

# An Empirical Study on the Comprehension of Stereotyped UML Class Diagram Layouts

Bonita Sharif and Jonathan I. Maletic

*Department of Computer Science*

*Kent State University*

*Kent Ohio 44242*

*bsimoes@cs.kent.edu, jmaletic@cs.kent.edu*

## Abstract

*An empirical study is presented that investigates how stereotype based layouts impact the comprehension of UML class diagrams. This work continues a previous study using eye-tracking equipment by replicating it using an alternative method. Here instead, online questionnaires were used as a means of collecting data. Subjects were given two types of tasks: one addressing UML syntax and the other addressing questions concerning software design. Three different layout strategies are compared. Along with general aesthetics, the layouts are primarily organized based on the class stereotypes of control, boundary, and entity. Besides the answers, a confidence value for each question was collected from the subjects to help validate the categorization of subjects. Results of the study are compared and contrasted to the eye-tracking study done with the same tasks and layouts. Results show a significant improvement in performance in both types of tasks with the multi-cluster stereotyped layouts.*

## 1. Introduction

UML class diagrams are one of the most popular visual presentations of software designs. A number of empirical studies have shown UML diagrams to be very useful during software maintenance [3, 5, 22]. The research here focuses on how we perceive and comprehend class diagrams. Specifically, we are trying to understand how the use of stereotypes in combination with different layout strategies impacts comprehension. Stereotypes are a standard extension mechanism provided by the UML that allow users to define semantics on the notation thereby extending the language [10].

Three class stereotypes are generally accepted and used: control, boundary, and entity [4]. A boundary class models the interaction between a system and the external world. Entity classes models persistent information in a system. A control class models the coordination and sequencing of other objects in the system.

Class stereotypes are important in the process of understanding the role and importance of each class in a system as well as the whole of the system. They point out architecturally important classes and relationships and we feel they can be used to improve automated layout methods for class diagrams. In a pilot study [2], conducted previously, we found class stereotype based layouts to be a promising technique for further evaluation. Additionally, the usefulness of stereotypes for object classification and layout was investigated and assessed using eye-tracking [24].

While neither of these previous studies was comprehensive or statistically conclusive, both seemed to demonstrate that stereotypes are potentially important for comprehension. In the work presented here, we continue this line of research by replicating the eye-tracking study [24] with an alternative methodology. A survey based evaluation study was conducted with a larger number of participants and these results are compared and contrasted to the eye-tracking study.

The paper is organized as follows. Section 2 describes the stereotyped layouts used. The experiment design is discussed in Section 3. Section 4 analyzes the results and reports the findings of the experiment. The main results are highlighted and contrasted to the eye-tracking study in Section 5. Section 6 addresses the threats to validity. Related work is discussed in Section 7. Section 8 concludes the paper and discusses future work in the area.

## 2. Focus of the Study

We conducted an experiment that assesses the comprehension of class diagram layouts for two sets of tasks, namely UML tasks and design tasks. UML tasks address the syntax and structure of visually representing class diagrams whereas design tasks address the design comprehension aspect of the system with respect to maintenance and evolution. This experiment is a replication of an earlier experiment [24] where eye-tracking equipment was used to determine how subjects examined a class diagram. The goal of this work is to

further validate the results from the eye-tracking experiment and our own pilot study [2] with a bigger sample and varied experience of the subject system used. Replication with an alternative experimental method will also help better understand the relationship between different ability levels within the subjects in the two types of tasks examined. The ability level of subjects is determined based on self-assessment questions used in our background analysis and on their performance in both types of tasks. The two main research questions this paper tries to address can be stated as follows:

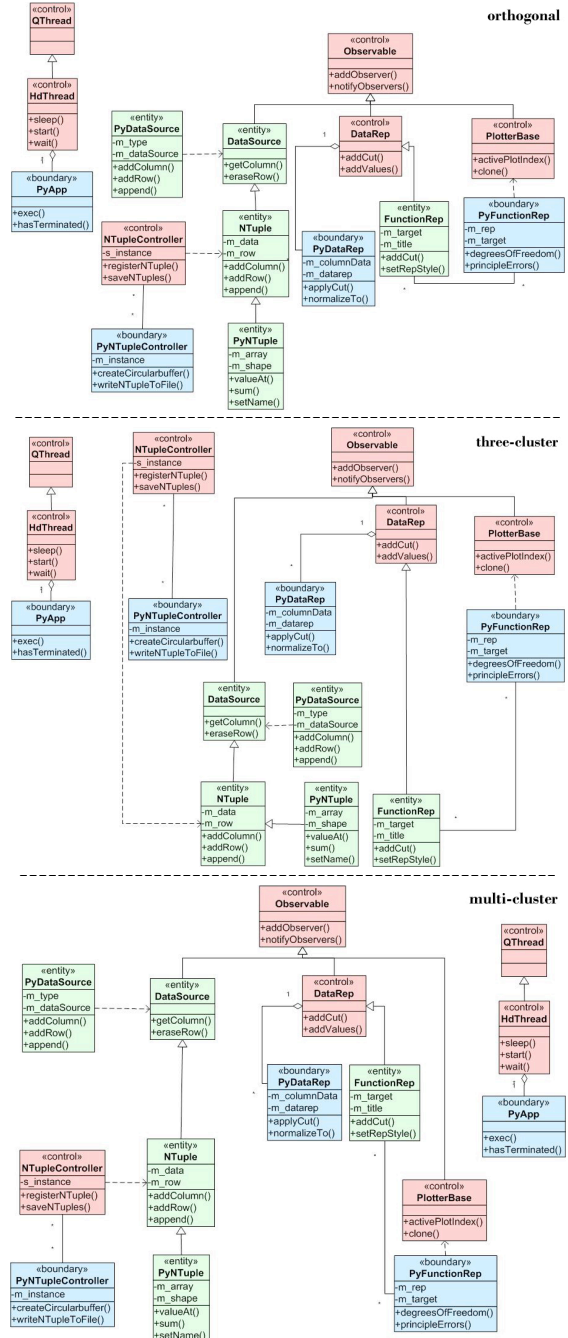
- RQ1: Is there an improvement in the comprehension of UML syntax based tasks and general design based tasks for the stereotyped class diagram layouts vs. layouts based on pure aesthetics?
- RQ2: How does the subject’s ability level affect the comprehension of stereotyped class diagram layouts vs. layouts based on pure aesthetics?

