Systematic Review of Quality in Class Diagrams for Software Engineering Competencies

Olivia Graciela Fragoso-Díaz, *Senior Member, IEEE*, José Antonio Sandoval-Acosta, Francisco Javier Álvarez-Rodríguez, Juan Carlos Rojas-Pérez, *Senior Member, IEEE*, and René Santaolaya-Salgado, *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 could lack information about the quality they contain. It may represent a drawback in the generation of the competencies since defects included in them can 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 must exist in class diagrams 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 in Class Diagrams 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, UML class diagram, systematic review.**

I. Introduction

NOWADAYS, electronic learning or E-learning is considered as a means that can solve education and training needs in different areas. In universities, it is frequently used because it offers facilities to deliver courses to students who do not necessarily have a condition of place and time. As for the industry, e-learning is a useful tool for training or empowering their employees. There are some differences when it comes to e-learning in universities and e-learning in training and coaching. However, a problem shared by the two areas is that learning resources (LR) do not necessarily meet the learning objectives for which they are used and it may have its origin in their quality. This work considers the definition of LR as described in [1], which defines it as an instrument to present and transmit educational material, such as images, maps, diagrams, videos, and written material such as books and scientific articles.

Manuscript received 16 March 2021; revised 16 June 2021 and 25 October 2021; accepted 29 December 2021. Date of publication 26 October 2022; date of current version 17 November 2022. This work was supported by CONACYT through a PhD Scholarship. *(Corresponding author: José Antonio Sandoval-Acosta.)*

Olivia Graciela Fragoso-Díaz, Juan Carlos Rojas-Pérez, and René Santaolaya-Salgado are with the Tecnológico Nacional de México/Cenidet, Cuernavaca, Morelos 62490, Mexico (e-mail: olivia.fd@cenidet.tecnm.mx; juan.rp@cenidet.tecnm.mx; rene.ss@cenidet.tecnm.mx).

José Antonio Sandoval-Acosta is with the Tecnológico Nacional de México/Guasave, Guasave, Sinaloa 81249, Mexico (e-mail: jose.sa@guasave.tecnm.mx).

Francisco Javier Álvarez-Rodríguez is with the Computer Science Department, Aguascalientes Autonomous University, Aguascalientes 20100, Mexico (e-mail: fjalvar@correo.uaa.mx).

A Spanish version of this article is available as supplementary material at https://doi.org/10.1109/RITA.2022.3217169.

Digital Object Identifier 10.1109/RITA.2022.3217169

In this paper, a review of Unified Modeling Language (UML) class diagrams (CD) available on the Internet is described to identify which are the most common defects in the diagrams and which generate some affectation regarding their quality and, therefore, the learning objectives and the desired competences may not be achieved. Defects are reviewed based on other related work that identifies the quality attributes that CD should have. Those defects span from a lack of readability to faults in the correct application of the object-oriented paradigm. This work first describes the problem being addressed, then shows the methodology for the review of defects in the CD and finally, the related works that propose attributes and some quality metrics for the diagrams are described.

II. Background and Problem Description

According to [2], the design of learning resources has been a subject of research. Through the use of computer technology and even more since the advent of the Internet, the proliferation of educational resources is growing year after year, hence the relevance of the review described here.

Open learning resources such as UML CD is a case that draws attention, since there are many UML CD that are available on the Internet and that can be used for teaching in universities and in training in the workplace. This is particularly applicable in the training of Software Engineers who also seek to achieve competencies to meet the demands of employers. However, the diagrams can have one or more defects and those defects that can prevent both the learning objective and Software Engineer's competencies from being achieved. The ultimate goal of this work is to use the information on the most common defects identified to define a quality model for CD, in which quality attributes and a way of measuring them are identified to offer quantitative information to users of the CD. This allows the use of the quality resources used in the teaching-learning processes.

CD defects are reviewed based on other works that identify quality attributes that UML CD should have.

Those defects range from a lack of readability to failures in the correct application of the object-oriented paradigm theory. Here, we first describe the methodology for reviewing defects in CD and then we will describe related work proposing attributes and some quality metrics for the diagrams.

III. Methodology

The methodology of the review is based on the work in [3], and consists of six general activities:

TABLE I
INCLUSION CRITERIA

Criterion
CD with readability defects, but they can be interpreted.
CD with letter, background or border colors, other than black and white.
CD with small, illegible print.
CD with text orientation different than landscape.
CD with intersections of lines of relations.
CD with relationship lines with more than two square corners.
CD lacking abstraction and showing redundancy of attributes or operations.
CD with relationship lines that are not identifiable under the UML notation.

