Assessing the Comprehension of UML Class Diagrams via Eye Tracking

Shehnaaz Yusuf, Huzefa Kagdi, and Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent Ohio 44242
{sdawoodi, hkagdi, jmaletic}@cs.kent.edu

Abstract

Eye-tracking equipment is used to assess how well a subject comprehends UML class diagrams. The results of a study are presented in which eye movements are captured in a non-obtrusive manner as users performed various comprehension tasks on UML class diagrams. The goal of the study is to identify specific characteristics of UML class diagrams, such as layout, color, and stereotype usage that are most effective for supporting a given task. Results indicate subjects have a variation in the eye movements (i.e., how the subjects navigate the diagram) depending on their UML expertise and software-design ability to solve the given task. Layouts with additional semantic information about the design were found to be most effective and the use of class stereotypes seems to play a substantial role in comprehension of these diagrams.

1. Introduction

The use of eye tracking to complement traditional usability assessments (e.g., surveys and questionnaires) is gaining popularity in a variety of domains [5, 8]. This can be attributed to a number of recent advancements in eye-tracking technology. High quality, extremely accurate, and user-friendly equipment are available today. These systems are relatively affordable and easy to use but their most noteworthy capability is the ability to collect a subject's eye gazes in a non-obtrusive manner. This accurate data can then be used for comprehending the cognitive process involved in the processing of visual data [3, 5, 16].

Pictorial representations such as the UML Class diagrams [6] are commonly used to model the design and structure of a software system. Representations of UML class diagrams are a general research topic with regards to software comprehension and maintenance activities. Investigations in the software visualization and program comprehension communities have primarily focused on effective layout schemes [1, 11, 25] and key aesthetics criteria [9, 10, 14] with the goal of enhancing the cognitive process. A number of usability studies have been reported that evaluate UML class diagrams, including those with additional semantic information (e.g., class stereotypes), for an effective representation in addressing various software evolution tasks [1, 2, 19, 24]. These studies typically form conjectures and/or draw conclusions from the data explicitly collected from subjects' via a combination of questionnaires, experience reports, and feedback comments after a designated task is completed. This raises a potential threat to the validity of the study, namely the match/disparity between the subjects' responses on completion of a task and the "reality" they observed while performing that task. For example, a subject may forget to report (or misreport) an observation after a lengthy task.

Here we take a different approach to assess UML class diagrams. We use eye-tracking equipment to implicitly collect a subject's activity data in a non-obtrusive way as they are interacting with the diagram in performing a given task. The equipment collects three forms of pertinent data including the eye-gazes with respect to the visual presentation and an audio/video recording of the subject during the session.

Here, we present our experience and results of a study that we conducted regarding some of the issues of how people see and understand UML class diagrams. We want to better understand how people explore, examine, and navigate class diagrams. With this understanding we can develop better layout mechanisms and other methods for presenting software design information. In support of this effort, we try to answer the following questions:

- Which UML class diagram layout is most effective for software comprehension and design tasks?
- Does the use of class stereotype information provide additional assistance?
- Is the use of colors to map semantic information on classes (entity, boundary, control) useful?
- What do people really look at in class diagrams?
- Is there a big difference between expert and novice?
- What items in the diagrams do people fixate on the most?
- How do people navigate through the diagrams?

The paper is organized as follows. The next section presents background on eye tracking. Section 3 describes our study on assessing how people comprehend UML class diagrams. Our findings and analysis of the study are presented in Section 4. Related work is presented in Section 5 followed by conclusions.

2. Eye Tracking

The fundamental design of eye-tracking equipment is based on the physiology of the human visual capability [8, 17]. These systems use cameras to track eye movement. Specifically, we used a *Tobii 1750* eye-tracker (www.tobii.se) to capture eye movements and collect eye gaze data. In this equipment, the two cameras used to track the eye are built into a 17 inch flat-panel screen. Therefore, no restraints such as wearing a headband or goggles are placed on the human subject. This was not the case in older eye tracking equipment. This provides a normal computer-operating environment during the study. Moreover, the *Tobii 1750* eye-tracker is very accurate with an error rate of less than 0.5 degrees and a sampling rate of 50MHZ. Software that records the XY screen coordinates of eye gazes and supports analysis of eye movements is also provided along with the eye-tracker system. An audio/video recording is also made of each study session.

The underlying basis is to capture various types of eye movements that occur while humans physically gaze at an object of interest. Among these, fixation and saccade are the two most widely used eye movements in these types of studies.

Definition: Fixation is the stabilization of eyes on an object of interest for a period of time.

Definition: Saccades are quick movements of the eyes that move interest from one location to the next (i.e., refixates).

Definition: Scanpath is a directed path formed by saccades between fixations.

The general consensus in the eye tracking research community is that the processing of visualized information occurs during fixations, whereas, no such processing occurs during saccades [17]. Humans use saccades to locate interesting parts in a visual scene to form a mental model.

