es subclase de una clase concreta;

Fórmula 10:

$$CASCC = \sum_{i=1}^{n} cascc_i$$

11. Definición de clases con herencia múltiple en el DC

Sea *chm* una clase que es subclase de varias superclases en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase que es subclase de varias superclases en un conjunto de elementos de un DC};
CHM es el conjunto de subclases que tienen varias clases padres en un conjunto de clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *M* el conjunto de clases de un DC;
Sea A un conjunto de superclases que forma parte de M;
Sea a un elemento de A;
Sea *D* una subclase de *M*;
Sea *R* un conjunto de relaciones de herencia entre la clase *D* una superclase en el DC;
Sea *r* un elemento de *R*;
Sea *F* una relación de herencia que conecta un elemento de A llamado a1 con *D* y que forma parte de *R*;
Donde D hereda de a1;
Sea *G* una relación de herencia que conecta un elemento de A llamado a2 con *D* y que forma parte de *R*;
Donde *D* hereda de a2;
Sea *H* una relación de herencia que conecta un elemento de A llamado a3 con *D* y que forma parte de R;
Donde *D* hereda de a3;
Sea *CHM* el subconjunto de clases en *M* que tienen herencia múltiple;
Sea *chm* un elemento de *CHM*;

$|CHM| > 0$;
Valor de calidad: 0;
El conjunto *CHM* tiene al menos un elemento, por lo tanto: *M* tiene al menos una clase que es subclase de varias superclases;

Fórmula 11:

$$CHM = \sum_{i=1}^{n} chm_i$$

12. Definición de relaciones en el DC

Sea *rs* una relación definida en un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación definida en un conjunto de elementos de un DC};
RS es el conjunto de relaciones definidas en un conjunto de elementos de un DC;
Sea *rs* un elemento de *RS*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
$C \neq \{\emptyset\}$;

Fórmula 12:

$$RS = \sum_{i=1}^{n} rs_i$$

13. Definición de relaciones de asociación en el DC

Sea *ras* una relación definida como relación de asociación en un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación definida como relación de asociación en un conjunto de elementos de un DC};

RAS es el conjunto de relaciones definidas como relaciones de asociación en un conjunto de elementos de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *C* un conjunto de relaciones de un DC;
Sea *c* un elemento de *C*;
Sea *RAS* el subconjunto relaciones de asociación contenidas en *C*;
Sea *ras* un elemento de *RAS*;
RAS ≠ {Ø};

Fórmula 13:

$$RAS = \sum_{i=1}^{n} ras_i$$

14. Definición de relaciones de asociación directa en el DC

Sea *rasd* una relación definida como relación de asociación directa en un conjunto de elementos de un DC;

Donde:

U = {X | X es relación definida como relación de asociación directa en un conjunto de *RASD* es la cantidad de relaciones definidas como relaciones de asociación directa en un conjunto de elementos de un DC};
Sea *C* un conjunto de relaciones de un DC;
Sea *c* un elemento de *C*;
Sea *RASD* un subconjunto relaciones de asociación directa contenidas en *C*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rasd* un elemento de *RASD*;
RASD ≠ {Ø};

Fórmula 14:

$$RASD = \sum_{i=1}^{n} rasd_i$$

15. Definición de relaciones de agregación en el DC

Sea *rag* una relación definida como relación de agregación en un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación definida como relación de agregación en un conjunto de elementos de un DC};
RAG es la cantidad de relaciones definidas como relaciones de agregación en un conjunto de elementos de un DC;
Sea *C* un conjunto de relaciones de un DC;
Sea *c* un elemento de *C*;
Sea *RAG* un conjunto subrelaciones de agregación contenidas en *C*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rag* un elemento de *RAG*;
$RAG \neq \{\emptyset\}$;

Fórmula 15:

$$RAG = \sum_{i=1}^{n} rag_i$$

16. Definición de relaciones de Composición en el DC

Sea *rcp* una relación de composición que se encuentra entre dos clases del diagrama, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación de composición que se encuentra entre dos clases del diagrama, y que forma parte de un conjunto de elementos de un DC};
RCP es el conjunto de relaciones de composición que se encuentran entre dos clases del diagrama, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de relaciones de un DC;
Sea *c* un elemento de *C*;
Sea *RCP* un subconjunto relaciones de composición contenidas en *C*;

i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rcp* un elemento de *RCP*;
RCP ≠ {Ø};

Fórmula 16:

$$RCP = \sum_{i=1}^{n} rcp_i$$

17. Definición de relaciones de dependencia en el DC

Sea *rdp* una relación definida como relación de dependencia en un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación definida como relación de dependencia en un conjunto de elementos de un DC};
RDP es el conjunto de relaciones definidas como relaciones de dependencia en un conjunto de elementos de un DC;
Sea *C* un conjunto de relaciones de un DC;
Sea *c* un elemento de *C*;
Sea *RDP* un subconjunto relaciones de dependencia contenidas en *C*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rdp* un elemento de *RDP*;
RDP ≠ {Ø};

Fórmula 17:

$$RDP = \sum_{i=1}^{n} rdp_i$$

18. Definición de relaciones de realización en el DC

Sea *rrz* una relación definida como relación de realización en un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación definida como relación de realización en un conjunto de elementos de un DC};
RRZ es el conjunto de relaciones definidas como relaciones de realización en un conjunto de elementos de un DC;
Sea *C* un conjunto de relaciones de un DC;
Sea *c* un elemento de *C*;
Sea *RRZ* un conjunto subrelaciones de realización contenidas en *C*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rrz* un elemento de *RRZ*;
RRZ ≠ {Ø};

Fórmula 18:

$$RRZ = \sum_{i=1}^{n} rrz_i$$

19. Definición de relación de Herencia en el DC

Sea *rg* una relación definida como relación de herencia en un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación definida como relación de herencia en un conjunto de elementos de un DC};
RG es la cantidad de relaciones definidas como relaciones de herencia en un conjunto de elementos de un DC;
Sea *C* un conjunto de relaciones de un DC;
Sea *c* un elemento de *C*;
Sea *RG* un subconjunto relaciones de herencia contenidas en *C*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rg* un elemento de *RG*;
RG ≠ {Ø};

Fórmula 19:

$$RG = \sum_{i=1}^{n} rg_i$$

20. Definición de relaciones reflexivas en el DC

Sea *rrf* es una línea que representa una relación reflexiva, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una línea que representa una relación reflexiva, y que forma parte de un conjunto de elementos de un DC};
RRF es la cantidad de líneas que representan relaciones reflexivas, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de relaciones de un DC;
Sea c un elemento de *C*;
Sea *RRF* un subconjunto relaciones reflexivas contenidas en *C*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rrf* un elemento de *RRF*;
RRF ≠ {Ø};

Fórmula 20:

$$RRF = \sum_{i=1}^{n} rrf_i$$

21. Definición de relaciones incompletas en el DC

Sea *rin* una relación en un conjunto de elementos de un DC y que no tiene una clase definida destino y/o una clase origen;

