Como citar este artículo: O. G. Fragoso-Díaz, J. A. Sandoval-Acosta, F. J. Álvarez-Rodríguez, J. C. Rojas-Pérez and R. Santaolaya-Salgado, "Systematic Review of Quality in Class Diagrams for Software Engineering Competencies," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 17, no. 4, pp. 351-357, Nov. 2022, doi: 10.1109/RITA.2022.3217169.

Calidad en Diagramas de Clases para Generar Competencias en Ingeniería de Software, una Revisión Sistemática

Fragoso-Díaz, Olivia, *Senior Member, IEEE*. Sandoval-Acosta, José. Álvarez-Rodríguez, Francisco. Rojas-Pérez, Juan, *Member, IEEE*. Santaolaya-Salgado, René, *Senior Member, IEEE*.

***Abstract*— Class diagrams may be used as learning resources for the generation of software engineers' competencies. However, when they are open learning resources, they may lack information about the quality they contain. It may represent a drawback in the generation of the competencies since defects included in them may hinder the learning objective that generates the competencies. This work reviews 109 open class diagrams to identify the most common defects, it also analyzes some related work to identify what are the quality attributes that class diagrams should contain and the metrics used to evaluate them. The review of the class diagrams is performed based on the attributes and values proposed in the related works. As a result, 15 defects and their frequency, a total of 323 defects were identified. In addition, four Class Diagram cases are presented and explained according to the evaluation of the quality attributes.**

***Index Terms*— E-learning, Learning Resources, Systematic Review, UML Class Diagram.**

I. Introducción

El aprendizaje electrónico (E-learning) es considerado como un 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 de entregar cursos a estudiantes que no necesariamente tienen una condición de lugar y tiempo. En cuanto a la industria, el aprendizaje electrónico es una herramienta útil para el entrenamiento o capacitación de sus empleados. Existen algunas diferencias en cuanto al aprendizaje electrónico en las universidades y el aprendizaje electrónico en capacitación y entrenamiento. Sin embargo, un problema que comparten los dos ámbitos es que los recursos de aprendizaje (RA) no necesariamente cumplen los objetivos de aprendizaje para los que son usados y esto puede tener su origen en la calidad de los mismos. Para este trabajo se considera la definición de recurso de aprendizaje descrita en [1] como un instrumento para presentar y transmitir material educativo, tales como, imágenes, mapas, diagramas, vídeos, material escrito como libros y artículos científicos.

En este trabajo, se presenta una revisión de diagramas de clases UML disponibles en Internet para identificar cuáles son los defectos más comunes en los diagramas y que generan alguna afectación respecto a su calidad y, por lo tanto, pueden prevenir que se logre el objetivo de aprendizaje y la generación de la competencia deseada.

Los defectos se revisan con base en otros trabajos que identifican atributos de calidad que deben tener los diagramas de clases. Esos defectos van desde falta de legibilidad hasta faltas en la aplicación correcta del paradigma orientado a objetos. En este trabajo primero se describe el problema que se atiende, luego la metodología de la revisión de los defectos en los diagramas de clase y, finalmente, se describen los trabajos relacionados que proponen atributos y algunas métricas de calidad para los diagramas.

II. Antecedentes del Problema

De acuerdo a [2], el diseño de los recursos de aprendizaje ha sido, y sigue siendo, tema de investigación. A través del uso de la tecnología informática; y más aún, desde el advenimiento de Internet, la proliferación de recursos educativos ha ido creciendo año tras año, de ahí la relevancia de la revisión que se presenta en este trabajo.

El caso de recursos de aprendizaje abiertos como son diagramas de clase UML, es un caso que llama la atención, ya que existen muchos diagramas de clases UML que están disponibles en Internet y que pueden ser usados para enseñanza en las universidades y en entrenamiento en el lugar de trabajo, en particular en la formación de Ingenieros de

Fragoso-Díaz, Olivia is with the *Tecnológico Nacional de México/Centro Nacional de Investigación y Desarrollo Tecnológico (TecNM/Cenidet)*. Cuernavaca, Mor. Mex. (mail: Olivia.fd@cenidet.tecnm.mx).

Sandoval-Acosta, Jose is with the *Tecnológico Nacional de México/Instituto Tecnológico Superior de Guasave (TecNM/ITSG)*. Guasave, Sin. Mex. (mail: jose.sa@guasave.tecnm.mx).

Álvarez-Rodríguez, Francisco is whit the *Univesidad Autónoma de Aguascalientes (UAA)*. Aguascalientes, Ags. Mex. (mail: fjalvar.uaa@gmail.com).

Rojas-Pérez, Juan is with the *Tecnológico Nacional de México/Centro Nacional de Investigación y Desarrollo Tecnológico* (*TecNM/Cenidet*). Cuernavaca, Mor. Mex. (mail: juan.rp@cenidet.tecnm.mx).

Santaolaya-Salgado, René is with the *Tecnológico Nacional de México/Centro Nacional de Investigación y Desarrollo Tecnológico* (*TecNM/Cenidet*). Cuernavaca, Mor. Mex. (mail: rene.ss@cenidet.tecnm.mx).

