On the Effects of Developers' Intuition on Measuring Similarity Between UML Models

Lucian José Gonçales
Graduate Program in Applied Computing (PPGCA), University of Vale do Rio dos Sinos (UNISINOS)
São Leopoldo, RS
lucianj@edu.unisinos.br

Kleinner Farias
Graduate Program in Applied Computing (PPGCA), University of Vale do Rio dos Sinos (UNISINOS)
São Leopoldo, RS
kleinnerfarias@unisinos.br

Vinicius Bischoff
Graduate Program in Applied Computing (PPGCA), University of Vale do Rio dos Sinos (UNISINOS)
São Leopoldo, RS
viniciusbischof@edu.unisinos.br

ABSTRACT

Software design models play a key role in many activities of information systems engineering, such as documenting software artefacts, communicating project decisions, and code generation. In this scenario, the techniques for comparison of software design models are used for several purposes, such as, for detecting clones, and model evolution. In the last decades, academia proposed different techniques for comparing software models. Even using these different techniques for model comparison, this process is still an activity of a subjective nature, because during this process, different developers can interpret the similarity differently. Thus, the problem is that it is still unknown if developers has the same intuition in order to resolve comparison of software design models. For this, the main objective of this work is to explore the effects of their experience level, i.e., experienced and inexperienced developers, relative to their effort and correctness for resolving activities of comparing software design models. Therefore, a controlled experiment was conducted to evaluate the developer's experience level regarding on similarities of UML Models. The results show that the developer's experience does not affect the understanding of similarities activities.

CCS CONCEPTS

• **Software and its engineering** → **Model-driven software engineering**; • **Human-centered computing** → *Empirical studies in HCI*;

KEYWORDS

Unified Modelling Language , UML , Software Engineering , Controlled Experiment , Information Systems

ACM Reference Format:
Lucian José Gonçales, Kleinner Farias, and Vinicius Bischoff. 2019. On the Effects of Developers' Intuition on Measuring Similarity Between UML Models. In *XV Brazilian Symposium on Information Systems (SBSI'19), May 20–24, 2019, Aracaju, Brazil.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3330204.3330238

SBSI'19, May 20–24, 2019, Aracaju, Brazil

ACM ISBN 978-1-4503-7237-4/19/05…$15.00
https://doi.org/10.1145/3330204.3330238

1 INTRODUCTION

Software design models play a key role in various on information systems [6, 9, 15, 21], such as documenting software artifacts, communicating project decisions, generating code, estimating development effort, and especially in the representation of project decisions to enhance communication among software development teams. In the context of model-driven software development, for example, developers use software models (such as UML class diagrams) as primary artefacts throughout the development process. These artefacts undergo a sequence of transformations to generate the application code. For this, these distributed development teams usually evolve and change these different parts of software design models. In this context, developers often need to identify the similarities of the changed models in parallel, as well as reconcile the parts of the models with conflicting information.

In this scenario, the similarities of software design models are used for several purposes, such as: clone detection [18], indicating if the models are clones for ensuring the true development authorship [5], identify which architectural patterns were used in the project [16]; identifying the overlaps between project models, and model composition [6] because similar elements must be identified to enable the integration of them. In particular, the term model comparison can be briefly defined as the activities that determine the correspondences between the elements of M_A and M_B input models. The correspondence between these elements is a relation S between the elements of M_A and M_B that is defined as similar or equivalent.

In the last decades, state-of-the-art literature has proposed different techniques for comparing software design models. Van den Brand et al. [20] based the comparison of software design models on the correspondences between the input models M_A and M_B. This approach assigns a unique identifier to each element of the models after receiving two input models, M_A and M_B. Therefore, the approach identifies the differences between the elements through these identifiers. Thus, equal elements have the same identifier in both input models, and new identifiers are detected by the technique when new elements are inserted into the model, and removed elements are identified by those identifiers that are no longer present in the model. Kolovos [11] defined that the process of model comparison is a matching of semantically equivalent elements between the M_A and M_B input models. For this, the user defines a set of semantic rules. Then the mechanism infers what are the equivalent elements between the input models based on these rules. Kpodjedo [13] defined that the comparison of software design models comprises on calculating the degree of overall similarity between the

elements of the input models M_A and M_B, considering syntactic and semantic aspects of the models. The general similarity degree is a value used to indicate how close the M_A and M_B models are.

Despite the existence of systematized methods to calculate the similarities of software design models [12], this is an activity of subjective nature. During this activity, developers with different levels of experience can interpret the similarity of software models in a different manner. Moreover, previous works already pointed the developers could comprehend software artefacts, such as a source code in a different manner [1, 2, 4, 22]. This difference could be motivated by the implicit use of intuition to establish the comprehension.

However, none of the previous studies investigated if developers interpret software artefacts with higher level of abstraction in the same manner. Therefore, it is necessary to investigate if developers level of expertise affects their interpretation. Thus, the main question this works aims to answer is "Does the developers' experience affects on the interpretation when they compare models?"