Donde:

U = {X | X es una relación en un conjunto de elementos de un DC y que no tiene una clase definida destino y/o una clase origen};
RIN es el conjunto de relaciones en un conjunto de elementos de un DC y que no tienen definidas una clase destino y/o una clase origen;
Sea *C* un conjunto de relaciones de un DC;

Sea *c* un elemento de *C*;
Sea *R* una relación contenida en *C*;
Sea *RIN* el subconjunto de relaciones en *R* que no tienen una clase origen o no tienen una clase destino o no tienen ambas;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea rin un elemento de *RIN*;
RIN ≠ {Ø};
|*RIN*| > 0;
Valor de calidad: 0;
El conjunto *RIN* tiene al menos un elemento, por lo tanto: *C* tiene al menos una relación en un conjunto de elementos de un DC y que no tiene una clase definida destino y/o una clase origen;

Fórmula 21:

$$RIN = \sum_{i=1}^{n} rin_i$$

22. Definición de relaciones no reconocidas por el estándar UML en el DC

Sea *rnr* una relación que se encuentra entre dos clases del diagrama, pero la relación no está definida por la notación UML, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación que se encuentra entre dos clases del diagrama, pero la relación no está definida por la notación UML, y que forma parte de un conjunto de elementos de un DC};
RNR es el conjunto de relaciones que se encuentran entre dos clases del diagrama, pero la relaciones no están definidas por la notación UML, y que forman parte de un conjunto de elementos de un DC;
Sea *R* el conjunto de relaciones en el DC;
Sea *r* un elemento de *R*;
Sea *RNR* el subconjunto de relaciones que no son un tipo de relación reconocido por la notación UML y que están contenidas en *R*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;

Sea *rnr* un elemento de *RNR*;
RNR ≠ {Ø};
|*RNR*| > 0;
Valor de calidad: 0;
El conjunto *RNR* tiene al menos un elemento, por lo tanto: R tiene al menos una relación que se encuentra entre dos clases del diagrama, pero la relación no está definida por la notación UML;

Fórmula 11:

$$RNR = \sum_{i=1}^{n} rnr_i$$

23. Definición de relaciones que repiten las mismas clases en sus extremos (no reflexivas) en el DC

Sea *rrce* una relación que conecta dos clases del diagrama, ambas clases están relacionadas con más de una relación del mismo o distinto tipo que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación que conecta dos clases del diagrama, pero dicha relación se encuentra repetida por otra relación de otro o del mismo tipo, y que forma parte de un conjunto de elementos de un DC};
RRCE es el conjunto de relaciones que conectan dos clases del diagrama, pero dichas relaciones se encuentran repetidas por otras relaciones de otro o del mismo tipo, y que forman parte de un conjunto de elementos de un DC;
Sea *A* y *B* clases contenidas en un DC;
Sea *R* el conjunto de relaciones en el DC;
Sea *r* un elemento de *R*;
Sea *RRCE* el conjunto de relaciones que conectan *A* con *B*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *rrce* un elemento de *RRCE*;
RRCE ≠ {Ø};
|*RRCE*| > 1;
Valor de calidad: 1;

El conjunto *RRCE* tiene al menos un elemento, por lo tanto: *R* tiene al menos una relación que conecta dos clases del diagrama, pero dicha relación se encuentra repetida por otra relación de otro o del mismo tipo;
Fórmula 23:

$$RRCE = \sum_{i=1}^{n} rrce_i$$

24. Definición de interfaces con relación distinta a realización con una clase concreta

Sea *cirdr* una clase definida como interfaz que no tiene definida por menos una relación de realización con otra clase en un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase definida como interfaz que no tiene definida por menos una relación de realización con otra clase en un conjunto de elementos de un DC};
CIRDR es el conjunto de clases definidas como y que no tienen definida por menos una relación de realización con otra clase en un conjunto de elementos de un DC;
Sea *C* un conjunto de clases de un DC;
Sea *A* una interfaz en *C*;
Sea *B* una clase en *C*;
A está conectada con *B*;
Sea *P* un conjunto de delaciones contenidas en el DC;
Sea *p* un elemento de *P*;
Sea *CIRDR* un subconjunto de relaciones contenidas en *P* que conecta *A* con *B,* y *R* no es un tipo de relación de realización;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *cirdr* un elemento de *CIRDR*;
CIRDR ≠ {Ø};
|*CIRDR*| > 0;
Valor de calidad: 0;
El conjunto *CIRDR* tiene al menos un elemento, por lo tanto: *C* tiene al menos una clase definida como interfaz que no tiene definida por menos una relación de realización con otra clase;

Fórmula 24:

$$CIRDR = \sum_{i=1}^{n} cirdr_i$$

25. Definición de atributos en el DC

Sea *atr* un atributo contenido en una clase que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un atributo contenido en una clase que forma parte de un conjunto de elementos de un DC};
ATR es el conjunto de atributos contenidos en clases que forman parte de un conjunto de elementos de un DC;
Sea *C* el conjunto de clases contenidas en un DC;
Sea *A* el conjunto de atributos contenidos en las clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *atr* un elemento de *ATR*;
ATR ≠ {Ø};

Fórmula 25:

$$ATR = \sum_{i=1}^{n} atr_i$$

26. Definición de operaciones en el DC

Sea *op* una operación contenida en una clase que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación contenida en una clase que forma parte de un conjunto de elementos de un DC};
OP es el conjunto de operaciones contenidas en clases que forman parte de un conjunto de elementos de un DC;
Sea *C* el conjunto de clases contenidas en un DC;

Sea *OP* el conjunto de operaciones contenidas en las clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea op un elemento de *OP*;
OP ≠ {Ø};

Fórmula 26:

$$OP = \sum_{i=1}^{n} op_i$$

27. Definición de operaciones abstractas en el DC

Sea *oab* una operación abstracta contenida en una clase que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación abstracta contenida en una clase que forma parte de un conjunto de elementos de un DC};
OAB es el conjunto de operaciones abstractas contenidas en clases que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *P* el conjunto de operaciones contenidas en un DC;
Sea *p* un elemento de *P*;
Sea *OAB* el conjunto de operaciones abstractas contenidas en *P*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *oab* un elemento de *OAB*;
OAB ≠ {Ø};

Fórmula 27:

$$OAB = \sum_{i=1}^{n} oab_i$$

28. Definición de atributos y operaciones estáticos en el DC

Sea *aoe* una operación estática o un atributo estático contenido en una clase que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación estática o un atributo estático contenido en una clase que forma parte de un conjunto de elementos de un DC};
AOE es el conjunto de operaciones estáticas o atributos estáticos contenidos en clases que forma parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *P* el conjunto de atributos y operaciones estáticas contenidas en un DC;
Sea *p* un elemento de *P*;
Sea *A* el conjunto de atributos y operaciones estáticas contenidas en las clases de un DC;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *aoe* un elemento de *AOE*;
$AOE \neq \{\emptyset\}$;

