Connection between UML use case diagrams and UML class diagrams: A matrix proposal

Bráulio Alturas

Instituto Universitário de Lisboa (ISCTE-IUL), ISTAR-Iscte (University Institute of Lisbon), Av. Forças Armadas 1649-026 Lisboa, Portugal
Email: braulio.alturas@iscte-iul.pt

Abstract: In recent years, the UML language has been one of the most used to conduct information system analysis and design. Being an object-oriented technique, UML provides a vast set of diagrams, in order to represent the various abstractions of the system, of which the most used are the use-case diagram and the class diagram. Often, in the modelling of less complex systems, only these two diagrams are used, where one represents the functionalities of the system (use case diagram) and the other the static structure of the system (class diagram). However, it is often difficult to make the connection between the two diagrams, and mainly, it is difficult to verify when one matches the other. In order to solve this problem, a matrix is proposed that links the two diagrams, using a case study to verify the utility of the matrix.

Keywords: UML; Use Case Diagram; Class Diagram; Modelling Technique; Matrix; Information Systems; Object-Oriented; Analysis and Design.

Reference to this paper should be made as follows: Alturas, B. (2023) 'Connection between UML use case diagrams and UML class diagrams: A matrix proposal', Int. J. of Computer Applications in Technology.

Biographical notes: Bráulio Alturas (PhD) is Associate Professor of Department of Information Science and Technology of Instituto Universitário de Lisboa (ISCTE-IUL) (University Institute of Lisbon), Lisbon, and researcher of the Information Systems Group of ISTAR (Information Sciences, Technologies and Architecture Research Center). He holds a PhD in Management with specialization in Marketing, a MSc in Management Information Systems, and a BSc in Business Organization and Management, all from ISCTE-IUL. His scientific research interests and publications are in acceptance and use of technology, digital marketing, social media, e-commerce, information management, information systems, direct selling and consumer behavior. ORCID: 0000-0003-0142-3737.

1 Introduction

Conceptual models, which are often graphically represented, are used by Information Systems professionals to denote both static and dynamic aspects of a particular domain. They play an increasingly important role during all phases of the information systems lifecycle (Fettke, 2009).

Model-based development, along with formal verification process, assures the developed model satisfies software requirements described in formal specifications (Jnanamurthy et al., 2021) and it becomes increasingly important to develop various artefacts, and the model-based artefacts co-evolution is another challenge when metamodel evolves (Sabraoui et al., 2022).

Since its creation in the late twentieth century, the UML language has been used by many academics and practitioners to build conceptual models. Object-oriented (OO) modelling techniques are nowadays fundamental for the analysis and design of information systems. Of the various OO techniques, the unified modelling language (UML) has emerged as the dominant modelling approach and has become an essential part of the toolset of any IS professional (Shen et al., 2018).

The latest version of UML considers 14 different diagrams in order to represent the various abstractions of a system; however, the use case diagram and the class diagram are still the most used by academics and practitioners. These are also the diagrams most taught in universities, in systems modelling courses, and students often learn to build both diagrams without understanding the connection between them. This connection is not always easy to understand, and above all it is not easy to verify if the two diagrams are in agreement with each other. The lack of integration between use cases and class models raises questions about the value of use cases in an object-oriented modelling approach (Dobing and Parsons, 2000).

In order to solve this problem, this article presents a matrix, inspired by the CRUD matrix, that links the two diagrams. The proposal is to build the matrix after making the two diagrams, to check if in any of the diagrams there are missing components or too many components, allowing their subsequent correction.

2 Theoretical Background

2.1 The UML language

The UML - Unified Modelling Language is a language that uses standard notation to specify, build, visualize and document object-oriented information systems (Nunes & O'Neill, 2004).

The usage of some modelling techniques has increased over the course of the last years, which applies to the use of UML (Fettke, 2009). UML is a visual language equipped with a rich set of diagrams. Each kind of UML diagram provides a different perspective of the system to be developed (Almendros-Jiménez & Iribarne, 2009).