The main objective of this work is to explore the effects of developers experience level relative to their effort and correctness, i.e., number of similar models correctly identified during the comparison of software design models. By identifying the unknown effects of the developers' experience level in the comparison of software design models. Thus, we seek to analyze the effects of their level of experience in relation to effort, and correctness, and verify if their experience level affect on their intuition during this activity.

For this, a controlled experiment was conducted to evaluate the developer's interpretation on similarity software design model's activities. Specifically, developers conducted some model comparison tasks in the context of model evolution with UML class diagrams. In addition, it is analyzed if the experience of the developer impacts on the understanding of these activities. In total, 37 developers participated in this experiment. The participants were composed by academic students, and software developers professionals. Then, their experience level were varied, and this allowed us to analyze the influence on the experience of the developers in the proposed activities. This results of this experiment points that the developer's experience does not affect the understanding of similarity of UML software models.

To investigate this problem this work is presented in six sections. Section 2 presents basic concepts about this research. Section 3 describes the related works. Section 4 describes the study methodology. Section 5 presents and discusses the results obtained regarding the research questions. Section 6 describes threats to validity. Finally, Section 7 presents the conclusion and future works.

2 MODEL COMPARISON OF SOFTWARE DESIGN MODELS

The comparison of software design models consists on evaluating the similarity between the input models M_A and M_B. Figure 1 shows the base model M_A, and two subsequent evolved delta models M_B. In order to compare these models, literature has provided many manners to identify these differences and common points. They are usually based on match-by-name approach, or similarity values. In a match-by-name approach, attributes and class names are matched based on their names [20]. Then M_B delta model would be fully similar to their source model M_A. This is due all class names of the both M_B models match exactly to the corresponding names in M_A. This result would change with a similarity technique, that also computes the differences in a unique similarity value, accouting the differences of some aspects such as the relationships between elements, and their neighbors [10, 13].

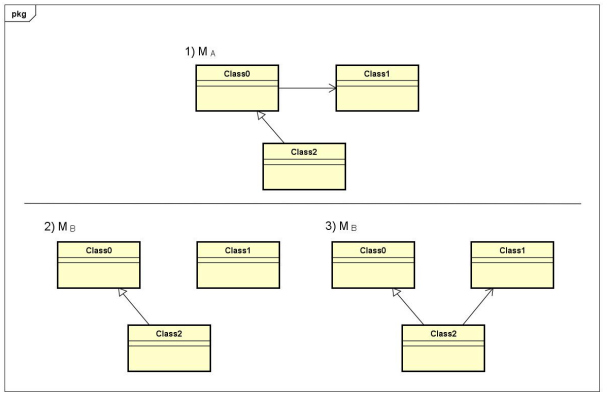

Figure 1: Examples of graphs connectivity.

Based on the relationship differences developers may interpret that the M_B model of Figure 1.2 would be more similar in relation to the M_A than the model M_B on Figure 1.3. This because they may apply different subjective rules to judge the similarity of models [12, 14]. For example, they may apply subjective rules that penalizes negatively the similarity value in the case of modifications that implies on disconnected components. Applying this intuitive property on the Figure 1.2 turns M_B less similar in relation to M_A because it generates a component that is not connected (*Class1*). However, in practice developers may disagree with this rule. They may argue that Figure 1.2 will maintain architecture specifications even with a disconnected component, and fewer relationships regarding M_B on Figure 1.3. Therefore, others may argue that M_B on Figure 1.2 would penalize more the similarity value in relation to M_B on Figure 1.3 because it it now has a component that its not used by the application.

3 RELATED WORKS

To the best of our knowledge, this study is the first to explore how developers comprehend the similarity between UML model's using intuitive properties [12]. For this, this work specifically collected experimental data from 37 professionals from industry. They solved a total of 370 tasks of model similarity. This experiment is a starting point to bridge a gap concerning empirical studies about the similarity of software design models.

Empirical evidence is lacking to demonstrate if the expectations of developers meet the results of similarity techniques. Despite the importance of measuring the similarity between software design models, little have been done to produce to investigate of similarity on software design models. Ricca et al. [17] evaluated the comprehension of developers in relation to the usage of stereotypes on the UML diagrams. The diagram's similarities were not considered during the evaluation, neither the evaluation of intuitive properties.

In contrast, this work is the first to produce a controlled experiment about similarity of software design models over the developer's perspective. This is because earlier works focused on generally to produce empirical evidence on model composition, and the effects of improvements on the UML representation. Furthermore, the other point neglected by previous studies is that developer influence over the results was not properly discussed, i.e., studies concerned producing techniques instead on developer's factors that can generate issues on comparison of software design models process.