TABLA III
CANTIDAD DE DEFECTOS ENCONTRADOS EN LOS DC POR ATRIBUTO DE CALIDAD

Atributo de Calidad	Cantidad	Descripción
Legibilidad	49	Nombres, atributos y operaciones con letra muy pequeña o ilegible.
Abstracción	28	Atributos u operaciones repetidos entre clases que no forman parte de una misma jerarquía de herencia.
Redundancia	42	Atributos u operaciones repetidos entre clases de una misma jerarquía de herencia.
Rel. Incompletas	2	La relación no surge o no llega hasta una clase en particular.
Rel. De Asociación	13	Relación o dirección equivocada.
Rel. De Herencia	43	Relación o dirección equivocada.
Rel. De Agregación	2	Relación o dirección equivocada.
Rel. De Composición	5	Relación o dirección equivocada.
Rel. De Realización	11	Relación o dirección equivocada.
Rel. De Dependencia	3	Relación o dirección equivocada.
Rel. Desconocida	1	Relación no reconocida por el estándar UML.
Cruces de Líneas	37	Una línea de relación cruza con otra.

Internet, cuáles son los defectos más comunes en los diagramas y que generan alguna afectación respecto a su calidad. Los defectos se revisan con base en otros trabajos que identifican atributos de calidad que deben tener los diagramas de clases UML. Esos defectos van desde falta de legibilidad hasta faltas en la aplicación correcta de la teoría del paradigma orientado a objetos. Aquí primero se describe la metodología de la revisión de los defectos en los diagramas de clase y luego se describen los trabajos relacionados que proponen atributos y algunas métricas de calidad para los diagramas.

III. METODOLOGÍA

La metodología de la revisión se basa en la propuesta de [3], y consistió de seis actividades generales:

1) Identificación del objeto del cual se requería obtener información: en este caso son diagramas de clases (DC) de UML de libre acceso, es decir, que se encuentran en Internet y disponibles para su descarga.
2) Uso de palabras clave para búsqueda de DC: se buscaron empleando palabras o cadenas de búsqueda claves, siendo las más utilizadas: “UML Class Diagram”, “Crossing lines Class Diagram” y “Diagrama de clases UML”. Utilizando estas cadenas de búsqueda se consiguió un conjunto muy grande de DC UML, aunque algunos de los DC estaban duplicados y como resultado se decidió en primera instancia revisar 145 casos de DC.
3) Criterios de inclusión y exclusión: Una vez que se seleccionaron las palabras o cadenas de búsqueda se

TABLA II
CRITERIOS DE EXCLUSIÓN DE LOS DC

Criterio
DC que no presentan claramente nombre o descripción de su proceso por lo que su interpretación se vuelve difícil.
DC que no son legibles por lo que su interpretación es imposible.
Aun cuando la falta de legibilidad en estos es en sí un gran defecto se decidió dejarlos fuera porque no permitían obtener más información sobre otros defectos.
DC que presentan diagramas UML muy grandes más de 30 clases o que tiene muchos elementos para que sean interpretados correctamente por un aprendiz de IS.
DC que presentan diagramas que contienen pocas clases (menos de 4 clases). Porque pueden no contener suficiente información para interpretar correctamente el diagrama.
DC que no están disponibles para su descarga.

4) Análisis de los DC. A partir de la aplicación de los criterios de inclusión/ exclusión, la cantidad de DC revisados se redujo de 145 a 109. Asimismo, se identificaron defectos tales como: inconsistencias, falta de legibilidad, falta de abstracción, cruces de líneas, uso excesivo de dobleces en líneas que representan relaciones, entre otros. En la Tabla III, se listan los 15 tipos de defectos y su frecuencia, 323 en total, identificados en los 109 DC analizados, siendo el uso de colores no acorde a recomendaciones de UML [4], el defecto que más veces se presenta en los DC.
5) Para la búsqueda de trabajos relacionados se utilizaron las cadenas: “Measurement and estimation for UML class diagram quality”, “UML class diagram quality assessment” y “Class diagram design quality”. Las fuentes o bases de datos científicas son: ACM, IEEE Xplore y Springer.
6) Preguntas de investigación:
   P1: ¿Cuáles son los defectos que los DC UML presentan con mayor frecuencia?
   P2: ¿Cuáles son los enfoques que atienden los autores en los trabajos relacionados?

IV. RESULTADOS

Respondiendo a la pregunta P1, el análisis de 109 DC permitió encontrar una serie de defectos que pueden prevenir el entendimiento de los mismos.

En la Tabla III, se enlistan los 15 tipos de defectos y su frecuencia, haciendo un total de 323, con un promedio de defectos encontrados de un 2.96 ocurrencias por cada DC, siendo el uso de colores no acorde a recomendaciones de [4], el defecto que más veces se presenta en los DC. Asimismo, la redundancia de atributos y operaciones y la legibilidad de los diagramas presentan gran cantidad de ocurrencias.

Se ha tenido como limitante el hecho de que los DC con pobre legibilidad son muy difíciles de interpretar por lo que debieron ser descartados. Otra limitante que impide la correcta interpretación de los DC es que en muchos casos no contienen anotaciones que expliquen a grandes rasgos su funcionalidad.

