Level of detail in UML models and its impact on model comprehension: a replication study

Ariadi Nugroho[1], Michel R.V Chaudron[2]
[1]Department of Computer Science, BINUS Graduate Program, Bina Nusantara University, Jakarta, Indonesia
[2]Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, Netherlands

Article Info

Article history:

Received Jun 17, 2025
Revised Dec 22, 2025
Accepted Mar 4, 2026

Keywords:

Level of detail
Model comprehension experiment
Model quality
UML model

ABSTRACT

This replication study examines the impact of level of detail (LoD) in unified modeling language (UML) on model comprehension, replicating a controlled experiment, which involved 53 MSc students at Eindhoven University of Technology. Using the same UML model and experimental design, we conducted the study with 23 MSc Computer Science students at Bina Nusantara University, Indonesia. Consistent with the original findings, higher LoD was found to enhance comprehension correctness. However, the effect on comprehension efficiency was weaker and not statistically significant, likely due to the smaller sample size and contextual differences in subjects' backgrounds. Furthermore, we found a potential disconnect between perception and actual comprehension performance in the subjects receiving UML model with low LoD. Specifically, while they viewed the model favourably, their actual understanding may have been impaired by the limited information and therefore the perceived clarity and ease of comprehension are not reflective of the true comprehension. Overall, this study reinforces the importance of LoD in UML modeling and highlights the need for further replication, particularly in contexts involving professional software engineers.

Corresponding Author:

Ariadi Nugroho
Department of Computer Science, BINUS Graduate Program, Bina Nusantara University
Jakarta, Indonesia
Email: ariadi.nugroho@binus.ac.id

1. INTRODUCTION

The unified modeling language (UML) is a standard notation widely adopted for visualizing, specifying, and documenting software systems. UML plays a critical role in both software engineering research and industrial practice. Previous studies, for example in [1], [2] have highlighted diverse modeling styles and levels of rigor in using UML; however, few have examined how these variations impact development outcomes such as model comprehension. Our previous study [3], involving 53 MSc students at Eindhoven University of Technology, demonstrated that a higher level of detail (LoD) in UML class and sequence diagrams significantly improves model comprehension, both in terms of correctness (percentage of correct answers) and efficiency (correct answers per unit time). This replication study aims to validate those findings by repeating the experiment with 23 MSc students from Bina Nusantara University, Indonesia.

In practice, the variation in UML modeling rigor manifests in differing degrees of completeness, granularity, and proportion within models. Despite its relevance, the effect of modeling style, particularly LoD, on model comprehension remains underexplored. In our previous study [3], we reviewed early works on UML visualization (e.g., Purchase et al. on class diagram notations [1]), comparative analyses of diagram types (e.g., Otero and Dolado on sequence vs. collaboration diagrams [2]), and modeling rigor (e.g., Briand et al. on OCL constraints [4]). At the time, the role of LoD in UML diagrams-that is, as a potential factor affecting model comprehension, was still largely unexplored. Since 2009, however, research on UML comprehension has evolved significantly. Recent work in this area can be categorized into three research areas: (1) the impact of the level of detail in UML diagrams, (2) the impact of using different UML diagram types, and (3) the impact of diagram layout and visualization techniques.

Fernandez-Saez *et al.* [5] investigated how LoD in use case, sequence, and class diagrams influences software maintenance. Using two Java systems and 11 student participants, they found a slight tendency toward better results with low-LoD diagrams. However, due to the small sample size, these results were considered preliminary. A series of related experiments by Scanniello et al. focused on how UML analysis and design models influence code comprehension and modification tasks [6]-[9]. Across 12 controlled experiments with participants of varying expertise, they found that analysis-phase UML models may hinder code comprehension and increase task completion time, while design-phase models improve comprehension outcomes. Another dimension of LoD involves the use of UML stereotypes. Cruz-Lemus *et al.* [10] evaluated the impact of stereotypes on sequence diagram comprehension, using the cognitive theory of multimedia learning (CTML) as a theoretical basis. Their experiments revealed that stereotypes significantly aid semantic comprehension and retention, especially among domain novices. Ricca *et al.* [11] found that while stereotypes do not generally improve comprehension, they help reduce the performance gap between less experienced and more experienced developers. Similar investigations on UML stereotypes include works by [12]-[14].

Several studies have assessed whether specific types of UML diagrams affect model comprehension. Torchiano *et al.* [9] examined the benefit of adding UML object diagrams to class diagrams. In a family of four controlled experiments with undergraduate and graduate students, they found that object diagrams improved comprehension only among more experienced participants, with no measurable time advantage. Felderer et al. compared activity diagrams and state machines in the context of test case derivation [15]. Their experiment with 84 students showed that while activity diagrams were more comprehensible, they led to more errors. The findings suggest that model understanding and error-prone behaviour in test design are not necessarily correlated. Similarly, Abrahao et al. conducted five experiments involving 112 participants (students and professionals) to assess the impact of UML sequence diagrams on understanding functional requirements. Results indicated that sequence diagrams significantly enhance comprehension for high-ability and experienced users [16].