Figure 1 shows the recording of eye positions superimposed on a UML class diagram. The numbered circles represent fixation and lines between them represent saccades. The size of a fixation (i.e., area of a circle) is proportional to its time duration. The numbering of circles represents the ordering of fixations. For example, in Figure 1, the fixation labeled with the number 35 on the class *NTuple* happened before the fixation labeled 36 on the class *NTupleController*. That is, the class *NTuple* was looked at before the class *NTupleController*. The scanpath in this case is directed to the left and downwards. A big circle on the class *PyNTuple* shows that a large amount of time was spent on this class. The eye-tracker captures fixation and saccades in the form of XY coordinates of the visual screen (in this case a UML class diagram) so that we can determine what was being looked at in a visual presentation.

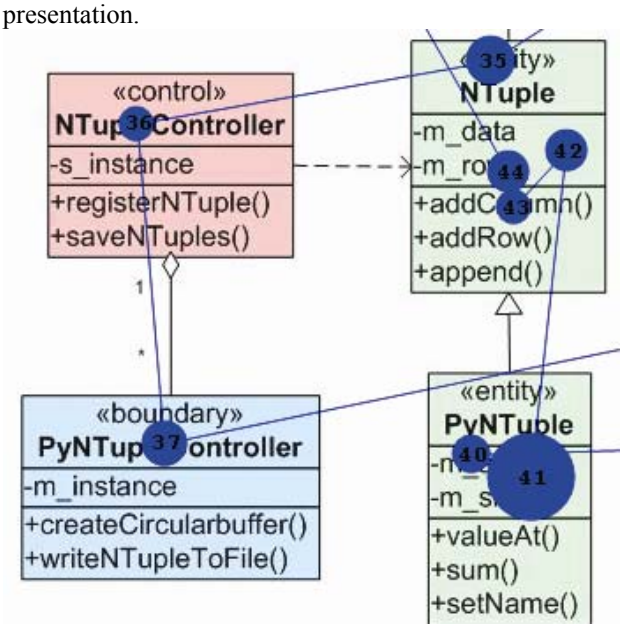

Figure 1. ScanPath of a user on the UML Class Diagram. Fixations are represented by the circles and saccades by the lines connecting the circles.

3. Assessment Study

The principal goal is to obtain an understanding of how human subjects use different types of information in UML class diagrams[1] in performing their tasks. In a nutshell, human subjects were given specific tasks to perform on diagrams. An eye-tracker was used to capture their activities in terms of fixation, saccades, audio, and video. The following is a more detailed description of the various components of our study.

3.1. UML Class Diagram Layout

We used UML class diagrams representing the design of the open source *HippoDraw* software (www.slac.stanford.edu/grp/ek/hippodraw). *HippoDraw* is a statistical data analysis application/framework that is primarily written in C++ and uses the *Qt* library for GUI.

We used three different layout techniques of UML class diagrams for our investigation. Our selection of these layout methods are based on previous work in

[1] The use of term diagram(s) means UML class diagram(s) unless specified otherwise.

assessing layouts [1]. These diagrams vary in layouts, semantic information (e.g., stereotype), and secondary notations (e.g., color).

Definition: The *orthogonal* layout focuses on the minimization of the edge crossings and bending. Multiples of 90 degree angles are used to position intersecting edges [10, 11, 22]. This layout is adopted from general graph drawing algorithms and is typically available in UML modeling and drawing tools.

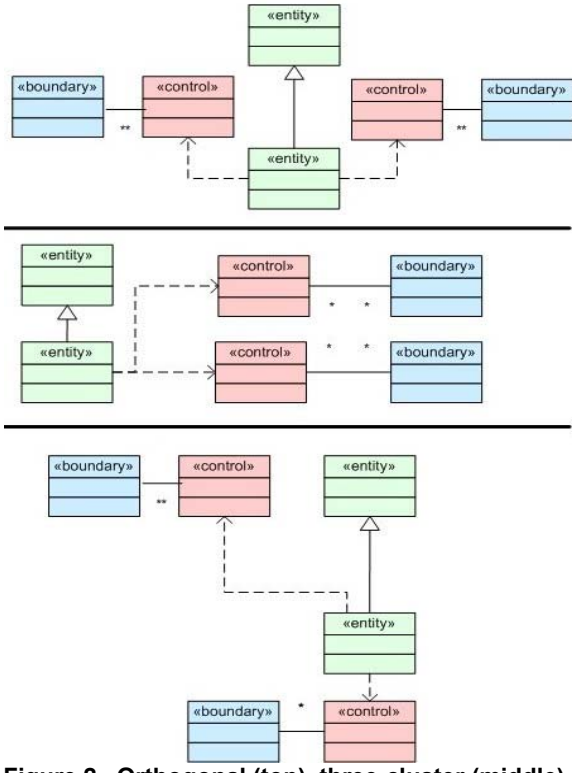

Figure 2. Orthogonal (top), three-cluster (middle), and multiple-cluster (bottom) layouts of a UML class model.