We make a distinction between a UML class model and a class diagram. A class diagram consists of a subset of classes and relationships in the class model and layout techniques. A class diagram need not consist of all possible classes and relationships and need not represent the entire system. Instead of developing a tool that uses our stereotype based layout techniques, we would first like to validate empirically the techniques that are most useful and base the tool on our validated results.

Reverse engineering tools [11, 21] can be used to generate the class model which represents the design of software. Class diagrams are then drawn based on these reverse engineered models by applying a particular layout algorithm to the classes and relationships in the diagram. The generation of class diagrams from reverse engineered class models is not necessarily automatic. Commercial tools such as Visual Paradigm (<http://www.visual-paradigm.com/product/vpum1/>) for UML automatically generate very rudimentary class diagrams after reverse engineering the class model. Most UML drawing editors pay more attention to reverse engineering the class model and less attention on the actual layout of the chosen classes to be displayed on the diagram.

In this study, the class stereotype is identified by textual annotations and color. In the UML, stereotypes are shown above the class name enclosed within guillemets. This is referred to as the textual annotation. For example, control classes will have the «control» textual annotation above their class name. In addition to textual annotations: control, boundary and entity classes are colored red, green, and blue respectively. Three types of layouts are used to draw class diagrams for the tasks. This selection is based on our previous work [2] in

assessing layout schemes for class diagrams. See Figure 1 for an example of the three layouts used.



**Figure 1. Class diagrams for the PythonWrappers module shown in three different layouts: orthogonal, three-cluster and multi-cluster**

The orthogonal layout is the current best practice method used in class diagram layout. The layout is based purely on general aesthetic criteria [1, 7, 9, 13, 16, 17] such as minimizing edge crossings, minimizing edge bends, minimizing edge length, maximizing symmetry,

and using 90 degree bends. This layout is used as a control layout in this experiment. It does not use information about the class stereotype in layout positioning.

The three-cluster layout uses information about the class stereotype to position classes into three clusters, one for each stereotype. This layout would be seen having control classes in the control cluster, entity classes in the entity cluster and boundary classes in the boundary cluster. This layout is based on the architectural importance of each class in the system.

The multi-cluster layout also uses information about the class stereotype to position classes into multiple clusters. Each cluster represents a cohesive unit where control, boundary and entity classes work together. This layout also depends on the types of relationships that exist between these classes. A cluster could represent a specific feature in the system. One can think of the multi-cluster layout as a specialization of the three-cluster layout.

Even though the orthogonal layout did not use stereotype information, classes in this layout still displayed the class stereotype and color in order to keep the visual design information the same across layouts. It is important to note that the three-cluster and multi-cluster layout do not ignore general aesthetics completely. Instead, they are given a lower priority with a higher priority given to stereotypes and their associated relationships. This means that we could introduce few edge crossings in the stereotyped layouts in order to better visualize the roles of classes and functional requirements of the system. This is unlike the orthogonal layout that tries to reduce edge crossings as its first priority.

### 3. Experimental Design

This section presents details on the logistics of the experiment. The overall design, hypotheses, subject system used, subjects, tasks, data collection and the basic running of the experiment is given.

#### 3.1. Experiment Goal and Hypotheses

The experiment seeks to *analyze* class diagram layouts primarily based on class stereotypes *for the purpose of* evaluating their usefulness in two categories of software comprehension tasks *with respect to* effectiveness (accuracy) and efficiency (time) *from the point of view of* the researcher *in the context of* students at Kent State University.

The high level hypothesis of this experiment is that the layouts based on the class stereotype increase the comprehension of the system. The null hypotheses are formulated below. The alternative hypotheses are easily derived from them.

- $H_{0u1}$ : There is no significant difference in UML task comprehension between class diagrams based on the orthogonal layout vs. the three-cluster layout
- $H_{0u2}$ : There is no significant difference in UML task comprehension between class diagrams based on the orthogonal layout vs. the multi-cluster layout.
- $H_{0d1}$ : There is no significant difference in design task comprehension between class diagrams based on the orthogonal layout vs. the three-cluster layout.
- $H_{0d2}$ : There is no significant difference in design task comprehension between class diagrams based on the orthogonal layout vs. the multi-cluster layout.

**Table 1. Experiment overview**

<b>Goal</b>	Study the comprehension effect of three types of class diagram layouts on two types of software tasks.
<b>Factor / Independent variable</b>	Class diagram layouts with three treatments: orthogonal layout, multi-cluster layout, three-cluster layout
<b>Dependent variables</b>	Quantitative variables: Accuracy, speed, confidence level
<b>Secondary factors</b>	Subject's ability level and task categories

The overview of the experiment is given in Table 1. The main factor being analyzed is the layout of class diagrams. We used three alternatives for the layout as discussed in Section 2. The dependent variables are described in Section 3.4. While analyzing the results we also looked at secondary factors such as the subject's ability level and the task category (UML and design).

