Evaluation of the ROBIX UML Web Modeler: Assessing the Level of Usability

Francisco Antonio Mejía Domínguez[1a], Brianda Raquel Campoy Esquer[1b], Ramón René Palacio Cinco[1c], Gilberto Borrego[2d], Manuel Alejandro Quintana García[2e]
[1]Unidad Navojoa, Instituto Tecnológico de Sonora
[2]Unidad Obregón, Instituto Tecnológico de Sonora
{[a]francisco.mejia216677, [b]brianda.campoy216578, [e]manuel.quintana133313}@potros.itson.edu.mx,{[c]ramon.palacio, [d]gilberto.borrego}@itson.edu.mx

***Abstract* — Robustness diagrams play a vital role in system development to ensure the alignment with customer requirements. However, the lack of specialized tools tailored for creating such diagrams has been a significant challenge. In response, this paper presents ROBIX, a purpose-built tool designed specifically for robustness diagrams that offers unique features, including the prevention of invalid connections between nodes. This study aims to evaluate the usability of ROBIX in comparison to a traditional tool, StarUML. We had two different groups of students from an academic institution, using between subjects study design.. Usability was evaluated using a Technology Acceptance Model (TAM) questionnaire. The results showed significant differences between the two groups. ROBIX achieved a higher average of perceived usability score compared to StarUML. Participants highlighted the productivity-enhancing aspects of ROBIX, such as improved organization and cloud project storage, confirming its potential to increase efficiency in system development processes. In terms of ease of use, ROBIX Modeler outperformed StarUML, receiving positive feedback from the participants who praised its simplicity and the clarity of its design and interface. The use of the t-student test to compare the means of the two groups confirmed the significant differences observed, further supporting the conclusion that ROBIX has superior usability. Valuable feedback from the participants included suggestions for improvements, such as eliminating errors in dragging functions and incorporating keyboard shortcuts, to enhance the overall user experience and efficiency of ROBIX.**

Keywords - UML; RUP; ICONIX; robustness diagram; modeler; usability.

I. Introduction

Robustness diagrams are important in software development because performing a robustness analysis helps us to ensure that use cases are robust enough to represent the usage requirements of the system being developed. However, UML has yet to add robustness diagrams to the UML standard but has allowed them as a UML version for elaborating processes in RUP and ICONIX [1]. These diagrams are necessary to be supported because they promote the diffusion of these processes. It also helps us identify potential objects or object responsibilities to support the logic requested in use cases. It effectively acts as a bridge to other diagrams, such as UML sequence diagrams and UML class diagrams.

Given the importance of these diagrams, the tool in which they are created must not only contain the necessary elements for the realization of the diagrams, but also facilitate their creation for users in terms of usability and help to make as few mistakes as possible.

There are different applications to help in the robustness diagrams creation, such as StarUML or Visual-Paradigm. Although these applications have an extensive repertoire of elements for developing different types of diagrams, they lack certain essential features to make this kind of diagram, such as applying the rules between the elements, which are very important either when someone is learning to make these diagrams, because they avoid errors and bad practices, or in development, which can have errors and cause problems in the future. For example, StarUML has the option to make diagrams, but it is a desktop application, which has the disadvantage that the work done in the application is saved locally. Nowadays, many applications are online and allow team collaboration. On the other hand, visual-paradigm is a web application with advantages over StarUML because it does not require installing any application. However, it is not aware of the rules to create a robustness diagram . It is worth mentioning that this is not only happening with StarUML and Visual-Paradigm, it is instead, in most cases, these applications try to cover as many types of diagrams as possible, which leads them to have numerous diagram-creation elements that, due to the nature of the application, can be combined even though they do not even belong to the same diagram type, thus allowing the user to create diagrams that completely ignore the rules of the language. Therefore, an application called ROBIX was created, which has essential features for creating robustness diagrams. However, since it is a new application, it will be tested in front of a group of software engineering students to measure its usability level to know whether it has enough usability to be considered a possible option for users.

