Examining the Effects of Layout and Working Memory on UML Class Diagram Defect Identification

Bonita Sharif
School of Computing
University of Nebraska - Lincoln
Lincoln, Nebraska USA
Email: bsharif@unl.edu

Kang-il Park
School of Computing
University of Nebraska - Lincoln
Lincoln, Nebraska USA
Email: kangil.park@huskers.unl.edu

Michael P. DeJournett
School of Computing
University of Nebraska - Lincoln
Lincoln, Nebraska USA
Email: mdejournett2@huskers.unl.edu

Isaac Baysinger
School of Computing
University of Nebraska - Lincoln
Lincoln, Nebraska USA
Email: isaacbaysinger@huskers.unl.edu

Mohammed Aly
Software Engineering Department
Assiut University
Assiut, Egypt
Email: mohamed.walid@compit.aun.edu.eg

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent, Ohio, USA
Email: jmaletic@kent.edu

***Abstract*—A controlled experiment investigating the effect layout has on how students find defects in UML class diagrams with respect to requirements is presented. Two layout schemes from prior literature namely, multi-cluster and orthogonal layouts, are compared with respect to two open source systems, Doxygen and Qt. The experiment is conducted with 89 students from two universities in a classroom lab setting. Each participant is placed in one of two groups where each group are given 2 defect detection tasks (with five sub-parts) with each task using one of the two layouts in each subject system. The only difference between groups is that the layouts were flipped between the two tasks. Feedback is collected after each task. A mental rotation and object memory task is conducted at the end of the two tasks to correlate their spatial and working memory skills to the task performance. Results indicate that the multi-cluster layout performed better in terms of accuracy of finding defects, but not significantly. There is also not much difference in time to find them. Furthermore, it is found that the object memory skills are sometimes correlated with the performance of the defect detection tasks. These results can be used to help improve the teaching of UML class diagram defect detection skills by incorporating clustered layouts and object memory tasks. In addition, they can help identify people who are best suited for finding critical defects in design.**

***Index Terms*—UML class diagrams, controlled experiment, layouts, working memory, defect detection, requirements**

I. Introduction

The Unified Modeling Language (UML) provides software engineers with a standardized means to design, specify, and visualize software [1]. UML consists of a variety of different diagrams and textual notation that represent static structure and dynamic behavior. The class diagram is the most commonly used diagram to design and specify how the different entities in a software system relate to each other. These diagrams are typically built during the analysis and design phase after requirements are elicited from customers. A diagram consists of a subset of the model (classes and relationships) drawn in a specific layout. There are various layouts that can be applied to a class model to enhance the way the diagram is drawn. The most commonly used layout is the orthogonal layout [2] used by most modeling software. The orthogonal layout prioritizes general aesthetic criteria such as minimizing edge crossings, edge bends, edge lengths, maximizing symmetry, and using 90-degree bends for edges. It does not use any semantic information about the classes or how they are related to position the classes on the diagram.

In 2005, a pilot study [3] was conducted by Andriyevska et al. that made use of class stereotypes of control, boundary, and entity [1] (visually represented by textual annotations and color) to produce a multi-cluster layout and a three-cluster layout for class diagrams. Entity classes store persistent information, control classes are responsible for main delegation of work, and boundary classes are what interface the system with the outside world. The three-cluster layout puts all control, entity, and boundary classes into three separate clusters. The multi-cluster layout uses information about the class stereotype semantics (of control, boundary, and entity) to position classes into multiple clusters on the diagram.

Each cluster represents a cohesive set of classes that depend on the types of relationships that exist between the classes in that cluster. A cluster can represent a specific concept or feature in the system. The study found that it is important to prioritize an architecturally meaningful UML class diagram rather than an aesthetically pleasing one because the clustered layouts performed better for their comprehension tasks. Since then, there are several studies [4]–[8] that show the clustered layouts performed better than the orthogonal layouts. A summary of the set of experiments assessing multi-cluster vs. orthogonal layouts is given by Sharif [9].

Prior studies done in the area of UML class diagrams are on high-level comprehension tasks with the exception of [6], which focuses on other tasks such as impact analysis, bug fix, feature addition, and refactoring besides reading and high-level overview tasks. However, none of the prior studies focus on finding defects in UML class diagrams (varying layout) with respect to requirements. This is important because UML class diagrams often decay or do not always match requirements during analysis. During design review, common practice is for a team member to review design documents to make sure they comply with requirements.

To bridge this gap, we conduct a study to understand how developers identify defects in UML class diagrams (presented in two different layouts) with respect to a requirements document (presented as text). The requirements and tasks are derived from two large C++ open-source systems. Besides the defect detection, we also want to determine if working memory capacity and spatial ability of the developers have any effect on their performance. Baum et al. found that working memory capacity has an effect on bug localization accuracy in code reviews [10]. Sharafi et al., found that spatial ability and data structure manipulation are related neural tasks [11]. For this reason, we include working memory capacity tasks in the study to determine if doing well on them transfers to defect detection skills between UML class diagrams and requirements documents. The study makes the following contributions: a) systematically studies if layout impacts defect detection accuracy and time, and b) determines if working memory capacity has an effect on defect detection performance.

The results show that there is no significant difference in defect detection score or time for neither the multi-cluster nor orthogonal layout in the Doxygen and Qt systems. When looking for correlations between working memory and defect detection times, we did not see any significant correlations. When looking for correlations between working memory and defect detection accuracy, the object memory score had a significant correlation for both systems. However, this is only in one of the layouts for each system (multi-cluster for Doxygen, orthogonal for Qt).

The rest of the paper is organized as follows. Section II lists our research questions and hypotheses. Related work is presented in Section III. The experiment design is given in Section IV. Section V presents the results for each research question. We discuss the impact of this work in Section VII and conclude the paper in Section VIII.

II. Research Questions and Hypotheses

This study seeks to address the following research questions.

- RQ1 Does layout affect the accuracy of detecting defects in UML class diagrams with respect to requirements?
- RQ2 Does layout affect the time taken to detect defects in UML class diagrams with respect to requirements?
- RQ3 Does working memory correlate with accuracy and time taken in finding defects in UML class diagrams with respect to requirements?

TABLE I
Null and Alternative Hypotheses

Null Hypothesis	Alternative Hypothesis
$H1_0$: There is no significant difference on defect detection *accuracy* in UML class diagrams between the orthogonal layout and the multi-cluster layout with respect to requirements.	$H1_a$: There is a significant improvement on defect detection *accuracy* in UML class diagrams following the multi-cluster layout with respect to requirements.
$H2_0$: There is no significant difference on defect detection *time* in UML class diagrams between the orthogonal layout and the multi-cluster layout with respect to requirements.	$H2_a$: There is a significant improvement on defect detection *time* in UML class diagrams following the multi-cluster layout with respect to requirements.
$H3_0$: There is no significant correlation between working memory and defect detection *accuracy* in UML class diagrams between the orthogonal layout and the multi-cluster layout with respect to requirements.	$H3_a$: There is a significant correlation between working memory and defect detection *accuracy* in UML class diagrams in both the orthogonal and the multi-cluster layouts with respect to requirements.
$H4_0$: There is no significant correlation between working memory and defect detection *time* in UML class diagrams between the orthogonal layout and the multi-cluster layout with respect to requirements.	$H4_a$: There is a significant correlation between working memory and defect detection *time* in UML class diagrams in both the orthogonal and the multi-cluster layouts with respect to requirements.