Another critical factor in UML comprehension is diagram layout. Examined how diagram size, used as a proxy for layout complexity, affects understanding [17]. The findings showed a negative correlation between diagram size and performance, leading to guidelines recommending 20–60 elements per diagram for optimal comprehension. Sharif *et al.* [18] extended this work by evaluating different layout strategies in UML class diagrams. They found that multi-cluster layouts improve comprehension accuracy, reduce completion time, and lower visual effort, especially for complex modeling tasks.

While there have been many studies conducted to evaluate the impact of different styles of using UML on software development and maintenance, we argue that the results are far from conclusive. Therefore in this study we extend previous research by replicating a controlled experiment on LoD originally conducted in [3]. Similar to the original study, LoD is defined as the amount of information used to represent UML modeling elements. For sequence diagrams, a message could be an informal label, a method name, or a method with parameters. For class diagrams, LoD includes class attributes, operations, association names, directionality, and multiplicity. Low LoD uses minimal elements (e.g., class names, basic associations), while high LoD adds detailed specifications. By conducting this replication experiment, we aim to validate findings in our original study in a different experimental context.

Replication serves a crucial role in strengthening empirical evidence by validating previous findings across different contexts and conditions [19]. Unfortunately, replication is relatively rare in software engineering research [20]. By conducting this study, we aim to contribute in addressing the well-recognized scarcity of replication studies in software engineering research, which are essential for validating the generalizability and robustness of empirical findings. Nevertheless, a successful replication does not imply identical outcomes; even differing results can yield valuable insights and contribute to a deeper understanding of the studied phenomena.

2. METHOD

This study involved 23 MSc students in Computer Science at Bina Nusantara University, Jakarta, Indonesia, in 2024. The experiment was a mandatory assignment with adjusted grading to account for LoD treatments. Subjects had basic UML training through coursework, comparable to the original subjects. Following the definition in [19], this paper presents an exact and dependent replication. An exact replication attempts to follow the original study's procedures as closely as possible, while a dependent replication retains the same or similar experimental conditions.

2.1. Variables in the experiment

Identical to the original experiment, the independent variable was LoD (low LoD vs. high LoD), manipulated by varying information in UML class and sequence diagrams. The dependent variable in the experiment was model comprehension. Model comprehension was defined as the ability of the subjects to understand concepts/constructs described in a UML model. We defined two aspects of model comprehension, namely comprehension correctness and comprehension efficiency:

- Comprehension correctness: Percentage of correct answers to a 15-question questionnaire.
- Comprehension fficiency: Number of correct answers divided by total time spent.

Both correctness and efficiency were measured in a ratio scale. We use the term comprehension correctness and comprehension efficiency to refer to correctness and efficiency, respectively.

2.2. Hypotheses formulation

In replicating the original study on the role of LoD in UML models, we adopted the same hypotheses to examine its impact on model comprehension.

Hypothesis 1 (Comprehension Correctness)

- H1,null: There is no significant difference in comprehension correctness between subjects working with UML diagrams modeled with high versus low LoD.
- H1,alt: The use of UML diagrams with high LoD significantly improves subjects' comprehension correctness.

Hypothesis 2 (Comprehension Efficiency)

- H2,null: There is no significant difference in comprehension efficiency between subjects working with UML diagrams modeled with high versus low LoD.
- H2,alt: The use of UML diagrams with high LoD significantly improves subjects' comprehension efficiency.

Note that we stated one-tailed hypotheses because we had prior predictions that LoD in UML diagrams will increase both comprehension correctness and comprehension efficiency.

2.3. Experiment instruments

This section presents the materials used in the replication experiment. We begin by describing the UML model artifacts, followed by the model comprehension questionnaire, background questionnaire, and feedback questionnaire. All materials used in this replication were based on those from the original study.

2.3.1. The UML model

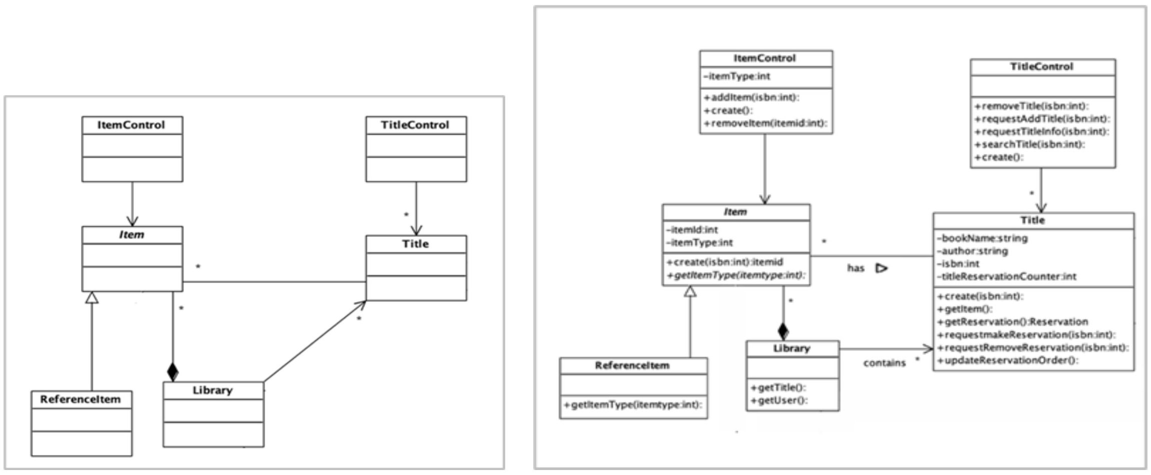
The subject of the replication experiment was a UML model representing a library system, originally adapted from the model described in [21]. Each subject was provided with a document containing the UML model. To examine the effects of LoD on model comprehension, two versions of the UML model were created (each model consisted of 23 UML diagrams), namely Model M-Low, which presents the library system with a lower level of detail, and Model M-High, which presents the same system with a higher level of detail. Four types of UML diagrams were used in the experiment. However, the experimental treatments were only applied to class and sequence diagrams. This decision reflects the findings in [22], which identified class and sequence diagrams as the most frequently used UML diagram types in practice.