TABLE II
EXCLUSION CRITERIA

Criterion
CD that do not clearly present a name or description of their process, so their interpretation becomes difficult.
CD that are not readable, so their interpretation is impossible. Even though the lack of legibility in these is itself a great defect, it was decided to excluded them because they did not allow more information on other defects.
CD or very large UML diagrams, over 30 classes or that have too many elements to be correctly interpreted by a Software Engineering learner.
CD that contain few classes (less than 4). Because they may not contain enough information to correctly interpret the diagram.
CD that are not available for download.

TABLE III
NUMBER OF DEFECTS FOUND IN CD BY QUALITY ATTRIBUTE

Quality Attribute	Quantity	Description
Readability	49	Names, attributes, and operations in very small font size or illegible type.
Abstraction	28	Repeated attributes or operations in classes that are not part of the same inheritance hierarchy.
Redundancy	42	Repeated attributes or operations in classes of the same inheritance hierarchy.
Incomplete Relationships	2	The relationship does not connect any particular class.
Association Relationships	13	Wrong relationship or wrong direction.
Inheritance Relationships	43	Wrong relationship or wrong direction.
Aggregation Relationships	2	Wrong relationship or wrong direction.
Composition Relationships	5	Wrong relationship or wrong direction.
Realization Relationships	11	Wrong relationship or wrong direction.
Dependency Relationships	3	Wrong relationship or wrong direction.
Unknown Relationships	1	Relationship not recognized by UML standard notation.
Line Crossings	37	A relationship line crosses another relationship line.
Multiple Square Corners	19	More than 2 bends in lines that represent relationships.
Colors	60	Colors in background, border or letter.
Isolated classes	8	Classes unrelated to other classes.

1) Identification of the object from which information is required: in this case, they are free access UML CD on the Internet and available for download.
2) Use of keywords for CD search: the search uses keywords or key search strings, the most commonly used being: "UML class diagram," "Line crossing class diagram," and "UML class diagram quality." Using these search strings, a very large set of UML CD was obtained, although some of the CD were duplicated and as a result, it was decided to review 145 CD.
3) Inclusion and exclusion criteria: Once the search words or strings were selected, the inclusion criteria described in Table I and the exclusion criteria described in Table II were determined.
4) Analysis of the CD: From the application of the inclusion/exclusion criteria, the number of CD reviewed was reduced from 145 to 109. In addition, defects such as inconsistencies, lack of readability, lack of abstraction, line crossings, and excessive use of "square corners" in lines that represent relationships, amongst others, were identified. In Table III, defect types and their frequency are listed; there are 323 in total, identified in the 109 CD analyzed, being non-complying colors [4] as the defect that occurs the most times in the CD.
5) The following strings were used to search for related works: "UML class diagram," "Line crossing class diagram," and "UML class diagram design quality." Scientific databases or sources are ACM, IEEE Xplore, and Springer.
6) Research questions:

Q1: What are the most common defects in UML CD?

Q2: What are the approaches for measuring CD quality used by the authors in related work?

IV. RESULTS

The answer to question Q1 is as follows. The analysis of 109 CD allowed the authors to find a set of defects that can prevent the understanding of them. In Table III, the 15 types of defects and their frequency are listed, making a total of 323, with an defects average found of 2.96 occurrences for each CD.

Also, the redundancy of attributes and operations as well as the readability of the diagrams are the defects with a large number of occurrences.

CDs with poor readability are very difficult to interpret and were discarded. A problem that prevents the correct interpretation of the CD is that in many cases, they do not contain annotations that explain their functionality.

For the purposes of explaining the defects more clearly, four analyzed CD test cases are described. Likewise, some original figures are shown with the same quality as they were found in Internet.

Case 1: Fire Extinguishing System. This case is shown in Fig. 1 [5], in which classes "*Hidrante*," "*Agua pulverizada*," "*Rociador automatico agua*," "*BIE*," and "*Espuma fisica*" have a long line association relationship with a class called "*Sistema Abastecimiento*."

Additionally, attributes are repeated in several classes, which means that the CD does not respect the abstraction