This article seeks to know the level of usability of ROBIX compared with StarUML, which is another tool of the same type. It is proposed by subjecting both tools to the same usability tests in two groups with similar characteristics. The test is intended to identify the ease of use of the tools and their usefulness.

II. Theoretical Framework

A. UML Modeling

In the ever-expanding digital landscape, software has permeated every aspect of our lives, becoming an essential component of our daily routines. As a result, the demand for skilled software developers has surged, accompanied by an increasing need for a deeper understanding of complex systems. Within the realm of software development, modeling has emerged as a critical aspect, enabling developers to conceptualize and design intricate systems effectively. To address this demand, the UML (Unified Modeling Language) was conceived by 1994 but it was not until 1997 that it became a standard. UML is a modeling language created to forge a visual modeling language. The fact that it is a common and unified language has facilitated the processes in many companies since it provides an easy way to capture the limits, structures and behavior of a system [2]. In terms of specific applications, UML describes the notation for classes, components, nodes, activities, workflows, use cases, objects, states and relationship models. UML also supports the idea of custom extensions such as robustness diagrams. Some of the key benefits of UML are

- Better overall development times.
- System modeling, not just software modeling.
- Establishes executable concepts and artifacts.
- Manage the scaling of complex systems.
- Better project planning and control.
- High reusability and cost minimisation.

The UML is a visual modeling language that goes beyond a simple set of graphical symbols. In fact, every symbol used in UML notation has a precise and defined meaning. This means that a developer can create a model using UML and another developer or tool can unambiguously interpret that model, thanks to the well-defined semantics behind each symbol used [3].

B. Robustness diagram

Robustness diagrams [4] are a hybrid between class diagrams and activity diagrams; they represent the behavior of a use case, showing both the classes involved and the behavior of the software. However, they do not describe which classes these activities correspond to. These diagrams are used to model the behavior of a system under different workloads and to ensure that the system is reliable, secure, and resilient. Robustness diagrams are described as a technique for representing the ability of a system to operate under different situations and conditions [3]. The basic idea of using robustness diagrams is that you can analyze the steps of a use case to validate business logic within it and ensure that the terminology is consistent with other use cases you have previously analyzed; in other words, ensure that use cases are robust enough to represent the system requirements [5].

C. ROBIX

Creating robustness diagrams in UML diagramming tools can be a difficult task, requiring a bit of ingenuity in working with the elements and options available. ROBIX, a web application that can draw robustness diagrams, was developed in React

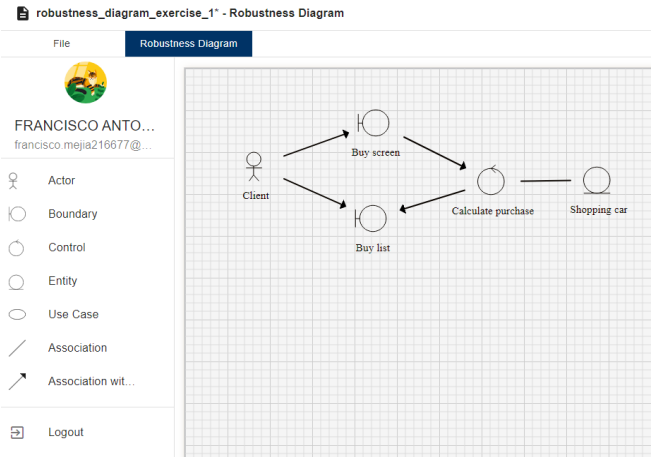

Figure 1. Overview of the ROBIX main window

using SVG plugins on the frontend and node.js and MySQL on the backend. This modeler applies the rules of robustness diagrams, thus it avoids invalid relationships, for example: creating a relationship between a boundary element and an entity element. This feature is a key differentiator from other modelers that do not apply the robustness diagram rules. ROBIX also has the ability to create projects that can be backed up with many robustness diagrams. These projects can be shared with other users of the modeler, allowing a development team to collaborate on the same project by creating or editing diagrams. Projects and diagrams are stored in the cloud, so you can work on them from anywhere.