Table 1 summarizes the LoD treatments applied in the experiment. For class diagrams, model M-high includes attributes, operations, and labeled associations, whereas model M-low omits these details. Some associations were labeled in m-low when essential for basic system understanding, to preserve information sufficiency. Across both versions, 20 classes were modeled. Model M-high included 23 attributes and 123 operations, 14% of which were simple getter methods, often present in entity classes. Although getters are generally trivial, they were retained in some cases to support realistic interactions in the corresponding sequence diagrams. For sequence diagrams, treatments focused on the details of messages. In m-high, messages precisely reflect the operations defined in class diagrams, including parameters and return values. In contrast, M-low used dummy messages-that is, simple text labels without parameters or return types. Aside from the treatments listed, all other aspects (e.g., diagram layout) were kept consistent across both versions. Sequence diagrams were linked to use cases to illustrate how system functionality is executed via object interactions. The class diagrams provided static representations to support these dynamic views.

Figure 1 shows an example of a class diagram represented using different levels of detail. A class diagram with Low LoD omits class attributes and methods. On the other hand, a class diagram with high LoD specifies all class attributes, methods, and other details such as method signatures and parameters.

Table 1. LoD treatments in the UML model treatments

Diagram types	Model elements	M-low	M-high	# Diagrams
Package diagram	Package name	Yes	Yes	1
Use case diagram	Use case name	Yes	Yes	2
	Actor name	Yes	Yes	
Class diagram	Class attributes	No	Yes	3
	Class operations	No	Yes	
	Association labels	No	Yes	
Sequence diagram	Real method names	No	Yes	17
	Message parameters	No	Yes	
	Message returns	No	Yes	



(a) (b)

Figure 1. Example of UML class diagram with (a) low LoD and (b) high LoD

2.3.2. Model comprehension questionnaire

A model comprehension questionnaire was used to assess subjects' understanding of the UML model. The questionnaire included questions specifically related to the library system and required subjects to derive their answers based solely on the information provided in the UML specifications. Its structure and content were aligned with those used in the original study to maintain consistency. To minimize ambiguity and potential bias, we followed questionnaire design guidelines as discussed by Oppenheim [23]. The question sheet contained 15 multiple-choice questions. Each question was composed of three main parts: (1) a diagram reference, (2) the question and options, and (3) a remarks section.

2.3.3. Background and feedback questionnaire

As in the original experiment, we employed the same background questionnaire to assess subjects' prior knowledge and experience across four key domains: object-oriented design, object-oriented programming, UML, and familiarity with library systems. These factors were identified as potential confounding variables that might influence subjects' performance in the model comprehension task. Subjects' self-assessed knowledge and experience were measured using 10 Likert-scale items. An even-point Likert scale was chosen to reduce central tendency bias by discouraging the selection of a neutral middle point. This design encourages subjects to make more deliberate self-assessments.

In addition, subjects were asked to complete a feedback questionnaire about their experience during the experiment. The same instrument as used in the original study was employed. This questionnaire aimed to gather qualitative and quantitative feedback that could inform further analysis and potential improvements to the experimental design. Subjects were asked to evaluate the UML model they received in terms of its perceived complexity, comprehensibility, consistency, detailedness, and clarity. They were also asked to rate various aspects of the experimental setup from their individual perspective. Except for the open-ended comment section, all items were assessed using a Likert scale.

2.4. Experimental design

As in the original study, this replication followed a single-factor design with two treatments using a completely randomized design [24]. Each subject was randomly assigned to one treatment and interacted with a single object (i.e., UML model), following a between-subjects design. This design choice was primarily driven by time constraints, as the experiment had to be completed in a single two-hour session. The experimental object was a UML model representing a library system. Subjects were randomly assigned to one of two treatment groups: Low LoD (L-LoD) or High LoD (H-LoD). Those receiving the less detailed model were placed in the L-LoD group, while those given a more detailed model were assigned to the H-LoD group. Both groups participated in the experiment in the same room, without any physical separation or visible group identifiers. This was done to reduce bias and avoid alerting subjects to the existence of multiple treatment conditions. To ensure statistical balance and simplify the subsequent analysis [24], we aimed for equal group sizes. The final group sizes were 11 for LLoD and 12 for H-LoD.