The experiment was conducted as a within-subjects design where all subjects were given all three types of treatments of the factor i.e., class diagram layout. We used this to compare our results to the eye-tracking experiment which also used the same design. Another reason was to gather more data points for each layout.

#### 3.2. Subject System

Hippodraw [14] is used as the subject system under investigation. It is an interactive and Python scriptable statistical data analysis application and framework written in C++ with Qt for the user interface. Hippodraw consists of nearly 96 KLOC in over 600 source and header files. Hippodraw does not come packaged with any design documents (class models or class diagrams). However, doxygen documentation is available. We reverse engineered Hippodraw source code using the srcTools framework [21] to generate the class model. This gave us the classes, associations, generalizations and dependencies between the classes. Some associations were changed to aggregations by manually inspecting the source code. Next, we manually constructed class diagrams using the class model in a UML drawing editor. A total of 100 unique classes are used in this study.

### 3.3. Tasks

The tasks used in this study are identical to the eye-tracking study [25] and we refer the reader there for detailed information about each task and only briefly describe the tasks here. This study consists of two types of tasks: UML tasks and design tasks. The subjects had to provide an answer to a total of 27 questions: 12 UML tasks and 15 design based tasks. The UML tasks were based on four Hippodraw modules whereas the design tasks were based on five modules. A total of six Hippodraw class modules were constructed based on related functionality which resulted in 18 diagrams (6 diagrams \* 3 layouts). Three of the modules were common between UML tasks and design tasks.

Three of the modules consisted of one class diagram drawn in three different layouts (9 diagrams). The other three modules had 2 diagrams drawn in three different layouts (18 class diagrams). The diagrams were manually engineered in a UML drawing editor using general aesthetic criteria [1, 7, 9, 13, 16, 17] and stereotype information (in the case of the three-cluster and multi-cluster layout). The number of classes used for each of the modules range from a minimum of 12 (XmlNode module) to a maximum of 21 (PlotterBase module) classes.

An example of a UML task would be asking the subject to select all the classes involved in dependency or to identify the kind of relationship between two classes. These types of tasks depend on the UML syntax/notation used for class diagrams.

A design related task required the subject to analyze the class diagram to answer specific questions about understanding Hippodraw. One example of a design task used was: Which class controls the active window of an application? The subjects did not need to be an expert in Hippodraw's design and/or implementation to answer these questions. The answers to questions could be found by analyzing the classes, relationships, attributes, methods, and stereotypes.

**Table 2. Three design tasks asked for the PythonWrappers module. The layouts used for these questions are shown in Figure 1**

ID	Layout	Question Text
Q13	Orthogonal	Select the class that a python wrapper uses to access data in the class NTuple
Q18	Three-cluster	Select the class that is a python wrapper for a class with the method name adduct.
Q23	Multi-cluster	Name the entity class that is responsible for storing data

Since this is a within-subjects study, the same task is not asked for more than one layout. This eliminates any learning bias involved in answering the same question

twice. Instead, similar questions were asked for the three layouts. Consider the following three questions from the PythonWrappers module. Each of these questions is similar in nature that allows analysis of subject's performance across three layouts. The class diagrams that accompany these questions contain the same information with the exception of the layout.

**Table 3. Debriefing Questionnaire**

ID	Question	Mapping
1	I had sufficient time to complete the questions	1-5
2	I think the questions were difficult to answer	1-5
3	The questions were clear to me	1-5
4	I was able to understand information in the class diagrams	1-5
5	I think the questions were realistic	1-5
6	I found UML class stereotypes useful in answering questions	1-5
7	Did you concentrate on the spatial layout while answering questions?	Yes = 1 No = 0 Not sure = 2

### 3.4. Data Collection

We used three online questionnaires to gather data in this experiment. The first questionnaire collected background information about the subjects. This information is presented in Section 3.5.

The second questionnaire consisted of the actual study tasks. Each task (UML and design) was given a score. We calculated the accuracy of answering UML tasks and design tasks based on the score. The speed i.e., time taken to complete each task was also recorded. Besides the accuracy and speed, we also collected a confidence level of the subject's answer for each task. The confidence level was on a Likert scale from 1 (not confident) to 5 (very confident). Finally, the third questionnaire was a debriefing questionnaire that collected data about the task and stereotypes used. See Table 3 for questions.

### 3.5. Subjects

We gathered twenty-nine subjects (14 undergraduate students and 15 graduate students) to participate in this experiment. The undergraduate students were within the age range of 18-25 years. Ten of the graduate students were between 25-35 years and five were between 35-45 years. They were all from the computer science department at Kent State University. There were 23 males and six females. Five of the subjects had used class diagrams in both academia and industry while the rest had theoretical knowledge of UML and applied it in an academic setting, typically in software engineering courses.

The subjects were informed that the purpose of the study was to understand how people interpret class diagrams (not their UML expertise). They were not aware of the experiment’s hypotheses or of the different layouts used. They were also instructed to answer the questions from the point of view of a maintainer trying to understand the system. We collected information about design and programming skills, number of years of experience in general programming and in OO programming and familiarity of Hippodraw. The programming and design skills were on a scale from 1 to 5, the others were on a 1 to 4 scale. A low rating indicates low experience/skill/familiarity. There was a large difference in the familiarity of Hippodraw among the participants. Most of the subjects (with the exception of 1) were not familiar with the design of Hippodraw. The subjects reported their self assessment of programming and design skills. This is correlated with the accuracy of UML and design questions in Section 4.