D. Usability

Software usability is the ability of the user to understand, use and learn the software. Usability issues relate to how easy it is to use a product, i.e. they are related to the "usability" of products. More formally, the International Organization for Standardization (ISO) defines usability as "...the effectiveness, efficiency, and satisfaction with which specified users can achieve specific goals in designated environments" (ISO DIS 9241-11) [6]. There are five very important attributes for usability, which are learnability, attractiveness, compressibility, operability, and usability behavior [7]. However, there are different definitions according to research standards. Another way of approaching usability is to distinguish the possibilities of an action perceived by the user, which may be more or less easy, according to mental models and previous lived experiences, which is very close to affordance. From this we can use usability tests to measure usability empirically and then analyze the quantitative and qualitative results in terms of effectiveness, efficiency and satisfaction. Effectiveness refers to the percentage of success in the tests, which refers to participants who correctly achieve each objective; efficiency is the metric that refers to the measured time needed to complete a task; satisfaction refers to how users feel and perceive the system on a subjective and objective level, how comfortable it was, and what its acceptability would be. Usability metrics are easy to define, but difficult to collect [8]. Usability testing is a very common research technique in the UX field. During a testing session, asks a participant to perform certain tasks, which. usually

involve interaction with one or more specific user interfaces [9]. Throughout the process, the researcher closely observes the participant's behavior and pays attention to his or her comments [10]. There are three points out that there are three main categories of usability testing: exploratory, evaluative and comparative. The exploratory category is used in the early stages of product development to evaluate the effectiveness and usability of a preliminary design or prototype, and to understand users' thought processes and conceptual understanding. The evaluation category is used in the middle of the product development process or as a general evaluation of the usability of a technology. During this test, the technology is evaluated in real time to determine user satisfaction, effectiveness, and overall ease of use. Finally, the Comparison category is used to compare two or more instructional technology products or designs and to distinguish the strengths and weaknesses.

III. Method

This section outlines the research methods used in this comparative study. The following areas include: Research Type, Participants, Materials, and Procedures. Each section highlights key components like research design, participant characteristics, equipment, and step-by-step approach.

A. Type of Research

We conducted a comparative study where participants were asked to rate the usefulness and ease of use of ROBIX and StarUML. The study used a between subjects design, with two subgroups of participants, one subgroup worked with ROBIX, and the other one worked with StarUML. In this way, we have the advantage that their responses are not influenced from one system to the other, and we can compare the participants' behavior with the two tools equally.

B. Participants

The test required the participation of two groups of under-graduated students (between 18 and 23 years old) from the Instituto Tecnológico de Sonora. All the students were enrolled in the Software Engineer educational program. We had 34 participants in total, where the first group comprised 20 participants (18 males and 2 females), the second group comprised 14 students, and the second group comprised 14 participants (13 males and 1 female). All participants were in their third semester and above. and they were familiar with the creation of UML diagrams. .

C. Materials and Instruments

The materials and tools used were the following:

- **ROBIX:** Web application that allows drawing robustness diagrams, from which it is necessary to the level of usability.
- **StarUML**: Desktop application that draws diagrams. UML diagrams are used as a reference for measuring the usability of the modeler.
- **Exercises:** A set of activities to be performed in each system. The exercises were similar for both software.
- **TAM questionnaire**: An instrument used to measure the usability and usefulness of both programs.

A. Procedure

To carry out the tests, we requested the participation of 2 subgroups of students. In the first subgroup there were 20 participants, while in the second one there were 14 participants. Next, we explained to them about the activity and their participation, as well as we explained about the robustness diagrams, and their usage to make it easier for them to understand the tasks they would perform. Then, we provided each participant access to the appropriate modeling software on their computer, along with a document that describes the steps required to use all of the diagram elements to create the diagram shown in Figure 1.