Each subject received two materials: (1) a UML model of the library system (varying in level of detail), (2) an online questionnaire consisting of a model comprehension questionnaire, background questionnaire, and feedback questionnaire (all are identical across both treatment groups). The model comprehension questionnaire was the first task aimed at assessing subjects' understanding of the UML model. Following the comprehension task, subjects completed the background questionnaire, which assessed their prior knowledge and experience related to object-oriented analysis and design, object-oriented programming, UML, and library systems. Lastly, the feedback questionnaire collected subjective impressions of the experiment from each subject.

The experiment was conducted in a single session, starting at 09:00 and lasting for 90 minutes. Subjects were instructed to complete all tasks within this time frame. Subjects were randomly assigned to treatment groups at the start. A brief orientation was provided to explain the procedure, with written instructions also included in the model comprehension questionnaire. Unlike the original paper-based experiment, this replication used online questionnaires equipped with time-tracking features. The UML models were also provided digitally via an online repository. Upon completion, all responses were collected, preprocessed, and stored in a spreadsheet for analysis.

2.5. Analysis method

The analysis began with manual preprocessing of the raw data collected from the questionnaires. Once processed, the data were imported into PSPP [25] for statistical analysis. The next step involved data exploration, including outlier detection and checking assumptions for statistical testing. Since the hypotheses focused on comparing group performance, we applied tests for differences between two independent groups: independent t-tests for parametric assumptions and Mann-Whitney tests otherwise. Given the higher statistical power of parametric tests, we prioritized them when assumptions were met. Normality and homogeneity of variance were assessed using the Shapiro-Wilk and Levene's tests, respectively. If normality was violated, data were normalized using area transformation [26]. A significance level of $\alpha = 0.05$ was used for all hypothesis tests. To assess the potential impact of subjects' background knowledge and experience on performance, we conducted a two-way ANOVA using the same significance threshold. Finally, qualitative responses–that is, subjects' written justifications for their answers were analyzed manually to identify patterns and reasoning differences across groups.

3. RESULTS AND DISCUSSION

To evaluate the effect of LoD on comprehension correctness and efficiency, we conducted an independent samples t-test. The summary statistics are presented in Table 2 and the test results are in Table 3.

3.1. Testing hypothesis 1: effect of LoD on comprehension correctness

As shown in Table 2, the H-LoD group exhibited higher comprehension correctness than the L-LoD group. The mean value for H-LoD exceeded that of L-LoD, indicating a consistent advantage for greater detail. To evaluate the statistical significance of this difference, we conducted an independent samples t-test. The H-LoD group achieved a mean comprehension correctness of 64.46 (Std. error mean = 3.70), compared to 52.13 (Std. error mean = 4.21) for the L-LoD group. The t-test results in Table 3, focusing on the row comprehension correctness, confirmed this difference to be statistically significant (p = 0.02, one-tailed), indicating that the observed effect is unlikely due to chance. Based on these results, we reject the null hypothesis (H1,null) and accept the alternative hypothesis (H1,alt): The use of UML diagrams with higher LoD significantly improves comprehension correctness.

3.2. Testing hypothesis 2: effect of LoD on comprehension efficiency

As shown in Table 2 there is no substantial difference in comprehension efficiency mean values between the H-LoD and L-LoD groups. Specifically, the H-LoD group achieved a mean efficiency of 0.18 (Std. error mean = 0.02), while the L-LoD group scored 0.19 (Std. error mean = 0.03). Consistent with the earlier analysis, we conducted a t-test to test for statistical significance. The t-test results presented in Table 3 (see row comprehension efficiency) shows that the difference in means was not statistically significant (p = 0.40, one-tailed). Based on these results, we reject the second alternative hypothesis (H2,alt) and accept the null hypothesis (H2,null): There is no significant difference in comprehension efficiency between subjects working with UML diagrams modeled at high or low levels of detail.

Table 2. group statistics for comprehension correctness and efficiency across low and high level of detail (L-LoD And H-LoD) groups

Metric	Group	N	Mean	Std. dev.	Std. error mean
Comprehension correctness	L-LoD	11	52.13	13.95	4.21
	H-LoD	12	64.46	12.82	3.70
Comprehension efficiency	L-LoD	11	0.19	0.11	0.03
	H-LoD	12	0.18	0.06	0.02

Table 3. Independent T-Test for comprehension correctness and efficiency

Metrics	Lavene's test		T-test statistics						
	F	Sig.	t	df	Sig. (1-tailed)	M. Diff.	SE Diff.	95% CI Diff. Lower	95% CI Diff. Upper
Compr. correctness	0.67	0.42	-2.21	21.00	0.02	12.33	5.58	-23.94	-0.73
Compr. efficiency	1.49	0.23	0.25	21.00	0.40	0.01	0.04	-0.07	0.09

3.3. Subjects' background knowledge and experience

Subjects' background knowledge and experience were assessed through a subject's background questionnaire. The questionnaire consisted of 10 items rated on a 6-point ordinal scale (1: No. knowledge/experience, 6: very good). To assess whether background differences might confound the experimental results, we compared overall knowledge/experience scores across groups. Each subject's score was computed as the sum of all questionnaire responses, yielding a range of 10 to 60. Given the ordinal nature of the data, the Mann-Whitney U test was used. The results of the Mann-Whitney U test are presented in Table 4 and Table 5. As shown in Table 4 and Table 5, there was no statistically significant difference in background knowledge and experience between the two groups (p = 0.264). Since the significance level exceeds the conventional threshold of 0.05, we conclude that differences in prior knowledge and experience are unlikely to have influenced the main outcomes of this experiment.