### 3.6. Study Instrumentation

The study was conducted online. A fixed amount of time up to one minute was allotted to each question. We did this to keep the subjects on task and to replicate the timing aspect of the eye-tracking experiment as close as possible. A couple of days before the experiment, subjects were asked to go through a class diagram tutorial. A short description of class stereotypes and their graphically representation was given. They were also informed of the colors used to differentiate between different class stereotypes. The tutorial was optional, however, all subjects with an exception of a few participated in the tutorial. The purpose of the tutorial was to make sure all subjects were on the same page with respect to understanding information presented in the class diagrams used in the study.

The following information was presented for each task: a question, answer choices and a class diagram in one of the three possible layouts. The subjects were asked to choose the answer for the question with respect to the class diagram. After all the tasks were completed, a debriefing questionnaire was presented for the subjects to complete. This concluded the experiment from the subjects’ viewpoint.

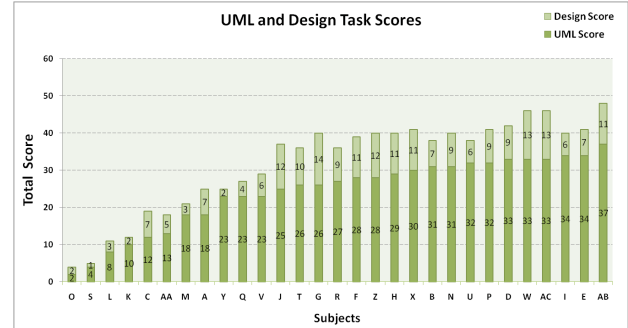
## 4. Experimental Results and Analyses

This section presents the results of this experiment. We first discuss the parts common to this study and the eye-tracking study: classification of subjects and questions. Next, we discuss exclusive observations pertaining to this study: effect of layout on UML and design tasks as well as the effect of ability and layout together. Confidence levels are correlated with

performance and skill level of subjects. Finally, we present results from the debriefing questionnaire.

### 4.1. Subject Classification

The accuracy and speed of 12 UML tasks and 15 design tasks were analyzed. Figure 2 shows the accuracy and speed for all 29 subjects. None of the subjects answered all UML questions and all design questions correctly.



**Figure 2. UML and Design task scores across all twenty nine subjects. The maximum possible score is 42 for UML and 18 for design. The subjects are first sorted by UML score and then by design score.**

Based on the performance of subjects in answering the questions, we classified them into eight groups. This was also done in the eye-tracking study [25]. Table 4 shows the categories. The main groups are: Agnostic (A), Inexperienced (I), Knowledgeable (K) and Expert (E). UML scores in the ranges of [0,23], [24,29] and [30,42] were mapped to the agnostic, knowledgeable and expert category respectively. Design scores in the ranges of [0,3], [4,6], [7,10] and [11,18] are mapped to the agnostic, inexperienced, knowledgeable, and expert category respectively.

**Table 4. Categorization of subjects for UML and Design tasks. UML and design tasks are shown by a U and D respectively.**

		Design Categories				Total
		DA	DI	DK	DE	
UML Categories	UA	6	3	2	x	11
	UK	x	x	2	5	7
	UE	x	2	5	4	11
	Total	6	5	9	9	29

The eight categories are shown in the table:

- The UADA group represents subjects with little knowledge of UML and software design. There are four subjects (K, L, M, O, Y, S) in this group. They took between 1 and 8 minutes to complete the study.
- The UADI group represents subjects with little knowledge of UML and some basic design knowledge. There are three subjects (V, Q, AA) in this group. They took between 11 and 15 minutes to complete the study.

- The UADK group represents subjects with little knowledge of UML and more knowledgeable in design. There are two subjects (C, A) in this group. They took 9 and 13 minutes to complete the study.
- The UKDK group represents subjects knowledgeable in UML and design. There are two subjects (R, T) in this group. They took 14 and 16 minutes to complete the study.
- The UKDE group was knowledgeable in UML and experts in software design. There are five subjects (F, G, Z, H, J) in this group. They took between 10 and 17 minutes to complete the study.
- The UEDI group was very proficient in UML with basic design knowledge. There are two subjects (I, U) in this group. They took 13 and 20 minutes to complete the study.
- The UEDK group was very proficient in UML and knowledgeable in software design. There are five subjects (B, N, D, P, E) in this group taking between 12 and 21 minutes to complete the study.
- The UEDE group was experts in both UML and design. Four subjects (W, AC, AB, X) fall into this category. They took between 9 and 19 minutes to complete the study.

The UML knowledgeable (UK) category was not present in the eye-tracking experiment due to the small focus group of subjects in that study. The classification of subjects into these groups shows a varying expertise in UML and software design skills. We ran a correlation test to determine if there is a match between self assessed design skills and UML and design scores. The Spearman rank correlation between self assessed design skills and design task scores ( $r_s = 0.39$  p-value=0.01) indicates a significant positive correlation between the two. There is also a positive correlation between programming skills and design task scores ( $r_s = 0.34$  p-value=0.03) and programming skills and UML task scores ( $r_s = 0.32$  p-value=0.04). No correlation was found between UML Scores and Design skills ( $r_s = 0.212$  p-value=0.135). This is not surprising since there are subjects who are experts in design but don't use UML on a regular basis.