Definition: The *three-cluster* layout positions classes into three clusters (i.e., boundary, control, and entity) based on their design or architectural roles. Classes that are stereotyped entities, in terms of UML vocabulary, are placed in a single cluster. Similarly, classes that are stereotyped boundary and control form the other two clusters. This is an example of layouts that use the general role of a class in the high-level design modeling and analysis of a software system via UML.

Definition: The *multiple-cluster* layout is a further specialization of the three-cluster layout. Related classes that are responsible for a specific functionality of a software system are positioned in a single cluster. This is an example of layouts that further use the responsibilities of classes in modeling, analysis, and realization of an application domain specific concept. For example, a cluster could map to a functional requirement of a system. Therefore, the number of clusters in a layout could be equivalent to the number of functional requirements.

Figure 2 shows examples of orthogonal (top), three-cluster (middle), and multiple-cluster (bottom) layouts for the same UML class model. Colors and textual annotations (i.e., <<*entity*>>, <<*control*>>, and <<*boundary*>>) are used to represent class stereotypes. Boundary, entity, and control classes are represented by three different colors (blue, green, and red colors in our study). The orthogonal layout does not use the semantic information such as stereotype of a class in positioning it on a diagram, whereas the other two do. However, we also made the stereotype information in the orthogonal layout with textual annotation so that all the diagrams exhibit the same design information.

3.2. Tasks

The tasks given to the subjects in our study consist of the subjects answering specific questions by viewing UML class diagrams. We designed two types of questions, one set dealing with basics of UML class diagram and the other set related to the software design. The set of diagram questions deal with the characteristics of the classes, attributes, methods, relationships, and general notations. For example, what is the type of relationship between two given classes? This set of questions is aimed at understanding the user activities in performing general exploration, explanatory, and navigation tasks in a UML class diagram.

The set of software-design related questions are concerned with general software design understanding, extensibility, and changeability. For example, name the class that could be extended to accommodate a new GUI functionality. These questions are aimed at providing insight as to how software developers approach, process, and accomplish design tasks by utilizing UML class diagrams. The questions in this set were planned in such a way that they needed minimal knowledge of the finer design, implementation, and domain minutia of *HippoDraw*, and knowledge of fundamental software design principles to address them.

Table 1 shows the set of 12 UML questions and Table 2 shows the set of 15 software design questions used in our study. Table 3 shows the distribution of questions that are asked for the six modules of *HippoDraw* using the three different types of layouts. Only UML questions are allocated to the module *High-Level* and only software design questions are allocated to the modules *XmlNode* and *Canvas*. The remaining three modules *Python Wrappers*, *PlotterBase*, and *Tuple* are allocated questions from both sets. Notice that the same question is not asked for two different layouts. This was done to avoid any learning bias that may occur due to the same

question asked twice. However, very similar questions were asked to give a fair coverage to all the layouts. We felt that this distribution allows us to analyze common and exclusive behavior of the three layouts in supporting two different types of tasks.

Table 1. UML questions used in the study

No.	Questions
1	Identify the kind of relationship between class *ViewBase* and class *PlotterBase*.
2	Name the classes involved in aggregation.
3	Name the derived classes of the class *PlotterBase*.
4	Name the class with the method name *getAverage*.
5	Identify the kind of relationship between class *NTuple* and class *DataSource*.
6	Name all the classes involved in dependency.
7	Count the number of derived classes of the class *Observer*.
8	Name the class with the method name *objectiveValue*.
9	Identify the kind of relationship between class *DataSource* and class *Observable*.
10	Name all the classes involved in generalization.
11	Count all the classes involved in aggregation.
12	Name the class with the method name *registerNtuple*.

Table 2. Software design questions used in the study

No.	Questions
13	Name the class that a python wrapper uses to access data in the class *NTuple*.
14	Name the class responsible for managing XML serialization.
15	Name the class that controls the active window of an application.
16	Name the base class for axis representation hierarchy.
17	Name the class through which a boundary class could access data in the class *NTuple*.
18	Name the class that is a python wrapper for a class with the method name *adduct.*
19	Name the classes that are specialized for XML processing in QT.
20	Name the class that responds to the toolbar events from windows and messages sent by the class *Inspector*.
21	Name the class that plots point in 2D.
22	Name the class through which a boundary class could access data in the class *DataSource*.
23	Name the entity class that is responsible for storing data.
24	Name the entity class that could be extended to specify a new property (besides *Font* and *Color*) in XML
25	Name the concrete class that displays data in a tabular format.
26	Name the class that sets the range and scale of the axis.
27	Name the class that gets data from the class *DataSource* objects and uses functions from the class *FunctionBase*.