Table 4. Ranks of background knowledge/experience scores

Group	N	Mean rank	Sum of ranks
L-LoD	11	10.36	114.00
H-LoD	12	13.50	162.00

Table 5. Mann-Whitney U test results

Statistic	Value
Mann-Whitney U	48.00
Wilcoxon W	114.00
Z	-1.12
Asymp. Sig. (2-tailed)	0.264

3.4. In-depth analyses

3.4.1. Per-question comprehension performance

To gain a deeper understanding of how the LoD treatments influenced the subjects' performance – that is, in terms of comprehension correctness, we conducted a qualitative analysis of their answers to the model comprehension questionnaire.

Figure 2 presents a comparison of correct responses across both L-LoD and H-LoD groups. Overall, subjects in the H-LoD group performed better on most questions. Notably, for five questions, namely Q2, Q3, Q7, Q11, and Q14, the H-LoD group outperformed the L-LoD group by a substantial margin (at least three points). These questions appear to be key contributors to the overall difference in comprehension scores between the two experimental conditions. We examined four of these questions in more detail, as they exhibited the most pronounced performance gaps.

- Q3: A common misconception among L-LoD subjects involved the interpretation of the pseudocode reservation.count within a sequence diagram. Many assumed that reservation referred to a class, whereas it actually denoted a conceptual entity, i.e., the total number of reservations. Although this misunderstanding was also observed among some H-LoD subjects, it was more prevalent in the L-LoD group, likely due to the lack of contextual clues in the low-detail diagrams.
- Q7: Both groups struggled to correctly identify the function of controller classes. The distribution of answers suggests limited familiarity with the model-view-controller (MVC) design pattern. However, the

L-LoD group was significantly affected, likely due to the absence of attribute and operation information in their class diagrams, which hindered their ability to reason about class roles.

- Q11: Only a few L-LoD subjects correctly identified the class responsible for instantiating the Reservation object in the Make Reservation sequence diagram. In contrast, H-LoD subjects benefited from additional visual cues, specifically the return message indicating the creation of a Reservation object by the Title class. This information was omitted in the low-detail version.
- Q14: L-LoD subjects had difficulty selecting the correct pseudo-code snippet that represents the deletion of a book title based on the sequence diagram. Common errors involved misidentifying the relevant objects or incorrectly sequencing the deletion steps. We attribute this confusion to the absence of key details in the L-LoD diagrams, namely the lack of explicit messages and parameter information.

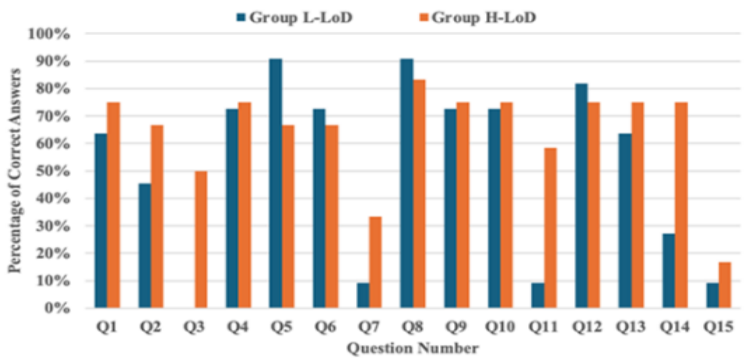


Figure 2. Scores of all questions for the UML model questionnaire

3.4.2. Subjects' feedback

In the feedback questionnaire, we asked the subjects to evaluate aspects of the UML model that comprise simplicity, comprehensibility, consistency, detailedness, and clarity. Data obtained from the questionnaire is presented in Figure 3. The figure displays the mode value of all questions, in which higher value represents better subject impression. Overall, subjects in both the L-LoD and H-LoD groups evaluated most aspects of the UML models favourably – particularly with respect to comprehensibility, consistency, detailedness, and clarity. However, a notable exception lies in the evaluation of simplicity: subjects in the H-LoD group rated the model's simplicity less favourably than those in the L-LoD group. Interestingly, despite the H-LoD group rating their model as more detailed (as expected), they perceived it as less simple. This suggests that the increase in model detail may come at the cost of perceived simplicity.

Conversely, L-LoD subjects generally gave more favourable ratings across most quality dimensions, despite their models containing less information. These results raise notable observations – that is, perceived simplicity appears to be only partially influenced by the actual LoD in the model. This prompts a broader question of whether perceived model complexity is truly dependent on the amount of information present, or if it is more closely linked to layout, structure, or individual cognitive bias. Furthermore, the positive evaluations from L-LoD subjects suggest a disconnect between perception and actual comprehension performance. Specifically, while subjects viewed the model favourably, their actual understanding may have been impaired by the limited information. This implies a form of illusion of understanding, where users are unaware that their perceived clarity and ease of comprehension are not reflective of their true comprehension.