## 4.2. Question Classification

We classified the UML and Design questions (tasks) based on the number of correct answers in each. UML questions that were answered correctly in the ranges of [0%, 59%), [59%, 69%), [69%, 76%), [76%, 100%) were classified as challenging, difficult, intermediate and easy respectively. All UML questions were classified as easy in the eye-tracking study. Design questions that were answered correctly in the ranges of [0%, 31%), [31%, 43%), [43%, 55%), [55%, 100%) were classified as challenging, difficult, intermediate and easy respectively. We used interquartile analysis to derive these ranges.

Table 5 shows the question classification in this study. The design questions in boldface font are classified at the same level in both this and the eye-tracking study. With respect to design questions, there is a 100% match in the challenging category, a 50% match in the easy and difficult category and a one question match in the intermediate category. From the mismatched items, four of them were placed into a higher category in this study and three were placed into a lower category.

**Table 5. UML and Design Question Classification**

Level	UML Questions	Design Questions
Easy	8, 9, 12	14, <b>15, 16, 26</b>
Intermediate	2, 4, 5, 6, 11	17, 19, 21, <b>22</b>
Difficult	1, 3, 10	13, 24, <b>25, 27</b>
Challenging	7	<b>18, 20, 23</b>

We use the question classification to generate a weighted score for UML and design tasks. Easy questions were given a lower weight and difficult questions were given a higher weight. The total weight equals 1. In the eye-tracking study, the classified questions were used to compare the effort needed based on the average number of fixations. We are not able to do this analysis due to the lack of eye tracking data. Instead we determine the layout performance based on accuracy shown in the next two sections.

## 4.3. Effect of Layout on UML Tasks

This section analyzes the accuracy of UML tasks for all subjects and determines if the layout had any effect on task accuracy. The first and second hypotheses ( $H_{0u1}$  and  $H_{0u2}$ ) seek to determine the effects of the orthogonal layout vs. the three-cluster layout and the orthogonal layout vs. the multi-cluster layout respectively for UML tasks. We use the paired Wilcoxon non-parametric test to determine the better layout. We conduct a pair wise comparison between the three layouts. The results are shown in Table 6.

**Table 6. Results of the pair wise Wilcoxon test for UML tasks. Significance is shown in bold. alpha = 0.05. For the 1-tailed values, the direction is given by the order of treatment pairs i.e., first < second.**

Treatment Pairs	Z-statistic	p-value (2 tailed)	p-value (1 tailed)	Diff Median
Orthogonal and three-cluster	1.43	0.1533	0.923	1.0
Orthogonal and multi-cluster	-2.90	<b>0.0037 *</b>	<b>0.0019 *</b>	-3.0
Three-cluster and multi-cluster	-3.98	<b>&lt;0.0001 *</b>	<b>&lt;0.0001 *</b>	-3.5

Results indicate a significant difference between the orthogonal layout vs. the multi-cluster with the multi-cluster layout performing better (1-tailed p-value=0.0019). No significant difference was found between



the orthogonal layout and the three-cluster layout. One of the reasons could be due to Question 7 (based on the three cluster layout) which was classified as challenging since it was not answered correctly by most participants. To determine this, we excluded Question 7 and its corresponding questions from the orthogonal and three-cluster layout from the analysis and ran the paired Wilcoxon test again on all the data. This gives a significant difference (1-tailed p-value = 0.0004) between the orthogonal and three-cluster layout with the three-cluster layout outperforming the orthogonal layout.

There is also a significant difference between the three-cluster layout and the multi-cluster layout with the multi-cluster layout outperforming the three-cluster layout (1-tailed p-value < 0.0001). We did not formulate any hypotheses between the three-cluster and multi-cluster layouts so this is a new observation.

#### 4.4. Effect of Layout on Design Tasks

This section analyzes the accuracy of design tasks for all subjects and determines if the layout had any effect on task accuracy. The third and fourth hypothesis ( $H_{0d1}$  and  $H_{0d2}$ ) seek to determine the effects of the orthogonal layout vs. the three-cluster layout and the orthogonal layout vs. the multi-cluster layout respectively for design tasks. Similar to UML task analysis, we use the paired Wilcoxon non-parametric test to determine the better layout. We conduct a pair wise comparison between the three layouts. The results are shown in Table 7.

**Table 7. Results of the pair wise Wilcoxon test for design tasks. Significance is shown in bold. alpha = 0.05. For the 1-tailed values, the direction is given by the order of treatment pairs i.e., first < second.**

Treatment Pairs	Z-statistic	p-value (2 tailed)	p-value (1 tailed)	Diff Median
Orthogonal and three-cluster	-2.58	<b>0.0099 *</b>	<b>0.005 *</b>	-0.20
Orthogonal and multi-cluster	-2.74	<b>0.0062 *</b>	<b>0.0031 *</b>	-0.20
3-cluster and multi-cluster	0.11	0.9138	0.5431	0.00

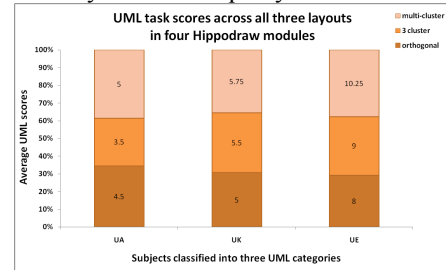
Results indicate a significant difference between the orthogonal layout vs. the three-cluster layout (1-tailed p-value = 0.005) with the three-cluster layout performing better. A significant difference between the orthogonal layout and the multi-cluster layout was also found (1-tailed p-value=0.0031) with the multi-cluster layout performing significantly better than the orthogonal layout. No significant difference was found between the three-cluster layout and the multi-cluster layout. This suggests that for the design tasks, both three-cluster and multi-cluster layouts performed equally well.