At the end of the tasks, each student was asked to answer a questionnaire based on the Technology Acceptance Model (TAM) [11] (with Likert-5 scale, from 1 (strongly disagree) to 5 (strongly agree)) to know their perception of the system's usability and ease of use. In addition, the students were asked to add comments indicating which parts of the system they liked or found pleasant and what they did not like or found unpleasant when using it.

IV. Results

Table I shows the results obtained from questions 1 to 6, which cover the application usefulness, and Table II from questions 7 to 12, which concern the user's ease of use perception.

These results are used to test whether ROBIX Modeler has a higher and better usability than a traditional tool like StarUML. In this case, the mean of the perceived usability in ROBIX was 3.72, while in StarUML, it was 3.22. As it can be seen, ROBIX is superior, and when we analyzed the data in detail, most of the participants think that ROBIX increases their productivity, and this can be noticed not only in Table I but also in the comments: "...it could help me to have a better organization...[participant 4]", "...save your projects in the cloud...[participant 14]".

TABLE I. Table of application utility

Question	***ROBIX***	***StarUML***
1. Using the system in my work would allow me to complete my tasks more quickly	3.6 (0.94)	3.1 (0.99)
2. Using the system would improve my job performance	3.8 (0.95)	3.3 (1.04)
3. Using the system would improve my effectiveness at work.	3.6 (0.99)	3.2 (1.03)
4. Using the system would increase my productivity.	3.95 (0.88)	3.2 (1.04)
5. Using the system would make my job easier.	3.7 (1.08)	3.3 (1.04)
6. I would find the system helpful in my work.	3.7 (1.03)	3.0 (1.2)
Average utility grade	3.72 (0.98)	3.22 (1.08)

Note. The first value is the mean value of the Likert scale responses. and the result in parentheses is the standard deviation from the mean value.

Regarding ease of use, ROBIX scored 3.81, much higher than StarUML, which scored only 3.08. As we can see, there is a big difference, which is also reflected in the participants'

comments: "...The design is simple and easy to understand... [Participant 8]", "...Simple interface... [Participant 8]".

Table II shows the results related to the usability of both applications.

TABLE II. TABLE OF EASE OF USE

Question	*ROBIX*	*StarUML*
7. I would find it easy to learn how to use the system.	3.9 (0.99)	3.5 (1.06)
8. I would find it easy to get the system to do what I want.	3.7 (0.91)	3.0 (0.96)
9. Interacting with the system would be straightforward to understand.	3.6 (0.98)	2.8 (1.06)
10. I would find it flexible to interact with the system	3.6 (1.09)	3.0 (1.06)
11. I would find it easy to learn how to use the system.	4.1 (1.11)	3.3 (1.11)
12. I would find it easy to use the system.	3.8 (1.03)	2.7 (0,96)
Average ease of use grade	3.81 (1.02)	3.08 (1.03)

Finally, Table III shows the results obtained by comparing the usefulness and ease of use of the two applications.

TABLE III. TABLE OF USABILITY

Question	*ROBIX*	*StarUML*
Application utility	3.72 (0.98)	3.22 (1.08)
Ease of use	3.81 (1.03)	2.7 (0.96)
Average usability grade	3.76 (1.00)	3.15 (1.05)

We applied a Kolmogorov-Smirnov normality test which reveals that the collected data were not adjusted to a normal distribution. This prompted us to employ the Mann-Whitney U test to determine whether the difference in perceived usefulness was significant. After we applied this test, we noticed that the perceived usefulness difference was statistically significant ($P = 0.00096$) with a z-score of 3.29757. Similarly, the ease of use exhibited a highly significant difference ($P < 0.00001$) with a z–score of 4.73753. Some exciting suggestions regarding the usefulness and ease of use were, "...the function of dragging things has a bug.... [participant 9]", "...maybe keyboard shortcuts I would have liked.... [participant 16]", "...how irresponsible the associative functions become... [participant 17]".

V. CONCLUSION

In this work, ROBIX, a software that helps to draw robustness diagrams, has been presented to know its usability level, applying TAM tests to a group of under-graduated students. Its usability was found to be highly favorable and relevant concerning other, more traditional modeling systems.