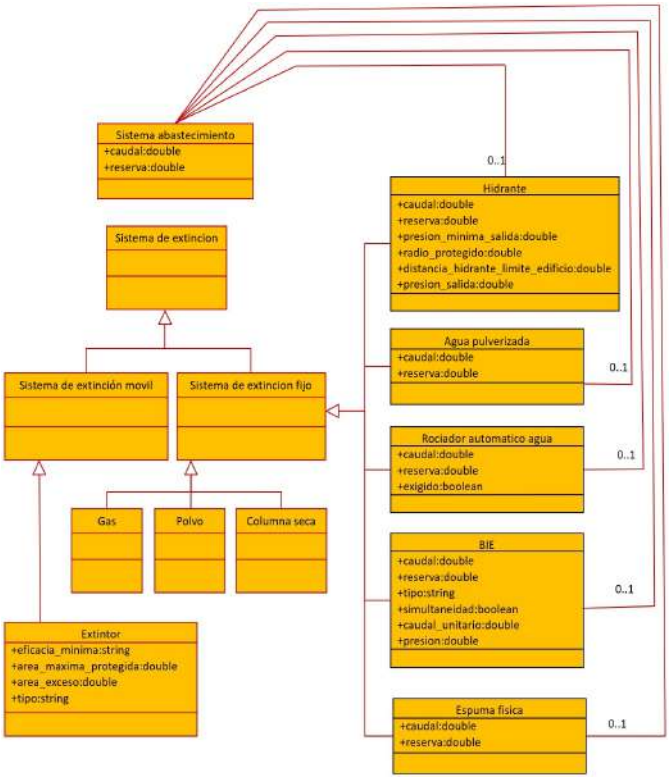

Fig. 1. Fire extinguishing system CD [5].

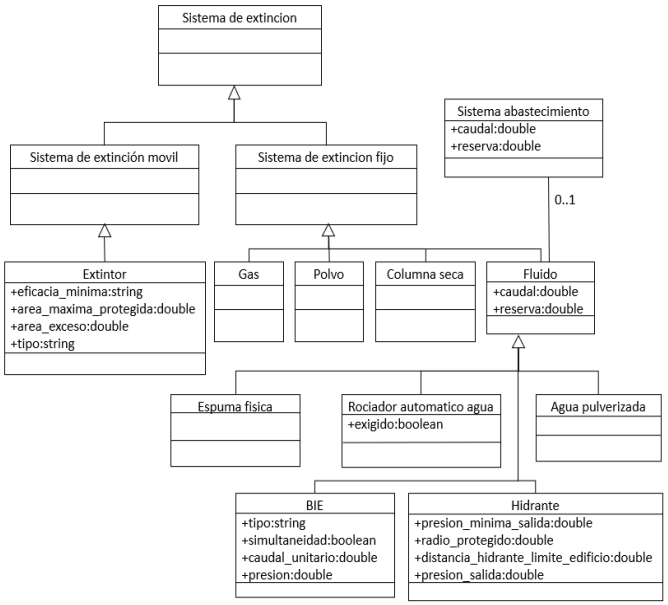

Fig. 2. Proposed solution to fire extinguishing system CD.

principle. That is, the classes that represent types of extinguishers that are fluid-based — "*Hidrante*," "*Agua pulverizada*," "*Rociador automatico agua*," "*BIE*," and "*Espuma fisica*" — have attributes "*Caudal*" and "*Reserva*" that are repeated in each class.

The diagram also represents the classes with a background color, which is not recommended according to [4]. In [6], it is recommended to minimize the "square corners" of the relationships since it is easier to follow straight lines than to follow lines that change direction. For the purpose of this review, those learning resources that show relationships with more than two "square corners" were considered.

For this case, the proposed solution consists of adding an abstraction or superclass called "*Fluido*" and inheriting the characteristics involved to the subclasses "*Hidrante*," "*Agua pulverizada*," "*Rociador automatico agua*," "*BIE*," and "*Espuma fisica*" corresponding, as shown in Fig. 2.

Case 2: Model of the design of a metric. In Fig. 3 [7], attributes in classes "CalculableConcept," "ConceptModel," "Unit," "Tool," "Method," "ElementaryModel," and "GlobaModel" are repeated. In the same way, long line relationships and multiple square corners may generate confusion when interpreting the diagram.

Likewise, the diagram presents colors such as red, purple, and yellow on the edges and background, which contravenes the recommendation of [4] to represent the diagrams in black with a white background.

Case 3: Model of a sports social network. In the example of Fig. 4 [8], background colors are used in classes and borders, which is not in accordance with the recommendations of [4].