#### 4.5. Effect of Ability and Layout

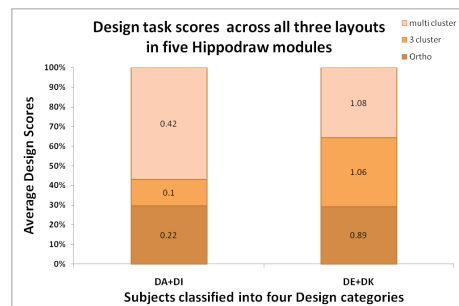
This section addressed our second research question. It investigates if the subject's ability level affects the comprehension of stereotyped or orthogonal class diagram layouts. The ability of a subject is the UML group (UA, UK, UE) and design group (DA, DI, DK, DE) they belong to. Figure 3 and Figure 4 show the average UML and design scores with respect to the three layouts used. We analyze the UML tasks and design tasks separately.

The UML task results indicate that in all the three UML categories, the multi-cluster layout performed better than the orthogonal and the three-cluster layouts. The three-cluster layout was the second best layout in all three UML categories with the exception of the UA category.

For design tasks, we based our comparison on low and high design abilities. The low design ability subjects were from groups DA and DI. The groups DK and DE were combined to form the high design ability group. From Figure 4, we see the performance for low design ability subjects to be higher for multi-cluster layouts. The same trend is seen in the high design ability group. There is not much difference between the multi-cluster layout and three-cluster layout for the high design ability group. This relates to the discussion in the above section where it was statistically shown that three-cluster and multi-cluster layouts were equally effective.



**Figure 3. UML task scores across layouts**



**Figure 4. Design task scores across layouts combined into low (DA+DI) and high (DK+DE) design ability**

We did not conduct a 2-way ANOVA to determine interaction between ability and layout due to low sample

size (n=29) and non-normality of the UML task scores. This is left for a future experiment.

#### 4.6. Confidence for Subject Categories

We collected the confidence level for each question to analyze the way subjects' rate each answer and whether this correlates with the subjects' self-assessment of their design and programming skills (See Section 3.4). The Spearman rank correlation between design skills and the average design task confidence level for all UML questions ( $r_s = 0.46$  p-value = 0.009) indicates a significant positive correlation between the two. However, no correlation was found between design skills and average UML task confidence for all design questions.

The Spearman rank correlation between UML scores and the average UML confidence level for all UML questions ( $r_s = 0.76$  p-value<0.001) indicates a significant positive correlation between the two. The same is shown between design scores and average design task confidence level for all design questions ( $r_s = 0.79$  p-value<0.001). As expected, we found that high levels of ability in UML and design result in higher confidence.

#### 4.7. Debriefing Questionnaire Results

Half of the subjects stated that the time given was not sufficient. These were subjects that fell into the low ability groups. We needed to time the questions to make a fair comparison to the eye-tracking study which did not involve any interruptions or distractions. The data also shows that the questions were considered somewhat difficult to answer. Again this rating was more prevalent in the low ability groups.

Overall, the subjects agreed that the questions were clear, realistic and the information in the class diagrams was understandable. They also considered stereotypes to be helpful while answering questions. Question 7 from the debriefing questionnaire asked the subjects if they concentrated on the spatial layout while answering the questions. Only seven subjects said that they did. However, this number is very subjective since even though they might have concentrated on the spatial layout, they might not have been aware of it. An eye-tracking analysis would have been useful here.

### 5. Discussion

The results of this experiment show a significant improvement in subjects' performance with the multi-cluster layout. In particular, for UML tasks, the multi-cluster layout was the best layout when compared to the orthogonal layout and three-cluster layout. We also compared the three-cluster layout with the multi-cluster layout and found the multi-cluster layout to be significantly better in performance for UML tasks.

For design tasks, the three-cluster layout and the multi-cluster layout outperformed the orthogonal layout. There was no significant difference in performance between the three-cluster and multi-cluster layout for design tasks.

In addition to the above results, we find that the multi-cluster layouts resulted in better performance at each subject ability level (agnostic, inexperienced, knowledgeable, expert). The three-cluster layout was the second best layout with the orthogonal considered to have the worst performance. For design tasks, in particular, there was no difference in performance between the three-cluster layouts and multi-cluster layouts in each subject group. This is consistent with the analysis without considering ability.

The subjects' confidence level for the tasks correlate positively with their design skills. We observed a positive correlation between task scores and design and programming skills. Since the subject's categorization into ability level groups was based on the task scores, we wanted to validate this against the subjects' self-assessment of programming and design skills.

Subjects in higher ability groups (UK, UE, DK, DE) were more confident of their answers compared to lower ability groups (DA, DI, UA) in both UML and design tasks. Finally, our qualitative assessment of the debriefing questionnaire revealed that the subjects found class stereotypes helpful in answering the questions. We did not find any significant effect between time taken to complete the task and the layouts used.

We now compare and contrast our study with the eye-tracking study. The common goal between these two studies was to find the layout that is most effective for comprehension tasks. One major difference is the method of data collection. We did not use eye tracking equipment, instead online questionnaires were used. The prediction, based on our previous work, was that the clustered layouts would result in better performance. This is proved using statistical significance tests in this study. The eye-tracking study did not produce such significance. Our subject classification resulted in the UML knowledgeable (UK) category that didn't exist in the eye-tracking experiment.