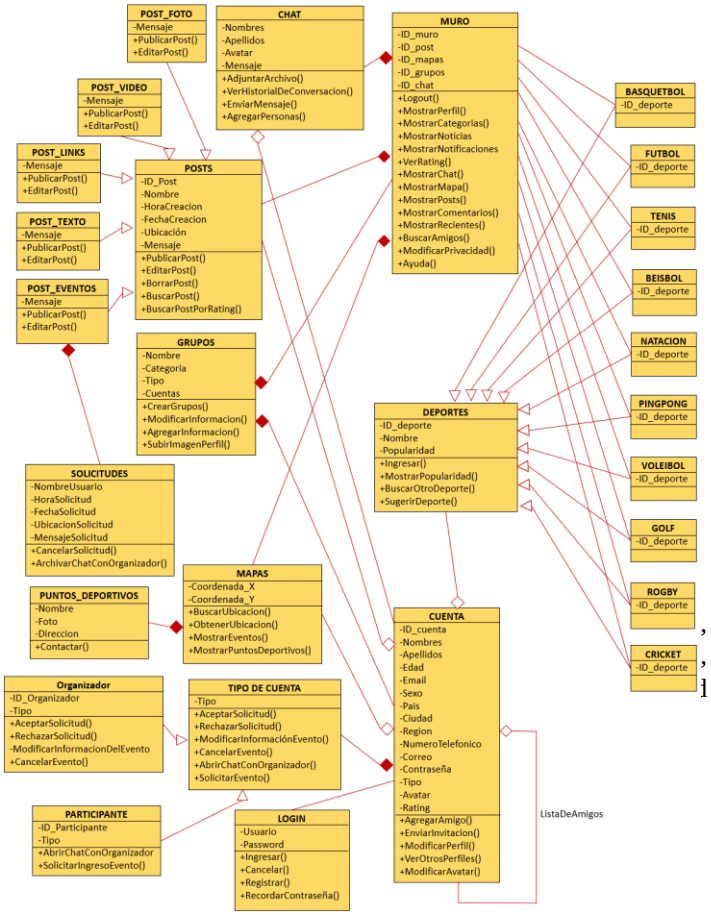

Fig. 4. Modelo de Clases de una Red Social Deportiva

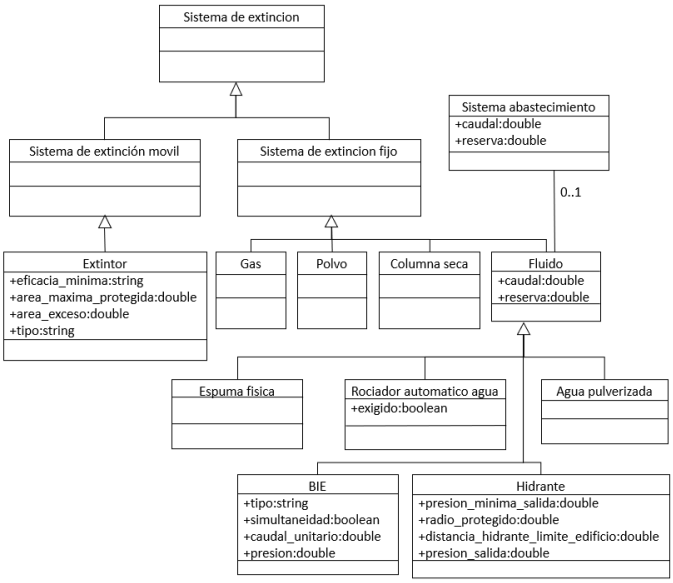

Fig. 2. Solución Propuesta al Diagrama de Sistema de Extinción

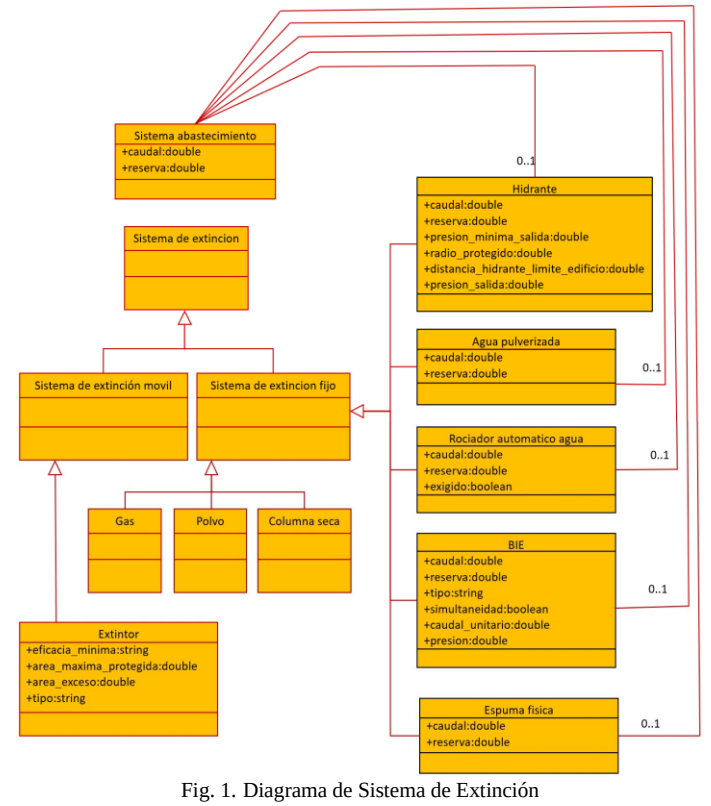

Fig. 1. Diagrama de Sistema de Extinción

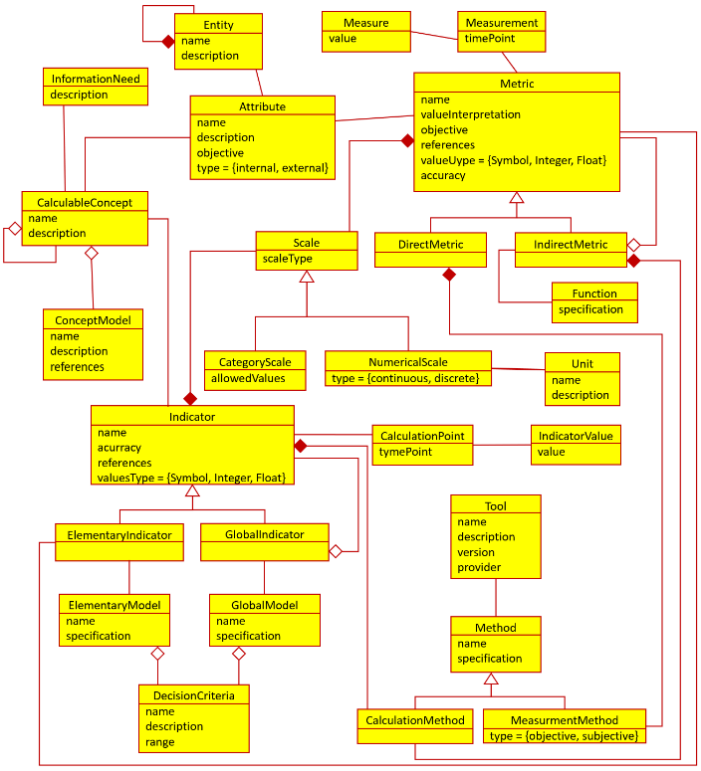

Fig. 3. Modelo de Diseño de una Métrica

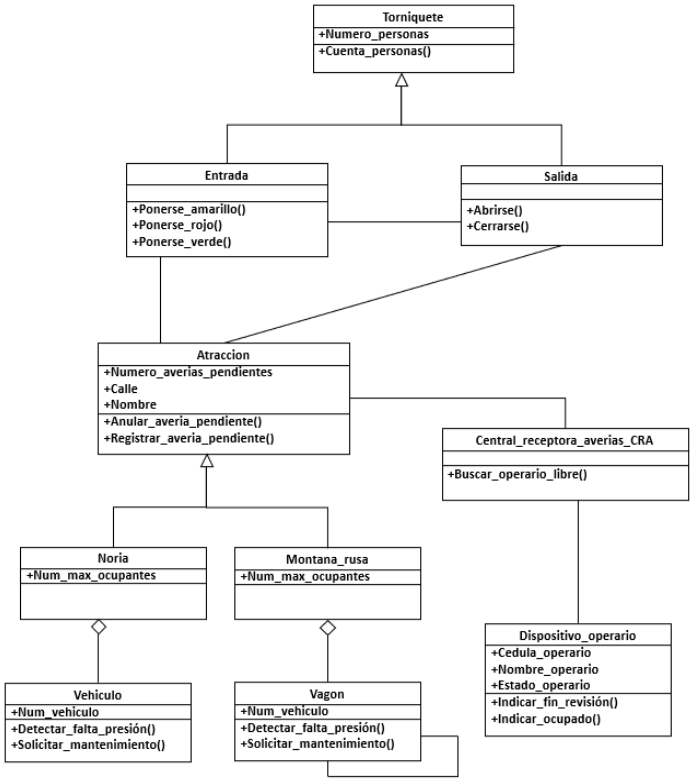

Fig. 5. Modelo de Clases de la Gestión del Parque

En el diagrama también se representan las clases con color de fondo que no es recomendable de acuerdo a [4]. En [6], se recomienda minimizar las "esquinas en escuadra" de las relaciones, debido a que es más fácil seguir líneas rectas que seguir líneas que van cambiando de dirección.

Para efectos de la revisión realizada se consideraron aquellos RA que presentan relaciones que tienen más de dos dobleces. Para este caso, la solución propuesta consiste en agregar una abstracción o superclase llamada "Fluido" y heredar las características involucradas a las subclases: "Hidrante", "Agua pulverizada", "Rociador automático agua", "BIE" y "Espuma física" correspondientes, como se muestra en la Figura 2.

Caso 2: Modelo del diseño de una métrica. En la Figura 3 [7] Se puede ver que existe repetición en los atributos de las clases "CalculableConcept", "ConceptModel", "Unit", "Tool", "Method", "ElementaryModel" y "GlobaModel". De igual manera se visualizan relaciones de gran longitud y múltiples dobleces en escuadra, las cuales pueden generar confusión al momento de interpretar el diagrama.

Asimismo, el diagrama presenta colores como rojo, morado y amarillo en bordes y fondo, lo cual contraviene la recomendación de [4] en representar los diagramas en color negro con fondo blanco.