We observed that although other UML modelers have more years of development than ROBIX and many more tools in the design of robustness diagrams, more modelers are needed for users to develop this type of diagram in the system. Still, presenting so many tools makes it more difficult for users to find the correct elements for realizing a specific type of diagram. Thus, we learned that simplified and task-focused systems are more efficient and more accessible for them to learn to use.

Several studies related to the paradox of choice can support the above-stated above. Robustness diagrams are essential in software engineering because they help to visualize how different system components interact and how the system performs in a given situation. It allows software engineers to understand the system's work, identify potential problems, and develop appropriate solutions [12].

Robustness diagrams are particularly suitable for software development in agile methodologies [13] because their simplicity makes them easy and quick to design, which saves a lot of time in the documentation process. Using robustness diagrams is essential in several areas of software engineering education, including software design, software architecture, and requirements engineering. Creating and understanding robustness diagrams is necessary for building efficient and scalable software systems. Given the importance of robustness diagrams in software engineering education, using ROBIX as a learning tool could have a positive impact since it is straightforward, facilitating students' learning.

ACKNOWLEDGMENT

This work was supported by the “Programa de Fortalecimiento a la Investigación 2023” (project numbers: PROFAPI 2023_063 and PROFAPI 2023-0397) of the Instituto Tecnológico de Sonora.

REFERENCES

[1] D. Rosenberg and M. Stephens, *Use Case Driven Object Modeling with UML: Theory and Practice*. Apress, 2008. [Online]. Available: https://books.google.com.mx/books?id=fPwlCD5JtaMC

[2] C. B. Reynoso, *Introducción a la Arquitectura de Software*, Universidad de Buenos Aires, 2004.

[3] G. Booch, *El lenguaje UML es un estándar OMG diseñado para visualizar, especificar, construir y documentar software orientado a objetos,* in Diseño y calidad de software, J. R. Hilera & M. J. Sánchez-Segura (Eds.), Universidad de Granada, 2010, pp. 371-380. Available: http://elvex.ugr.es/decsai/java/pdf/3e-uml.pdf

[4] D. Rosenberg and M. Stephens, Use Case Driven Object Modeling with UML: Theory and Practice. Apress, 2008. [Online]. Available: https://books.google.com.mx/books?id=fPwlCD5JtaMC

[5] C. Cornejo, *Diagramas de robustez*, Assembla.com, Apr. 6, 2011. [Online]. Available: https://app.assembla.com/wiki/show/iw_10-11/Diagramas_de_robustez

[6] P. W. Jordan, *An Introduction to Usability*, London, CRC Press, 1998.

[7] A. A. Almazroi, *A systematic mapping study of software usability studies*, International Journal of Advanced Computer Science and Applications: IJACSA, vol. 12, no. 9, 2021. DOI: 10.14569/ijacsa.2021.0120927

[8] J. Nielsen, *Nielsen Norman Group*, Jan. 20, 2001. [Online]. Available: https://nngroup.com/articles/usability-metrics/

[9] Moran, K.. *Nielsen Norman Group*. Dec. 1, 2019. [Online]. Available: https://nngroup.com/articles/usability-testing-101/

[10] J. Cáceres, *Apañados*, Nov. 6, 2012. [Online]. Available: https://apañados.es/319-introduccion-a-pruebas-de-usabilidad-web

[11] F. D. Davis, “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology,” MIS Q., vol. 13,no. 3, pp. 319–340, 1989, doi: 10.2307/249008.

[12] B. Schwartz, *The Paradox of Choice: Why More Is Less*, Sage Publications, 2017. [Online]. Available: https://journals.sagepub.com/doi/book/10.4135/9781452220681

[13] B. Selic, “Agile Documentation, Anyone?,” IEEE Softw., vol. 26, no. 6, pp. 11–12, 2009, doi:10.1146/annurev.ps.29.020178.002001.