The literature provided measures to analyze the comprehension of developers on software engineering tasks [17][19]. However, these works diverged what measures to apply. Ricca et al. [17] not evaluated the time effort. This because the experiment had a time limit of two hours in which almost participants used. They also applied the precision and recall for evaluating answer's correctness. Uesbeck et al. [19] collected the effort that developers apply to solve questions. However, they defined a time limit for respondents solving tasks. This was defined because authors were afraid that inexperienced developers may solve the tasks of the experiment. The authors also evaluated the correct answers, and for this, they analysed the output from the compiler. In contrast, this work analysed the time developers spent to solve tasks. As in the previous works, the time also is an important factor for analysing their applied effort to interpret questions. However, we did not set any time limit because the tasks developers performed were small. In addition, we also evaluated the answer's correctness. The correctness is an important variable to corroborate their comprehension on the similarity tasks together with their effort.

To sum up, the experiments produced so far evidence a lack of empirical evidence about the model comparison research field. Moreover, this problem contributes to a lack of understanding about the effects of the developer's interpretation has on model similarity. In particular, regarding the intuitive properties implemented in the similarity tools. Therefore, this work conducted a controlled experiment to verify whether developers comprehend the similarity of models. This experiment was a starting point to understand (1) if intuitive properties help developers understand the model similarity, and confirm if those properties are proper to implement in a comparison tool, and (2) verify if the developer's expectations are adherent to the results of similarity.

4 STUDY METHODOLOGY

This section describes the study methodology. Section 4.1 presents the objective and research questions that are explored in our study. Section 4.2 formulates the study hypothesis. Section 4.3 discusses details about the study variables and their quantification method. Finally, Section 4.4 shows the adopted experimental design to run the controlled experiment.

4.1 Objective and Research Questions

This study investigates how software developers comprehend intuitive properties in the context of model comparison activities. We analyze if software developers expect the similarity adherent to the definitions of intuitive properties. For this, their respective levels of experience in relation to the model comparison activities. The effort, and correctness of comparison activities were measured to evidence these differences. Thus, the objective of this study is organized in the GQM (Goal Question Method) model [3]:

Analyze *developers experience level*
for the purpose of *investigating the effects*
with respect to *correctness and effort*
from the perspective of *developers*
in the context of *model comparison.*

In this way, we aim to investigate whether the developers' experience level, i.e., inexperienced and experienced, has the same intuition on resolving model comparison activity. For this, their intuition is analyzed through the effort, and correctness in the context of model comparison. In addition, the level of experience is another factor that can affect the understanding of comparison activities. Therefore, the level of experience is a factor analyzed in evidence. In this way, the research questions (RQ) are defined as follows:

- **RQ1:** What is the impact of the level of experience in the effort to identify the similarity of the models?
- **RQ2:** What is the impact of the level of experience on the correctness of model comparison?

4.2 Hypothesis Formulation

This section tries to formulate the hypotheses that aim to guide the controlled experiment. In this paper two hypotheses were formulated. Hypothesis 1 deals specifically with analyzing the effort, while hypothesis 2 analyzing the correctness.

Hypothesis 1: this hypothesis assumes that experienced developers compare design model in a more systematic way, since its assumed that their understanding about the process of similarity of software models are better than inexperienced ones, and implies in less effort during the activity of model comparison. That is, experienced developers do not invest more effort to identify correspondences between design models. In contrast, it is assumed that inexperienced developers tend to apply more effort when comparing software models. However, this hypothesis may not be maintained due to inexperienced developers could comprehend design models more quickly due the simplicity on dealing with design models. In addition, the size of the software design models could also improve the understanding of inexperienced developers. Based on this assertion, the hypotheses (null and alternative) are presented as follows:

Null Hypothesis 1 H_{1-0}: *Experienced developers (Exp) apply the same or more effort to compare software design models than inexperienced developers (Inex).*

$$H_{1-0}: Effort(M_A, M_B)_{Exp} \geq Effort(M_A, M_B)_{Inex}$$

Alternative Hypothesis 1 H_{1-1}: *Experienced developers (Exp) apply less effort to compare software design models than inexperienced developers (Inex).*

$$H_{1-1}: Effort(M_A, M_B)_{Exp} < Effort(M_A, M_B)_{Inex}$$

Thus, we seek to understand if the level of experience is critical in relation to the effort in order to identify the equivalences between elements, generating empirical evidence about the effects

of the level of experience in model comparison activities. After performing this analysis, a first perception about the comprehensibility of the developers in the model comparison activity will be created. The next hypothesis investigates the impact of the developers experience level on the correctness of similarity between design models.