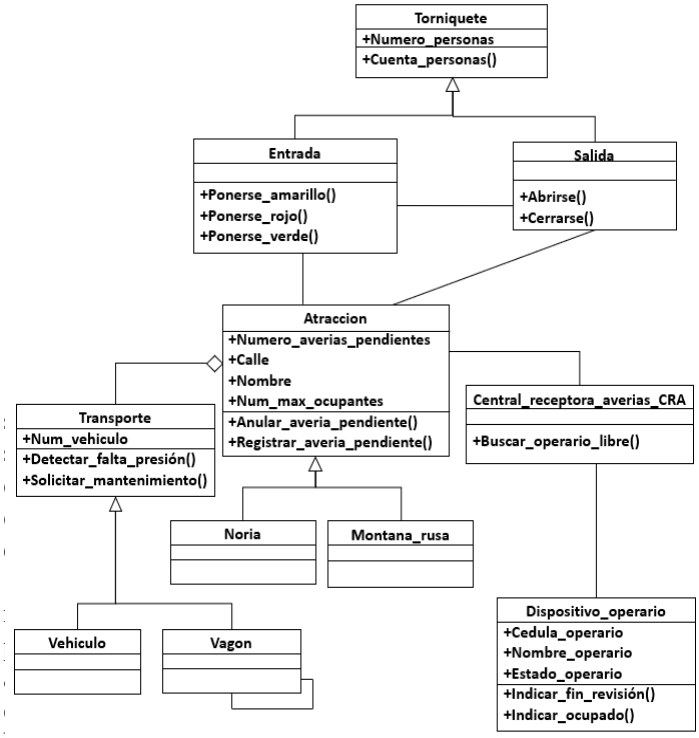

Fig. 6. Modelo Propuesto de Clases para la Gestión del Parque

recomendación de [10], 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. En este caso de DC, es importante asegurarse que el sistema que se empleó para construir el diagrama permita ser configurado para que los tipos de línea puedan dibujarse de forma diferente de tal manera que se minimicen los cruces.

Caso 4: Modelo de Clases de la Gestión de un Parque. Este ejemplo muestra en la Figura 5 [11], el diagrama para la gestión de un parque de diversiones. Como ejemplo, se detecta en el mismo que no hay una abstracción que corresponda a una clase padre referente a las clases de medios de transporte "Vehiculo" y "Vagon", provocando que exista redundancia en los atributos; por lo que la solución propuesta es agregar la abstracción correspondiente creando la clase "Transporte", la cual, contiene los atributos comunes de los medios de transporte en las atracciones, como se muestra en la Figura 6, heredando de esta forma los atributos y operaciones comunes.

En la Figura 5 se observan atributos y operaciones que son comunes en dos clases "Vehiculo" y "Vagon" que no están relacionadas entre sí. Además, las clases "Noria" y "Montana_rusa" tiene el atributo "Num_max_ocupantes" en común y ambas son subclases de la clase "Atraccion" por lo que puede decirse que están repitiendo un atributo que podría ser heredado por la clase padre.

La solución propuesta para esto es colocar dichos atributos y operaciones en una nueva clase abstracta que herede esas características comunes a las clases "Vehiculo" y "Vagon". Además, el atributo "*Num_max_ocupantes*" puede ser colocado en la clase "Atraccion" para ser heredado a todas sus subclases como se muestra en la Figura 6.

V. Trabajos Relacionados

Respondiendo a la pregunta P2, los trabajos analizados se clasifican en cuatro enfoques: A) Estética de los DC UML; B) Grado de Entendimiento; C) Métricas para la Medición de la Calidad del DC UML; y D) Herramientas para la Medición de la Calidad del DC UML.

La Tabla IV, ubica a los trabajos relacionados dentro de los enfoques identificados, el recuadro verde indica que el trabajo at
at

Tabla IV
Enfoques Detectados en los Trabajos

Artículo	Estética de los Diagramas de clases UML	Grado de Entendimiento del Diagrama de Clases UML	Métricas para la Medición de la Calidad del Diagrama de Clases UML	Herramientas para la Medición de la Calidad del Diagrama de Clases UML
[9], [12]	☑	☒	☒	☒
[13], [14]	☒	☑	☒	☒
[16], [19], [20], [21], [22], [23]	☒	☒	☑	☒
[24], [25], [26], [27]	☒	☒	☒	☑

En el segundo de los trabajos [9], los autores presentan un enfoque en el que el tamaño del DC UML es lo fundamental respecto a su entendimiento. La hipótesis planteada se refiere que entre más grande es el diagrama el individuo que lo modela tiene un desempeño más pobre, mientras que un diagrama pequeño es más fácil de usar e interpretar independientemente de su calidad.

Los autores proponen 4 principios para la construcción de los diagramas de clase, como son: 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.

B. Grado de Entendimiento del Diagrama de Clases UML.

En [13], los autores realizan una revisión del estado del arte, en la que muestran un estudio enfocado principalmente en la comprensión de los modelos de procesos de software y analizan diferentes tipos de diagramas y aspectos de UML. Los autores concluyen que se requieren más estudios para madurar el conocimiento sobre este tema.