The creation of UML was originally motivated by the desire to standardize the disparate notational systems and approaches to software design. Efforts to create UML began in October 1994, when the company Rational promoted the meeting between the North Americans Rumbaugh and Booch, with the aim of unifying the method of modelling by Booch and OMT (Object Modelling Technique) created by Rumbaugh. The Object Management Group (OMG) first standardized the Unified Modelling Language in 1997. The software industry quickly accepted it as the standard modelling language for specifying software and system architectures. Although UML is primarily intended for general purpose modelling, it is receiving extensive use in various specialized areas, such as business process modelling and real-time systems modelling (Björkander & Kobryn, 2003).

UML offers a powerful set of notations and diagrams that allow the capture of both static and dynamic aspects of processes and can increase their understandability (Bendraou et al., 2010).

In most projects, UML models are the first artifacts to systematically represent a software architecture. They're subsequently modified and refined in the development process. Their importance has increased with the advent of model-driven architecture methodology (Lange & Chaudron, 2006). Actually, the Unified Modelling Language (UML) provides a graphical representation of a software system, which can be used for both forward engineering and reverse engineering (Wong & Sun, 2006). A 2019 study, of more than 5,400 entries dealing with the execution of the UML model, concluded that there is growing scientific interest in the execution of the UML model (Ciccozzi et al., 2019).

It has been observed that graphical representation of model is easily accessible and understandable to the user. The primary gap between the developer and the user has been easily filled by the graphical description. In UML, Use Case diagram defines the behaviour of a system. Classes diagrams are used to capture the information about the system to be developed (Singh et al., 2016).

2.2 Use case diagram

A use case represents system functionality, and is the UML technique for capturing a system's requirements. Each use case provides one or more scenarios that indicate how the system must interact with end users, or with other systems, to achieve a specific business objective. Use cases typically avoid technical jargon, preferring instead the language of the end user or a domain expert. Use cases are often co-authored between analysts and users (Alturas, 2022). And so, a use case is a piece of the system's functionality, describing the possible interactions between the system and a user entity external to it called "actor", for the purpose of achieving a goal of that actor (Shoval et al., 2006).

In general, requirements expressed in natural language are the first step in the software development process and are documented in the form of use cases (Zaman et al., 2020). The use-case diagrams are built when collecting and specifying requirements and represent the functional requirements, which, when implemented, will be the functionalities of the system, that is, they show what the system does and not how it does it. Bennet, McRobb, and Farmer (2010) identified three categories of requirements (Bennett et al., 2010):

- Functional requirements - Describe what the system should do, that is, the description of the processing, inputs, and outputs, in the interaction with people and other systems;
- Nonfunctional requirements - Describe the quality with which the system must provide functional requirements, that is, performance measures, response times, data volume, security considerations;
- Usability requirements - These are the guarantee of a good connection between the system and the users, as well as the tasks they perform with the support of the system.

The use case notation comprises actors, use cases and associations. An actor reflects a role that is played by a human (or non-human) with respect to a system. Actors execute use cases. The link between actors and use cases is shown by an association, which indicates that there is communication between the actor and the use case, such as the sending and receiving of messages (Vidgen, 2003).

A use case diagram is used to provide a visual summary of the use cases, actors, and their relationships. Since use case modelling is performed early in the software development cycle, any defects in a use case model will propagate to subsequent development phases and artifacts (Khan & El-attar, 2016).

2.3 Class diagram

Class diagrams are the oldest part of UML (Rumbaugh, 2006). Class diagrams are concise, easy to understand, and work well in practice (Alturas, 2022).

The class diagram represents not only the different classes, but also the relationships that exist between them, through simple graphical notation, in an object-oriented approach. The simplicity of UML diagrams is a

fundamental aspect in systems modeling because 'the essential task of the software development process is to offer the user the illusion of simplicity, protecting him from the frequent complexity of the system' (Booch et al., 2005).

The UML class diagram is essentially designed to represent the structural component of a system, namely the information structure that supports it. This structure is usually static for a considerable period of time (Ramos, 2007).