Hypothesis 2: developers may do not identify the correct equivalences between software design models. This is import because subsequent activities can be negatively affected, such as the model composition. If a wrong description of commonalities and differences of software design model elements be identified before the composition process, an inconsistent model would be generated. This hypothesis assumes that experienced developers identifies with more accuracy the commonalities and differences between software design models than inexperienced developers. Therefore, the second hypothesis evaluates whether experienced developers can actually identify similar models correctly in relation to inexperienced developers in the same activity. In addition, this hypothesis assumes that high experienced developers outperform the number of identification of similar models in relation to not experienced developers. Based on this, the null and alternative hypotheses are defined as follows:

Null Hypothesis 2 H_{2-0}: Inexperienced developers (Inex) identify more or equal correct answers (Cor) in relation to experienced developers (Exp) during similarity of software design models activities.

$$H_{2-0} : Cor(M_A, M_B)_{Inex} \geq Cor(M_A, M_B)_{Exp}$$

Alternative Hypothesis 2 H_{2-1}: Inexperienced developers (Inex) identify less correct answers (Cor) in relation to experienced developers (Exp) during similarity of software design models activities.

$$H_{2-1} : Cor(M_A, M_B)_{Inex} < Cor(M_A, M_B)_{Exp}$$

To run the analysis of the above hypothesis some variables were defined. These variables are detailed in the next section.

4.3 Study Variables

The independent variable of the hypotheses 1 and 2 is the experience level of the developers, respectively. As earlier mentioned, the experience level of developers was controlled in this experiment to understand its impact on the dependent variables. The dependent variable of the first hypothesis is the effort applied by the developers to solve the similarity between two input models M_A and M_B. This variable was measured in seconds. The dependent variable of the second hypothesis is the number of correct answers.

4.4 Experiment Workflow

Figure 2 shows the adopted experimental process. This process was adopted because it was already used on previous studies [8][7]. This experimental process is composed of three phases which are described below:

- **First phase: Selection and Training of Participants.** In the first step, we executed the activity invite and select the participants. In this activity, we concerned on inviting participants to execute the experiment. Candidates who had experience and involvement with modeling and development of information systems were selected to participate in the experiment. Next, in the training activity, all participants that accepted to execute the experiment were trained. This training consists on presenting to them the contextualization about how to execute the experiment, and the kind of activities they would perform. This preparation ensured that they acquired the necessary familiarity with the upcoming tasks.
- **Second phase: Experiment Execution.** The participants executed the experiment. A unique activity composes this step, i.e., Identify Similar Models. In this activity selected participants executed ten tasks of model similarity. In this questionnaire, each similarity scenery is equivalent to a question. Practitioners answered a total of ten questions. Each question required to users choose one of the four delta models M_B. Specifically, practitioners must choose one of the M_B that they judge be more similar in relation to the base model M_A. The models used in this study were UML class diagrams that contained about 8 classes and 7 relationships. The size of the models is due to the time limitation to perform this experiment, making it impossible for participants to solve very large models. The output of this activity is the (1) set of correct answers (***Cor***); (2) the time spent to perform the similarity task (***f***); and (3) the rate of correct answers (***RCor***). The questionnaire with activities they performed are available as additional material [1].

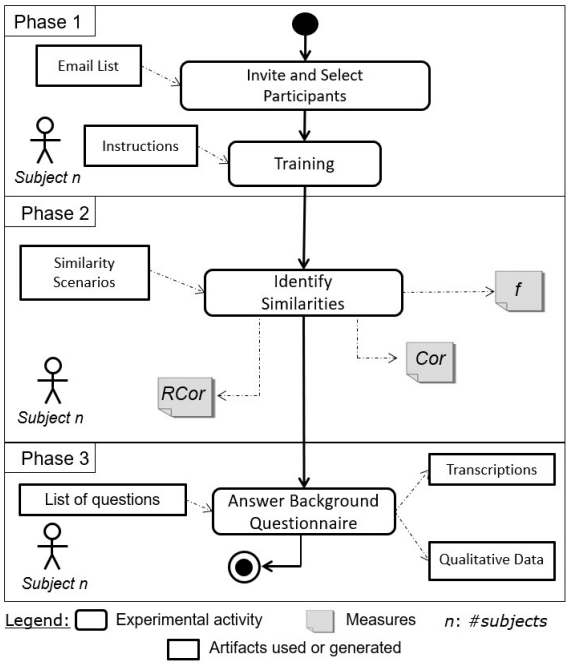

Figure 2: The Experimental Process.

[1] https://drive.google.com/open?id=1laOVril1WJe5CGY14DiVYUK8yQook35P

- **Third phase: Respondents Background.** Finally, in the last step we executed the activity respond background questionnaire. In this activity, we distributed a list of questions asking information about the background of the participants. This questionnaire ask their current positions, years of experience with software modelling and development. In addition, they opined about the experiment. To sum up, they fulfilled a questionnaire about their knowledge backgrounds, and skills. The output of this activity is the transcriptions of their reports about the experiment, and qualitative data about their experience profile, and knowledge background.

5 RESULTS

This section details the results of the experiment. Section 5.1 presents an overview of the participants profile. Section 5.2 shows the results of the first hypothesis, that investigates the developer's effort. Section 5.3 shows the results of the second hypothesis that investigates the developer's correctness.

5.1 Participants Background

A total of 37 participants were recruited to perform this experiment. All participants has knowledge background related to software development, and modelling of information systems. In other words, almost all of them at least attended or actually attend in an undergraduate course related to systems analysis and development. In particular they attended on graduate courses such as computer science, development and system analysis, information systems, and computer engineering. One of the participant has a bachelor's degree in mathematics, and another participant was enrolled on a psychology course. But both of them were involved on academic projects that required them to perform activities related to software modelling and development.