En [14], los autores presentan un estudio enfocado al entendimiento del DC UML principalmente desde un punto de vista semántico; realizan revisiones a través de una herramienta llamada *SDMetrics*® [15], y por medio de un diccionario de datos que fue creado por expertos. La valoración va de 0 a 1, siendo para este estudio el valor cero una valoración pobre, y el valor 1 una valoración excelente. Los autores concluyen que deben realizar estudios posteriores para validar todas las hipótesis.

C. Métricas para la Medición de la Calidad del Diagrama de Clases UML.

En [16], los autores presentan un trabajo en el cual hacen una revisión de varios grupos de métricas propuestas por [17] y [18]. Tiene como objetivo proponer algoritmos para hacer el cálculo de las métricas basándose en el mapeo de los metadatos de archivos XML del DC UML. Los autores presentan los algoritmos para realizar los cálculos y mediciones de las métricas con base en los datos extraídos de los DC;

incorporando dichos algoritmos a una herramienta mencionada como *UML Refactoring*.

En [19], los autores se enfocan en analizar los Recursos Educativos Abiertos (REA) que benefician a la Ingeniería de Software IS. Mencionan tanto los REA físicos como los electrónicos; enfocándose principalmente en los recursos electrónicos como lo son: vídeos; y cursos abiertos masivos en línea y juegos multimedia. Su principal búsqueda es para identificar recursos que puedan ayudar a mejorar el aprendizaje de IS estableciendo criterios para encontrar recursos relacionados con: requerimientos de software; gestión de proyectos; pruebas y refactorización. Se establecen criterios de valoración de los REA, como lo son: calidad del contenido; usabilidad y valor educativo, sin mencionar la forma de evaluar esos atributos.

En [20], los autores tienen como objetivo mejorar la calidad de los DC UML a través del uso de Algoritmos Genéticos. Se utiliza una métrica de acoplamiento entre objetos para medir el grado de dependencia de los componentes del sistema. Se enfoca principalmente en el DC UML asignándole clasificaciones a las clases y estableciendo sus relaciones por medio de algoritmos genéticos. Como resultado, se demuestra que los algoritmos genéticos tienen potencial para evaluar el DC UML; sin embargo, los autores mencionan que se requieren más estudios al respecto.

En [21], los autores presentan un análisis de 22 diagramas UML en el cual se miden 11 atributos o métricas de calidad como: número de atributos, número de clases hijas, número de relaciones y número de jerarquías de generalización. Los autores concluyen que en futuros trabajos se requiere incrementar el tamaño de los DC, la variedad en tipos de casos, así como trabajar con profesionales de IS para una mejor validación de los resultados.

Similarmente a [16]-[21], en [22] Los autores presentan un estudio que tiene como objetivo realizar un análisis de métricas para el desarrollo de DC UML en el modelo orientado a objetos. Los autores realizan la revisión de un conjunto de métricas y sus relaciones encontradas en el DC, aunque la revisión se valida con base en código. El estudio se justifica en el análisis de métricas de software con patrones de dependencia para proveer una interpretación de cada una de estas dependencias.

En [23], 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 tipos de diagramas UML, el tema principal es el DC. Los autores 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 que los diagramas estén libres de defectos, con base en integridad, consistencia, y simetría.

D. Herramientas para la Medición de la Calidad del Diagrama de Clases UML.

El trabajo [24] presenta la descripción de una serie reglas y herramientas para IS particularmente para el DC UML. Se enfoca principalmente en el DC, y menciona el enfoque que tienen las herramientas CASE fuertemente hacia la implementación no hacia los diagramas visuales. Enumera una serie de reglas respecto al *layout* de los DC y también sobre el dibujado de los DC y la estética de los mismos. Se revisan diferentes herramientas entre ellas librerías de Java y gráficas. El artículo aporta información respecto a las reglas actuales que se usan en la representación de los DC UML y algunas herramientas para trabajarlos, pero enfatiza que estas herramientas no consideran todos los aspectos del diagrama.

En [25], los autores en este trabajo presentan un análisis entre dos tipos de diagramas del UML, DC 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 de los autores es desarrollar un algoritmo y la herramienta correspondiente. Para ello, los autores establecen reglas que deben contener los diagramas tanto de clase como de secuencia los cuales están desarrollados en *Enterprise Architect* y son exportados a formato XML para su posterior análisis; el cual, se realiza por medio de una herramienta de software.

En [26], aunque los autores plantean el grado de entendimiento del DC UML como uno de los atributos a evaluar, su propuesta principal es una herramienta para recolección de la información referente a las métricas sobre el entendimiento de DC, pero sin hacer mención de las técnicas utilizadas para recolectar y procesar la información.

El trabajo [27] muestra una herramienta llamada *TUPUX*. 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 el resultado que los estudiantes identifican incorrectamente relaciones entre clases resultando en un cálculo incorrecto de los puntos función. El autor considera realizar futuros trabajos con casos de estudio distintos para obtener resultados más precisos respecto a la aplicabilidad del modelo en la industria.