Fórmula 28:

$$AOE = \sum_{i=1}^{n} aoe_i$$

29. Definición de clases cuyos nombres no están en letra *bold* en el DC

Sea *cnb* un nombre de clase cuya letra no se encuentra representada en tipo bold y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un nombre de clase cuya letra no se encuentra representada en tipo bold y que forma parte de un conjunto de elementos de un DC};
CNB es el conjunto de nombres de una clase cuya letra no se encuentra representada en tipo bold y que forman parte de un conjunto de elementos de un DC;

Sea *C* un conjunto de clases de un DC;
Sea *Y* un conjunto de nombres de clase contenidos en *C*;
Sea *y* un elemento de *Y*;
Sea *CNB* un subconjunto de nombres de clase contenidos en Y que no tienen texto bold.
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *cnb* un elemento de *CNB*;
CNB ≠ {Ø};
|*CNB*| > 0;
Valor de calidad: 0;
El conjunto *CNB* tiene al menos un elemento, por lo tanto: *C* tiene al menos una clase con nombre de clase cuya letra no se encuentra representada en tipo bold;

Fórmula 29:

$$CNB = \sum_{i=1}^{n} cnb_i$$

30. Definición de atributos u operaciones cuyos nombres están texto *bold* en el DC

Sea *natb* un nombre de atributo u operación cuya letra se encuentra en tipo bold y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un nombre de atributo u operación cuya letra se encuentra en tipo bold y que forma parte de un conjunto de elementos de un DC};
NATB es el conjunto de nombres el nombre de atributos u operaciones cuya letra se encuentra en tipo bold y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases de un DC;
Sea *Y* un conjunto de atributos y operaciones contenidos en *C*;
Sea *y* un elemento de *Y*;
Sea *NATB* un subconjunto de atributos y operaciones contenidos en *Y* que tienen nombres con texto bold.
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;

Sea *natb* un elemento de *NATB*;
NATB ≠ {Ø};
|*NATB*| > 0;
Valor de calidad: 0;
El conjunto *NATB* tiene al menos un elemento, por lo tanto: *C* tiene al menos nombre de atributo u operación cuya letra se encuentra en tipo bold;

Fórmula 30:

$$NATB = \sum_{i=1}^{n} natb_i$$

31. Definición de operaciones no implementadas entre una interfaz y una clase

Sea *oimi* una operación que forma parte de una interfaz y que no se implementó en al menos una clase del diagrama y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación que forma parte de una interfaz y que no se implementó en al menos una clase del diagrama y que forma parte de un conjunto de elementos de un DC};
OIMI es la cantidad de operaciones que forman parte de una interfaz y que no se implementaron en al menos una clase del diagrama y que forma parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases en un DC;
Sea *A* una interfaz en *C*;
Sea *B* una clase en *C*;
Sea *R* una relación de realización en el DC que conecta *A* con *B*
Donde B implementa el comportamiento de *A*;
Y es el subconjunto de operaciones contenidas en *A*;
y es un elemento de *Y*;
OIMI es el subconjunto de operaciones implementada por *B* desde *A*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *oimi* un elemento de *OIMI*;
Y ≠ *OIMI*;
Valor de calidad: Y debe ser igual a *OIMI*;

El conjunto Y es distinto al conjunto *OIMI*, por lo tanto: C tiene al menos una operación que forma parte de una interfaz y que no se implementó en al menos una clase del diagrama;

Fórmula 31:

$$OIMI = \sum_{i=1}^{n} oimi_i$$

32. Definición de Operaciones que no se declararon públicas en interfaces

Sea *opi* una operación que forma parte de una interfaz y que no se declararon pública y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación que forma parte de una interfaz y que no se declararó pública y que forma parte de un conjunto de elementos de un DC};
OPI es el conjunto de operaciones que forman parte de una interfaz y que no se declararon públicas y que forma parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases en un DC;
Sea *A* una clase definida como interfaz en C;
Sea *O* un conjunto de operaciones de *A*;
Sea *o* una operación contenida en *O*;
Sea *OPI* un subconjunto de operaciones de *O* que no se declararon públicas;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *opi* una operación contenida en *OPI*;
OPI ≠ {Ø};
|*OPI*| > 0;
Valor de calidad: 0;
El conjunto *OPI* tiene al menos un elemento, por lo tanto: *C* tiene al menos una operación que forma parte de una interfaz y que no se declaró pública;

Fórmula 32:

$$OPI = \sum_{i=1}^{n} opi_i$$

33. Definición de Atributos contenidos en interfaces en el DC

Sea *aci* un atributo que se encuentra definido en una interfaz y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un atributo que se encuentra definido en una interfaz y que forma parte de un conjunto de elementos de un DC};
ACI es el conjunto de atributos que se encuentran definidos en interfaces y que forman parte de un conjunto de elementos de un DC;
Sea C un conjunto de clases en un DC;
Sea *IN* un conjunto clases definidas como interfaces en C;
Sea *in* un elemento de *IN*;
Sea *ACI* un conjunto de atributos contenidos en *IN*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *aci* un elemento de *ACI*;
ACI ≠ {Ø};
|*ACI*| > 0;
Valor de calidad: 0;
El conjunto *Y* tiene al menos un elemento, por lo tanto: *C* tiene al menos una operación que forma parte de una interfaz y que no se declaró pública;

Fórmula 33:

$$ACI = \sum_{i=1}^{n} aci_i$$

34. Definición de operaciones repetidos entre Subclase y Superclase

Sea *orcs* una operación que se encuentra definido en una clase que es subclase de una abstracción y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación que se encuentra definido en una clase que es subclase de una abstracción y que forma parte de un conjunto de elementos de un DC};
ORCS es el conjunto de operaciones que se encuentran definidas en clases que son subclases de abstracciones y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases en un DC;
Sean *A y B* clases en *C*;
Sea *R* una relación de herencia de un DC;
Donde A está conectada con *B*;
Donde B hereda de *A*;
Sea *Y* es el subconjunto de operaciones contenidas en *A*;
Sea *y* es un elemento de *Y*;
Sea *Z* es el subconjunto de operaciones contenidas en *B*;
Sea *z* es un elemento de *Z*;
Sea *ORCS* el conjunto resultado de la intersección de *Y* con *Z*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *orcs* un elemento de *ORCS*;
ORCS = Y ∩ Z;
ORCS ≠ {Ø};
|*ORCS*| > 0;
Valor de calidad: 0;
El conjunto *ORCS* tiene al menos un elemento, por lo tanto: los conjuntos *Y y Z* comparten al menos una operación;