Table 4 shows the number of classes in a UML class diagram that are used from the corresponding modules of *HippoDraw*. Overall 100 unique classes are used from the *Hippodraw* system. We selected six class models that represent six logical subsystems or a set of related functionalities. We manually engineered three class diagrams with orthogonal, three-cluster, and multiple-cluster layouts to represent each model. Each of the resultant 18 diagrams occupies approximately the same amount of physical screen space and consists of between 12 and 21 classes. The bound on the number of classes in a diagram is guided from Purchase's [22] results on the optimal number of the classes beyond which there is a substantial cognitive overhead for comprehension tasks. Also, Sun et al. [25] showed that a diagram with very dense information leads to difficulty in its readability. Therefore, our diagram shows only selective methods and attributes that are considered most relevant to the designated tasks. Further, we considered various advocated aesthetics criteria such as fewer edge bends and crosses, shorter edge lengths, and maximization of symmetry in the literature [9, 10, 22].

Table 3. Distribution of questions for the three UML class diagram layouts and their corresponding modules in HippoDraw software.

Modules	Orthogonal	Three-Cluster	Multiple-Cluster
High-Level	1	5	9
Python Wrappers	2, 13	6, 18	10, 23
PlotterBase	3, 16	7, 21	11, 26
Tuple	4, 17	8, 22	12, 27
XmlNode	14	19	24
Canvas	15	20	25

Table 4. Number of classes used from the design of corresponding HippoDraw software.

Modules	Number of Classes
High-Level	14
Python Wrappers	15
PlotterBase	21
Tuple	19
XmlNode	12
Canvas	19

3.3. Stimuli

Using the eye tracking terminology, an object that is viewed by a subject is known as the *stimulus*. We combine a question and the corresponding diagram into a single stimulus. The question is placed at the top-left corner and the diagram occupied the remaining space. Research on the use of eye tracking for a variety of domains show a human bias for the top-left corner [5, 12] and/or reading from the left to right [4, 18]. Therefore,

our chosen arrangement of the question and the diagram should help eliminate or drastically reduce this bias. Figure 3 shows an example of a portion of a stimulus meeting our criteria. In our study, a total of 27 stimuli are formed from the combinations shown in Table 3.

3.4. The Subjects

Volunteers who had completed undergraduate and/or graduate level of software engineering coursework and used UML class diagrams for academic and/or industry projects were used as subjects. We secured nine such subjects: three faculty, four doctoral students, one master student, and one undergraduate student. These subjects were all from computer science but had varying degrees of software design and programming experience.

Additionally, we had three non Computer Science graduate students who had no knowledge of UML and very little or no software development experience. We incorporated these subjects in the study to compare results from these two groups and see if there is any inherent difference in their eye movements.

3.5. Running the Study

The study consisted of subjects viewing the stimuli and verbally responding to the stated questions. The entire study was conducted over a two-day period. The subjects were informed well in advance of the schedule of their sessions. On the day of the study, subjects were given a single page UML notation guide along with introductory information of *HippoDraw*. Also the subjects were briefed on the eye-tracking equipment as to how it works and what information would be recorded. They were informed that the eye tracking system automatically records their audio, video, and eye movements on the class diagram.

All the subjects were given the 27 stimuli (comprehension tasks). Only one subject at a time performed the study and it took between 10 and 20 minutes to complete. The subject was stationed comfortably in front of the eye tracker at a distance of approximately 60 cm and the eye-tracker was calibrated for their individual use to verify that the system was working properly. This process takes less than a couple minutes to complete. After this, the environment in front of them was just a common desktop Windows operating environment.

The subjects were then instructed to read the question on a stimulus loudly and verbally answer it so that they could be recorded. There was no time limit on individual stimulus or the entire session. After concluding the task on a stimulus, the subjects were asked to say "next" so that the auditor could make them transit to the next stimulus. The set of 12 UML questions stimuli was presented before the set of 15 software design questions stimuli. The auditor verbally warned the subjects of the transition from one set to the other. The same diagram was not presented in consecutive stimuli in order to avoid immediate learning bias occurring due to a mental picture in the short-term memory. The subjects were encouraged to verbally provide their observations, comments, and feedback during and after the study.

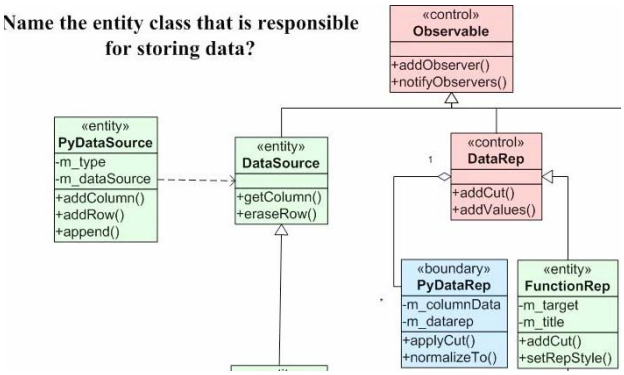

Figure 3. A portion of stimulus used in the study with a question in the top-left corner and the UML class diagram occupying the rest of the visual space.