Another difference between our study and the eye-tracking study is the subjects' familiarity with Hippodraw and the sample size. In this study, we used a bigger sample (n=29). This is a larger sample compared to our pilot study [25] (20 subjects) which was run as a between-subjects study. The eye tracking study had 9 subjects. Several subjects in the eye-tracking study were familiar with Hippodraw's design or had used it before. In our study we have a more varied sample with most subjects not familiar with Hippodraw. Another difference is that the eye-tracking study uses information about the average number of fixations for each question



to determine the effort required by subjects. It then compares this effort with the difficulty level of each question to determine if the effort is at the same, higher or lower level. They find the most effort was required with the orthogonal layouts. This differs from our study, where we determine the usefulness of stereotyped layouts using the accuracy of answers and statistical significance tests.

## 6. Threats to Validity

Internal validity refers to the presence of other factors besides the main factor that might have an effect on the results. Since this was a within-subjects experiment we had to make sure that there was no learning effect involved when comparing the results of three layouts. We address this by asking a very similar question for each of the three layouts to have a fair and unbiased comparison between them. The questions were presented to each subject in a randomized order to further reduce any learning effect that might occur. Since the experiment was part of the subjects' grade in a course, they were sufficiently motivated to do well.

External validity deals with generalizing our results. We used students as subjects in our study. Many of the subjects had worked with UML in academia and industry. The high ability group of students had real world experience in designing and maintaining software systems. We can liken this group of subjects to mid-level to senior level developers. The subject system we used is a real life system not a toy application. Subjects tend to agree that tasks were realistic and typical of ones they would ask themselves during maintenance.

To ensure conclusion validity, we use the non-parametric paired Wilcoxon statistical test to determine significance of stereotyped layouts vs. the orthogonal layout due to the small sample size.

## 7. Related Work

The related work broadly falls into two categories: the proposal of new layout techniques for class diagrams and the empirical validation of class diagrams for comprehension. Eiglsperger et al. [8, 9] present a topology-shape-metrics automatic layout method for class diagrams based on graph aesthetic criteria. Eichelberger et al. [6, 7] investigated the effect of object oriented design, cognitive psychology and human computer interactions on UML aesthetics criteria for class diagrams. They suggest incorporating annotated complexity stereotypes, spatial distribution, scaling based on complexity and coloring into the set of aesthetic criteria for layout of class diagrams. No quantitative evidence has been shown for any of the proposed criteria. Gutwenger et al. [13] also propose an algorithm for layout of UML class diagrams that balances the

following aesthetic criteria: minimize crossings, minimize bends, uniform direction of arcs, no hierarchy nesting, more orthogonal, merging multiple inheritance edges and labeling edges. Sun et al. [20] propose key graph layout criteria for class diagrams based on the laws of perceptual theories. They list a set of fourteen criteria for class diagrams. No priorities for layout criteria are given. von Gudenberg et al. [23] propose an evolutionary algorithm for class diagrams. Inheritance and associations are evolved to determine their position. The disadvantage is that it is very slow.

Purchase et al. [16, 17] identified that the most important aesthetic preferences for class diagrams were minimizing crossings, minimizing bends, horizontal labels, joined inheritance arcs and more orthogonal layout. Kuzniarz et al. [15] [19] investigated the role and effect of domain stereotypes in the comprehension of class and collaboration diagrams for the telecommunication domain. The results of this study statistically prove that the use of stereotypes helped in system comprehension. Ricca et al. [18] conducted a series of experiments to determine the usefulness of Conallen's stereotyped class diagrams vs. standard class diagrams. Conallen's stereotypes did not help graduate students but did significantly help undergraduates with little experience in design. None of these studies dealt with the layout of class diagrams.

In our pilot study [2], it is shown that layouts based on architectural importance i.e., control, boundary and entity stereotypes, help the comprehension of class diagrams than those based on pure aesthetics. Yusuf et al. [25] conducted the eye-tracking study we compare this study to. Their goal was to assess the comprehension of UML class diagrams using an eye tracker. The use of layout, color, and class stereotypes were all assessed to determine their effectiveness in program comprehension. The results indicate variation of eye movements between experts and novices in both UML expertise and software design ability. We are aware of only one other eye-tracking study by Guéhéneuc et al. [12] that studies how software engineers obtain design information from class diagrams during program comprehension.

## 8. Conclusions and Future Work

This research presents the results of an empirical study to measure the effect of stereotyped class diagram layouts on two types of comprehension tasks. The first set of tasks dealt with UML syntax and the second set of tasks dealt with elements of design. Results show a significant improvement in performance accuracy when multi-cluster layouts were used, for both UML tasks and design tasks. The second best layout was the three-cluster layout for UML tasks with the orthogonal layout having the worst score in both UML and design tasks.

The three-cluster and multi-cluster layouts performed equally well for the design tasks. We do not claim to generalize our results to different types of tasks. This would require further empirical analysis.

These results repeat and add to the findings of the eye-tracking study and our own pilot study which suggests that stereotyped layouts have a positive effect on the comprehension of class diagrams. This experiment further validates the results of an eye-tracking experiment on the same set of class diagram layouts and tasks. This shows that eye-tracking and online questionnaires are complementary techniques of obtaining comprehension performance. In future work, we intend to investigate the comprehension aspect of sub-categories of design tasks with respect to different stereotyped layouts via both eye-tracking methods and online questionnaires.