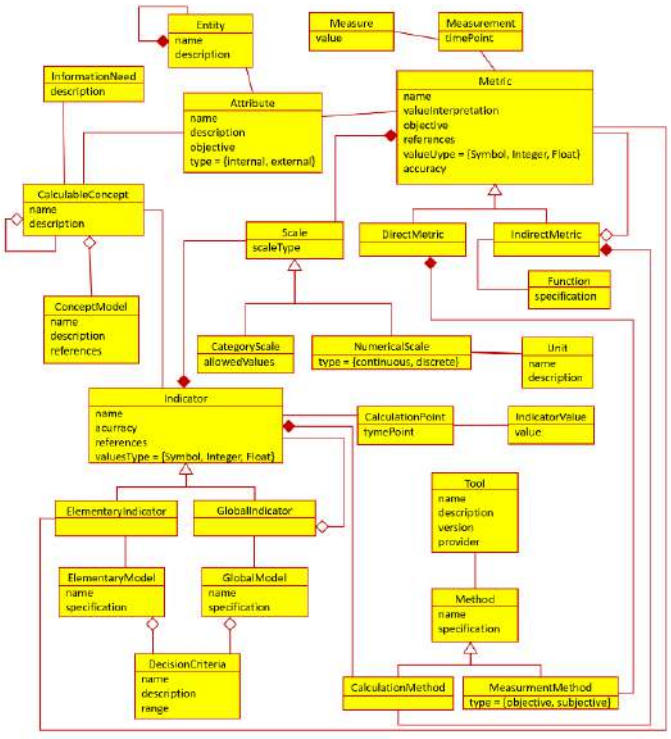

Fig. 3. Design model of a metric [7].

In addition, it shows inheritance and association relationships that repeatedly intersect with each other, which make it difficult to keep track of every relationship in the diagram; it contravenes the recommendation of [9], minimize the number of line crossings, the length of lines, and the number of bends or square corners in the same relationship.

The diagram in Fig. 3 is difficult to understand since it shows many long relationship lines with multiple square

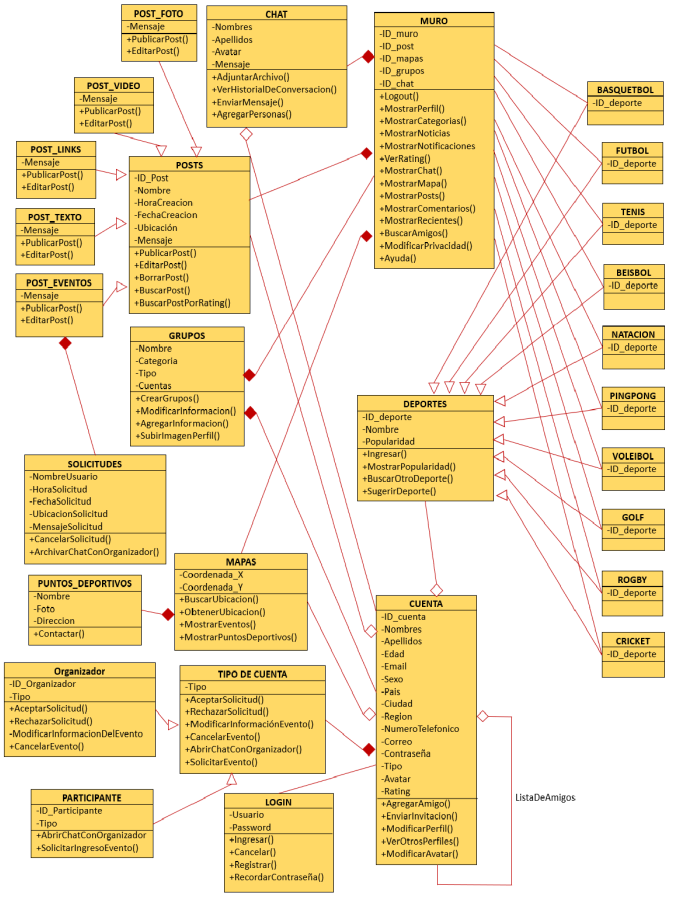

Fig. 4. Model of a sports social network [8].

cornered lines. Fig. 4 shows a large number of crossing associations, inheritance relationship lines, and aggregation and composition lines.

Following the recommendation of [10], to minimize the number of line crossings since the greater the number of crossings the more difficult it is for human eye to detect which classes are connected to each other, making it difficult to interpret the CD. Nonetheless, it is important to make sure that the system used to build the diagrams is configured so that the line types can be drawn differently in a way that crossings are minimized.

Case 4: Park Management Class Model. The example in Fig. 5 shows a diagram for the management of an amusement park. An example of a defect is the lack of abstraction of a class corresponding to a super class of transport "*Vehiculo*" and "*Vagon*," and as a consequence, there is attributes redundancy.

So, the proposed solution is to add the corresponding abstraction creating the class "*Transporte*," which contains the common attributes of the means of transportation, as shown in Fig. 6, and thus allows attributes and operations inheritance.

The proposed solution for this case is to place the attributes and operations in a new abstract class that inherits those characteristics common to the classes "*Vehiculo*" and "*Vagon*." In addition, the attribute "*Num_max_ocupantes*" can be placed in the class "*Atraccion*" to be inherited to all its subclasses as it is shown in Fig. 6.