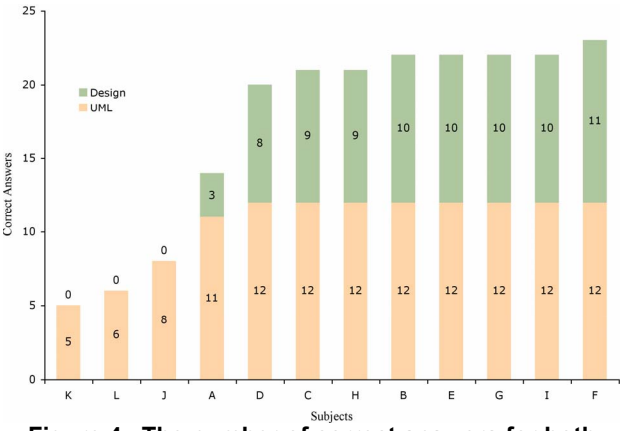

Figure 4. The number of correct answers for both UML and Design sets of questions.

4. Analysis and Results

We analyzed the data collected from our study to obtain an understanding of subjects' visual activities in answering questions with the three layouts.

4.1. Subject and Question Classification

We analyzed the accuracy and response time of the answers to the 27 stimuli using the audio and video recordings of the experiments. Figure 4 shows the number of correctly answered questions. The remainder of the questions were either incorrectly answered or skipped. Eight of the nine computer science subjects

answered all the 12 UML questions correctly. No one answered all of the 15 design questions correctly.

The subjects with no UML knowledge prior to the study were able to answer a number of UML questions after reading the one-page description of the notation. Based on the performance of subjects in answering the questions, we classified them into the following groups:

- Both UML and design agnostic (UADA): Subjects that demonstrated very little knowledge of UML and software design. Three subjects (K, L, and J) are found in this category. These subjects took between 14 and 16.5 minutes each to complete the study.
- UML expert but inexperienced designer (UEDI): Subjects that seem very skillful in UML but seem to exhibit a lack of software design experience. Only one subject (A) is found in this category. This subject took approximately 13.9 minutes to complete the entire study.
- UML expert and knowledgeable designer (UEDK): Subjects that seem to be expert in UML and knowledgeable in software design. Three subjects (D, C, and H) are found in this category. These subjects took between 8.5 and 14 minutes to complete the entire study.
- Both UML and design expert (UEDE): Subjects that exhibited commendable knowledge on both UML and software design. Five subjects (B, E, G, I, and F) are found in this category. These subjects took between 6.5 and 11.5 minutes to complete the study.

It should also be noted that subjects had very different reading speeds. Some were very fast readers while others read slowly and carefully. This is one of the main reasons why we cannot compare performance based purely on the time to complete a particular task.

The classification of subjects shows that we have representatives with varying UML and software design skills. Also, our questions were effective enough to enable this classification and this information is used in further analysis presented in the following sections. We now classify the tasks based on the performance of subjects to gauge the difficulty level in answering the questions. Figure 4 shows that most subjects with the exception of the UADA group answered all the UML questions. Therefore, we believe that the UML questions were quite easy to handle and are not classified further.

We classified the 15 design questions based on the distribution of subjects answering them correctly and excluded the UADA group from this analysis. Questions that were answered correctly by subjects in the ranges [0%, 25%), [25%, 70%), [70%, 80%), and [80%, 100%] were classified as *easy*, *intermediate*, *difficult*, and *challenging* respectively. Table 5 shows the specific design questions in the respective categories. No subject answered the question numbered 20 correctly. Other questions were correctly answered by at least one subject.

Table 5. Classification of design questions based on the percentages of subjects correctly answering them. The question numbers correspond to the questions in Table 2

Level	Questions
Easy	15, 16, 19, 21
Intermediate	13, 14, 22, 24
Difficult	17, 25, 26, 27
Challenging	18, 20, 23

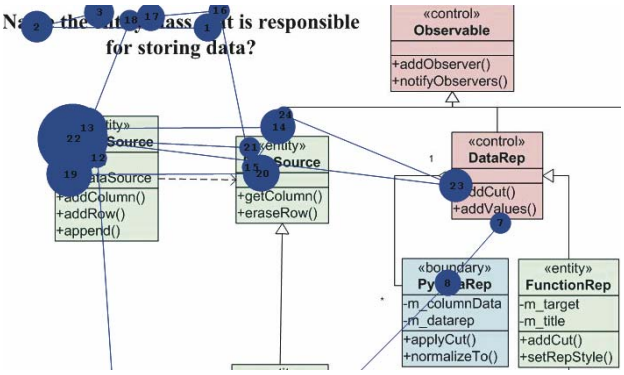

Figure 5. A gaze plot for a portion of the stimulus shown in Figure 3

4.2. Exploration, Examination, and Navigation

Here, we focus on trying to understand how subjects use their eye movements for:

- Exploration of visual space: How they perform searches on the UML class diagram to locate objects required for a given task.
- Examination of visual objects: How they visualize, in detail, whole or parts of classes and relationships while accomplishing a given task.
- Navigation: How they move from one object of interest to the next after their discovery.