In a study published in 2011 it was found that the class diagram was the most used of all UML diagrams. In this study, the class diagram was considered most useful with the use case diagram least useful (Dobing & Parsons, 2011). In 2019, another study proposes to use the UML use case diagram to model business requirement level, and UML class diagram for software system level (Habba et al., 2019).

Classes represent things; relationships represent the connections between things. UML caters for three types of relationship: association, generalization, and aggregation. An association is a structural relationship between things showing that one can navigate from the instances of one class to the instances of another (and possibly vice versa) (Vidgen, 2003).

3 Matrix proposal

UML is not a formal model, and hence ambiguities may arise in design specifications between models that represent overlapping but different aspects of the same system. Ensuring traceability of requirements in different phases of its life cycle and verifying consistency among different models representing these phases are of utmost importance (Chanda et al., 2009).

The structural model of a problem domain may be specified by the conceptual class model with abstractions (classes) from that problem domain and relationships between them (most notably, associations and generalizations). On the other hand, the behavioural aspect of the problem is specified by use cases (Milicev, 2002).

One question that can be asked is what is the best order to create the diagrams, that is, the main analysis tasks in a use case-driven approach: creating a class diagram to model the problem domain, and creating use cases to describe the functional requirements of the system. An experiment published in 2006 reveal that starting the analysis by creating a class diagram leads to a better class diagram (Shoval et al., 2006). But another study suggests that, when teaching UML, it may be good to start off with teaching the use case diagram, as it is probably easier for students to comprehend than the class diagram (Siau & Lee, 2004).

A research published in 2004, centres on investigating the roles of use case diagrams and class diagrams in requirements analysis. The findings of this research have some implications for practitioners. The results of the study show that use-case diagrams were interpreted more completely than class diagrams. This seems to imply that use case diagrams are easier to interpret than class diagrams, thus enabling the subjects to attain a more complete understanding of the model (Siau & Lee, 2004).

VanderMeer and Dutta (2009) assert that UML is complex and difficult to learn. They evaluated the UML sequence diagram, which may be viewed as a bridge between a use case and its class diagram (VanderMeer & Dutta, 2009).

But there may be other solutions to make the link between the two diagrams so that users can better understand the link. The modeling languages and techniques must adapt to the needs of their users as spoken human languages change in response to the needs of their speakers (Erickson, 2008).

The two diagrams described previously (use-case diagram and class diagram) must be logically related (Surmsuk & Thanawastien, 2008; Whitten & Bentley, 2005). Although there are other diagrams that can represent the relationship between the two, one of the simplest ways to relate them, and to evaluate their coherence, is through a simple matrix: the use cases / classes matrix.

The use cases in the use case diagram represent system functionality. In order for these functionalities to perform the service for which they were created, they must have access to data, that is, in the object classes of the class diagram. This data can be consulted or modified by use cases.

The matrix is built by placing the functionalities (use cases) in line and the classes in columns. Sometimes, for simplicity, classes are numbered (and the same can be done in the class diagram itself). Then, in the boxes where a feature crosses with a class, it can be marked "C" if the functionality Consult class data or "M" if the functionality Modify (insert, update or delete) the class data. The proposed matrix is shown in Figure 1.

Figure 1 Use Cases/Classes Matrix

CLASSES / USE CASES	1	2	3	4	5	6	7	8	9	10	11	12
A												
B												
C												
D												
E												
F												
G												
H												
I												
J												

In line, all the main functionalities of the system must be placed, and in column all the classes, including the associative classes, are placed. Then it is marked with C when a functionality queries the data of objects of a certain class, and / or with M when a functionality modifies the data of objects of a certain class. After the matrix is completed, it is checked if there are any columns or rows without a C or

an M, and if this occurs, it means that at least one of the diagrams is not completely correct.

This matrix was inspired by the CRUD matrix. A CRUD matrix is a table in which rows indicate attributes of the entities of a relation, and columns indicate different locations of the applications (Surmsuk & Thanawastien, 2008; Whitten & Bentley, 2005).