A total of 67.5% of the participants (25/37) work in the software industry, on companies such as TOTVS, SAP, INMETRO, ADP, CWI software, Correios, and PROCERGS. Their position varies from software analysts (24.3%, 9/37), software developers (35.1%, 13/37), project managers (5.04%, 2/37), and software architects (2.7%, 1/37). 32.4% (12/37) of participants are undergraduate (16/2%, 6/37), and graduate students (16/2%, 6/37) enrolled on computer science related courses, and that attended in software industry previously. 51.1% (19/37) has at least four years of experience on software development, and 48.6% (18/37) of the participants has more than five years of experience on software development. Moreover, 78.3% (29/37) of participants has until four years of experience on software modelling, and a minority (21.6%, 8/37) of participants has more than five years of experience on software modelling. However, as participants highlighted that the activity of software modelling was secondary to their daily tasks, i.e., they perform with less frequency than dealing with programming tasks, their experience level was classified according their software development experience. It was assumed that classifying their experience level according to their amount of years on software development activities is somewhat equivalent to the experience on modelling software, as they deal with software design models to plan the implementation of the software system.

5.2 RQ1:developers Effort

Descriptive statistics. A total of 370 responses were collected from 37 participants. These participants answered 10 questions. Table 1 presents the statistical description of the data related to developer's effort. Inexperienced developers performed 190 samples of similarity activities, and while experienced developers produced 180 samples. These values are presents in column N of Table 1 specifically. The results appear show that there isn't a group that applied less effort on comparison activity of software design models. Except the mean value (average) time, the maximum value (Max), and the standard deviation (SD) indicates that the group of experienced developers applied more effort. In particular, the maximum effort value shows that experienced developers applied a 50% more effort than inexperienced developers. Whereas, the median value points to the same direction, i.e., experienced developers dedicating more time on some similarity tasks.

Table 1: Descriptive statistics of hypothesis 1.

	Effort	
	Experienced	**Inexperienced**
N	180	190
Min	45	45
25th	45	45
Median	60	60
75th	60	60
Max	360	240
Mean	67,25	65,13
Standart deviation (SD)	43,06	30,02

Hypothesis Testing. Table 2 shows the results of hypothesis tests using the Mann-Whitney test. Specifically, Table 2 presents, for each question, the rank in seconds of each group, the Mann-Whitney value, and the significance value (p-value). The Mann-Whitney test showed that there was no significant difference in effort applied between experienced and non-experienced developers, as the overall result pointed that Mann-Whitney U = 16569.5 and p-value = 0.563, with an average effort rank of 188.29 for the group of inexperienced developers and 182.55 for experienced developers. Therefore, this test failed to reject the null hypothesis H_{1-0}.

In general, the p-values of all questions were above 0.05, demonstrating that the mean of the two groups does not have a significant difference. In question 10, the p-value of 0.061 is closer to the confidence interval of 0.05, as the average effort of the inexperienced developers was slightly lower than the average effort applied by the experienced developers.

We also applied the Kruskall-Wallis statistical test to reinforce the results presented on the the Mann Whiteney U test. The Kruskall-Wallis test is a nonparametric test for independent variables, and as in the Mann Whitney U test, it is also based on the mean ranking to determine if there is a significant difference between the developers' experience groups. The Kruskall-Wallis test is an extension of the Mann Whitney U test, as well as being designed for the same purpose, this test also evaluates two or more independent variables. Therefore, this test was used to support the results presented in Table 2.

Table 3 presents the results obtained for each question through the Kruskall-Wallis test. The Kruskal-Wallis test confirms that there

Table 2: Results of Mann-Whitney U Test

Questions	Statistics	Inexperienced	Experienced
General	Rank	188,29	182,55
	Mann-Whitney U	16569,5	
	p-value	0,563	
1	Rank	18,21	19,83
	Mann-Whitney U	156	
	p-value	0,605	
2	Rank	20,42	17,5
	U de Mann-whitney	144	
	p-value	0,339	
3	Rank	19,89	18,06
	U de Mann-whitney	154	
	p-value	0,532	
4	Rank	20,18	17,75
	U de Mann-whitney	148,5	
	p-value	0,458	
5	Rank	19,16	18,83
	U de Mann-whitney	168	
	p-value	0,918	
6	Rank	21,05	16,83
	U de Mann-whitney	132	
	p-value	0,193	
7	Rank	19,26	18,72
	U de Mann-whitney	166	
	p-value	0,86	
8	Rank	18,84	19,17
	U de Mann-whitney	168	
	p-value	0,912	
9	Rank	19,39	18,58
	U de Mann-whitney	163,5	
	p-value	0,799	
10	Rank	16,16	22
	U de Mann-whitney	117	
	p-value	0,061	