To answer RQ1, we rely on the score of each of the tasks and compare the two groups of multi-cluster and orthogonal layouts for each task across the same system. For RQ2, we test if time plays a factor in how they answer the two tasks in each respective layout across the same system. It has been shown that working memory capacity has an effect on how problem-solving tasks are performed. We operationalize working memory by testing the participants on two sub-tasks: object memory [12] and mental rotation [13]. The rationale for the object memory tasks (where participants had to recall letters and numbers) is to determine if that skill transferred to recalling the requirements they read. The rationale for the mental rotation tasks (where the participants are asked to determine if the image is the same when rotated) is to determine if the spatial recognition skill transferred to finding objects in a diagram when different layouts are used. The corresponding null and alternative hypotheses for each of the research questions are given in Table I.

III. Related Work

This section presents related work in the areas of defect detection in UML diagrams and empirical studies using UML models and layouts for comprehension.

Ondik et al. discuss the creation and use of a new software tool to better analyze defects in UML diagrams [14] based on developers' activities over models. They do this by detecting defects in real time and provide a way to incrementally synchronise the models. Laurent et al. suggests extending fUML [15] to include built-in error detection and correction. They provide a prototype for debugging UML models and present a small case study showing better understanding of the

models to novices (for proper use of constructs) and experts (to better understand designs). Enckevort sought to detect defects in UML class diagrams using custom OCL queries [16]. These studies do not empirically test comprehension of specific layouts.

Lange and Chaudron empirically investigated the impact of defects in UML models [17]. Their primary objective was to determine if defects caused misinterpretation of the models. The researchers' goal was to improve modeling practices, ensure clearer communication, and minimize risks in software design and development processes. 111 Masters students participated in the experiment. Key findings of the research include that there's a notable variation in defect detection rates across different defect types. 96% of participants were able to detect the "Class not in Sequence Diagram" defect, but only 10 % could spot the "Multiple Definitions of the same Class" defect. The risks for misinterpretations stemming from these defects were found to be significant. Overall, the study calls for enhanced clarity and training in software modeling to ensure effective communication and flawless development.

There have been other studies focusing solely on layout [18] such as using page rank algorithms to emphasize important classes [19], evolutionary algorithms [20], topology shape metrics [21], and force directed layouts [22]. Purchase et al. conducted a study to identify important aesthetic criteria for the automatic layout of UML class diagrams from a human perspective [23]. They concluded that the aesthetics depend on the actual semantics of the diagram and entities. They also studied five notational variants for class diagrams showing that the task might be a factor in determining the best performing notation [24].

The layouts used in this study stem directly from work done by Andriyevska et al. [3]. They found that architectural important layouts that are based on stereotypes assist in comprehension more compared to ones based solely on aesthetics [3]. Our layouts are directly based off of the architectural importance guidelines in [3]. Yusuf et al. conducted one of the first eye tracking studies to assess the comprehension of UML class diagrams [4]. They tested layouts, color, and stereotype usage and found that layouts with semantic information were found most effective with stereotypes playing an important role.

Sharif and Maletic replicated this study using online questionnaires and a bigger sample [5]. Results show a significant improvement in performance when multi-cluster layouts are used for UML notation and design tasks In 2009, Sharif and Maletic conducted a controlled experiment with 45 participants of varied experience levels to determine the effect of two different layout strategies i.e. orthogonal layout and multi-cluster layout in UML class diagrams [6]. Their aim was to investigate whether the layout of class diagrams influences the accuracy and efficiency of six different categories of software comprehension tasks. The results indicated that the multi-cluster layout demonstrates higher accuracy and faster task completion across a majority of task categories also indicating its effectiveness in supporting software comprehension.

In 2010, Sharif and Maletic investigated how design patterns in class diagrams are identified by students [7]. The layout was determined to have a significant effect on both the accuracy of identification and also speed of correct identification. Later Sharif and Maletic sought to identify how design patterns effect comprehension of UML class diagrams via eye tracking [8]. Four design patterns were shown to participants as their eyes were tracked. The participants also answered a questionnaire about the diagrams they were shown. Results showed a significant increase in accuracy regarding multi-cluster layouts with the Strategy pattern, the same is true for the Observer pattern.

In 2018, Störrle et al. investigated how modelers read UML diagrams and if there are any strategies to be identified [25]. 28 modelers viewed 18 diagrams while their eyes were being tracked, along with several other metrics. The reading strategies vary with expertise and diagram type but not with layout quality. Störrle et al. sought to replicate previous findings, and determine if size has an impact on the readability of UML diagrams [26]. Computer Science students were made to answer questions about UML diagrams of varying size. The correlation between size and readability was found to be significant, and an optimal size was found. Razali et al. analyzes two previous experiments and determine the efficacy of UML class diagrams in comparison to other diagram standards [27]. Using the datasets from the two previous studies the researchers were able to determine many metrics and compare them between modeling styles. The paper found that semi-formal graphical styling could improve accessibility in formal notation. [28] reviews three experiments to determine the effectiveness of UML class stereotypes. Data from the three experiments was aggregated and analyzed. It found that there was a significant increase in performance among low experience participants, and stereotypes have great prospects as training tools.

Savary analyzed a large amount of synthetic UML diagrams to determine overall trends in modeling [29]. Tools were created to automatically collect data, which was used to determine various quality metrics which could then be analyzed. The researchers successfully gathered data on large trends such as diagram size and case type. This data can now be used to further advance how UML diagrams are made. Pourali et al. asked students to fix errors in UML diagrams and found that tools did not offer enough support and were ineffective for users who were dissatisfied [30]. Bergström et al. provide various criteria that can be used to evaluate layouts (with no semantics considered) [31]. Their results point to the orthogonal layout as the one that is most liked by users.

Prior studies have not studied how developers find defects in UML class diagrams with respect to a requirements specification. The study presented in this paper seeks to add to the body of evidence supporting the use of UML class modeling via empirically validated layouts that are based on architectural importance.

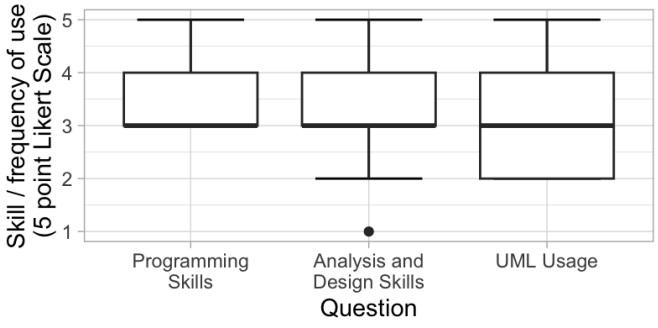

Fig. 1. Distribution of values of participant skill self assessment.

IV. Experimental Design

The experiment seeks to *analyze* the effect of two class diagram layouts (orthogonal and multi-cluster) *for the purpose of* evaluating defect detection in class diagrams related to a requirements document *with respect to* effectiveness (accuracy) and efficiency (time) *from the point of view of* the researcher *in the context of* students at two universities. Table III provides an overview of the experimental factors. The study follows a within-subjects group design, where each participant sees both layouts albeit in two different systems. This allowed us to collect more data points for each of the layout categories.

A. Participants

The participants are students recruited from two Midwestern universities in the USA. The recruitment was done from software engineering classes that had exposure via several lecture modules on UML diagrams including design patterns. There are 89 participants who took part in the study.Figure 1 shows the skill distribution collected from the pre-questionnaire prior to the study, with the full set of questions on self-reported knowledge shown in Table II.