The CRUD matrix is constructed in such a way that the functionalities are listed in one of its axes and the entities in the other. The intersecting cells denote the type of existing interaction, that is: they show which entity will be affected by the execution of a certain functionality and it explains the CRUD properties for that intersection: Create (inclusion), Read (query), Update (change) and Delete (exclusion).

4 Case study

4.1 Description of the case study

In order to test the matrix, a case study of a small company that sells home appliance parts was used. The company needed to develop a small information system for order management. Interviews were conducted with the CEO, the warehouse manager and the purchasing manager, who would be the main users of the system to be created. From the interviews, the requirements were collected and specified, resulting in the following text.

The company has suppliers that supply several parts, and each part can be supplied by several suppliers. Suppliers (providers) are assigned a code that uniquely identifies them. The same procedure is used for the parts.

Suppliers are further characterized by name, address, and telephone number. Parts are characterized by designation and colour. The quantity in stock of each piece is also known. There are two types of parts: the plastic parts in relation to which the material of which they are made is kept, and the metallic parts on which the respective measure is kept. Supplies are made at a given price and on a given date.

Each supplier fulfils several part orders. The orders are sequentially numbered, and each order has a variable number of positions (order lines), each of which concerns a part to be ordered. Each part can only be in one position of each order. The positions are numbered from one onwards, within each order.

The order has a date and status (made, confirmed, received, checked or completed) and is addressed to a single supplier. Positions include the position number, the unit price, and the quantity ordered, as shown in Figure 2.

Figure 2 Order Form

Order No. _____		Date: _____		Supplier: _____	
Position no.	Part code	Designation	Quantity	Unit price	Total

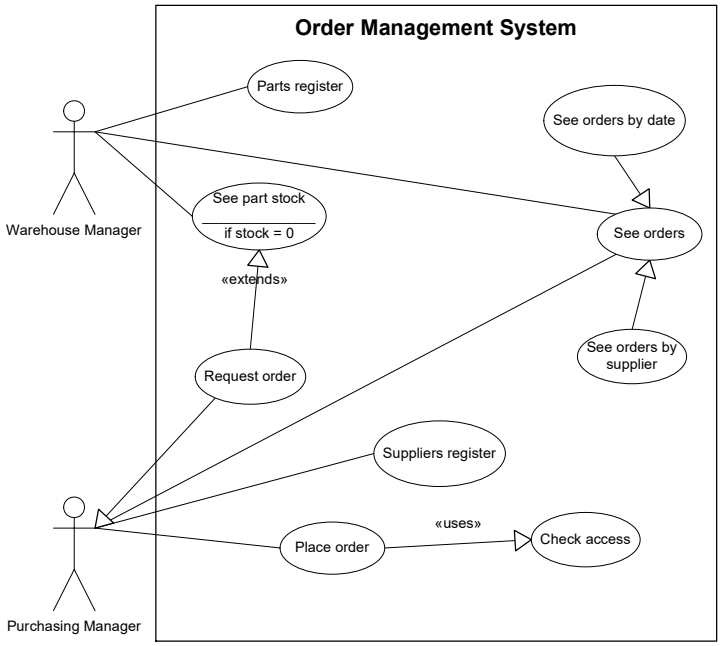
The Warehouse Manager is responsible for registering all parts in the system. He should also check the stock periodically, and in the event of a zero stock check, the system automatically sends a message to the Purchasing Manager to place an order.

The purchasing manager is responsible for registering suppliers in the system and also for placing orders, and for that purpose, they must validate their access to the system. Both the Purchasing Manager and the Warehouse Manager can consult the orders registered in the system, which can be done by date or by supplier.