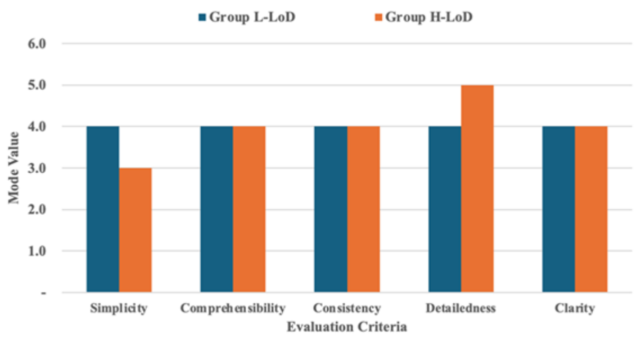


Figure 3. Subjects' perception of the UML model

3.5. Discussion

In this study, we have found that UML diagrams with high LoD significantly enhance comprehension correctness. However, the effect on comprehension efficiency was not statistically significant. In the original study, higher LoD in UML diagrams significantly improved both comprehension correctness and efficiency. We identify two reasons that might explain the weaker effect on comprehension efficiency. First, a weaker effect on comprehension efficiency might be attributed to the smaller sample size (23 subjects) compared to the original (53 subjects), which likely reduced the statistical power. Second, it is also plausible that the students at Bina Nusantara University, who had primarily theoretical exposure to UML through coursework, may not have benefited from the higher LoD to the same extent as the students at TU/e. The latter resonates with the findings of Cruz-Lemus et al. [13], which suggest that increased detail may overwhelm novice users, particularly in smaller or less experienced samples. This discrepancy in practical experience may have moderated the impact of LoD on comprehension efficiency.

From a research perspective, the result of this study resonates well with previous work conducted by the researcher [6]-[8], [10]. Further, this replication contributes to the relatively limited pool of empirical replication studies in software engineering, thereby strengthening the evidence base regarding the role of LoD in model comprehension. The results also demonstrate that prior findings are transferable across different contexts-that is, our study was conducted with a distinct subject population (Indonesian MSc students) and more than a decade after the original experiment.

From a practical standpoint, one important takeaway of this study is that software engineering educators and practitioners should prioritize including sufficient detail in UML diagrams, particularly when they are intended for communication or instructional purposes. Although simplified models may reduce visual complexity, they risk omitting critical semantic cues that aid comprehension, especially for novice or less experienced stakeholders. It is also important to note that UML models with low LoD might cause an illusion of understanding, where users are unaware that their perceived clarity and ease of comprehension are not reflective of their true comprehension.

Therefore, we suggest three practical recommendations for using UML diagrams. First, use multiple UML diagram types to represent multiple viewpoints, namely use case diagram (system requirements), sequence diagram (component interactions), and class diagrams (structural relationships). Second, specify important attributes or concepts across UML diagrams. For example, all classes in a class diagram must specify class attributes and methods essential for understanding the system. Similarly, all messages in sequence diagrams must also specify critical message parameters. Finally, maintain consistency across all UML diagrams – for example, messages appearing in a sequence diagram must also be consistently specified in the class diagram. It is also important to note that UML modeling tools could be enhanced to support flexible views, allowing users to toggle between low and high LoD presentations depending on the task or audience needs.

Considering the aforementioned points, further work is still needed to investigate LoD in UML models. In particular, we underline the importance of conducting more experimental replications to validate the results of this study. Furthermore, with the advancement of tooling that aids software engineers in creating and maintaining UML diagrams, including those powered by Generative AI, it is important to investigate the impact of LoD in such contexts.

3.6. Threats to validity

We recognize several potential threats to the validity of our findings. First, internal validity may be affected by uncontrolled factors, such as individual differences in prior UML knowledge, reasoning ability, or familiarity with the problem domain. While we ensured that all subjects had received comparable UML training, we did not assess their prior experience in depth. Second, the fact that the subjects were MSc students may limit external validity, i.e., reducing generalizability to industrial practitioners. Third, the measures for comprehension correctness and efficiency rely on a 15-question questionnaire. While the questions were carefully designed to reflect understanding of the UML model, they may not capture the full complexity of model comprehension in real-world settings, which may affect construct validity. Finally, statistical power may be limited due to the relatively small sample size, which may affect the conclusion validity. Despite these limitations, our study provides valuable insights into the role of LoD in UML model comprehension and offers a solid foundation for further empirical work in this area.

4. CONCLUSION

This study replicates an earlier experiment investigating the role of LoD in UML models on model comprehension. While the original study concluded that higher LoD significantly improves both comprehension correctness and efficiency, it also called for further replication. Our replication confirms that higher LoD improves comprehension correctness among 23 MSc Computer Science students at Bina

Nusantara University. However, the effect on comprehension efficiency was not statistically significant, likely due to the smaller sample size and contextual factors such as students' theoretical background and limited practical modeling experience. These findings reinforce the relevance of LoD and highlight the need for carefully balancing detail and clarity in UML diagrams. Based on these results, we also highlight three practical recommendations for using UML diagrams. First, use multiple UML diagram types to represent different viewpoints. Second, specify important attributes or concepts across UML diagrams. Finally, maintain consistency across all UML diagrams.