Fórmula 34:

$$ORCS = \sum_{i=1}^{n} orcs_i$$

35. Definición de atributos repetidos entre Subclase y Superclase

Sea *arcs* un atributo que se encuentra definido en una clase que es subclase de una abstracción y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un atributo que se encuentra definido en una clase que es subclase de una abstracción y que forma parte de un conjunto de elementos de un DC};
ARCS es el conjunto de atributos que se encuentran definidas en clases que son subclases de abstracciones y que forman parte de un conjunto de elementos de un DC;
Sean *A y B* clases de un DC;
Sea *R* una relación de herencia de un DC;
Sea *r* un elemento de *R*;
Donde B hereda de *A*;
Sea *Y* es el subconjunto de atributos contenidos en *A*;
Sea *y* es un elemento de *Y*;
Sea *Z* es el subconjunto de atributos contenidos en *B*;
Sea *z* es un elemento de *Z*;
Sea ARCS el conjunto resultado de la intersección de *Y* con *Z*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *arcs* un elemento de *ARCS*;
ARCS = Y ∩ Z;
ARCS ≠ {Ø};
|*ARCS*| > 0;
Valor de calidad: 0;
El conjunto *ARCS* tiene al menos un elemento, por lo tanto: los conjuntos *Y y Z* comparten al menos un elemento;

Fórmulas 35:

$$ARCS = \sum_{i=1}^{n} arcs_i$$

36. Definición de operaciones repetidas entre clases asociadas en el DC

Sea *orcas* una operación que se encuentra definida en una clase que tiene relación de asociación con otra clase que contiene la misma operación y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación que se encuentra definida en una clase que tiene relación de asociación con otra clase que contiene la misma operación y que forma parte de un conjunto de elementos de un DC};
ORCAS es el conjunto de operaciones que se encuentran definidas en clases que tienen relación de asociación con otras clases que contienen la misma operación y que forman parte de un conjunto de elementos de un DC;
Sean *A y B* clases de un DC;
Sea *R* una relación de asociación;
Sea *r* un elemento de *R*;
A está conectada con *B*;
Sea *Y* es el subconjunto de operaciones contenidas en *A*;
Sea *y* es un elemento de *Y*;
Sea *Z* es el subconjunto de operaciones contenidas en *B*;
Sea *z* es un elemento de *Z*;
Sea *ORCAS* el conjunto resultado de la intersección de *Y* con *Z*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *orcas* un elemento de *ORCAS*;
ORCAS = Y ∩ Z;
ORCAS ≠ {Ø};
|*ORCAS*| > 0;
Valor de calidad: 0;
El conjunto *ORCAS* tiene al menos un elemento, por lo tanto: los conjuntos *Y y Z* comparten al menos una operación;

Fórmula 36:

$$ORCAS = \sum_{i=1}^{n} orcas_i$$

37. Definición de atributos repetidos entre clases asociadas en el DC

Sea *arcas* un atributo que se encuentra definido en una clase que tiene relación de asociación con otra clase que contiene el mismo atributo y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un atributo que se encuentra definido en una clase que tiene relación de asociación con otra clase que contiene el mismo atributo y que forma parte de un conjunto de elementos de un DC};
ARCAS es el conjunto de operaciones que se encuentran definidas en clases que tienen relación de asociación con otras clases que contienen la misma operación y que forman parte de un conjunto de elementos de un DC;
Sean *A y B* clases de un DC;
Sea *R* una relación de asociación;
Sea *r* un elemento de *R*;
Donde A está conectada con *B*;
Sea *Y* es el subconjunto de atributos contenidos en *A*;
Sea *y* es un elemento de *Y*;
Sea *Z* es el subconjunto de atributos contenidos en *B*;
Sea *z* es un elemento de *Z*;
Sea *ARCAS* el conjunto resultado de la intersección de *Y* con *Z*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea arcas un elemento de ARCAS;
ARCAS = Y ∩ Z;
ARCAS ≠ {Ø};
|*ARCAS*| > 0;
Valor de calidad: 0;
El conjunto *ARCAS* tiene al menos un elemento, por lo tanto: los conjuntos *Y* y *Z* comparten al menos un atributo;

Fórmula 37:

$$ARCAS = \sum_{i=1}^{n} arcas_i$$

38. Definición de operaciones repetidas entre clases no relacionadas

Sea *orcna* es una operación que se encuentra repetida entre dos clases que no tienen ninguna línea de relación entre ellas, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación que se encuentra repetida entre dos clases que no tienen ninguna línea de relación entre ellas, y que forma parte de un conjunto de elementos de un DC};
ORCNA es el conjunto de operaciones que se encuentra repetida entre dos clases que no tienen ninguna línea de relación entre ellas, y que forman parte de un conjunto de elementos de un DC;
Sean *A* y *B* clases de un DC;
Donde A no está conectada con *B*;
Sea *Y* un subconjunto de operaciones contenidas en *A*;
Sea *y* un elemento de *Y*;
Sea *Z* un subconjunto de operaciones contenidas en *B*;
Sea *z* un elemento de *Z*;
Sea *ORCNA* el conjunto resultado de la intersección de *Y* con *Z*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *orcna* un elemento de *ORCNA*;
ORCNA = Y ∩ Z;
ORCNA ≠ {Ø};
|*ORCNA*| > 0;
Valor de calidad: 0;
El conjunto *ORCNA* tiene al menos un elemento, por lo tanto: *A* y *B* comparten al menos una operación;

Fórmula38:

$$ORCNA = \sum_{i=1}^{n} orcna_i$$

39. Definición de atributos repetidos entre clases no relacionadas

Sea *arcna* es un atributo que se encuentra repetido entre dos clases que no tienen ninguna línea de relación entre ellas, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un atributo que se encuentra repetido entre dos clases que no tienen ninguna línea de relación entre ellas, y que forma parte de un conjunto de elementos de un DC};

ARCNA es el conjunto de atributos que se encuentran repetidos entre dos clases que no tienen ninguna línea de relación entre ellas, y que forman parte de un conjunto de elementos de un DC;
Sean *A y B* clases de un DC;
Donde A no está conectada con *B*;
Sea *Y* un subconjunto de atributos contenidos en *A*;
Sea *y* un elemento de *Y*;
Sea *Z* un subconjunto de atributos contenidos en *B*;
Sea *z* un elemento de *Z*;
Sea *ARCNA* el conjunto resultado de la intersección de *Y* con *Z*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *arcna* un elemento de *ARCNA*;
ARCNA = Y ∩ Z;
ARCNA ≠ {Ø};
|*ARCNA*| > 0;
Valor de calidad: 0;
El conjunto *ARCNA* tiene al menos un elemento, por lo tanto: *A* y B comparten al menos un atributo;