Gaze plots, such as shown in Figure 5, that provide fixations, saccades, and scanpaths are used in this analysis. We found the following:

- The eye-tracker captured the fixations at the granularities of class, attribute, and method textual names in the diagram. Most subjects directly explored only the part of the diagram that contained the names specified in the questions. For example, when a class containing a specific method name X was required, subjects only searched the parts of the class containing methods.
- A wide majority of the fixations are found on classes and relationships, and very few on the empty spaces.

- The first fixations were found only on the end of relationship symbols (e.g., diamond edge for aggregation) for questions regarding or involving relationships. Therefore, subjects start examining from the relationship-ends for answering specific questions about them. Only saccades were found on the rest of a relationship symbol (i.e., the lines). So the line parts of relationship notations are used only for navigation purposes.
- All subjects in the UEDE and UEDK groups start exploring the diagrams from the center and moved towards the periphery.
- Subjects in the UADA and UEDI groups explore the diagrams from top-to-bottom, and left-to-right.

Figure 6. A heatmap showing the cumulative fixations of subjects on a specific stimulus. The colors red, orange, yellow and green indicate the decrease in number of fixations from highest to lowest. Best viewed in color.

4.3. Stereotype Usage

Here, we discuss the use of explicit stereotype information that was provided in the form of textual annotations and color in the diagrams. Gaze plots and video recording were used to facilitate the analysis. We found the following:

- All subjects in the UEDE group and majority of the subjects in the UEDK group visually examined the textual annotations used for stereotypes in answering the majority of the design questions. This was evident by the number and size of the fixations on text.
- All subjects in the UEDE group and majority of the subjects in the UEDK group used the distinct class colors indicating their stereotypes to facilitate exploration and navigation through the diagrams.
- None of the subjects in the UADA and UEDI groups used the stereotype textual annotations and colors. Since they did not use this information they explored and examined almost all classes in the diagram.
- Subjects in the UEDE group divided the visual space of the UML diagram into clusters based on the stereotype color information. They used clusters as units of navigation (and not classes). They narrowed down their search to the cluster potentially containing the answer and examined that cluster in detail.
- Subjects that used the above strategy answered more questions correctly and quickly than others.
- When answering questions that involved both stereotypes and relationships, majority of the subjects in the UEDE group used stereotype to narrow down to the possible solution, and then located the appropriate relationship to complete the answer.

Also, we analyzed all the heatmaps consisting of cumulative fixations of all the subjects for a particular stimulus (i.e., task) and found support for all of the above findings. For example, the heatmap in Figure 6 for the question 23 shows a large number of fixations on the textual annotations of stereotypes.

4.4. Efficient Layouts

There is a wide variety of eye tracking metrics in the literature [15]. The most frequently used metric is the number of fixations. A large number of fixations is an indicator of poor arrangements of objects in a stimulus. The determination of the efficiency of a layout is based on the total number of fixations on a stimulus. In our study, each stimulus corresponds to a diagram with one of the three layouts. Fewer total number of fixations on a stimulus means that the subject needs less effort to answer the associated question. We conjecture that if the total number of fixations is high then the classes and relationships are laid out in a way that leads to inefficient visual exploration, explanation, and navigation. Such poor arrangement spans the attention of the subject across a number of objects instead of systematically narrowing down the visual space to only the relevant area of interest. Similar measures are used to assess the arrangement of objects in a visual environment in other domains that use eye-tracking methodology for assessment [12, 16, 18, 21, 27].

The average number of fixations for a specific task is computed from the fixations of all the subjects (excluding the group UADA) on the associated stimulus. The column Average Fixations in Table 6 shows the average fixations for all the UML and design questions used in our study. In order to determine the relative effort required in answering the questions, four categories *low*, *intermediate*, *high*, and *extreme* are formed from the analysis of the average number of

fixations. The median of all the average number of fixations of the stimuli is 34.33. The stimuli with average number of fixations in the range [0, 34), [34, 42), [42, 50), [50, 67) are classified as low, intermediate, high, and extreme respectively. The classification of the questions based on the accuracy of answers from Table 5 is also shown in Table 6 to facilitate comparison of difficulty level and the required effort.

Table 6. Classification of effort required to answer questions based on the average fixations taken over the number of expert subjects for each stimulus. The table is ordered by effort/average fixation. The column Levels is taken from Table 5.