Future research should consider: (1) conducting replication studies with larger and more diverse samples, including professional software engineers, (2) investigating the impact of advanced tooling that can be used to assist software engineers in maintaining UML models of varying levels of detail, including those augmented with Generative AI. Such efforts would contribute to a deeper understanding of how visual detail supports model comprehension and inform better modeling practices in both academia and industry.

FUNDING INFORMATION

The authors state that there was no funding for the development of the research.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ariadi Nugroho	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	
Michel R.V Chaudron	✓									✓				

C :	**C**onceptualization	I :	**I**nvestigation	Vi :	**Vi**sualization
M :	**M**ethodology	R :	**R**esources	Su :	**Su**pervision
So :	**So**ftware	D :	**D**ata Curation	P :	**P**roject administration
Va :	**Va**lidation	O :	Writing - **O**riginal Draft	Fu :	**Fu**nding acquisition
Fo :	**Fo**rmal analysis	E :	Writing - Review & **E**diting		

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no financial, personal, or professional conflicts of interest that could have influenced the results or interpretations of this research.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, Ariadi Nugroho, upon reasonable request.

REFERENCES

[1] H. C. Purchase, L. Colpoys, M. McGill, D. Carrington, and C. Britton, "UML class diagram syntax: an empirical study of comprehension," in *Proceedings of the 2001 Asia-Pacific symposium on Information visualisation-Volume 9*, 2001, pp. 113–120. [Online]. Available: http://dl.acm.org/citation.cfm?id=564054

[2] M. C. Otero and J. J. Dolado, "Evaluation of the comprehension of the dynamic modeling in UML," *Information and Software Technology*, vol. 46, no. 1, pp. 35–53, Jan. 2004, doi: 10.1016/S0950-5849(03)00108-3.

[3] A. Nugroho, "Level of detail in UML models and its impact on model comprehension: a controlled experiment," *Information and Software Technology*, vol. 51, no. 12, pp. 1670–1685, Dec. 2009, doi: 10.1016/j.infsof.2009.04.007.

[4] L. C. Briand, Y. Labiche, H. D. Yan, and M. Di Penta, "A controlled experiment on the impact of the object constraint language in UML-based maintenance," in *IEEE ICSM*, 2004, pp. 380–389, doi: 10.1109/ICSM.2004.1357823.

[5] A. M. Fernández-Sáez, M. Genero, and M. R. V. Chaudron, "Does the level of detail of UML models affect the maintainability of source code?," in *International Conference on Model Driven Engineering Languages and Systems*, 2012, pp. 134–148. doi: 10.1007/978-3-642-29645-1_15.

[6] G. Scanniello, C. Gravino, M. Genero, J. A. Cruz-Lemus, and G. Tortora, "On the impact of UML analysis models on source-code comprehensibility and modifiability," *ACM Transactions on Software Engineering and Methodology*, vol. 23, no. 2, pp. 1–26, Mar. 2014, doi: 10.1145/2491912.

[7] G. Scanniello *et al.*, "Studying the effect of UML-based models on source-code comprehensibility: results from a long-term investigation," in *Product-Focused Software Process Improvement*, vol. 9459, 2015, pp. 311–327. doi: 10.1007/978-3-319-26844-6_23.

[8] G. Scanniello *et al.*, "Do software models based on the UML aid in source-code comprehensibility? aggregating evidence from 12 controlled experiments," *Empirical Software Engineering*, vol. 23, no. 5, pp. 2695–2733, Oct. 2018, doi: 10.1007/s10664-017-9591-4.

[9] M. Torchiano, G. Scanniello, F. Ricca, G. Reggio, and M. Leotta, "Do UML object diagrams affect design comprehensibility? Results from a family of four controlled experiments," *Journal of Visual Languages and Computing*, vol. 41, pp. 10–21, Aug. 2017, doi: 10.1016/j.jvlc.2017.06.002.

[10] J. A. Cruz-Lemus, M. Genero, D. Caivano, S. Abrahão, E. Insfrán, and J. A. Carsí, "Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A family of experiments," *Information and Software Technology*, vol. 53, no. 12, pp. 1391–1403, Dec. 2011, doi: 10.1016/j.infsof.2011.07.002.

[11] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato, "How developers' experience and ability influence web application comprehension tasks supported by UML stereotypes: A series of four experiments," *IEEE Transactions on Software Engineering*, vol. 36, no. 1, pp. 96–118, Jan. 2010, doi: 10.1109/TSE.2009.69.

[12] B. Sharif and J. I. Maletic, "An empirical study on the comprehension of stereotyped UML class diagram layouts," in *IEEE International Conference on Program Comprehension*, IEEE, May 2009, pp. 268–272. doi: 10.1109/ICPC.2009.5090055.