is no significant difference about the effort between experienced and inexperienced developers, since χ^2 = 0.334 and p-value = 0.563, with an average effort rank of 188.29 in the group of inexperienced developers, and 182.55 for experienced developers in the general row of Table 3. Therefore, through this test also fails to reject the null hypothesis H_{1-0}. In the previous test, the p-value of question 10 resulted in 0.061, which was the value closest to the significance value. In this case, the Kruskal-Wallis test also pointed that there is no significant difference in effort applied between experienced and inexperienced developers, since χ^2 = 3.505, and p = 0.061, with an average effort rank of 16.16 in the developer group and 22 for the group of experienced developers in the field of question 10 of Table 3. Finally, we also highlighted that the general results of p-values in Table 3 presented values practically equal to the p-values in Table 2, confirming the results in the Mann Whitney U test.

5.3 RQ2:developers Correctness

Descriptive statistics. Correctness specifically refers to the amount of correct answers participants produced. Thus, this descriptive analysis is based on the quantity of correct answers produced in each question. A total of 37 participants answered 10 questions about model similarity. Specifically, 18 participants were experienced developers, and 19 inexperienced developers. Table 4 presents the descriptive analysis of these data. Overall, data from the descriptive analysis point out that there is no significant difference between experienced developers and inexperienced developers in

Table 3: Results of Kruskall Whallis Test on hypothesis 1.

Questions	Statistics	Inexperienced	Experienced
General	Rank	188,29	182,55
	χ^2	0,334	
	p-value	0,563	
1	Rank	18,21	19,83
	χ^2	0,268	
	p-value	0,605	
2	Rank	20,42	17,5
	χ^2	0,914	
	p-value	0,339	
3	Rank	19,89	18,06
	χ^2	0,391	
	p-value	0,532	
4	Rank	20,18	17,75
	χ^2	0,551	
	p-value	0,458	
5	Rank	19,16	18,83
	χ^2	0,011	
	p-value	0,918	
6	Rank	21,05	16,83
	χ^2	1,693	
	p-value	0,193	
7	Rank	19,26	18,72
	χ^2	0,031	
	p-value	0,86	
8	Rank	18,84	19,17
	χ^2	0,012	
	p-value	0,912	
9	Rank	19,39	18,58
	χ^2	0,065	
	p-value	0,799	
10	Rank	16,16	22
	χ^2	3,505	
	p-value	0,061	

relation to the results correctness. In addition, Table 4 also indicates inexperienced developers have more correct answers compared to the experienced developers. However, this difference is not large. The mean value, shows an advantage for the inexperienced group with 1.3 higher correct answers compared to the group of experienced developers.

Table 4: Descriptive statistics of the quantity of correct answers.

	Correctness	
	Experienced	Inexperienced
N	18	19
Min	10	12
25th	12	13,25
Median	13	14
75th	14	15
Max	16	18
Mean	13.1	14,4
Standart Deviation (SD)	1.73	1.96

Figure 3 shows the number of correct answers per question according to the developers' experience. Experienced developers achieved more correct answers on questions 6 and 10 in relation to

inexperienced developers. Inexperienced developers achieved more correct answers in general, but without a significant difference. Thus, the inexperienced developers had a superior performance in 8 questions. This points that experienced developer neglects the simplest activities.

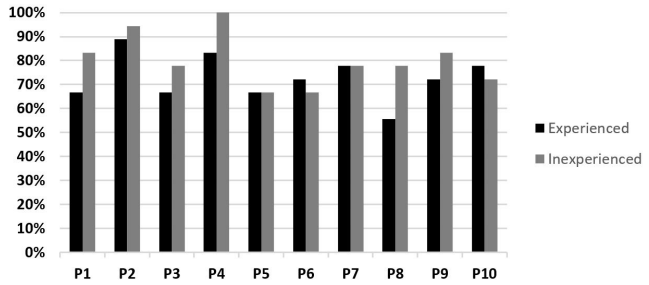

Figure 3: Quantity correct answers by question.

Hypothesis Testing. The chi-square test was chosen to test Hypothesis 2. This test fulfilled the two main requirements for analyzing this hypothesis: (1) the data are categorical and nominal, that is, the response is defined as correct or incorrect; (2) the two variables consist of two independent groups (Experienced and Inexperienced Developers). Specifically, this test verifies if the developer experience level (experienced / inexperienced) is significantly associated with the type of response (correct or incorrect). Thus, there are two nominal variables being evaluated: developer category (experienced / not experienced) and the answer type (correct or incorrect). Data are significantly associated when p-value < 0.05. It is possible to note that the general result of the Chi-square test in Table 5 shows that χ^2 = 2.2178, and p-value = 0.1364. This data points that there is no statistically significant association between the developer experience category and the response type, that is, developers experience has no influence on the amount of correct answers. Specifically, the p-value = 0.1334 is above the significance value of 0.05, and therefore the null hypothesis H_{2-0} is not rejected. It is also possible to associate that, where p-values are equal to 1 in Table 5, the questions correspond with the proximity of correctness between experienced and inexperienced developers in Figure 3.