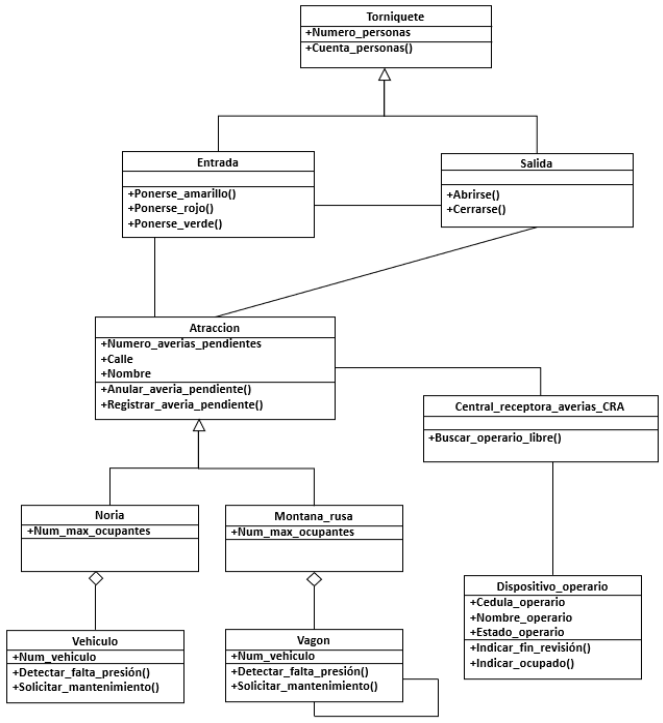

Fig. 5. Theme park management class model.

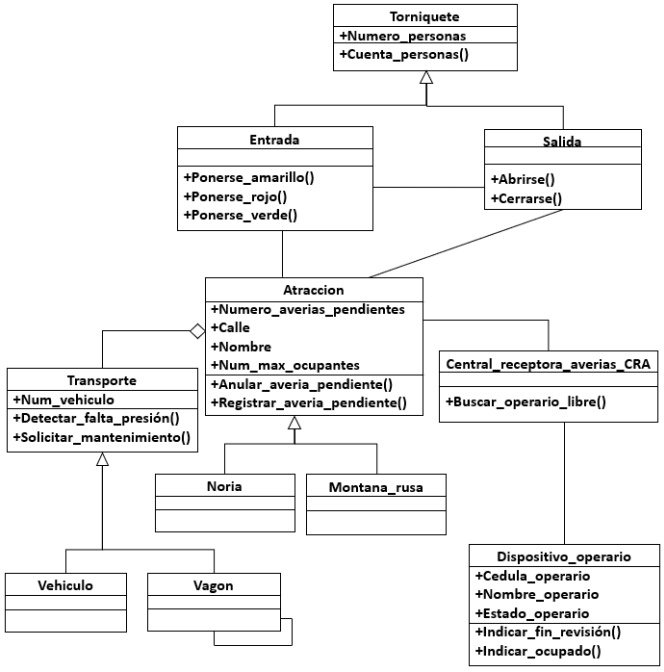

Fig. 6. Proposed solution to theme park management class model.

The answer to question Q2 is as follows. The analyzed works are classified into four approaches: a) Aesthetics of UML CD, b) Degree of Understanding, c) Metrics for the Quality of the UML CD, and d) Tools to Measure the Quality of the UML CD. Table IV classifies the related jobs within the identified approaches, the green box indicates that the job addresses the approach, and the black box with the X indicates the opposite.

TABLE IV
DETECTED APPROACHES IN RELATED WORK

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]	☒	☒	☒	☑

V. RELATED WORK

A. Aesthetics of UML CDs

In this section, two works were identified. The first work [12] proposes a set of guidelines regarding the aesthetics of the UML diagram presented. It mainly focuses on improving the quality of diagrams through compliance with established rules, as well as time spent on building the diagrams. The characteristics that are considered in the diagram are: design, structure, layout, and semantics. The issue of correctness is also addressed. The authors of this work conclude that more work and experimentation is required since the results are not conclusive; it is not yet possible to determine if a better-quality CD may be obtained with the application of the proposed guidelines.

In the second work [9], the authors present an approach in which the size of the UML CD is the fundamental thing regarding its understanding. The described hypothesis refers to the fact that the larger the diagram, the designer who models it has a poor performance, while a small diagram is easier to use and interpret regardless of its quality. The authors propose four principles for the construction of CD: 1) Principles of graphic design and visualization. 2) Minimization of crossings, bends, and line size. 3) Notations such as the order of visual elements focused on visual flows, minimization of visual clutter. 4) The practical part, use of colors, sizes, and positions that serve as a guide for readers.

B. Degree of Understanding of the UML CD

In [13], the authors carry out a review of the state of the art, in which they show a study focused mainly on the understanding of software process models and analyze different types of diagrams and aspects of UML. The authors conclude that more studies are required to mature the knowledge on this topic.