4.2 UML case study modelling

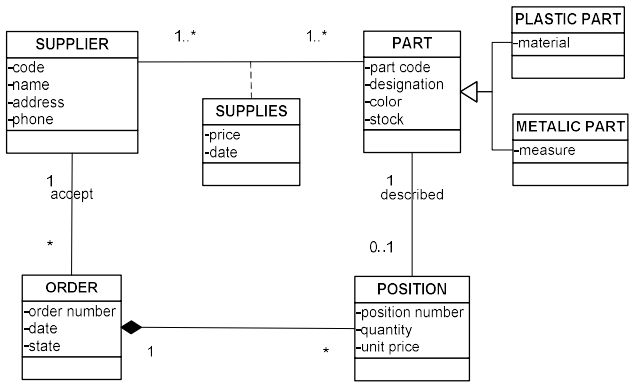
From the requirements specification, the system was modelled using UML. First, from the necessary functionalities, the use case diagram was drawn (see Figure 3) and then, from the necessary data, the class diagram was drawn (see Figure 4). For the construction of both diagrams, MS Visio was used.

Figure 3 Use Cases Diagram



As can be seen in Figure 3, the use case diagram represents two actors: the warehouse manager and the purchasing manager, and seven main use cases, one of which has two particularizations. The class diagram represents four main classes (one with two particularizations) and an associative class (see Figure 4).

Figure 4 Class Diagram



Once the two diagrams were built, the matrix was made, placing the use cases in line and the classes in a column (including associative classes), as shown in Figure 5.

Figure 5 Case Study Use Cases/Classes Matrix

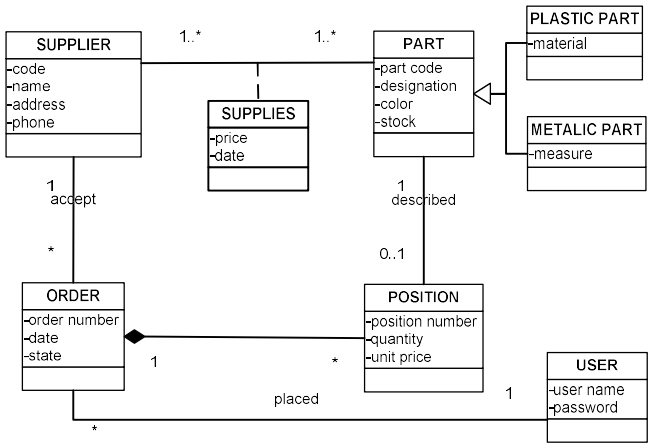
Use Cases \ Classes	SUPPLIER	SUPPLIES	PART	ORDER	POSITION
Parts register			M		
See orders				C	C
See part stock			C		
Request order			C		
Suppliers register	M	M			
Place order	C	C	C	M	M
Check access					

As can be verified in Figure 5, the line regarding the use case "Check access" is not written, neither a C nor an M. This means one of two things: or the use case "Check access" is not necessary and must be removed from the use case diagram and also from the Matrix; or the use case "Check access" is necessary and it will be necessary to add a class representing objects whose data are necessary for this use case in the class diagram.

Likewise, if a column for a class did not have any C or M, it could mean one of two things: either the class is not needed for any functionality in the system and must therefore be removed from the class diagram; or it will be necessary to add to the diagram of use cases a use case that uses the data of the objects of that class.

In this case, after consulting the company's managers again, it was concluded that it was necessary to validate access to the system to place orders, and thus it was necessary to store information about the users of the system, so the class diagram was redesigned, according to Figure 6.

Figure 6 Redesigned Class Diagram



Obviously, it was also necessary to redesign the matrix, including the USER class (see Figure 7).

Figure 7 Redesigned Use Cases/Classes Matrix

Use Cases \ Classes	SUPPLIER	SUPPLIES	PART	ORDER	POSITION	USER
Parts register			M			
See orders				C	C	
See part stock			C			
Request order			C			
Suppliers register	M	M				
Place order	C	C	C	M	M	
Check access						C

Now it can be said with certainty that the two diagrams are in agreement with each other. Of course, this does not mean that they represent exactly the reality that was intended to model. As is known, the task of system modelling is iterative and during the analysis and design of the system it may always be necessary to add other use cases or other classes.

5 Evaluation

To evaluate the utility of the matrix, the case study was presented to a sample of 40 students, from a post-graduate program, who had attended a systems modeling course with UML. A questionnaire was built to find out if the students had understood the diagrams of use cases and classes, and if the matrix had helped them to understand the connection between the diagrams. 35 valid responses to the questionnaire were obtained.