[13] M. Genero, J. A. Cruz-Lemus, D. Caivano, S. Abrahão, E. Insfran, and J. A. Carsí, "Assessing the influence of stereotypes on the comprehension of UML sequence diagrams: A controlled experiment," in *Model Driven Engineering Languages and Systems*, vol. 5301 LNCS, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 280–294. doi: 10.1007/978-3-540-87875-9_20.

[14] M. Staron, L. Kuzniarz, and C. Wohlin, "Empirical assessment of using stereotypes to improve comprehension of UML models: A set of experiments," *Journal of Systems and Software*, vol. 79, no. 5, pp. 727–742, May 2006, doi: 10.1016/j.jss.2005.09.014.

[15] M. Felderer and A. Herrmann, "Comprehensibility of system models during test design: a controlled experiment comparing UML activity diagrams and state machines," *Software Quality Journal*, vol. 27, no. 1, pp. 125–147, Mar. 2019, doi: 10.1007/s11219-018-9407-9.

[16] S. Abrahão, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora, "Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments," *IEEE Transactions on Software Engineering*, vol. 39, no. 3, pp. 327–342, Mar. 2013, doi: 10.1109/TSE.2012.27.

[17] H. Störrle, "On the impact of layout quality to understanding UML diagrams: Size matters," in *Model-Driven Engineering Languages and Systems*, vol. 8767, 2014, pp. 518–534. doi: 10.1007/978-3-319-11653-2_32.

[18] B. Sharif, "Empirical assessment of UML class diagram layouts based on architectural importance," in *IEEE International Conference on Software Maintenance, ICSM*, IEEE, Sep. 2011, pp. 544–549. doi: 10.1109/ICSM.2011.6080828.

[19] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in Empirical Software Engineering," *Empirical Software Engineering*, vol. 13, no. 2, pp. 211–218, Apr. 2008, doi: 10.1007/s10664-008-9060-1.

[20] A. Brooks, M. Roper, M. Wood, J. Daly, and J. Miller, "Replication's role in software engineering," in *Guide to Advanced Empirical Software Engineering*, London: Springer London, 2008, pp. 365–379. doi: 10.1007/978-1-84800-044-5_14.

[21] L. Briand and Y. Labiche, "A UML-Based Approach to System Testing," *Software and Systems Modeling*, vol. 1, no. 1, pp. 10–42, Sep. 2002, doi: 10.1007/s10270-002-0004-8.

[22] B. Dobing and J. Parsons, "How UML is used," *Communications of the ACM*, vol. 49, no. 5, pp. 109–113, May 2006, doi: 10.1145/1125944.1125949.

[23] A. N. Oppenheim, *Questionnaire design and attitude measurement*. Oxford, England: Basic Books, 1966.

[24] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in software engineering: An introduction*, no. 2. 2000. doi: 10.1016/s0898-1221(00)90203-7.

[25] "GNU PSPP Statistical Analysis Software, Version 2.0.1." [Online]. Available: https://lists.gnu.org/archive/html/info-gnu/2024-03/msg00004.html

[26] D. J. Krus and P. H. Krus, "Lost: McCall's T scores: Why?," *Educational and Psychological Measurement*, vol. 37, no. 1, pp. 257–261, Apr. 1977, doi: 10.1177/001316447703700134.

BIOGRAPHIES OF AUTHORS

Ariadi Nugroho is a senior researcher and lecturer at the Department of Computer Science, BINUS Graduate Program, Bina Nusantara University, Jakarta, Indonesia. He holds a Ph.D. degree in Computer Science from Leiden University, where he specialized in model-driven software development. His research spans empirical software engineering, software architecture, and technical debt management. With a career that bridges academia and industry, Ariadi brings a unique combination of scholarly rigor and real-world insight. He is the founder and CEO of KED Consulting (ked-consulting.com), a Jakarta-based advisory firm that helps organizations align technology strategies with business goals. Prior to this, he held senior IT leadership roles across the consulting, banking, and telecom sectors, giving him hands-on experience in architecting and managing complex systems in dynamic business environments. Ariadi is a member of IEEE and IEEE Computer Society. He can be reached at ariadi.nugroho@binus.ac.id.

Michel R.V Chaudron is Full Professor and Chair of the Software Engineering group at the TU Eindhoven which is part of the Department of Mathematics and Computer Science. Prior to this, he worked at Universities in Gothenburg (Chalmers|GU), Leiden and Eindhoven in the Netherlands. He obtained his Ph.D. in the area of formal methods and programming calculi for parallel computing. His research interests are in software architecture, software design, software modeling with a special focus on UML, software composition and knowledge sharing. Recently, use of AI (Artificial Intelligence) for Software Development. He has an interest in empirical studies in software engineering especially in the aforementioned areas and preferably in industrial contexts. He supports several conferences and journals including (Conf:) ICSE, MODELS and Euromicro SEAA, FAMECSE and (Jnl:) SoSyM and Empirical Studies in Software Engineering (EMSE). He has given tutorials and guest lectures on Software Architecture Design and Modeling as well as on Empirical Software Engineering Research Methods as guest lecturer in (o.a.) Spain, Tunesia, France, Finland, Slovakia and The Netherlands. He can be reached at m.r.v.chaudron@tue.nl.