Of all 89 participants in this study 45% of them started with C as their first language. 51% of the participants rated themselves "Average" skilled in programming, with a low level of standard deviation of 0.62. 54% of the participants rated themselves as "Average" when evaluating their ability to do analysis and design of programming systems, with a small standard deviation of 0.8. 59% of the programmers have been programming for 2-5 years. 49% of programmers studied 2-5 years of object oriented programming. 37% of participants report occasionally using UML, 32% report using UML frequently. 62% report having not learning a specific layout for UML. 82% of the participants use UML exclusively in academia.

B. Study Variables

An overview of the study variables in the experiment is shown in Table III. The main factor (independent variable) in the study is the UML layout with two treatments: multi-cluster or orthogonal layouts. The dependent variables in the study are task accuracy and task time. Secondary variables include working memory capacity measured by object memory task score and mental rotation task score as separate measures. The accuracy has a maximum score of 10 points (5 for each task). Time is measured in seconds. The object memory score can be up to 14 points and the mental rotation score goes up to 4 points.

C. Subject Systems and Layouts

The subject systems used for the defect detection tasks are two open-source systems written in C++, namely, Doxygen (v. 1.6.1) and Qt (v. 4.3.3), each with 100+ classes. Doxygen is a documentation generation application and Qt is a GUI application. The systems are reverse-engineered using a static analysis reverse engineering tool in MS Visual Studio to generate the corresponding UML models. We manually refined the models after inspecting the code to include other relationships that are missed such as associations and aggregations. In addition, these relationships are verified are the tool by Sutton et al. [32], [33].

For each system, a UML class diagram is drawn with a subset of the model and relationships relevant to the requirements being tested. The diagram is presented in both the orthogonal layout and the multi-cluster layout. In total, there are 4 diagrams drawn, 2 for Doxygen and 2 for Qt. An example of the orthogonal and multi-cluster layouts for Doxygen is shown in Figure 2. The diagram shows a subset of the UML model along with all the relationships between the classes. Only certain attributes and methods relevant to the tasks are shown that were relevant to the task at hand. These diagrams were hand crafted to fit on a printed 8.5 by 11 page. The diagrams are drawn in MS Visio using the imported reversed engineered model. Refer to our replication package for the layouts in Qt. The multi-cluster layout is designed to meet the heuristics for the multi-cluster layout. Note that it is possible for someone to devise a different set of clusters for the multi-cluster layout based on what they deem important. The orthogonal layout is generated using MS Visio's in-built orthogonal layout with minor adjustments to fit the page. These minor adjustments did not alter the layout drastically.

Both the orthogonal and the multi-cluster layouts are drawn with the stereotype information via textual annotation and color to avoid any confounding factors or biases even though the only layout that actually used this information to position classes and relationships is the multi-cluster layout. The orthogonal layout does not use the stereotype information.

D. Tasks

The experiment is split into two parts: defect detection tasks and working memory capacity tasks. A requirements document approximately 2 pages in length was written for each system in which the features and modules of each system and the relationship between classes are explained. The requirements are derived from existing documentation of the open-source system. Enough information is given to help participants understand what the system does.

TABLE II
SELF-REPORTED LEVEL OF KNOWLEDGE FOR EACH ATTRIBUTE

Question \ Response	No Knowledge	Theoretical Knowledge	Applied in Academic Environment	Applied in 1 Industrial Project	Applied in >1 Industrial Project
UML	0	7	44	4	6
Designing Software	1	17	30	7	5
Documenting Software	3	12	33	5	7
Implementing Software	0	11	31	7	11
Reviewing Source Code	2	15	32	6	4
Reviewing Designs	4	20	26	8	3
Maintaining/Debugging Software	2	20	23	6	10
Software Inspections	18	20	17	1	3
Quality Assurance for Software	13	30	11	3	3
OO Programming	1	4	39	7	10

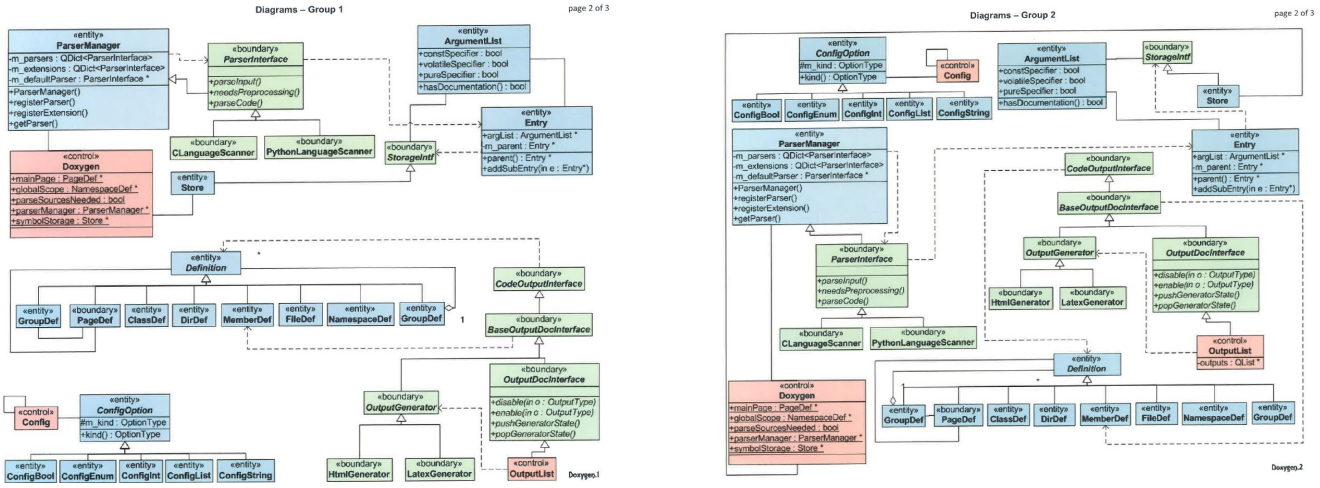

Fig. 2. The Doxygen UML class diagrams in the multi-cluster (left) and orthogonal (right) layouts

TABLE III
EXPERIMENT OVERVIEW

Goal	Study the effect of two types of class diagram layouts on defect detection in class diagrams with respect to a requirements document
Independent variable	Class diagram layouts with two treatments: orthogonal layout and multi-cluster layout
Dependent variables	Accuracy, Time
Secondary factors	Working Memory Capacity (object memory and mental rotation)

TABLE IV
DEFECTS INJECTED IN DOXYGEN AND QT. EACH OF THE 5 QUESTIONS HAD SPECIFIC ITEMS (CLASSES AND RELATIONSHIPS) TO CHECK. ONLY ONE OF THE FIVE QUESTIONS WAS A MATCH FOR EACH SYSTEM.

Q#	Doxygen	Qt
1	Match	Defect (subclass missing)
2	Defect (missing section field, incorrect relationship)	Match
3	Defect (incorrect class stereotype and class relationship)	Defect (incorrect relationship)
4	Defect (incorrect relationship, reversed dependency)	Defect (missing dependency)
5	Defect (redundant classes, incorrect hierarchy)	Defect (missing relationship)

A defect detection task with 5 sub-parts (questions) was created for each system i.e., Doxygen and Qt. Each sub-part is a checklist of classes, methods, and corresponding relationships to be checked for compliance with the requirements document of that system. For each sub-part the participant can choose Match or Defect for their answer based on their inspection of the items to be checked. The participants are encouraged to explain why they chose Defect in words for each sub-part.