We also applied Fisher's test in this hypothesis. Basically, this test is also suitable for this type of test, i.e., for categorical and independent variables. But the main reason for this test being used is to reinforce the values presented in the previous test. In the Fisher test, the data were also significantly associated when p-value < 0.05. Table 6 shows the p-values obtained in this test. The general evaluation of the questions points out a p-value = 0.1362, i.e., this test also points out that there is no association between the developer experience category and the response type. Therefore, the null hypothesis H_{2-0} is not rejected because the p-value is above 0.05.

6 TREATS TO VALIDITY

This study presents a series of threats to validity. For this, some measures were taken in order to guarantee the validity, and the reproducibility of this work. Therefore, the internal threats, external threats, statistical conclusion threats, and threats to experiment were treated specifically.

Table 5: Chi-Square test.

Question	Experienced vs Inexperienced	
General	χ^2	22.178
	P-value	0.1364
1	χ^2	0,59259
	P-value	0,4414
2	χ^2	0
	P-value	1
3	χ^2	0.13846
	P-value	0.7098
4	χ^2	14.545
	P-value	0.2278
5	χ^2	0
	P-value	1
6	χ^2	0
	P-value	1
7	χ^2	0
	P-value	1
8	χ^2	1.125
	P-value	0,2888
9	χ^2	0.16071
	P-value	0.6885
10	χ^2	0
	P-value	1

Table 6: Fisher test.

Question	P-value
General	0.1362
1	0.443
2	1
3	0.7112
4	0.2286
5	1
6	1
7	1
8	0.289
9	0.6906
10	1

Internal validity. In this kind of study is usual to validate the inferences between the independent variable and the dependent variables are valid internally. This study presents a valid inference because the study meets all three requirements for internal validity. These requirements are: (1) the temporal precedence criteria was met, i.e., the participants experience influences the similarity effort, and correctness; (2) the covariation observed, i.e., the correct answers rate of the comparison between the models varied according to the participants experience; and (3) there is no extra cause for the detected covariation.

External validity. External validity refers to the validity of the results obtained in other wider contexts. For replicating this study is necessary that: (1) the participants must be able to make comparisons between models to carry out the activities of the questionnaire, as well as (2) know the concepts and UML notation, such as UML class diagrams, (3) comparison must be implemented to identify the

evolution of software models based on the addition, and deletion of class relationships, and (4) models must be small.

Conclusion validity. To validate the statistical conclusion it was checked whether the dependent and independent variables have been subjected to appropriate statistical methods. These methods are useful to analyze whether the experiment variables covariates or not. For this, it was verified that the sample data has normal distribution, through the Kolmogorov-Smirnov and Anderson-Darling tests.

Experiment validity. The validity of the experiment aims to verify whether we are really measuring what we think are measuring. The dependent variable was quantified based on seconds number to perform each question, and the concept of correct answers rate, and comprehension rate is well-known in the literature; the quantification was done accurately by the authors, i.e., the variables were properly evaluated since it was found that the data collected were in agreement with the hypothesis according the study objectives; and different types of similarity tools can produce different results, but in this experiment participants did not used any different version of similarity tool. Therefore the treatment was the same for all participants.

7 CONCLUSION

This research sought to investigate if the developers experience level influence on the comprehension on the similarity of software design models. Therefore, a controlled experiment was conducted to evaluate the understanding of the developers in the activities of comparison of software design models. Activities all of which were elaborated respecting the intuitive properties. Then the understanding was investigated through the analysis of the effort and correctness applied in the activities of comparison of software design models, and in addition, if there is influence of their experience level in the understanding of these activities. Therefore, two hypotheses were elaborated to analyse the effort and correctness respectively. Moreover, this work mitigate factors that can affect the comprehension of developers during the development and modelling of information systems. Information systems are usually complex to understand, then integrating their features requires the appropriated identification of different and overlapping parts by developers.

The main conclusions regarding the two hypotheses formulated were: (1) The developers experience does not influence the effort in the comparison of software design models, (2) nor does it influence the quantity of correct answers. Thus, although the difference in experience level, developers have a common understanding on UML software design models, even in a similarity calculation activity, that is a complex task. Finally, this controlled experiment contributed specifically to how the factor of the developer's experience affects understanding; and also in relation to the importance of empirical studies in the area of software development to investigate the developers capability of interpretation, as well as, the importance of higher abstraction languages on the comprehension of software developers. Further research will focus on replicate this study with more participants aiming to increase and improve the significance of obtained results.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior -Brasil (CAPES) - Finance Code 001.

REFERENCES

[1] Mujtaba Alshakhouri, Jim Buchan, and Stephen G MacDonell. 2018. Synchronised visualisation of software process and product artefacts: Concept, design and prototype implementation. *Information and Software Technology* 98 (2018), 131–145.