Fórmula 39:

$$ARCNA = \sum_{i=1}^{n} arcna_i$$

40. Definición de operaciones repetidas entre clases coasociadas

Sea *orcca* es una operación que se encuentra repetida entre dos clases que no tienen ninguna línea de relación entre ellas, pero ambas tienen una relación de asociación con una misma tercera clase, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación que se encuentra repetida entre dos clases que no tienen ninguna línea de relación entre ellas, pero ambas tienen una relación de asociación con una misma tercera clase, y que forma parte de un conjunto de elementos de un DC};
ORCCA es el conjunto de operaciones que se encuentran repetidas entre dos clases que no tienen ninguna línea de relación entre ellas, pero ambas tienen una relación de asociación con una

misma tercera clase, y que forman parte de un conjunto de elementos de un DC;
Sean *A*, *B* y *C* clases de un DC;
Donde A tiene una relación de asociación con B;
Donde A tiene una relación de asociación con *C*;
Donde B no tiene establecida ninguna relación con *C*;
Sea *P* el conjunto de operaciones contenidas en *B*;
Sea *p* un elemento de *P*;
Sea *Q* el conjunto de operaciones contenidas en *C*;
Sea *q* un elemento de *Q*;
Sea *ORCCA* el conjunto resultado de la intersección de *P* con *Q*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *orcca* un elemento de *ORCCA*;
ORCCA = P ∩ Q;
ORCCA ≠ {Ø};
|*ORCCA*| > 0;
Valor de calidad: 0;
El conjunto *ORCCA* tiene al menos un elemento, por lo tanto: *B* y *C* comparten al menos una operación;

Fórmula 40:

$$ORCCA = \sum_{i=1}^{n} orcca_i$$

41. Definición de atributos repetidos entre clases coasociadas

Sea *arcca* es un atributo que se encuentra repetido entre dos clases que no tienen ninguna línea de relación entre ellas, pero ambas tienen una relación de asociación con una misma tercera clase, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un atributo que se encuentra repetido entre dos clases que no tienen ninguna línea de relación entre ellas, pero ambas tienen una relación de asociación con una misma tercera clase, y que forma parte de un conjunto de elementos de un DC};
ARCCA es el conjunto de atributos que se encuentran repetidas entre dos clases que no tienen ninguna línea de relación entre ellas, pero ambas tienen una relación de asociación con una

misma tercera clase, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sean *A*, *B* y *C* clases contenidas en *C*;
Sea *R* un conjunto de relaciones contenidas en el DC;
Donde A tiene una relación de asociación con *B*;
Donde A está conectada con *B*;
Donde A tiene una relación de asociación con *C*;
Donde A está conectada con *C*;
Donde B no tiene establecida ninguna relación con *C*;
Sea *P* el conjunto de atributos contenidos en *B*;
Sea *p* un elemento de *P*;
Sea *Q* el conjunto de atributos contenidos en *C*;
Sea *q* un elemento de *Q*;
Sea *ARCCA* el conjunto resultado de la intersección de *P* con *Q*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *arcca* un elemento de *ARCCA*;
ARCCA = P ∩ Q;
ARCCA ≠ {Ø};
|*ARCCA*| > 0;
Valor de calidad: 0;
El conjunto *ARCCA* tiene al menos un elemento, por lo tanto: *B* y *C* comparten al menos un atributo;

Fórmula 41:

$$ARCCA = \sum_{i=1}^{n} arcca_i$$

42. Definición de operaciones repetidas entre 2 o más clases que se ubican en distinto árbol de herencia

Sea *ordah* es una operación que se encuentra repetida entre dos clases que no tienen ninguna conexión de relación entre ellas, y ambas clases se encuentran ubicadas en distintos árboles de herencia, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación que se encuentra repetida entre dos clases que no tienen ninguna conexión de relación entre ellas, y

ambas clases se encuentran ubicadas en distintos árboles de herencia, y que forma parte de un conjunto de elementos de un DC};
ORDAH es el conjunto de operaciones que se encuentran repetidas entre dos clases que no tienen ninguna línea de relación entre ellas, pero ambas tienen una relación de asociación con una misma tercera clase, y que forman parte de un conjunto de elementos de un DC;
Sea *C* el conjunto de clases de un DC;
Sean *A* un subconjunto de clases que pertenece a un árbol de herencia que pertenece a *C*;
Sean *B* un subconjunto de clases que pertenece a un árbol de herencia que pertenece a *C*;
Sea *P* una clase que forma parte del conjunto *A*;
Sea *p* un elemento de *P*;
Sea *Q* una clase que forma parte del conjunto *B*;
Sea *q* un elemento de *Q*;
Donde P no está conectada con *Q*;
Sea *E* el subconjunto formado por las operaciones de la clase *P*;
Sea e un elemento de *E*;
Sea *F* el subconjunto formado por las operaciones de *Q*;
Sea *f* un elemento de *F*;
Sea *ORDAH* el conjunto formado por la intersección de *E* con *F*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea ordah un elemento de *ORDAH*;
ORDAH = E ∩ F;
ORDAH ≠ {Ø};
|*ORDAH*| > 0;
Valor de calidad: 0;
El conjunto *ORDAH* tiene al menos un elemento, por lo tanto: *P* y *Q* comparten al menos una operación;

Fórmula 42:

$$ORDAH = \sum_{i=1}^{n} ordah_i$$

43. Definición de párrafos de texto con disposición distinta a horizontal

Sea *pth* un párrafo de texto contenido en una clase y que tiene disposición distinta a horizontal, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {Todos los elementos de la notación UML contenidos en un diagrama de clases};
PTH es el conjunto de párrafos de texto contenidos en clases y que tienen disposición distinta a horizontal, y que forman parte de un conjunto de elementos de un DC;
U = {X | X es un párrafo de texto contenido en una clase y que tiene disposición distinta a horizontal, y que forma parte de un conjunto de elementos de un DC};
Sea *P* un conjunto de elementos contenidos en un DC
Sea *Q* un conjunto de párrafos de texto contenidos en *P*;
Sea *PTH* un subconjunto de *Q* que se integra por párrafos cuyo texto tiene disposición distinta a horizontal;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *pth* un elemento de *PTH*;
$PTH \neq \{\emptyset\}$;
$|PTH| > 0$;
Valor de calidad: 0;
El conjunto *PTH* tiene al menos un elemento, por lo tanto: *Q* tiene al menos un párrafo de texto con disposición distinta a horizontal;

Fórmula 43:

$$PTH = \sum_{i=1}^{n} pth_i$$

44. Definición de nombres de clase con alineación distinta a centro

Sea *ncc* es un nombre de clase que tiene alineación distinta a centro, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un nombre de clase que tiene alineación distinta a centro, y que forma parte de un conjunto de elementos de un DC};

NCC es el conjunto de nombres de clase que tienen alineación distinta a centro, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenido en un DC;
Sea *N* un conjunto de nombres de clases contenidos en *C*;
Sea *NCC* un subconjunto de N que se integra por nombres de clases cuyo texto tiene alineación distinta a centro;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *ncc* un elemento de *NCC*;
$NCC \neq \{\emptyset\}$;
$|NCC| > 0$;
Valor de calidad: 0;
El conjunto *NCC* tiene al menos un elemento, por lo tanto: *N* tiene al menos un nombre de clase con alineación distinta a centro;