Stimuli (Questions)	Average Fixations	Effort	Levels
5	23.00	Low	Easy
23	23.56	Low	Challenging
11	24.67	Low	Easy
26	25.22	Low	Difficult
15	27.67	Low	Easy
8	28.00	Low	Easy
12	29.56	Low	Easy
6	29.89	Low	Easy
7	30.22	Low	Easy
18	30.56	Low	Challenging
9	31.22	Low	Easy
3	32.00	Low	Easy
19	32.56	Low	Easy
14	34.33	Medium	Intermediate
1	36.22	Medium	Easy
21	38.00	Medium	Easy
2	40.56	Medium	Easy
4	41.00	Medium	Easy
16	42.56	High	Easy
24	42.56	High	Intermediate
25	42.78	High	Difficult
10	43.56	High	Easy
22	44.22	High	Intermediate
13	62.44	Extreme	Intermediate
20	63.67	Extreme	Challenging
27	65.22	Extreme	Difficult
17	66.33	Extreme	Difficult

In order to compare the three types of layouts we compared the level of question and the effort needed in answering them. The baseline of comparison is that the level and effort should be directly related. That is, easy questions should require low effort, intermediate questions should require medium effort, difficult questions should require high effort, and challenging questions should require extreme effort. We refer to such questions having this property as *equal-effort*. Also, questions that require more effort than the corresponding baseline level are referred as *more-effort*, whereas those that require less effort than the corresponding baseline level are referred as *less-effort*.

Using Table 3 and Table 6 we can map questions and stimuli to the corresponding layouts. Table 7 shows that the multiple-cluster layout supports the highest number of questions at the equal-effort and the orthogonal layout supports the lowest number of questions at the equal-effort. Similar performance is seen in favor of multiple-cluster and three-cluster layouts, and against the orthogonal layout with respect to the more-effort category. Moreover, multiple-cluster and three-cluster layouts show support at less-effort, whereas no such support is found in the orthogonal layout. Clearly, the multiple-cluster layout outperforms the other two layouts, and the orthogonal layout is outperformed by the other two layouts, for both sets of UML and design questions.

Table 7. Distribution of questions based on level and effort. The multiple-cluster layout outperforms the others with respect to effort.

Layout Types	Equal-Effort	More-Effort	Less-Effort
Orthogonal	3	6	0
Three-cluster	4	4	1
Multiple-cluster	5	2	2

4.5. Threats to Validity

We discuss the internal and external validities of our approach with regards to the results obtained from its evaluation.

Internal validity refers to addressing the possible factors in our evaluation that bias the results one-way or the other and as such do not represent reality. All our subjects were from academia and volunteers. This raises the threat that they may not have been motivated enough to perform to their fullest capability and interest. Also, some subjects may have apriori knowledge of the *Hippodraw* system. We believe that this was less of an issue as there are no UML design documents publicly available (with the exception of *Doxygen* documents). The number of subjects (12) in our study may appear to be low, however, this range is typically found in eye-tracking studies [3, 12, 16].

External validity refers to addressing the general applicability of our approach and conclusions to any given dataset. We assessed UML diagrams with subjects from academia, 27 questions, 3 layouts, and one system. We tried to take adequate measures so that our study represents commonly found comprehension and design scenarios, however we do not claim that our results will

generalize to any arbitrary task, layout, system, and subject combination.

5. Related Work

There are two research areas that are related to our work. We discuss representative works in the UML and eye-tracking usability studies.

5.1. UML Class Models

Very recently, Guehéneuc [13] used eye-tracking to study the comprehension of the software engineers on the class diagrams. However, their study was more limited with regard the questions and the scope. Additionally, they used a head mounted system that is quite intrusive and no more accurate than what we used.

Sun et al. [25] proposed key criteria and guidelines for the effective layouts of UML Class diagrams based on the perceptual theories. Kurniaz et al. [19] and Staron et al. [24] evaluated the influential role of stereotypes in understanding UML class and collaboration diagrams. Andriyevska et al. [1] found that the layouts based on design and architectural information assists more in comprehension of UML class diagram than those solely based on graph drawing aesthetics. Eichelberger [9, 10] proposed a set of aesthetic criteria and semantic clustering of nodes to increase the readability of UML class diagrams. Purchase et al. [22, 23] conducted user studies to evaluate the effect of aesthetics criteria (i.e., minimize bends, edge crossing, orthogonal) on the UML diagrams. Tilley et al. [26] investigated the use of UML syntax, semantics, spatial layout, and domain knowledge in system evolution tasks. Eiglsperger et al. [11] proposed an automatic layout algorithm for UML class diagrams. Their algorithm is based on the topology and shape metrics that try to minimize the edge crossing, bends, and occupied area. Briand et al. [7] showed that the combined use of OCL and UML offers significant benefits in terms of defect detection, comprehension, and maintenance of UML analysis documents. Arisholm et al. [2] showed that the UML documentation can provide significant improvements in the functional correctness of changes and overall quality of the design for complex tasks.