VI. CONCLUSIONES Y TRABAJO FUTURO

Los procesos de desarrollo de software tienen la inherente característica de ser procesos difíciles, derivado de que el principal producto del software es intangible. Por lo tanto, es necesario orientar esfuerzos hacia facilitar el logro del aprendizaje de los elementos de los procesos de desarrollo, de los cuales, la construcción de DC en las fases de diseño de cualquier modelo de proceso es una actividad que requiere del conocimiento del paradigma orientado a objetos y de suficiente creatividad para poder aplicar el paradigma en la generación de una solución a un problema que se quiere atender.

El aprendizaje de la actividad de diseño de software puede verse inhibido derivado de la calidad de los recursos de aprendizaje que se emplean y por lo tanto previenen la generación de la competencia de un Ingeniero de Software con el rol de diseñador de DC. La revisión que aquí se presenta pone en evidencia que los recursos de aprendizaje llamados diagramas de clases abiertos, bajo los criterios de inclusión y exclusión, contienen diversos errores que van desde la forma,

por ejemplo, las líneas, colores, tipos de texto, etc., hasta el fondo; es decir, no se cumple con la teoría y principios del paradigma orientado a objetos, de tal manera que cuando alguien quiere aprender con estos recursos de aprendizaje, posiblemente tendrá un mal o incompleto aprendizaje previniendo la formación de las competencias de los Ingenieros de Software. Muchas veces, un tipo de defecto puede llevar a otro tipo de defecto. Por ejemplo, un problema de aplicar mal el concepto de abstracción puede llevar a un problema de legibilidad, y ese mismo, puede llevar a un problema de comprensibilidad, y así, se puede tener una secuencia de defectos.

Los cuatro casos aquí analizados pueden tener más defectos que los que se indican en su sección, sin embargo, no se corrigen todos los defectos ya que sólo se pretende mostrar algunos ejemplos. Los trabajos descritos en la sección IV se tomaron como base para establecer lo que puede ser un defecto en diagramas de clase.

Tomar en cuenta la simplicidad como un atributo que debe existir, posiblemente reduciría la cantidad de defectos de los diagramas, pero diseñar con simplicidad es una competencia difícil de obtener.

A partir del análisis realizado se visualiza que no se toman en cuenta algunos principios de modelado conceptual que indican que un modelo solo presente lo necesario para ser entendido.

Adicionalmente, a partir de los 109 casos estudiados se puede sugerir que una buena práctica es poner o identificar la clase cliente en un diagrama.

Asimismo, un defecto muy sencillo resultó en algo muy importante para facilitar el entendimiento de la solución que representa el DC, es la falta de anotaciones en la gran mayoría de los DC que faciliten el análisis y la comprensión de su propuesta de diseño.

Atendiendo al problema descrito en la sección II, como trabajo futuro se reconoce la necesidad de buscar un modelo de calidad en donde se definan los atributos de calidad requeridos en los DC complementarios a los ya propuestos en los trabajos relacionados, se defina la forma de medirlos, se desarrollen las herramientas necesarias para realizar automáticamente la evaluación de los DC sin depender de la presencia de expertos en el tema, y en su caso realizar la refactorización de los DC y realizar las correcciones en la etapa de diseño y no cuando ya se ha pasado a etapas posteriores del desarrollo de software. Muy notable es que algunos trabajos relacionados concluyen sobre la necesidad de extender los estudios en el tema.

Adicionalmente, se puede mencionar que las diferencias en las notaciones de los diagramas hacen difícil el entendimiento de los DC, por lo que se sugiere se establezcan equivalencias y estas puedan ser usadas en todas las herramientas de desarrollo de DC.

Finalmente, se concluye que este trabajo puede ser una guía en la selección de recursos abiertos para formación de los Ingenieros de Software, no solamente de DC sino recursos de otro tipo.

AGRADECIMIENTOS

Queremos agradecer en primer lugar al Concejo Nacional de Ciencia y Tecnología (CONACYT), y también, al Tecnológico Nacional de México (TecNM) Campus Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) por su invaluable apoyo para este trabajo.

REFERENCIAS

[1] R. Bušljeta, “Effective Use of Teaching and Learning Resources,” *Czech-Polish Historical and Pedagogical Journal*, vol. 5, pp. 55-70, 2013.

[2] J. L. Navarro, “Objetos de aprendizaje : formación de autores con el modelo redes de objetos.,” 2005.

[3] B. Kitchenham y S. Charters, “Guidelines for Performing Systematic Literature Reviews in Software Engineering,” Keele University and Durham University. EBSE Technical Report, Durham, 2007.

[4] Object Management Group, OMG Unified Modeling Language, 2017.

[5] Instituto Nacional de Seguridad e Higiene en el Trabajo, Automatización del Reglamento de Seguridad Contra Incendios en Establecimientos Industriales, Madrid, 2004.

[6] K. Sugiyama, S. Tagawa y M. Toda, “Methods for Visual Understanding of Hierarchical System Structures,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, pp. 109-125, 1981.

[7] M. d. l. Á. Martín, Sistema de Catalogación de Métricas e Indicadores con Potencia de Web Semántica, 2004.

[8] Blogspot, “Diagrama de una red social deportiva,” Junio 2012. [En línea]. Available: http://rsdplus.blogspot.com/2013/06/diagrama-de-clases-rsdplus.html. [Último acceso: Marzo 2021].

[9] 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.