For each system, four defects are injected in different parts of the diagram. The defects included the following: missing class data members, incorrect relationship types, incorrect class stereotypes, incorrect hierarchies, reversed dependency relationships, redundant classes, missing classes and missing relationships. There is only 1 task in each system that is a Match and not a Defect. See Table IV for a list of defects injected. The replication package provides more details on the Classes and items to check along with more details on the missing and incorrect items.

There are two types of working memory capacity tasks: mental rotation tasks and object memory tasks. There are four questions for the mental rotation tasks and two questions for the object memory tasks. In the first two mental rotation

questions, the prompt read "In the figures below, one of the shapes (A-D) is identical to the first figure but has been rotated. Which figure is identical to the first?". The first two questions are on 2D figures. The next two mental rotation questions are similar but with 3D shapes. The prompt read "Which of the three comparison shapes on the right is identical to the shape on the left except for its rotation angle?". In the object memory tasks, there are two questions. The first question asked the participants to study an array of letters and numbers of size 9 in length for 1 minute. The prompt read "Circle the items in the array that have been circled or swapped.". The second question is similar but now the participants had to study an array of five object shapes in order for 1 minute. The prompt for the second question read "Number each item in the array according to its position in the original array from 1 to 5."

E. Grouping

The participants are randomly split into two groups. Group 1 had 46 participants and Group 2 had 43 participants. See Table V for how the groups are split across the two layouts and subject systems. With this design, we are able to expose each participant to each layout albeit in a different system. We use a different system for the second task to avoid learning effects. The comparisons that we seek to make are between the groups. Our analysis compares the Doxygen multi-cluster layout with the Doxygen orthogonal layout across groups. Similarly, the Qt multi-cluster layout is compared with the Qt orthogonal layout across groups.

TABLE V
PARTICIPANT GROUPINGS AND LAYOUTS USED FOR EACH DEFECT DETECTION TASK.

Task	**Group 1 (N=46)**	**Group 2 (N=43)**
Task 1	Doxygen in multi-cluster layout	Doxygen in orthogonal layout
Task 2	Qt in orthogonal layout	Qt in multi-cluster layout

F. Study Procedure and Data Collection

The study was conducted at two universities in the USA. Approval for the study was obtained from the Institutional Review Board before the study was conducted. The entire study is performed in one sitting in a classroom-like setting using a paper-based questionnaire. We direct the reader to our replication package for an example of the study booklet.

Since this study is conducted during a software engineering class during a semester, the instructor designated lab time for the students to complete the study. Students are given extra credit for participating in the study. The students are all seated at their individual tables and given the paper booklet. The computer in front of them had the time, which they use to record the start and end time for each task sub-part. The computer is not logged in and only displayed the time. They are not allowed to use the web to find the answers to the questions. The lab is monitored by a lab assistant to make sure everyone is on task.

The study started with the students completing the pre-questionnaire on paper. After that they proceed to read a one-page instruction list to prepare them for what the study is about. They are told that the requirements document given to them is correct with no defects. The diagram however may or may not contain defects. Their task is to check if certain parts of the class diagram are compliant with the requirements document. They are given guidance to first read the requirements document and then asked to look for defects in the diagram with respect to the following.

- Missing classes, relationships, attributes.
- Incorrect class/attribute visibility, incorrect class stereotype, incorrect relationship between classes, incorrect multiplicity, wrong directionality, incorrect attribute types.
- Redundant and misleading information in the class diagram that causes ambiguity.
- Any other kind of inconsistency observed compared to the requirements document.

Next, a sample tutorial task is presented on an Automated Teller System (ATM) system. A 1-page description of an ATM system is given followed by a class diagram of an ATM system. The tutorial task has only 3 sub-parts and is solved for them. For example, there is one match and two defects in the tutorial task. When it is a defect, an explanation is required to be given. One of the defects in the ATM system is that a subclass for BalanceInquiry is missing even though the requirements document specified it. Another defect is that the multiplicities between the ATM and UIConsole are incorrectly shown as 1 to many instead of 1 to 1. Once this task is done, the end of the page instructed them that the actual study will begin on the following page. All participants then continue to do Doxygen task (in the layout assigned to their group) followed by the Qt task (in the layout assigned to their group). The start time and end time for each of the sub-parts in Doxygen and Qt is recorded using wall clock time shown in front of them on the monitor. After the Doxygen and Qt task, they are asked a short feedback questionnaire on whether they understood the diagram, requirements document, question, found all defects, level of difficulty, and defect detection confidence. There is also space for open-ended comments if they wished to write more. This feedback between each task is not timed. Note that the participants are not aware of the different layouts used between tasks.

After both the defect detection tasks, they are asked to complete four trials of mental rotation tasks - two in 2D and two in 3D. These are also timed for each set of 2D and 3D tasks. Following this, they completed two object memory tasks - the first is on the letter and number array and the second is on shapes. Finally, they completed a short post-questionnaire on their experience in the study.

G. Verifiability

The study replication package is at https://osf.io/2u3p5. Provided in the package are the paper-based questionnaires for each group, the digitized responses along with the mapping

Fig. 3. Answer accuracy of each sub-task question on the Doxygen system for the multi-cluster layout and orthogonal layout.

Fig. 4. Answer accuracy of each sub-task question on the Qt system for the multi-cluster layout and orthogonal layout.

scheme used, correct answers for all tasks, and the scripts used to analyze the data.

V. Experimental Analyses and Results

This section discusses the steps taken to first pre-process the data in a form that is ready for statistical analysis. Results for each of the research questions are presented next. Finally results from the post questionnaire are presented.

A. Data Pre-processing

Before the results are analyzed, all the paper-based questionnaire data was digitized into a spreadsheet. The data from pre-questionnaires, post-questionnaires, defect detection tasks, object memory tasks, and mental rotation tasks are entered into electronic format. We divided this work among five undergraduate researchers. A lead was assigned to collate all the results and double-check the data for accuracy. A scheme of tabulating each answer for all the tasks was carefully crafted to make sure all the data was coded correctly. The replication package is listed in Section IV-G and contains a complete mapping of the column information that matches the paper-based tests. Trustworthiness of the data was established through splitting parts across multiple data entry coders. An R script was written to process this digitized file to grade the answers for the defect detection tasks based on Table IV. Based on the start and end time recorded, the time elapsed for each question was calculated by the script as well. The working memory tasks were also graded in a similar way. We now present the results for each research question.

B. RQ1 Results: Accuracy

1) Doxygen System: The task accuracy for each question sub-task on the Doxygen layouts can be seen in Figure 3. We observed that overall, participants who read the Doxygen UML class diagram in the multi-cluster layout had a higher accuracy in answering questions correctly with a mean score of 3.543 out of 5. On the other hand, participants who read the diagram in the orthogonal layout had a mean score of 3.256 out of 5.

The feedback after answering each question for the Doxygen system is shown in Figure 5. When looking at the feedback for the multi-cluster layout on average, we saw a higher level

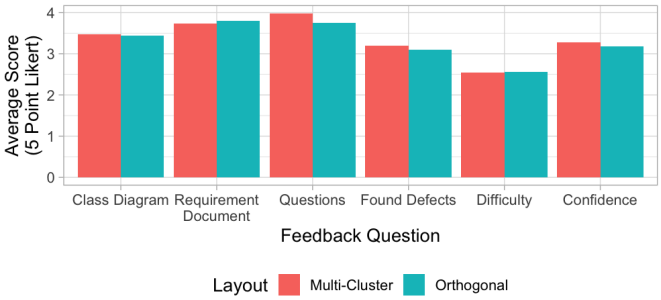

Fig. 5. Average participant feedback for the Doxygen system for each layout type.

of confidence and a higher level of understanding of the class diagram, questions, and defect finding compared to the orthogonal layout.