To evaluate UML knowledge, respondents were asked to rate a set of nine items on a 5-point Likert scale from 1 (strongly disagree) to 5 (strongly agree). These items were considered by the teachers of information systems, as being the most adequate. Respondents were also asked to provide answers on some demographic characteristics, namely gender and age. The results show that 54% of the respondents were women and 46% men, the average age being 33 years.

Regarding the nine items (which are now our nine ordinal variables) it was found that all of them have an average higher than 4 (on a scale of 1 to 5), with the highest average relative to the questions "What I learned from UML is useful "and" What I learned from UML helped me to better understand the analysis and design of information systems " with 4.77 (Table 1).

Finally, to check which items are most correlated with item 9 (the matrix allowed me to better understand the relationship between the use case diagram and the class diagram), a Pearson correlation was run (Table 2). The results show significant correlations with all items, except item 5 (I am able to draw a use case diagram correctly), and, on the other hand, item 4 (I understand what a use case diagram is) was the one that presented stronger correlation. That is, for the respondents it is more important to understand what a use case diagram is, than to know how to draw it correctly.

Table 1 Descriptive Statistics

	N	Mean	Std. Deviation
1. What I learned from UML is useful	**35**	**4,77**	**0,426**
2. What I learned from UML helped me to better understand the analysis and design of information systems	**35**	**4,77**	**0,426**
3. I understand most UML diagrams	**35**	**4,43**	**0,502**
4. I understand what a use-case diagram is	**35**	**4,60**	**0,497**
5. I am able to draw a use case diagram correctly	**35**	**4,20**	**0,759**
6. I understand well what a class diagram is	**35**	**4,63**	**0,547**
7. I am able to draw a class diagram correctly	**35**	**4,34**	**0,684**
8. I understand the relationship between the use-case diagram and the class diagram	**35**	**4,31**	**0,758**
9. The matrix allowed me to better understand the relationship between the use-case diagram and the class diagram	**35**	**4,23**	**0,731**
Valid N (listwise)	**35**		

Table 2 Correlations between variables

		1.	2.	3.	4.	5.	6.	7.	8.	9.
1. What I learned from UML is useful	**Pearson Correlation**	**1**								
	Sig. (2-tailed)									
	N	**35**								
2. What I learned from UML helped me to better understand the analysis and design of information systems	**Pearson Correlation**	**,838****	**1**							
	Sig. (2-tailed)	**0,000**								
	N	**35**	**35**							
3. I understand most UML diagrams	**Pearson Correlation**	**,334***	**0,196**	**1**						
	Sig. (2-tailed)	**0,050**	**0,258**							
	N	**35**	**35**	**35**						
4. I understand what a use-case diagram is	**Pearson Correlation**	**,389***	**0,250**	**,589****	**1**					
	Sig. (2-tailed)	**0,021**	**0,147**	**0,000**						
	N	**35**	**35**	**35**	**35**					
5. I am able to draw a use case diagram correctly	**Pearson Correlation**	**0,145**	**0,055**	**,386***	**,530****	**1**				
	Sig. (2-tailed)	**0,404**	**0,756**	**0,022**	**0,001**					
	N	**35**	**35**	**35**	**35**	**35**				
6. I understand well what a class diagram is	**Pearson Correlation**	**0,256**	**0,130**	**,597****	**,736****	**,397***	**1**			
	Sig. (2-tailed)	**0,138**	**0,457**	**0,000**	**0,000**	**0,018**				
	N	**35**	**35**	**35**	**35**	**35**	**35**			
7. I am able to draw a class diagram correctly	**Pearson Correlation**	**0,176**	**0,075**	**,588****	**,589****	**,487****	**,744****	**1**		
	Sig. (2-tailed)	**0,312**	**0,668**	**0,000**	**0,000**	**0,003**	**0,000**			
	N	**35**	**35**	**35**	**35**	**35**	**35**	**35**		
8. I understand the relationship between the use-case diagram and the class diagram	**Pearson Correlation**	**0,229**	**0,229**	**0,177**	**,421***	**,654****	**,432****	**,410***	**1**	
	Sig. (2-tailed)	**0,186**	**0,186**	**0,310**	**0,012**	**0,000**	**0,010**	**0,014**		
	N	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	
9. The matrix allowed me to better understand the relationship between the use-case diagram and the class diagram	**Pearson Correlation**	**,456****	**,456****	**,446****	**,664****	**0,233**	**,586****	**,427***	**,503****	**1**
	Sig. (2-tailed)	**0,006**	**0,006**	**0,007**	**0,000**	**0,178**	**0,000**	**0,010**	**0,002**	
	N	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**