In [14], the authors present a study focused on understanding the UML CD mainly from a semantic point of view; they carry out reviews through a tool called SDMetrics® [15] and through a data dictionary that was created by experts. The assessment values range from 0 to 1, with a value of 0 being a poor assessment for this study and a value of 1 being an excellent assessment. The authors conclude that they must carry out further studies to validate all of the hypotheses.

C. Metrics to Measure the Quality of the UML CD

In [16], the authors present a work in which they review several groups of metrics proposed by [17] and [18]. The authors' objective is to propose algorithms to calculate the metrics based on the mapping of the metadata of UML CD XML files. The authors present the algorithms to perform the calculations and measurements of the metrics based on the data extracted from the CD, incorporating these algorithms to a tool mentioned as UML Refactoring.

In [19], the authors analyze Open Educational Resources (OER) that benefit Software Engineering. They mention both physical and electronic OERs, focusing mainly on electronic resources such as videos, massive, open online courses and multimedia games. The main interest is to identify resources that can help to improve learning in Software Engineering by establishing criteria related to software requirements, project management, testing, and refactoring. OER evaluation criteria such as content quality, usability, and educational value are taken into account; however, they do not mention how to evaluate those attributes.

In [20], the authors aim to improve the quality of UML CD through the use of Genetic Algorithms. An inter-object coupling metric is used to measure the degree of dependency of system components. The work focuses mainly on classify classes and establish their relationships through genetic algorithms. As a result, it is shown that genetic algorithms have the potential to evaluate UML CD; however, the authors mentioned that more studies are required in this regard.

In [21], the authors present an analysis of 22 UML diagrams in which 11 attributes such as: number of attributes, number of child classes, number of relationships, and number of generalization hierarchies. The authors also describe the quality metrics used and conclude that future work is required in order to evaluate when the size of CD is increased, the variety of types of cases, as well as working with Software Engineering professionals for better validation of the results.

Similar to [16] and [21], in [22] the authors present a study that aims to perform an analysis of metrics for the development of UML CD in the object-oriented model. The authors review a set of interrelated metrics found in the CD, although the review is realized on code. The study is justified in the analysis of software metrics with dependency patterns to provide an interpretation of each of these dependencies.

In [23], the authors propose a set of metrics based on CMMI, taking into account three quality models: syntax, semantics, and aesthetics. Although 13 types of UML diagrams are mentioned in the work, the main topic is the CD. The authors establish a connection between the quality models considering that a possible defect in one of them would produce defects and incorrect interpretations of the others. The review of the elements of the diagrams focuses mainly on the correctness of the same, based on integrity, consistency, and symmetry.

D. Tools to Measure the Quality of the UML CD

The work in [24] presents the description of a set of rules and tools for Software Engineering, particularly for the UML CD. In particular, the deal with CASE tools is strongly oriented to implementation instead of visual diagrams. It lists a set of rules regarding the layout of the CD and also about the drawing of the CD and their aesthetics. Different tools are reviewed, including Java libraries and graphics. The article provides information regarding the current rules used in the representation of UML CD and some tools to work with them, but emphasizes that these tools do not consider all aspects of the diagram.

In [25], the authors present an analysis between two types of UML diagrams — CD and sequence diagram — making a comparison in the inconsistencies found when passing the content of the first diagram to the second. The main idea of the authors is to develop an algorithm and the corresponding tool. In order to comply with it, the authors establish certain rules that both class and sequence diagrams developed in *Enterprise Architect* must contain and be exported to XML for later analysis. This work also proposes that the rules should consider other types of UML diagrams.

In [26], although the authors propose the degree of understanding of the UML CD as one of the attributes to be evaluated, their main proposal is a tool for collecting information regarding the metrics on the understanding of CD, but without mentioning the techniques used to collect and to process the information.

The work [27] describes a tool called TUPUX. The tool works with a set of rules that refer to the association, aggregation, generalization, and association relationships between classes. A comparison is made between the result of graduate and undergraduate students to determine if the established rules are correctly applied; as a result, the students wrongly identified the relationships between classes and as a consequence, an incorrect calculation of function points metric. The authors are considering carrying out future work with different case studies to obtain more precise results regarding the applicability of the model in the industry.

VI. Conclusion and Future Work

Software development processes have the inherent characteristic of being difficult processes; this is derived from the fact that the main software product is intangible. Therefore, it is necessary to direct efforts towards facilitating the achievement of learning the elements of the development processes, of which the construction of CD in the design phases of any process model is an activity that requires knowledge of the object-oriented paradigm and sufficient creativity to be able to apply the paradigm in generating a solution to a problem that needs to be addressed.