To see if there is any statistical difference, we first performed the Shapiro-Wilk test on each participant's scores on the Doxygen system. We found the scores are not normally distributed ($W = 0.878, p < 0.001$), and therefore we use the Wilcoxon rank sum test. There is no significant effect of layout on score observed ($W = 1096, p = 0.357$, Cohen's $d = 0.258$), and therefore we fail to reject the null hypothesis $H1_0$.

2) Qt System: The task accuracy for each question sub-task on the Qt layouts can be seen in Figure 4. Similar to the Doxygen layouts, we observe a higher overall question

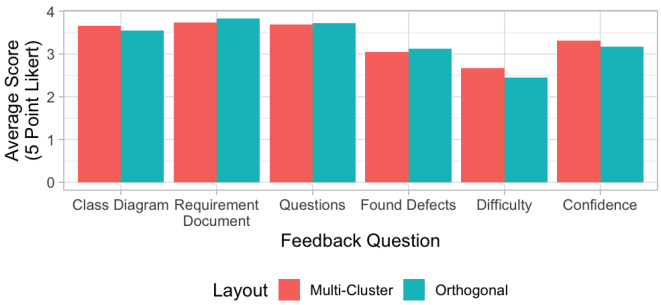

Fig. 6. Average participant feedback for the Qt system for each layout type.

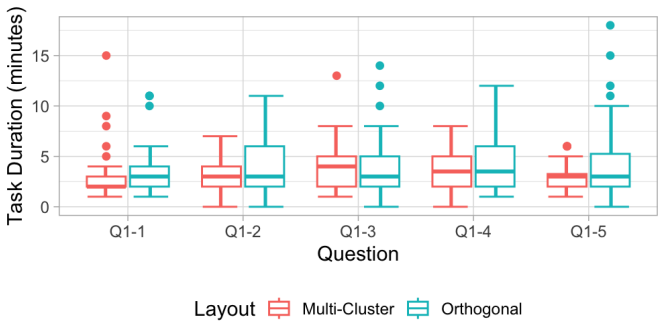

Fig. 7. Durations of each sub-task for the Doxygen system.

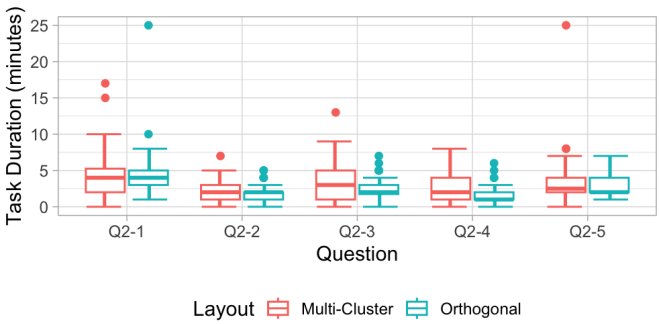

Fig. 8. Durations of each sub-task for the Qt system.

accuracy on the multi-cluster layout vs. the orthogonal layout (mean scores of 2.581 out of 5 vs. 2.456 out of 5). However, the scores on both are overall lower compared to the Doxygen layouts.

The feedback after answering the set of questions for the Qt system can be see in Figure 6. On average, we see a higher level of self-reported confidence, difficulty, and level of understanding of the class diagram, but the perceived level of understanding is lower for the requirement document, questions, and finding defects.

To determine if the difference is statistically significant, we again test the normality with the Shapiro-Wilk test. We again observe the scores to be not normally distributed ($W = 0.935, p < 0.001$), and therefore use the Wilcoxon rank sum test. We did not observe a significant effect ($W = 1030.5, p = 0.731$, Cohen's $d = 0.093$), and therefore we do not reject the null hypothesis $H1_0$.

C. RQ2 Results: Time

The average time taken for participants to complete the entire study was 38.9 minutes. Of these, the average time participants took for the Doxygen system questions was 18.6 minutes while the Qt system took on average 14.9 minutes.

1) Doxygen System: The task duration in minutes for each sub-task on the Doxygen layouts can be seen in Figure 7. We observe that overall, participants who read the Doxygen UML class diagram in the orthogonal layout spent more time answering each question (mean = 4.045) compared to the multi-cluster layout (mean = 3.385).

We look to see if these observations are statistically significant. However, when we performed the Shapiro-Wilk test on the question answer times for the Doxygen system, we see that it is significantly non-normal ($W = 0.824, p < 0.001$). Because of this, we use the Wilcoxon rank sum test to see if the differences in answer times for each treatments are significant. However, we do not observe a significant effect ($W = 1054.5, p = 0.626$, Cohen's $d = -0.288$). Therefore, we do not reject the null hypothesis $H2_0$.

2) Qt System: The task duration in minutes for each sub-task on the Qt layouts can be seen in Figure 8. Unlike the Doxygen layout, we observe participants spending less times answering each question when they read the Qt UML class diagram in the orthogonal layout (mean = 2.6) compared to the multi-cluster layout (mean = 3.322).

When looking for statistical significance, we observe that times are also non-normal when performing the Shapiro-Wilk test ($W = 0.697, p < 0.001$). We therefore also use the Wilcoxon rank sum test to compare between the layouts, but find the differences to be not significant ($W = 891, p = 0.075$, Cohen's $d = 0.450$). Because of this, we do not reject the null hypothesis $H2_0$.

D. RQ3 Results: Working Memory

We want to correlate our working memory results with defect detection task accuracy for each system and see if one system correlates more than the other. Each of the participants completed two working memory tasks: the *object memory* task and the *mental rotation* task. The object memory task had a total of 14 points, and the mean score was 12.989 across all participants with a mean task time of 1.472 minutes. The mental rotation score had a total of 4 points, and the mean score was 3.011 across all participants, with a mean task time of 3.629 minutes.

Our data for the defect detection score, defect detection duration, object memory test score, and mental rotation test score were not normally distributed when running the Shapiro-Wilk test on each metric ($W = 0.952, p = 0.002$, $W = 0.867, p < 0.001$, $W = 0.611, p < 0.001$, $W = 0.818, p < 0.001$ respectively). Because of this, we will use Spearman's correlation to determine any significant correlations.

1) Doxygen System:

a) Object Memory Test: The distribution of object memory scores versus defect detection scores on the two Doxygen system layouts can be seen in Figure 9. When we test for a statistically significant correlation, we find that there is a significant positive correlation between object memory scores and defect detection scores ($r_s = 0.325, p = 0.027$) for defect detection tasks on the Doxygen system with the multi-cluster layout. However, there was no significant correlation when the layout was orthogonal ($r_s = 0.133, p = 0.394$). From this, we have partial support for the alternate hypothesis $H3_a$.