****. Correlation is significant at the 0.01 level (2-tailed).**

***. Correlation is significant at the 0.05 level (2-tailed).**

6 Conclusion

Although analysts are known to use similar matrices, no scientific publications have been found that formalize a matrix with these characteristics. In this study, a matrix was used to check whether a use-case diagram matches the class diagram in a given case. The matrix allows one to check what may be missing in the diagrams, allowing them to be easily corrected. The case study provide empirical evidence of the difficulties to connect the two diagrams and how this difficulty can be overcome. Are presented concrete recommendations to address the learning challenges. Therefore, valuable information and recommendations on teaching UML were provided.

The case study demonstrated the utility of the matrix, allowing us to verify what may be missing (and which should be added) or what is not necessary (and should be removed) from the diagrams under study. It was found that in the columns of the matrix, it is not enough to represent the classes, but also the associative classes and the many to many associations, must be placed.

In addition, the case study was presented to a sample of students from a post-graduate program who had attended a system modelling course with UML. Most of the respondents said they strongly agree with the statement 'The matrix allowed me to better understand the relationship between the use case diagram and the class diagram', which validates the utility of the matrix for this sample of respondents.

This study had limitations, the fact that it was based only on a single case study and was validated by a sample of only 35 students. In future studies, more cases should be analyzed and validated with more representative samples, not only from students, but also from professionals.

References

Almendros-Jiménez, J.M. and Iribarne, L. (2009) 'UML modeling of user and database interaction', *Computer Journal*, Vol. 52, No. 3, pp. 348–367.

Alturas, B. (2022) *Introdução Aos Sistemas de Informação Organizacionais – 2ª Edição [Introduction to Organizational Information Systems – 2nd Edition]*, Edições Sílabo, Lisboa.

Bendraou, R., Jézéquel, J.M., Gervais, M.P. and Blanc, X. (2010) 'A comparison of six UML-based languages for software process modeling', *IEEE Transactions on Software Engineering*, Vol. 36, No. 5, pp. 662–675.

Bennett, S., McRobb, S. and Farmer, R. (2010) *Object-Oriented Systems Analysis and Design Using UML*, 4th ed., McGraw-Hill Education.

Björkander, M. and Kobryn, C. (2003) 'Architecting systems with UML 2.0', *IEEE Software*, Vol. 20, No. 4, pp. 57–61.

Booch, G., Rumbaugh, J. and Jacobson, I. (2005) *The Unified Modeling Language User Guide*, 2nd edition, Addison-Wesley Professional.

Chanda, J., Kanjilal, A., Sengupta, S. and Bhattacharya, S. (2009) 'Traceability of requirements and consistency verification of UML UseCase, activity and class diagram: A formal approach', *Proceedings of International Conference on Methods and Models in Computer Science, ICM2CS09*, available at:https://doi.org/10.1109/icm2cs.2009.5397941.

Ciccozzi, F., Malavolta, I. and Selic, B. (2019) 'Execution of UML models: a systematic review of research and practice', *Software & Systems Modeling*, Vol. 18, pp. 2313–2360.

Dobing, B. and Parsons, J. (2000) 'Understanding the Role of Use Cases in UML', *Journal of Database Management*, Vol. 11, No. 4, pp. 28–36.