## 9. References

- [1] Ambler, S. W., *The Elements of UML Style*, New York, NY, Cambridge University Press, 2002.
- [2] Andriyevska, O., Dragan, N., Simoes, B., and Maletic, J. I., "Evaluating UML Class Diagram Layout based on Architectural Importance", in *Proceedings of 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT'05)*, Budapest, Hungary, September 25th 2005, pp. 14-19.
- [3] Arisholm, E., Briand, L. C., Hove, S. E., and Labiche, Y., "The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation", *IEEE Trans. on Software Engineering (TSE)*, vol. 32, no. 6, 2006, pp. 365-381.
- [4] Booch, G., Jacobson, I., and Rumbaugh, J., *The Unified Software Development Process*, Addison-Wesley, 1999.
- [5] Briand, L. C., Labiche, Y., Penta, M. D., and Yan-Bondoc, H., "An Experimental Investigation of Formality in UML-Based Development", *IEEE Transactions on Software Engineering (TSE)*, vol. 31, no. 10, 2005, pp. 833.
- [6] Eichelberger, H., "Aesthetics of Class Diagrams", in *Proceedings of 1st International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT)*, Washington, DC, USA, 2002, pp. 23 - 31.
- [7] Eichelberger, H., "Nice Class Diagrams Admit Good Design?" in *Proceedings of SOFTVIS 2003*, 2003, pp. 159-167.
- [8] Eiglsperger, M., Gutwenger, C., Kaufmann, M., Kupke, J., Jünger, M., Leipert, S., Klein, K., Mutzel, P., and Siebenhaller, M., "Automatic Layout of UML Class Diagrams in Orthogonal Style", in *Proceedings of Info Visualization*, 2004, pp. 189-208.
- [9] Eiglsperger, M., Kaufmann, M., and Siebenhaller, M., "A Topology-Shape-Metrics Approach for the Automatic Layout of UML Class Diagram", in *Proceedings of SoftVis*, San Diego, CA, USA, 2003, pp. 189-198.
- [10] Gogolla, M. and Henderson-Sellers, B., "Analysis of UML Stereotypes within the UML Metamodel", in *Proceedings of UML*, 2002, pp. 84-99.
- [11] Guéhéneuc, Y.-G., "A Reverse Engineering Tool for Precise Class Diagrams", in *Proceedings of Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*, Canada, Oct 4-7 2004, pp. 28-41.
- [12] Guéhéneuc, Y.-G., "TAUPE: towards understanding program comprehension", in *Proceedings of conference of the Center for Advanced Studies on Collaborative research (CASCON)*, Canada, 2006.
- [13] Gutwenger, C., Jünger, M., Klein, K., Kupke, J., Leipert, S., and Mutzel, P., "A New Approach for Visualizing UML Class Diagrams", in *Proceedings of 2003 ACM symposium on Software visualization (SoftVis)*, 2003, pp. 179-188.
- [14] Kunz, P. F., "HippoDraw Users Guide", <http://www.slac.stanford.edu/grp/ek/hippodraw/>, 2006.
- [15] Kuzniarz, L., Staron, M., and Wohlin, C., "An Empirical Study on Using Stereotypes to Improve Understanding of UML Models", in *Proceedings of 12th International Workshop on Program Comprehension (IWPC) 2004*, pp. 14-23.
- [16] Purchase, C. H., Allder, J.-A., and Carrington, D., "Graph Layout Aesthetics in UML Diagrams: User Preferences", *Journal of Graph Algorithms and Applications*, vol. 6, no. 3, 2002, pp. 255-279.
- [17] Purchase, C. H., McGill, M., Colpoys, L., and Carrington, D., "Graph Drawing Aesthetics and the Comprehension of UML Class Diagrams: An Empirical Study", in *Proceedings of Australian Symposium on Information Visualisation*, Sydney, Australia, December 2001, pp. 129-137.
- [18] Ricca, F., Di Penta, M., Torchiano, M., Tonella, P., and Ceccato, M., "An empirical study on the usefulness of Conallen's stereotypes in Web application comprehension", in *Proceedings of 8th IEEE International Symposium on Web Site Evolution (WSE)*, Philadelphia, PA, September 23-24 2006.
- [19] Staron, M., Kuzniarz, L., and Wohlin, C., "Empirical assessment of using stereotypes to improve comprehension of UML models: a set of experiments", *Journal of Systems and Software*, vol. 79, 2006, pp. 727-742.
- [20] Sun, D. and Wong, K., "On Evaluating the Layout of UML Class Diagrams for Program Comprehension", in *Proceedings of 13th IEEE International Workshop on Program Comprehension*, St. Louis, Missouri, USA, 2005, pp. 317-328.
- [21] Sutton, A. and Maletic, J. I., "Recovering UML Class Models from C++: A Detailed Explanation", *Information and Software Technology*, vol. 49, no. 3, Jan 2007, pp. 212-229.
- [22] Tryggeseth, E., "Report from an Experiment: Impact of Documentation on Maintenance", *Empirical Software Engineering* vol. 2, no. 2, 1997, pp. 201-207.
- [23] von Gudenberg, J. W., Niederle, A., Ebner, M., and Eichelberger, H., "Evolutionary Layout of UML Class Diagrams", in *Proceedings of SoftViz*, Brighton, UK, Sept 4 - 5 2006, pp. 163-164.
- [24] Wirfs-Brock, R., "Adding to your Conceptual Toolkit: What's Important about Responsibility-Driven Design", *Report on Object Analysis and Design (ROAD)*, vol. 1, no. 2, July-Aug 1994, pp. 27-34.
- [25] Yusuf, S., Kagdi, H., and Maletic, J. I., "Assessing the Comprehension of UML Class Diagrams via Eye Tracking", in *Proceedings of 15th IEEE International Conference on Program Comprehension (ICPC 2007)*, Banff Canada, June 26-29 2007, pp. 113-122.