[10] C. Batini, L. Furlani y E. Nardelli, What is a Good Diagram? A Pragmatic Approach., Chicago, Illinois.: IEEE Computer Society and North-Holland, 1985, pp. 312-319.

[11] U. d. Cauca, *Programa de Ingeniería de Sistemas – Ingeniería de Software I. Diagrama de Clases*, Cauca, 2012, p. 8.

[12] H. Eichelberger y k. Schmid, “Guidelines on the aesthetic quality of UML class diagrams,” *Information and Software Technology*, vol. 5, nº 12, pp. 1686-1698, 2009.

[13] A. Dikici, “Factors Influencing the Understandability of Process Models: A Systematic Literature Review,” *Information and Software Technology*, 2017.

[14] Y. Nakamura, K. Sakamoto, K. Inoue, H. Washizaki y Y. Fukazawa, “Evaluation of Understandability of UML Class Diagrams by Using Word Similarity,” pp. 178-187, 2011.

[15] SDMetrics, “SDMetrics,” [En línea]. Available: https://www.sdmetrics.com/. [Último acceso: Mar 2021].

[16] 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, nº 4, 2018.

[17] S. R. Chidamber y C. F. Kemerer, “A Metrics Suit for Object Oriented Design,” *IEEE Transactions on Software Engineering*, vol. 20, pp. 476-493, 1994.

[18] F. A. Brito, L. Ochoa y M. Goulao, “The MOOD metrics set,” *INESC/ISEG*, 1998.

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

[20] O. Deryugina, “Improving the structural quality of UML class diagrams with the genetic algorithm,” *ITM Web of Conferences*, 2016.

[21] 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.

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

[23] M. Sharma y R. Vishwakarma, CMMI Based Software Metrics to Evaluate OOAD, 2012.

[24] H. Eichelberger y J. Wolff, "UML Class Diagrams - State of the Art in Layout Techniques," *VISSOFT,* 2003.

[25] E. Ekanayake y S. Kodituwakku, "Consistency Checking of UML Class and Sequence Diagrams," *International Conference on Ubi-Media Computing (UMEDIA),* pp. 098-103, 2015.

[26] S. Rajesh y A. Chandrasekar, "Metrics measurement model: To measure the object oriented design metrics," *2015 Seventh International Conference on Advanced Computing (ICoAC),* pp. 1-6, 2015.

[27] 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,* pp. 126-134, 2013.

Olivia Gaciela Fragoso Díaz. Obtuvo el grado de Doctora en Ciencias de la Computación 2012 en el Centro Nacional de Investigación y Desarrollo Tecnológico (TECNM/CENIDET), México. Desde septiembre de 1995 es investigadora en el área de Ingeniería de Software en TECNM/CENIDET. Actualmente sus áreas de investigación son: Ingeniería de Software, Tecnologías de Software para E-learning, Reusabilidad del Software, Clasificación y Recuperación de Servicios Web, Calidad de Software y Procesos de Software. Es miembro del Sistema Nacional de Investigadores nivel I y es Senior Member del IEEE desde 2004.

José Antonio Sandoval Acosta. Obtuvo el grado de maestro en Sistemas de Información en la UACH, Campus Cd. Juárez, México en el 2010. Profesor del Instituto Tecnológico Superior de Guasave (TECNM/ITSG). Está matriculado en el programa de doctorado en Ciencias de la Computación en Centro Nacional de Investigación y Desarrollo Tecnológico (TECNM/CENIDET), México. Sus áreas de interés incluyen tecnologías de software para E-learning, Arquitectura Orientada a Servicios y Calidad en Software.

Francisco Javier Alvarez Rodríguez. Doctor en Ingeniería por la UNAM (México). Profesor de Ingeniería de Software adscrito al Departamento de Ciencias de la Computación, Universidad Autónoma de Aguascalientes (U.A.A.). Miembro de núcleos académicos de diversos posgrados de la U.A.A. Es autor de libros y artículos sobre la línea Objetos de Aprendizaje y Procesos de Desarrollo de Software. Actualmente es presidente del Consejo Nacional de Acreditación de programas de Informática y Computación, A. C.

Juan Carlos Rojas Pérez. Doctor en Ciencias de la Computación por el Centro Nacional de Investigación y Desarrollo Tecnológico (TECNM/CENIDET), México. Sus áreas de interés son: Patrones de Diseño, Inteligencia de Negocios, Bases de Datos Aplicadas a Ingeniería de Software, Procesamiento de Lenguaje Natural e Ingeniería de Software Aplicada al Big Data. Es Member del IEEE desde 2016.

René Santaolaya Salgado. Doctor en Ciencias de la Computación por el Centro de Investigación en Computación del Instituto Politécnico Nacional (CIC-IPM), México. Actualmente es investigador del Centro Nacional de Investigación y Desarrollo Tecnológico (TECNM/CENIDET), México. Su área de interés es la Ingeniería de Software, específicamente en: Modelos de Procesos de Software, Reingeniería de Software Legado, Reusabilidad de Software, Arquitecturas de Software, Arquitecturas Orientadas a Servicios, Servicios Web, Microservicios y Aplicaciones Novedosas de TI. Es Senior Member del IEEE desde 2004.