Fórmula 44:

$$NCC = \sum_{i=1}^{n} ncc$$

45. Definición de atributos con alineación distinta a izquierda en el DC

Sea *aai* es un nombre de atributo que tiene alineación distinta a izquierda, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un nombre de atributo que tiene alineación distinta a izquierda, y que forma parte de un conjunto de elementos de un DC};
AAI es el conjunto de nombres de atributos que tienen alineación distinta a izquierda, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *A* un conjunto de atributos contenidos en las clases que forman parte de *C*;
Sea *AAI* un subconjunto de *A* que se integra por nombres de atributos cuyo texto tiene alineación distinta a izquierda;
i: es el punto de partida de elementos a contar;

n: es el límite de elementos a contar;
Sea *aai* un elemento de *AAI*;
AAI ≠ {Ø};
|*AAI*| > 0;
Valor de calidad: 0;
El conjunto *AAI* tiene al menos un elemento, por lo tanto: *A* tiene al menos un atributo de texto con alineación distinta a izquierda;

Fórmula 45:

$$AAI = \sum_{i=1}^{n} aai_i$$

46. Definición de operaciones con alineación distinta a izquierda en el DC

Sea *oai* es un nombre de operación que tiene alineación distinta a izquierda, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un nombre de operación que tiene alineación distinta a izquierda, y que forma parte de un conjunto de elementos de un DC};
OAI es el conjunto de nombres de operaciones que tienen alineación distinta a izquierda, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *O* un conjunto de operaciones contenidas las clases que forman parte de *C*;
Sea *OAI* un subconjunto de *O* que se integra por nombres de operaciones cuyo texto tiene alineación distinta a izquierda;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *oai* un elemento de *OAI*;
OAI ≠ {Ø};
|*OAI*| > 0;
Valor de calidad: 0;
El conjunto *OAI* tiene al menos un elemento, por lo tanto: *O* tiene al menos una operación de texto con alineación distinta a izquierda;

Fórmula 46:

$$OAI = \sum_{i=1}^{n} oai$$

47. Definición de párrafos de texto con tamaño de letra pequeño en el DC

Sea *ptp* es un párrafo de texto que contiene letra pequeña, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un párrafo de texto que contiene letra pequeña, y que forma parte de un conjunto de elementos de un DC};
PTP es el conjunto de párrafos de texto que contienen letra pequeña, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *P* un conjunto de párrafos de texto contenidos en las clases que forman *C* y que tienen tamaño de letra más pequeño al recomendado;
Sea *PTP* un subconjunto *P* de a que se integra por párrafos de texto con tamaño de letra más pequeño al recomendado;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea ptp un elemento de *PTP*;
$PTP \neq \{\emptyset\}$;
$|PTP| > 0$;
Valor de calidad: 0;
El conjunto *PTP* tiene al menos un elemento, por lo tanto: *P* tiene al menos un párrafo de texto que contiene letra pequeña;

Fórmula 47:

$$PTP = \sum_{i=1}^{n} ptp_i$$

48. Definición de párrafos de texto con tamaño de letra grande en el DC

Sea *ptg* es un párrafo de texto que contiene letra grande, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un párrafo de texto que contiene letra grande, y que forma parte de un conjunto de elementos de un DC};
PTG es el conjunto de párrafos de texto que contienen letra grande, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *P* un conjunto de párrafos de texto contenidos en las clases que forman *C* y que tienen tamaño de letra más grande al recomendado;
Sea *PTG* un subconjunto *P* de a que se integra por párrafos de texto con tamaño de letra más grande al recomendado;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea ptg un elemento de *PTG*;
$PTG \neq \{\emptyset\}$;
$|PTG| > 0$;
Valor de calidad: 0;
El conjunto *PTG* tiene al menos un elemento, por lo tanto: *P* tiene al menos un párrafo de texto que contiene letra grande;

Fórmula 48:

$$PTG = \sum_{i=1}^{n} ptg_i$$

49. Definición de párrafos de texto con color de letra distinto al recomendado

Sea *ptc* es un párrafo de texto que contiene color de letra distinto al recomendado, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es un párrafo de texto que contiene color de letra distinto al recomendado, y que forma parte de un conjunto de elementos de un DC};

PTC es el conjunto de párrafos de texto que contienen color de letra distinto al recomendado, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *P* un conjunto de párrafos de texto contenidos en las clases que forman *C* y que tienen letra distinta al recomendado;
Sea *PTC* un subconjunto *P* de a que se integra por párrafos de texto con color de letra distinto al recomendado;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *ptc* un elemento de *PTC*;
$PTC \neq \{\emptyset\}$;
$|PTC| > 0$;
Valor de calidad: 0;
El conjunto *PTC* tiene al menos un elemento, por lo tanto, *P* tiene al menos un párrafo con color de letra distinto al recomendado;

Fórmula 49:

$$PTC = \sum_{i=1}^{n} ptc_i$$

50. Definición de clases con color de fondo distinto al recomendado

Sea *cccf* es una clase con color de fondo distinto al recomendado, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una clase con color de fondo distinto al recomendado, y que forma parte de un conjunto de elementos de un DC};
CCCF es el conjunto de clases con color de fondo distinto al recomendado, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases contenidas en un DC;
Sea *CCCF* un subconjunto C de a que se integra por clases que tienen color de fondo distinto al recomendado;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *cccf* un elemento de *CCCF*;

CCCF ≠ {Ø};
|*CCCF*| > 0;
Valor de calidad: 0;
El conjunto *CCCF* tiene al menos un elemento, por lo tanto: *C* tiene al menos una clase con color de fondo distinto al recomendado;

Fórmula 50:

$$CCCF = \sum_{i=1}^{n} cccf_i$$

51. Definición de figuras con color de borde distinto al recomendado

Sea *fcb* es una figura con color de borde distinto al recomendado, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una figura con color de borde distinto al recomendado, y que forma parte de un conjunto de elementos de un DC};
FCB es el conjunto de figuras con color de borde distinto al recomendado, y que forman parte de un conjunto de elementos de un DC;
Sea *F* un conjunto de figuras contenidas en un DC;
Sea *FCB* un subconjunto *F* de a que se integra por figuras que tienen color de borde distinto al recomendado;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *fcb* un elemento de *FCB*;
FCB ≠ {Ø};
|*FCB*| > 0;
Valor de calidad: 0;
El conjunto *FCB* tiene al menos un elemento, por lo tanto: *F* tiene al menos una figura con color de borde distinto al recomendado;

Fórmula 51:

$$FCB = \sum_{i=1}^{n} fcb_i$$

52. Definición de figuras con bordes con grosor distinto al recomendado

Sea *fgb* es una figura con grosor de borde distinto al recomendado, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una figura con grosor de borde distinto al recomendado, y que forma parte de un conjunto de elementos de un DC};
FGB es el conjunto de figuras con grosor de borde distinto al recomendado, y que forman parte de un conjunto de elementos de un DC;
Sea *F* un conjunto de figuras contenidas en un DC;
Sea *FGB* un subconjunto *F* de a que se integra por figuras que tienen borde con grosor distinto al recomendado;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *fgb* un elemento de *FGB*;
$FGB \neq \{\emptyset\}$;
$|FGB| > 0$;
Valor de calidad: 0;
El conjunto *FGB* tiene al menos un elemento, por lo tanto: F tiene al menos una figura con grosor de borde distinto al recomendado;