We then evaluate the correlation between object memory scores versus the time it took to complete defect detection tasks on the two Doxygen system layouts. When testing

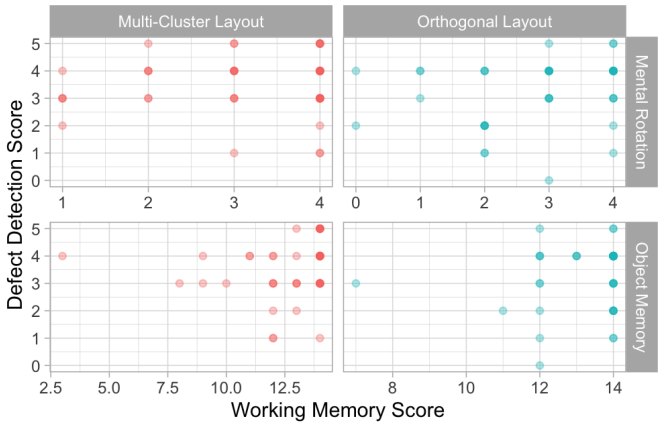

Fig. 9. Distribution of the total defect detection score vs. working memory scores on the Doxygen System.

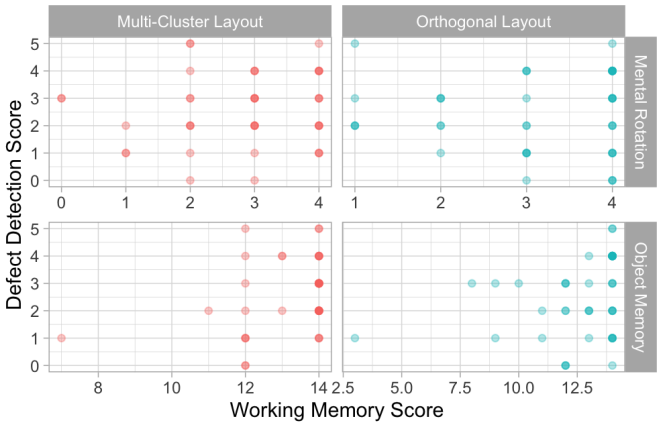

Fig. 10. Distribution of the total defect detection score vs. working memory scores on the Qt System.

for significant correlations, we did not find any significance between defect detection times and object memory scores for either the multi-cluster layout ($r_s = -0.06, p = 0.681$) or the orthogonal layout ($r_s = -0.183, p = 0.277$). Because of this, we fail to reject the null hypothesis $H4_0$.

b) Mental Rotation Task: The distribution of mental rotation scores versus defect detection scores on the two Doxygen system layouts can be seen in Figure 9. We were not able to find a significant correlation for either the multi-cluster layout ($r_s = 0.074, p = 0.623$) or the orthogonal layout ($r_s = 0.277, p = 0.073$). Therefore, we fail to reject the null hypothesis $H3_0$.

We also evaluate the correlation between mental rotation scores versus the time it took to complete defect detection tasks on the two Doxygen system layouts. When testing for significant correlations, we did not find any significance between defect detection times and mental rotation scores for neither the multi-cluster layout ($r_s = 0.232, p = 0.121$) nor the orthogonal layout ($r_s = 0.130, p = 0.445$). As such, we fail to reject the null hypothesis $H4_0$.

2) Qt System:

a) Object Memory Test: The distribution of object memory scores versus defect detection scores on the two Qt system layouts can be seen in Figure 10. When testing for significance, we observe a significant correlation between object memory scores and defect detection scores on the orthogonal layout ($r_s = 0.338, p = 0.022$) but not the multi-cluster layout ($r_s = 0.280, p = 0.069$). Therefore, we have partial support for the alternative hypothesis $H3_a$. We then evaluate the correlation between object memory scores versus the time it took to complete defect detection tasks on the two Qt system layouts. When we test for correlation, there was no significance for neither the multi-cluster layout ($r_s = -0.003, p = 0.983$) nor orthogonal layout ($r_s = -0.140, p = 0.387$). Because of this, we fail to reject the null hypothesis $H4_0$.

b) Mental Rotation Test: We also evaluated whether there was a correlation between mental rotation scores and defect detection scores on the two Qt system layouts. There was no significant correlation for neither the multi-cluster layout ($r_s = 0.106, p = 0.498$) nor the orthogonal layout ($r_s = 0.072, p = 0.634$). As such, we fail to reject the null hypothesis $H3_0$. When evaluating a correlation between mental rotation scores and defect detection task times, no significant correlations were found for neither the multi-cluster layout ($r_s = 0.031, p = 0.846$) nor orthogonal layout ($r_s = 0.121, p = 0.455$). This fails to reject the null hypothesis $H4_0$.

E. Post-Questionnaire Results

In the post questionnaire, 79% of participants chose the option of"I know and understand UML class diagrams". 38% of participants rated Doxygen defects as difficult, and 33% rated finding defects in Doxygen layout as very easy. 35% of participants rated Qt defect detection as difficult, and 34% rated it as undecided. 36% of the participants do not believe they found the most defects in the study, another 35% are undecided. 40% of participants found it harder to find defect inside of design patterns, with 37% being undecided. 70% of participants found the study a nice exercise. 54% of participants believe they performed average with 27% rating they performed below average. 48% of participants used to Doxygen but were not familiar with the design and 38% were very familiar with the design and structure. 49% of participants heard of Qt but were not familiar with the design, and 38% were familiar with the design and structure of Qt. 43% of participants disagreed with the tutorial helping them understand UML concepts, with 42% strongly disagreeing. 46% of participants somewhat disagree that they understand UML class stereotypes, with 28% strongly disagreeing, and 22% being neutral. 81% of participants replied yes when asked if this exercise gave them useful insights into class diagrams in real life. Other comments include that the study is a good practical exercise to see how class diagrams are used in real world software. Some comment on how they found the use of design patterns in real world software shown in the diagram

quite refreshing. They also commented that the tutorial helped them calibrate for what the study is about.

VI. Threats to Validity

We discuss some of the threats to validity pertaining to this study and discuss ways in which we try to mitigate them.

A. Internal Validity

With respect to *internal validity*, to avoid learning effects of the system as a whole, we have two groups and switch the layouts for each system between them. This means we perform paired analyses across two groups, where one group used the orthogonal layout and the other used the multi-cluster layout for the same system. We acknowledge mistakes could have occurred during the digitizing of the data from paper questionnaires. To minimize this, we use multiple people code in the answers and had one person double-check all the data. The entire study is done in one sitting in a classroom-like setting and the paper-based test is the best option we had at the time. The participants are not aware of the hypothesis of the study or that they are tested on the layout of the diagrams.

To make sure there are no confounding variables with diagram layout, models chosen in the diagram, and accuracy/time, each sub-part of each task in each subject system pertained to a different part of the diagram so as to avoid major learning effects within each task. The models used are reverse-engineered and double-checked for accuracy. The diagrams are manually engineered following heuristics for the multi-cluster and orthogonal layouts [5]–[8] presented in prior literature with no other differences. There is however, a possibility for someone to create a slightly different variation within each layout type. Also, it could be the case that the layout could have been biased to the questions. The multi-cluster layouts are drawn to represent different features/modules of the system. Care is taken to make sure the aesthetic criteria is met in all diagrams.

B. External Validity

With respect to *external validity*, we use real open-source systems and realistic tasks to validate the two layout schemes instead of toy applications. A subset of the models are carefully chosen to represent the main high-level features for each system. Another threat is that we use students in this study. We do not claim these results generalize to industry participants or to other tasks besides defect detection between requirements when UML class diagrams with these layouts are used.