Dobing, B. and Parsons, J. (2011) 'Dimensions of UML Diagram Use', *Journal of Database Management*, Vol. 19, No. 1, pp. 1–18.

Erickson, J. (2008) *A Decade and More of UML: An Overview of UML Semantic and Structural Issues and UML Field Use*.

Fettke, P. (2009) 'How conceptual modeling is used', *Communications of the Association for Information Systems*, Vol. 25, No. 1, pp. 571–592.

Habba, M., Fredj, M. and Benabdellah Chaouni, S. (2019) 'Alignment between Business Requirement, Business Process, and Software System: A Systematic Literature Review', *Journal of Engineering (United Kingdom)*, Hindawi, Vol. 2019, pp. 1–19.

Jnanamurthy, H.K., Henskens, F., Paul, D. and Wallis, M. (2021) 'Formal specification at model-level of model-driven engineering using modelling techniques', International Journal of Computer Applications in Technology, Vol. 67 No. 4, pp. 340–350.

Khan, Y.A. and El-attar, M. (2016) 'Using model transformation to refactor use case models based on antipatterns', *Information Systems Frontiers*, Vol. 18, No. 1, pp. 171–204.

Lange, C.F.J. and Chaudron, M.R.V. (2006) 'UML Software Architecture and Design Description', *IEEE Software*, Vol. 23, No. 2, pp. 40–46.

Milicev, D. (2002) 'Automatic Model Transformations Using Extended UML Object Diagrams in Modeling Environments', *IEEE Transactions on Software Engineering*, Vol. 28, No. 4, pp. 413–431.

Nunes, M. and O'Neill, H. (2004) *Fundamental de UML [UML Fundamental]*, 3rd ed., FCA – Editora de Informática, Lisboa, Portugal.

Ramos, P.N. (2007) *Desenhar Bases de Dados Com UML [Design Databases with UML]*, 2nd ed., Sílabo, Lisboa, Portugal.

Rumbaugh, J. (2006) 'ER Is UML', *Journal of Information Systems Education*, Vol. 17, No. 1, pp. 21–25.

Sabraoui, A., Abouzahra, A. and Afdel, K. (2022) 'Metamodel extension approach applied to the model-driven development of mobile applications', International Journal of Computer Applications in Technology, Vol. 68 No. 2, pp. 114–131.

Shoval, P., Yampolsky, A. and Last, M. (2006) 'Class diagrams and use cases - Experimental examination of the preferred order of modeling', *CEUR Workshop Proceedings*, Vol. 364, pp. 91–102.

Siau, K. and Lee, L. (2004) 'Are use case and class diagrams complementary in requirements analysis? An experimental study on use case and class diagrams in UML', *Requirements Engineering*, Vol. 9, No. 4, pp. 229–237.

Singh, M., Sharma, A.K. and Saxena, R. (2016) 'Formal Transformation of UML Diagram: Use Case, Class, Sequence Diagram with Z notation for Representing the Static and Dynamiv Perspectives of System', *Advances in Intelligent Systems and Computing*, Vol. 409, pp. 25–38.

Surmsuk, P. and Thanawastien, S. (2008) 'The Integrated Strategic Information System Planning Methodology', pp. 467–467.

VanderMeer, D. and Dutta, K. (2009) 'Applying Learner-Centered Design Principles to UML Sequence Diagrams', *Journal of Database Management*, Vol. 20, No. 1, pp. 25–47.

Vidgen, R. (2003) 'Requirements Analysis and UML: Use Cases and Class Diagrams', *Computing & Control Engineering*, No. April, pp. 12–17.

Whitten, J. and Bentley, L. (2005) *Systems Analysis and Design Methods*, 7th ed., McGraw-Hill.

Wong, K. and Sun, D. (2006) 'On evaluating the layout of UML diagrams for program comprehension', *Software Quality Journal*, Vol. 14, No. 3, pp. 233–259.

Zaman, Q. uz, Nadeem, A. and Sindhu, M.A. (2020) 'Formalizing the use case model: A model-based approach', *PLoS ONE*, Vol. 15, No. 4, pp. 1–30.