Fórmula 52:

$$FGB = \sum_{i=1}^{n} fgb_i$$

53. Definición de relaciones con línea punteada pero distintas a DASH

Sea *lcd* es una relación con línea punteada que es distinta a DASH, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una relación con línea punteada que es distinta a DASH, y que forma parte de un conjunto de elementos de un DC};
LCD es el conjunto de relaciones con línea punteada que son distintas a DASH, y que forman parte de un conjunto de elementos de un DC;
Sea *R* un conjunto de relaciones con línea punteada contenidas en un DC;
Sea *LCD* un subconjunto *R* de a que se integra por relaciones que tiene línea punteada distinta a DASH;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea lcd un elemento de *LCD*;
$LCD \neq \{\emptyset\}$;
$|LCD| > 0$;
Valor de calidad: 0;
El conjunto *LCD* tiene al menos un elemento, por lo tanto: *R* tiene al menos una relación con línea punteada que es distinta a DASH;

Fórmula 53:

$$LCD = \sum_{i=1}^{n} lcd_i$$

54. Definición de cruces entre líneas que representan relaciones en el DC

Sea *clr* es una línea que representa una relación que cruza con otra línea que representa una relación, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una línea que representa una relación que cruza con otra línea que representa una relación, y que forma parte de un conjunto de elementos de un DC};
CLR es el conjunto de líneas que representan relaciones que cruzan con otra línea que representa una relación, y que forman parte de un conjunto de elementos de un DC;
Sea *R* un conjunto de relaciones contenidas en un DC;
Sea *r* un elemento de *R*;
Sea *CLR* un subconjunto *R* de a que se integra por relaciones que tiene al menos un cruce con otra relación;

i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *clr* un elemento de *CLR*;
CLR ≠ {Ø};
|*CLR*| > 0;
Valor de calidad: 0;
El conjunto *CLR* tiene al menos un elemento, por lo tanto: *R* tiene al menos una línea que representa una relación que cruza con otra línea que representa una relación;

Fórmula 54:

$$CLR = \sum_{i=1}^{n} clr_i$$

55. Definición de líneas que representan relaciones y que tienen más de dos dobleces (excepto reflexivas)

Sea *lrd* es una línea que representa una relación no reflexiva y que tiene más de dos dobleces entre la clase origen y la clase destino, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una línea que representa una relación no reflexiva y que tiene más de dos dobleces entre la clase origen y la clase destino, y que forma parte de un conjunto de elementos de un DC};
LRD es el conjunto de líneas que representan relaciones no reflexivas y que tienen más de dos dobleces entre la clase origen y la clase destino, y que forman parte de un conjunto de elementos de un DC;
Sea *R* un conjunto de relaciones contenidas en un DC;
Sea *r* un elemento de *R*;
Sea *LRD* un subconjunto R de a que se integra por relaciones que tiene más de dos dobleces;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *lrd* un elemento de *LRD*;
LRD ≠ {Ø};
|*LRD*| > 0;
Valor de calidad: 0;

El conjunto *LRD* tiene al menos un elemento, por lo tanto: *R* tiene al menos una relación no reflexiva que tiene más de dos dobleces entre la clase origen y la clase destino;

Fórmula 55:

$$LRD = \sum_{i=1}^{n} lrd_i$$

56. Definición de operaciones abstractas no implementadas en el DC

Sea *oaim* es una operación definida como abstracta en una clase y que no se implementó en ninguna otra clase, y que forma parte de un conjunto de elementos de un DC;

Donde:

U = {X | X es una operación definida como abstracta en una clase y que no se implementó en ninguna otra clase, y que forma parte de un conjunto de elementos de un DC};
OAIM: es el conjunto de operaciones definidas como abstractas en clases y que no se implementaton en ninguna otra clase, y que forman parte de un conjunto de elementos de un DC;
Sea *C* un conjunto de clases en un DC;
Sea *A* un conjunto de operaciones contenidas en las clases que integran *C*;
Sea B un conjunto de operaciones abstractas contenidas en *A*;
Sea *OAIM* un subconjunto de B que se integra por operaciones abstractas que no se implementaron en al menos una clase que pertenece a *C*;
i: es el punto de partida de elementos a contar;
n: es el límite de elementos a contar;
Sea *oaim* un elemento de *OAIM*;
OAIM ≠ {Ø};
|*OAIM*| > 0;
Valor de calidad: 0;

El conjunto *OAIM* tiene al menos un elemento, por lo tanto: *A* tiene al menos una operación definida como abstracta en una clase y que no se implementó en ninguna otra clase;

Fórmula 56:

$$OAIM = \sum_{i=1}^{n} oaim_i$$

7.2. Anexo 2: Tabla de Trabajos Relacionados

No.	AÑO	PUBLICACIÓN	ENFOQUE
1	2016	IEEE	Análisis de métricas de calidad.
2	2016	Google Scholar	Análisis de métricas de calidad.
3	1998	Google Scholar	Análisis de métricas de calidad.
4	1999	Google Scholar	Análisis de métricas de calidad.
5	2001	Google Scholar	Análisis de métricas de calidad.
6	2003	IEEE	Análisis de métricas de calidad.
7	2003	IEEE	Análisis de métricas de calidad.
8	2004	Google Scholar	Análisis de métricas de calidad.
9	2004	Google Scholar	Análisis de métricas de calidad.
10	2005	IEEE	Análisis de métricas de calidad.
11	2005	IEEE	Análisis de métricas de calidad.
12	2010	IEEE	Análisis de métricas de calidad.
13	2014	Google Scholar	Análisis de métricas de calidad.
14	2014	Google Scholar	Análisis de métricas de calidad.
15	2000	ACM	Análisis de métricas de calidad.
16	2002	ACM	Análisis de métricas de calidad.
17	2001	Google Scholar	Análisis de métricas de calidad.
18	2005	Springer	Análisis de métricas de calidad.
19	2001	Google Scholar	Análisis de métricas de calidad.
20	2007	Google Scholar	Análisis de métricas de calidad.
21	2016	Google Scholar	Análisis de métricas de calidad.
22	2003	Google Scholar	Herramientas para la medición de calidad en diagramas UML.
23	2015	IEEE	Herramientas para la medición de calidad en diagramas UML.
24	2013	IEEE	Herramientas para la medición de calidad en diagramas UML.
25	2015	Google Scholar	Herramientas para la medición de calidad en diagramas UML.
26	2013	IEEE	Herramientas para la medición de calidad en diagramas UML.
27	2013	IEEE	Herramientas para la medición de calidad en diagramas UML.
28	2016	IEEE	Herramientas para la medición de calidad en diagramas UML.
29	2018	IEEE	Herramientas para la medición de calidad en diagramas UML.
30	2011	Google Scholar	Herramientas para la medición de calidad en diagramas UML.
31	2015	Google Scholar	Herramientas para la medición de calidad en diagramas UML.
32	2009	IEEE	Herramientas para la medición de calidad en diagramas UML.
33	2009	IEEE	Herramientas para la medición de calidad en diagramas UML.
34	2010	Google Scholar	Herramientas para la medición de calidad en diagramas UML.
35	2011	IEEE	Grado de entendimiento del diagrama UML.
36	2017	Google Scholar	Grado de entendimiento del diagrama UML.
37	2014	ACM	Grado de entendimiento del diagrama UML.
38	1981	IEEE	Grado de entendimiento del diagrama UML.
39	2019	Google Scholar	Grado de entendimiento del diagrama UML.