Learning software design is an activity which can greatly improve the quality of the learning resources that are used and thus have a positive impact on the competence of the next generation of Software Engineers with the role of CD designers. The review presented here shows that the learning resources called open CD, under the inclusion and exclusion criteria, contain several errors that range from the shape (e.g., the lines, colors, types of text, etc.) to the background. In other words, the theory and principles of the object-oriented paradigm are not fulfilled in such a way that when someone wants to learn with these learning resources, they will possibly have poor or incomplete learning, thus preventing the training of Software Engineers' competencies. Sometimes, one type of defect can lead to another type of defect. For example, a problem of misapplying the concept of abstraction can lead to a readability problem and that same time, it can lead to a comprehensibility problem and then there may be a sequence of defects.

The four cases analyzed may have more defects than those indicated in their section; however, not all defects are corrected as the intention is to show some examples.

The works described in Section IV were taken as a basis to establish what can be a defect in CD.

Taking simplicity into account as an attribute that must exist could possibly reduce the number of defects in diagrams, but designing with simplicity is a difficult competence to obtain. From the analysis carried out, it may be concluded that some conceptual modeling principles are not considered, which indicates that a model only presents what is necessary to be understood. Additionally, from the 109 cases studied, it can be suggested that a good practice is to put or identify the client class in a diagram.

Likewise, a very simple defect such as the lack of annotations resulted in something very important to facilitate the understanding of the solution that the CD represents.

Considering the problem described in Section II, a future work is the need for a quality model in which the quality attributes required in the CD are defined, complementary to those already proposed in the related works, and the way to measure them. Also, tools are needed to automatically perform the evaluation of the CD without depending on the presence of experts in the field, to refactor the CD, and to make corrections at the design stage of software development. It is very notable that some related works conclude on the need to extend studies on the subject.

Additionally, it can be mentioned that the differences in the notations of the diagrams make it difficult to understand the CD, so it is suggested that equivalences be established and these can be used in all CD development tools.

Finally, we conclude that this work can be a guide in the selection of open learning resources for Software Engineers, not only from CD but other types of resources.

Acknowledgment

The authors would like to thank the National Council of Science and Technology (Conacyt) and the Tecnológico Nacional de México/Centro Nacional de Investigación *y* Desarrollo Tecnológico (TecNM/Cenidet) for their support.

References

[1] R. Busljeta, "Effective use of teaching and learning resources," *Czech-polish historical pedagogical J.*, vol. 5, no. 2, pp. 55–70, Jul. 2013.

[2] J. L. Navarro, "Objetos de aprendizaje : Formación de autores con el modelo redes de objetos," UDGVirtual, Guadalajara, Mexico, Tech. Rep., 2005.