C. Construct Validity

To make sure we minimize any threats to *construct validity*, we chose dependent measures that accurately reflect our accuracy and time data. A script is run to score the answers as well as calculate elapsed time for each question sub-part. The mental rotation and object memory tasks are standardized instruments to gauge memory capacity.

D. Conclusion Validity

Finally, to support *conclusion validity*, we use the appropriate non-parametric tests due to the non-normality of the data and perform paired Wilcoxon analyses to verify our hypotheses.

VII. Discussion

The main findings of this study are not significant. However, we observed that on average the multi-cluster layout had a higher accuracy than the orthogonal layout in both Doxygen and Qt. The multi-cluster layout also took less time compared to the orthogonal one for Doxygen but not Qt. The participants also reported a higher level of confidence and understanding for the multi-cluster layouts. The only significant correlation for working memory capacity is between accuracy and object memory scores for the multi-cluster layout only on Doxygen and orthogonal layout only on Qt.

Defect detection is a more complicated task than just comprehension and hence prone to more individual differences in cognitive processing than just comprehension. Furthermore, there are not too many studies in the literature that focus on UML class diagram comprehension on realistic systems. We posit this being one of the factors why we did not see major differences in the results. A simpler (memory only) recall task might have seen more differences but is left as a future exercise. In hindsight, we also believe the experiment design was too complicated to produce any conclusive findings. Reflecting on the design further, we would focus each defect detection task to just one question and not five sub-parts. The design pattern questions should also be posed as separate questions from the syntax related ones.

The fact that the scores are higher for the multi-cluster layouts could mean that with a larger sample size we might tend to see more significant results in support of the multi-cluster layout. Prior research showns that UML documentation helps in improving correctness and quality in more complex tasks [34]. Other researchers show that the use of UML modeling potentially reduces defect density in software systems [35]. Raghuraman et al. uses statistical modeling to show that projects that create UML models are less likely to be defect prone [36]. Given this evidence of the importance of UML modeling, the inconsistencies of these results call for additional studies using other data collection methods that give us more insight into how developers actually read the layouts while solving defects. We also observe that the task type can affect how the layout is being used. In other task types in earlier studies [6], the multi-cluster is significantly better in a majority of tasks (reading, overview, impact analysis, bug fix) but not all tasks (feature addition, refactoring). This paper presents a first study to test if layout plays a role in defect detection with respect to requirements. Given that previous experiments have shown the multi-cluster layout to be better with this current study showing it is not worse than the orthogonal layout, perhaps indicates that the recommendation for layout should be that multi-cluster be used for UML class diagrams.

Educators can use these results to help improve the teaching of UML class diagram defect detection skills by using the multi-cluster layout to help with comprehension and incorporating object memory tasks into the curriculum. In addition, a triager can use these results to identify people who might be better at finding critical bugs in model design based on their object memory score. More studies are needed to extrapolate these results to industry professionals.

VIII. Conclusions and Future Work

The paper presents a controlled experiment examining the effects of two layouts namely, orthogonal and multi-cluster on defect detection in class diagrams with respect to a requirements document. The working memory capacity is also measured and correlated to performance. Results do not show any significance in time or accuracy towards either the multi-cluster or the orthogonal layouts for either system, even though we find the average scores on tasks using the multi-cluster layout to be slightly higher, albeit statistically insignificant. The object memory score is significantly correlated with accuracy for the Doxygen multi-cluster layout and the Qt orthogonal layout tasks. Due to these inconsistent results, we believe future studies should be done to further look into other tasks and systems to pinpoint perhaps other features that could have played a role. As part of future work, we plan to conduct eye-tracking studies on UML diagrams from open source systems [37] with different expertise groups to gain further insight into how developers actually read the diagrams to uncover any differences that may exist in gaze measures. In addition, collecting qualitative feedback about what participants did (not) like about the diagram could give valuable additional information.

Acknowledgment

This work is supported in part by the US National Science Foundation under Grant Number CCF 18-55756. The authors are grateful to undergraduate researchers Aaron Linnell, Nathaniel Liess, Anthony Vinton, and Saicharith Vaitla for helping with digitizing the paper-based questionnaires and verification of data entry. We are thankful to all the participants who took part in the study.

References

[1] G. Booch, J. Rumbaugh, and I. Jacobson, *The unified modeling language user guide*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2005.

[2] M. Eiglsperger, C. Gutwenger, M. Kaufmann, J. Kupke, M. Jünger, S. Leipert, K. Klein, P. Mutzel, and M. Siebenhaller, "Automatic layout of UML class diagrams in orthogonal style," *Information Visualization*, vol. 3, no. 3, pp. 189–208, 2004. [Online]. Available: https://doi.org/10.1057/palgrave.ivs.9500078

[3] O. Andriyevska, N. Dragan, B. Simoes, and J. Maletic, "Evaluating UML class diagram layout based on architectural importance," in *3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis*, 2005, pp. 1–6.

[4] S. Yusuf, H. Kagdi, and J. I. Maletic, "Assessing the comprehension of UML class diagrams via eye tracking," in *15th IEEE International Conference on Program Comprehension (ICPC '07)*, 2007, pp. 113–122.

[5] B. Sharif and J. I. Maletic, "An empirical study on the comprehension of stereotyped UML class diagram layouts," in *The 17th IEEE International Conference on Program Comprehension, ICPC 2009, Vancouver, British Columbia, Canada, May 17-19, 2009*. IEEE Computer Society, 2009, pp. 268–272. [Online]. Available: https://doi.org/10.1109/ICPC.2009.5090055

[6] ——, "The effect of layout on the comprehension of UML class diagrams: A controlled experiment," in *Proceedings of the 5th IEEE International Workshop on Visualizing Software for Understanding and Analysis, VISSOFT 2009, Edmonton, Alberta, Canada, September 25, 2009*, H. A. Müller, M. Lanza, and M. D. Storey, Eds. IEEE Computer Society, 2009, pp. 11–18. [Online]. Available: https://doi.org/10.1109/VISSOF.2009.5336430

[7] ——, "The effects of layout on detecting the role of design patterns," in *Proceedings 23rd IEEE Conference on Software Engineering Education and Training, CSEE&T 2010, Pittsburgh, Pennsylvania, USA, 9-12 March 2010*. IEEE Computer Society, 2010, pp. 41–48. [Online]. Available: https://doi.org/10.1109/CSEET.2010.23

[8] ——, "An eye tracking study on the effects of layout in understanding the role of design patterns," in *26th IEEE International Conference on Software Maintenance (ICSM 2010), September 12-18, 2010, Timisoara, Romania*, R. Marinescu, M. Lanza, and A. Marcus, Eds. IEEE Computer Society, 2010, pp. 1–10. [Online]. Available: https://doi.org/10.1109/ICSM.2010.5609582

[9] B. Sharif, "Empirical assessment of UML class diagram layouts based on architectural importance," in *IEEE 27th International Conference on Software Maintenance, ICSM 2011, Williamsburg, VA, USA, September 25-30, 2011*. IEEE Computer Society, 2011, pp. 544–549. [Online]. Available: https://doi.org/10.1109/ICSM.2011.6080828

[10] T. Baum, K. Schneider, and A. Bacchelli, "Associating working memory capacity and code change ordering with code review performance," *Empirical Software Engineering*, vol. 24, no. 4, pp. 1762–1798, 2019.

[11] Z. Sharafi, Y. Huang, K. Leach, and W. Weimer, "Toward an objective measure of developers' cognitive activities," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 30, no. 3, pp. 1–40, 2021.