40	2006	ACM	Grado de entendimiento del diagrama UML.
41	2015	ACM	Grado de entendimiento del diagrama UML.
42	2001	ACM	Grado de entendimiento del diagrama UML.
43	2017	IEEE	Grado de entendimiento del diagrama UML.
44	2000	ACM	Grado de entendimiento del diagrama UML.
45	2016	Google Scholar	Grado de entendimiento del diagrama UML.
46	2020	IEEE	Grado de entendimiento del diagrama UML.
47	2007	Google Scholar	Grado de entendimiento del diagrama UML.
48	2015	Google Scholar	Grado de entendimiento del diagrama UML.
49	2019	Google Scholar	Grado de entendimiento del diagrama UML.
50	2011	Google Scholar	Grado de entendimiento del diagrama UML.
51	1985	IEEE	Grado de entendimiento del diagrama UML.
52	2004	IEEE	Grado de entendimiento del diagrama UML.
53	2005	Google Scholar	Grado de entendimiento del diagrama UML.
54	2006	Google Scholar	Grado de entendimiento del diagrama UML.
55	2007	IEEE	Grado de entendimiento del diagrama UML.
56	2010	Google Scholar	Grado de entendimiento del diagrama UML.
57	2015	ACM	Estética del Diagrama UML.
58	2017	Springer	Estética del Diagrama UML.
59	2002	Google Scholar	Estética del Diagrama UML.
60	2002	Google Scholar	Estética del Diagrama UML.
61	2006	IEEE	Estética del Diagrama UML.
62	2018	Google Scholar	Estética del Diagrama UML.
63	2001	Springer	Estética del Diagrama UML.
64	2016	IEEE	Estética del Diagrama UML.
65	2001	ACM	Estética del Diagrama UML.
66	2011	Google Scholar	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
67	2010	IEEE	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
68	2016	Google Scholar	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
69	2007	IEEE	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
70	2007	IEEE	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
71	2016	Google Scholar	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
72	2000	Google Scholar	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
73	2012	ACM	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
74	2018	Google Scholar	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
75	2005	IEEE	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
76	204	ACM	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
77	1999	Google Scholar	Herramientas para la Medición de Expresiones y Emociones en E-Learning.
78	2013	IEEE	Herramientas para la Medición de Expresiones y Emociones en E-Learning.

7.3. Anexo 3: Glosario de Acrónimos y Tecnicismos

ACM	Acrónimo en inglés de *Association for Computing Machinery*. Sociedad científica y educativa para educar acerca de la Computación.
ACM-DL	Acrónimo en inglés de Association for *Computing Machinery Digital Library*.
Apache-POI	Proyecto ejecutado por *Apache Software Foundation*, y anteriormente un subproyecto del Proyecto Jakarta, proporciona bibliotecas Java puras para leer y escribir archivos en formatos de *Microsoft Office*.
C#	Acrónimo en inglés del lenguaje de programación de alto nivel orientado a objetos C-Sharp de Microsoft®.
CMMI	Acrónimo en inglés de *Capability Maturity Model Integration*.
Código SMELL	Código de programación que puede tener un problema en su diseño.
DAT	Extensión de archivos que deriva de la abreviación de "datos" e indica que el archivo contiene datos que pueden estar bajo cualquier formato.
DC	Acrónimo de Diagrama de Clases.
DOO	Acrónimo de Diseño Orientado a Objetos.
E-Training	Procesos de capacitación que se llevan a cabo a través de Internet y con el uso de las TIC.
E-Learning	Procesos de enseñanza-aprendizaje que se llevan a cabo a través de Internet y con el uso de las TIC.
Google Scholar	Buscador de Google enfocado y especializado en la búsqueda de contenido y bibliografía científico-académica.
HTTP	Acrónimo en inglés de *Hypertext Transfer Protocol*.
IEEE	Acrónimo en inglés de *Institute of Electrical and Electronics Engineers*.
IS	Acrónimo de Ingeniería de Software.
Java	Lenguaje de alto nivel orientado a objetos propiedad de Oracle Corporation.
JCR	Acrónimo en inglés de Journal Citation Reports. Base de datos multidisciplinar; es una herramienta de análisis de revistas científicas de *Clarivate Analytics*.
JSON	Acrónimo en inglés de *Javascript Object Notation*.
Layout	Forma en que están distribuidos los elementos y las formas dentro de un diseño.
Metadatos	Conjunto de datos que describen la información contenida en un recurso electrónico, de archivos o de información de los mismos.
Microsoft .NET	Entorno de ejecución administrado para Windows que proporciona diversos servicios a las aplicaciones en ejecución.
MOOC	Acrónimo en inglés de *Massive Online Open Courses* o Cursos en Línea Masivos y Abiertos.
MySQL	Sistema de gestión de bases de datos relacional desarrollado bajo licencia dual: Licencia pública general/Licencia comercial por *Oracle Corporation*.

OER	Acrónimo en inglés de *Open Educational Resources* o Recursos Educativos Abiertos.
OCL	Acrónimo en inglés de *Object Constraint Language*.
OMT	Acrónimo en inglés de *Object Model Diagram*.
OWL	Acrónimo en inglés de *Web Ontology Language*.
POO	Acrónimo de Paradigma Orientado a Objetos o Programación Orientada a Objetos.
PPTX	Formato de archivo en que se almacenan presentaciones formadas por diapositivas con imágenes, texto, animaciones, audio, vídeo, efectos, gráficas, diagramas y otros elementos.
RA	Acrónimo de Recurso de Aprendizaje.
REA	Acrónimo de Recursos Educativos Abiertos.
REST	Acrónimo en inglés de *Representational State Transfer*.
RITA	Acrónimo de Revista Iberoamericana de Tecnologías del Aprendizaje. Es parte de IEEE Education Society.
Springer	Editorial global que publica libros, libros electrónicos y publicaciones científicas de revisión por pares relacionados con ciencia, tecnología y medicina.
SWA	Acrónimo de Servicios Web de Aprendizaje.
TIC	Acrónimo de Tecnologías de la Información y Comunicación
UML	Acrónimo en inglés de *Unified Modeling Language*.
UXF	Acrónimo en inglés de *UML eXchange Format*.
XML	Acrónimo en inglés de *Extensible Markup Language*.
XMI	Acrónimo en inglés de *XML Metadata Interchange*.