[3] B. Y. S. Kitchenham Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele Univ. and Durham Univ., EBSE, Durham, U.K., Tech. Rep. EBSE-2007-01, 2007.
[4] *Object Management Group*, OMG Unified Modeling Language, Needham, MA, USA, 2017.
[5] *Automatización del Reglamento de Seguridad Contra Incendios en Establecimientos Industriales*, Instituto Nacional de Seguridad e Higiene en el Trabajo, Madrid, Spain, 2004.
[6] K. Sugiyama, S. Tagawa, and M. Toda, "Methods for visual understanding of hierarchical system structures," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-11, no. 2, pp. 109–125, Feb. 1981.
[7] M. D. L. Á. Martín, "Sistema de catalogación de métricas e indicadores con potencia de web semántica," Ph.D. dissertation, Magister en Ingenieria de Softw., Univ. Nacional de La Plata, La Plata, Argentina, 2004.
[8] Blogspot. (Jun. 2012). *Diagrama De Una Red Social Deportiva*. Accessed: Mar. 2021. [Online]. Available: http://rsdplus.blogspot.com/2013/06/diagrama-de-clases-rsdplus.htmlx
[9] H. Störrle, "On the impact of layout quality to understanding UML diagrams: Size matters," in *Proc. 17th Int. Conf. Model Driven Eng. Lang. Syst.*, 2017, pp. 518–534.
[10] C. Batini, L. Furlani, and E. Nardelli, "What is a good diagram? A pragmatic approach," in *Proc. 4th Int. Conf. Entity-Relationship Approach*, 1985, pp. 312–319.
[11] *Universidad del Cauca, Programa de Ingeniería de Sistemas—Ingeniería de Software I. Diagrama de Clases*, Universidad del Cauca, Popayán, Colombia, 2012, p. 8.
[12] H. Eichelberger and K. Schmid, "Guidelines on the aesthetic quality of UML class diagrams," *Inf. Softw. Technol.*, vol. 51, no. 12, pp. 1686–1698, Dec. 2009.
[13] A. Dikici, O. Turetken, and O. Demirors, "Factors influencing the understandability of process models: A systematic literature review," *Inf. Softw. Technol.*, vol. 93, pp. 112–129, Jan. 2018.
[14] Y. Nakamura, K. Sakamoto, K. Inoue, H. Washizaki, and Y. Fukazawa, "Evaluation of understandability of UML class diagrams by using word similarity," in *Proc. Joint Conf. 21st Int. Workshop Softw. Meas. 6th Int. Conf. Softw. Process Product Meas.*, Nov. 2011, pp. 178–187.
[15] SDMetrics. *SDMetrics*. En Linea. Accessed: Mar. 2021. [Online]. Available: https://www.sdmetrics.com/
[16] O. Deryugina, "UML class diagram object-oriented metrics: Algorithms of calculation," in *Proc. 7th Seminar Ind. Control Syst., Anal., Model. Comput. (ICS)*, 2018, vol. 18, n. 4, p. 03001.
[17] S. R. Chidamber and C. F. Kemerer, "A metrics suit for object oriented design," *IEEE Trans. Softw. Eng.*, vol. 20, no. 6, pp. 476–493, Jun. 1994.
[18] F. A. Brito, L. Ochoa, and M. Goulao, "The MOOD metrics set," INESC/ISEG, Internal Rep., 1998.
[19] M. Villavicencio, V. Revelo, and J. Pincay, "Towards the evaluation of open educational resources for learning software engineering," in *Proc. 42nd Latin Amer. Comput. Conf.*, 2016, pp. 1–9.
[20] O. Deryugina, "Improving the structural quality of UML class diagrams with the genetic algorithm," in *Proc. ITM Web Conf.*, 2016, p. 03003.
[21] B. Mathur and M. Kaushik, "Empirical analysis of metrics using UML class diagram," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 5, pp. 32–37, 2016.
[22] S. Sarica and T. Ovatman, "Software design metric based analysis of dependency patterns," in *Proc. 2nd Int. Conf. Informat. Appl. (ICIA)*, Sep. 2013, pp. 317–322.
[23] M. Sharma and R. Vishwakarma, "CMMI based software metrics to evaluate OOAD," in *Proc. 2nd Int. Conf. Comput. Sci., Eng. Inf. Technol.*, 2012, pp. 19–25.
[24] H. Eichelberger and J. Wolff, "UML class diagrams—State of the art in layout techniques," in *Proc. VISSOFT*, 2003, pp. 30–34.
[25] E. Ekanayake and S. R. Kodituwakku, "Consistency checking of UML class and sequence diagrams," in *Proc. 8th Int. Conf. Ubi-Media Comput. (UMEDIA)*, Aug. 2015, pp. 098–103.
[26] S. Rajesh and A. Chandrasekar, "Metrics measurement model: To measure the object oriented design metrics," in *Proc. 7th Int. Conf. Adv. Comput. (ICoAC)*, Dec. 2015, pp. 1–6.
[27] J. A. Pow-Sang, D. Villanueva, L. Flores, and C. Rusu, "A conversion model and a tool to identify function point logic files using UML analysis class diagrams," in *Proc. Joint Conf. 23rd Int. Workshop Softw. Meas. 8th Int. Conf. Softw. Process Product Meas.*, Oct. 2013, pp. 126–134.

Olivia Graciela Fragoso-Díaz (Senior Member, IEEE) received the Ph.D. degree in computer science in 2012. In 1995, she joined the Tecnológico Nacional de México/Cenidet (TecNM/CENIDET), Cuernavaca, Mexico, as a Researcher in software engineering. Her research interests include software engineering and software technologies for e-learning, software reusability, web services classification and retrieval, software quality, and software processes.

José Antonio Sandoval-Acosta received the master's degree in information systems in 2010. He is currently pursuing the Ph.D. degree in computer science with the TecNM/CENIDET, Guasave, Mexico. His research interests include software engineering, software technologies for e-learning, and software quality assurance.

Francisco Javier Álvarez-Rodríguez received the Ph.D. degree in engineering from the Universidad Nacional Autónoma de México (UNAM). He joined the Aguascalientes Autonomous University (UAA) as a Researcher in software engineering. He is also the President of the National Council for Accreditation of Informatics and Computing Degrees.

Juan Carlos Rojas-Pérez (Senior Member, IEEE) received the Ph.D. degree in computer science in 2009. In 2010, he joined the Tecnológico Nacional de México/Cenidet (TecNM/CENIDET), Cuernavaca, Mexico, as a Researcher in software engineering. His research interests include software engineering and software engineering applied to big data, service-oriented architectures, web services metrics, micro-services, natural language processing, and databases.

René Santaolaya-Salgado (Senior Member, IEEE) received the Ph.D. degree in computer science in 2003. In 1992, he joined the Tecnológico Nacional de México/Cenidet (TecNM/CENIDET), Cuernavaca, Mexico, as a Researcher in software engineering. His research interests include software reuse, refactoring software, software quality, cloud computing, service-oriented architectures and microservices, and software processes.