[2] Ameer Armaly, Paige Rodeghero, and Collin McMillan. 2018. A comparison of program comprehension strategies by blind and sighted programmers. *IEEE Transactions on Software Engineering* 44, 8 (2018), 712–724.

[3] Victor Caldiera and Dieter Rombach. 1994. The goal question metric approach. *Encyclopedia of software engineering* 2, 1994 (1994), 528–532.

[4] Igor Crk, Timothy Kluthe, and Andreas Stefik. 2016. Understanding programming expertise: an empirical study of phasic brain wave changes. *ACM Transactions on Computer-Human Interaction (TOCHI)* 23, 1 (2016), 2.

[5] Jing Dong, Yongtao Sun, and Yajing Zhao. 2008. Design pattern detection by template matching. In *Proceedings of the 2008 ACM symposium on Applied computing*. ACM, 765–769.

[6] Guilherme Ermel, Kleinner Farias, Lucian José Gonçales, and Vinicius Bischoff. 2018. Supporting the Composition of UML Component Diagrams. In *Proceedings of the XIV Brazilian Symposium on Information Systems*. ACM, 56.

[7] Kleinner Farias, Alessandro Garcia, and Carlos Lucena. 2014. Effects of stability on model composition effort: an exploratory study. *Software & Systems Modeling* 13, 4 (01 Oct 2014), 1473–1494. https://doi.org/10.1007/s10270-012-0308-2

[8] Kleinner Farias, Alessandro Garcia, Jon Whittle, Christina von Flach Garcia Chavez, and Carlos Lucena. 2015. Evaluating the effort of composing design models: a controlled experiment. *Software & Systems Modeling* 14, 4 (2015), 1349–1365.

[9] Vinícius Soares Fonseca, Monalessa Perini Barcellos, and Ricardo de Almeida Falbo. 2016. Tools integration for supporting software measurement: a systematic literature review. *iSys-Revista Brasileira de Sistemas de Informação* 8, 4 (2016), 80–108.

[10] Marouane Kessentini, Ali Ouni, Philip Langer, Manuel Wimmer, and Slim Bechikh. 2014. Search-based Metamodel Matching with Structural and Syntactic Measures. *J. Syst. Softw.* 97, C (Oct. 2014), 1–14. https://doi.org/10.1016/j.jss.2014.06.040

[11] Dimitrios S Kolovos. 2009. Establishing Correspondences between Models with the Epsilon Comparison Language. *ECMDA-FA* 9 (2009), 146–157.

[12] Danai Koutra, Neil Shah, Joshua T Vogelstein, Brian Gallagher, and Christos Faloutsos. 2016. D elta C on: principled massive-graph similarity function with attribution. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10, 3 (2016), 28.

[13] Segla Kpodjedo, Filippo Ricca, Philippe Galinier, Giuliano Antoniol, and Yann-Gael Gueheneuc. 2013. MADMatch: Many-to-Many Approximate Diagram Matching for Design Comparison. *IEEE Trans. Softw. Eng.* 39, 8 (Aug. 2013), 1090–1111.

[14] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. 2018. Graph Summarization Methods and Applications: A Survey. *ACM Computing Surveys (CSUR)* 51, 3 (2018), 62.

[15] Johnatan Oliveira, Eduardo Fernandes, Maurício Souza, and Eduardo Figueiredo. 2017. A method based on naming similarity to identify reuse opportunities. *iSys-Revista Brasileira de Sistemas de Informação* 10, 1 (2017), 99–121.

[16] Jan Reimann, Mirko Seifert, and Uwe Aßmann. 2013. On the reuse and recommendation of model refactoring specifications. *Software and Systems Modeling* (2013), 1–18.

[17] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato. 2010. How Developers' Experience and Ability Influence Web Application Comprehension Tasks Supported by UML Stereotypes: A Series of Four Experiments. *IEEE Transactions on Software Engineering* 36, 1 (Jan 2010), 96–118.

[18] Harald Störrle. 2013. Towards clone detection in UML domain models. *Software & Systems Modeling* 12, 2 (2013), 307–329.

[19] Phillip Merlin Uesbeck, Andreas Stefik, Stefan Hanenberg, Jan Pedersen, and Patrick Daleiden. 2016. An Empirical Study on the Impact of C++ Lambdas and Programmer Experience. In *Proceedings of the 38th International Conference on Software Engineering (ICSE '16)*. ACM, New York, NY, USA, 760–771.

[20] Mark van den Brand, Zvezdan Protic, and Tom Verhoeff. 2010. Fine-grained metamodel-assisted model comparison. In *Proceedings of the 1st international workshop on model comparison in practice*. ACM, 11–20.

[21] Bernhard Westfechtel. 2014. Merging of EMF models. *Software & Systems Modeling* 13, 2 (2014), 757–788.

[22] Xin Xia, Lingfeng Bao, David Lo, Zhenchang Xing, Ahmed E Hassan, and Shanping Li. 2018. Measuring program comprehension: A large-scale field study with professionals. *IEEE Transactions on Software Engineering* 44, 10 (2018), 951–976.