[12] A. R. Conway, M. J. Kane, M. F. Bunting, D. Z. Hambrick, O. Wilhelm, and R. W. Engle, "Working memory span tasks: A methodological review and user's guide," *Psychonomic bulletin & review*, vol. 12, no. 5, pp. 769–786, 2005.

[13] S. G. Vandenberg and A. R. Kuse, "Mental rotations, a group test of three-dimensional spatial visualization," *Perceptual and Motor Skills*, vol. 47, no. 2, pp. 599–604, 1978, pMID: 724398. [Online]. Available: https://doi.org/10.2466/pms.1978.47.2.599

[14] J. Ondik, M. Olejár, K. Rástočný, and M. Bieliková, "Activity-based model synchronization and defects detection for small teams," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2017, pp. 8–15.

[15] Y. Laurent, R. Bendraou, and M.-P. Gervais, "Executing and debugging UML models: an fUML extension," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 1095–1102. [Online]. Available: https://doi.org/10.1145/2480362.2480569

[16] T. v. Enckevort, "Refactoring UML models: using openarchitectureware to measure UML model quality and perform pattern matching on UML models with ocl queries," in *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*, ser. OOPSLA '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 635–646. [Online]. Available: https://doi.org/10.1145/1639950.1639959

[17] C. F. J. Lange and M. R. V. Chaudron, "Effects of defects in UML models: An experimental investigation," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 401–411. [Online]. Available: https://doi.org/10.1145/1134285.1134341

[18] H. Störrle, "On the impact of layout quality to understanding uml diagrams: Diagram type and expertise," in *2012 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2012, pp. 49–56.

[19] H. Hu, J. Fang, Z. Lu, F. Zhao, and Z. Qin, "Rank-directed layout of UML class diagrams," in *Proceedings of the First International Workshop on Software Mining*, ser. SoftwareMining '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 25–31. [Online]. Available: https://doi.org/10.1145/2384416.2384420

[20] J. W. v. Gudenberg, A. Niederle, M. Ebner, and H. Eichelberger, "Evolutionary layout of UML class diagrams," in *Proceedings of the 2006 ACM Symposium on Software Visualization*, ser. SoftVis '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 163–164. [Online]. Available: https://doi.org/10.1145/1148493.1148525

[21] M. Eiglsperger, M. Kaufmann, and M. Siebenhaller, "A topology-shape-metrics approach for the automatic layout of UML class diagrams," in *Proceedings of the 2003 ACM Symposium on Software Visualization*, ser. SoftVis '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 189–ff. [Online]. Available: https://doi.org/10.1145/774833.774860

[22] T. Dwyer, "Three dimensional UML using force directed layout," in *Proceedings of the 2001 Asia-Pacific Symposium on Information Visualisation - Volume 9*, ser. APVis '01. AUS: Australian Computer Society, Inc., 2001, p. 77–85.

[23] H. C. Purchase, M. McGill, L. Colpoys, and D. Carrington, "Graph drawing aesthetics and the comprehension of UML class diagrams: an empirical study," in *Proceedings of the 2001 Asia-Pacific Symposium on Information Visualisation - Volume 9*, ser. APVis '01. AUS: Australian Computer Society, Inc., 2001, p. 129–137.

[24] 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*, ser. APVis '01. AUS: Australian Computer Society, Inc., 2001, p. 113–120.

[25] H. Störrle, N. Baltsen, H. Christoffersen, and A. M. Maier, "How do modelers read UML diagrams? preliminary results from an eye-tracking study," in *Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings*, ser. ICSE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 396–397. [Online]. Available: https://doi.org/10.1145/3183440.3195025

[26] H. Störrle, "Diagram size vs. layout flaws: Understanding quality factors of UML diagrams," in *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '16. New York, NY, USA: Association for Computing Machinery, 2016. [Online]. Available: https://doi.org/10.1145/2961111.2962609

[27] R. Razali, C. F. Snook, and M. R. Poppleton, "Comprehensibility of UML-based formal model: a series of controlled experiments," in *Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies: Held in Conjunction with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE) 2007*, ser. WEASELTech '07. New York, NY, USA: Association for Computing Machinery, 2007, p. 25–30. [Online]. Available: https://doi.org/10.1145/1353673.1353680

[28] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato, "The role of experience and ability in comprehension tasks supported by UML stereotypes," in *Proceedings of the 29th International Conference on Software Engineering*, ser. ICSE '07. USA: IEEE Computer Society, 2007, p. 375–384. [Online]. Available: https://doi.org/10.1109/ICSE.2007.86

[29] M. Savary-Leblanc, X. L. Pallec, P. Palanque, C. Martinie, A. Blouin, F. Jouault, M. Clavreul, and T. Raffaillac, "Mining human factors general trends from +100k UML class diagrams," in *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, ser. MODELS '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 913–922. [Online]. Available: https://doi.org/10.1145/3550356.3559098

[30] P. Pourali and J. M. Atlee, "An empirical investigation to understand the difficulties and challenges of software modellers when using modelling tools," in *Proceedings of the 21th ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 224–234. [Online]. Available: https://doi.org/10.1145/3239372.3239400

[31] G. Bergström, F. Hujainah, T. Ho-Quang, R. Jolak, S. A. Rukmono, A. Nurwidyantoro, and M. R. Chaudron, "Evaluating the layout quality of uml class diagrams using machine learning," *Journal of Systems and Software*, vol. 192, p. 111413, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016412122200125X

[32] A. M. Sutton and J. I. Maletic, "Mappings for accurately reverse engineering UML class models from C++," in *12th Working Conference on Reverse Engineering, WCRE 2005, Pittsburgh, PA, USA, November 7-11, 2005*. IEEE Computer Society, 2005, pp. 175–184. [Online]. Available: https://doi.org/10.1109/WCRE.2005.21

[33] A. Sutton and J. I. Maletic, "Recovering UML class models from C++: A detailed explanation," *Information and Software Technology*, vol. 49, no. 3, pp. 212–229, 2007, 12th Working Conference on Reverse Engineering. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584906001844

[34] E. Arisholm, L. Briand, S. Hove, and Y. Labiche, "The impact of UML documentation on software maintenance: an experimental evaluation," *IEEE Transactions on Software Engineering*, vol. 32, no. 6, pp. 365–381, 2006.

[35] A. Nugroho and M. R. Chaudron, "Evaluating the impact of UML modeling on software quality: An industrial case study," in *Proceedings of the 12th International Conference on Model Driven Engineering Languages and Systems*, ser. MODELS '09. Berlin, Heidelberg: Springer-Verlag, 2009, p. 181–195. [Online]. Available: https://doi.org/10.1007/978-3-642-04425-0_14

[36] A. Raghuraman, T. Ho-Quang, M. R. V. Chaudron, A. Serebrenik, and B. Vasilescu, "Does UML modeling associate with lower defect proneness? A preliminary empirical investigation," in *Proceedings of the 16th International Conference on Mining Software Repositories*, ser. MSR '19. IEEE Press, 2019, p. 101–104. [Online]. Available: https://doi.org/10.1109/MSR.2019.00024

[37] G. Robles, T. Ho-Quang, R. Hebig, M. R. V. Chaudron, and M. A. Fernandez, "An extensive dataset of UML models in github," in *Proceedings of the 14th International Conference on Mining Software Repositories*, ser. MSR '17. IEEE Press, 2017, p. 519–522. [Online]. Available: https://doi.org/10.1109/MSR.2017.48