5.2. Eye Tracking and Usability

Jacob [17] discusses the human factors and technical considerations in using eye tracking in Human Computer Interaction (HCI). Beymer et al. [4] developed the tool *WebGazeAnalyzer* to record and analyze eye gazes on web browsing sessions. Uwano et al. [27] used eye tracking to characterize the individual's performance in reviewing source code. Nakamichi et al. [20] advocates the use of gaze-point velocity to detect the low usability web pages. Khiat et al. [18] studied the relation between subjects understanding and their eye movements on the text in a non-native language. Pan et al. [21] studied factors such as gender information, web page viewing order, and different types of website (news and shopping) by using eye tracking measures. Whalen et al. [28] conducted a study to determine the elements in web browsers that are viewed (and ignored), and how easily they can be noticed. Bednarik et al. [3] applied eye tracking to study comprehension of java programs. Iqbal et al. [16] investigated the mental workload demanded by computer-based tasks perform by users in an eye tracking study. Additionally, our findings corroborate with the prior results [1, 13, 19] obtained by traditional evaluation methods, and further provide reasoning behind those results.

6. Conclusions

This work, along with the work by Guehéneuc [13], are the first studies to use eye-tracking equipment to assess how people comprehend UML class diagrams in the context of software design problems. The advent of new eye-tracking technology makes the use of this equipment easier and unobtrusive. This method of data acquisition is implicit and more objective compared to traditional usability study methods. It also opens the door for the creation of objective assessment metrics of class diagram layout.

Our findings showed that experts tend to use such things as stereotype information, coloring, and layout to facilitate more efficient exploration and navigation of class diagrams. Additionally, experts tend to navigate/explore from the center of the diagram to the edges whereas novices tend to navigate/explore from top-to-bottom and left-to-right.

We made some observations that need further study. Even if subjects could not answer the question correctly, they got very close to the answer by using stereotype and color information. Defining standards for the use of this type of additional information could lead to more readable and effective diagrams. Also, we observed that the close similarity in the notations for generalization and aggregation relationships could cause undue effort to differentiate. Using less similar visual notations may reduce the effort to understand diagrams.

We thank Dr. David Robbins, Dr. Jason Holmes, and Aaron Rosenberg for their assistance in the use of the Tobii eye-tracker.

7. References

[1] 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.

[2] Arisholm, E., Briand, L. C., Hove, S. E., and Labiche, Y., "The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation", IEEE Transactions on Software Engineering (TSE), vol. 32, no. 6, 2006, pp. 365.

[3] Bednarik, R. and Tukiainen, M., "An eye-tracking methodology for characterizing program comprehension processes", in Proceedings of the Sym. on Eye tracking research \& applications (ETRA), San Diego, CA, 2006, pp. 125-132.

[4] Beymer, D. and Russell, D. M., "WebGazeAnalyzer: a system for capturing and analyzing web reading behavior using eye gaze", in Proceedings of CHI '05 extended abstracts on Human factors in computing systems, Portland, OR, USA, 2005, pp. 1913-1916.

[5] Bojko, A., "Eye Tracking in User Experience Testing: How to Make the Most of It. " in Proceedings of 14th Annual Conference of the Usability Professionals Association (UPA), Montréal, Canada, 2005.

[6] Booch, G., Rumbaugh, J., and Jacobson, I., Unified Modeling Language User Guide, NJ, Addison Wesley, 2005.

[7] 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 Enginering (TSE), vol. 31, no. 10, 2005, pp. 833.

[8] Duchowski, A. T., Eye Tracking Methodology: Theory and Practice, London, Springer-Verlag, 2003.

[9] Eichelberger, H., "Aesthetics of class diagrams", in Proceedings of First IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT), Washington, DC, USA, 2002, pp. 23-31.

[10] Eichelberger, H., "Nice Class Diagrams Admit Good Design?" in Proceedings of SOFTVIS'03, 2003, pp. 159-167.

[11] Eiglsperger, M., Kaufmann, M., and Siebenhaller, M., "A topology-shape-metrics approach for the automatic layout of UML class diagrams", in Proceedings of 2003 ACM Sym on Software visualization, San Diego, CA, 2003, pp. 189-199.

[12] Goldberg, J. H., Stimson, M. J., Lewenstein, M., Scott, N., and Wichansky, A. M., "Eye tracking in web search tasks: design implications", in Proceedings of 2002 symposium on Eye tracking research & applications (ETRA), New Orleans, Louisiana, 2002, pp. 51-58.

[13] Guehénéuc, Y. G., "TAUPE: Towards Understanding Program Comprehension", in Proceedings of The Conference of the Center for Advanced Studies on Collaborative Research (CASCON'06), Toronto, Canada, Oct 16-19, 2006, pp. 1-13.

[14] Gutwenger, C., Junger, 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, San Diego, CA, 2003, pp. 179-188.

[15] Hyona, J., Radach, R., and Deubel, H., The Mind's Eye:Cognitive and Applied Aspects of Eye Movement Research, 0-444-51020-6 ed., Amsterdam, 2003.

[16] Iqbal, S. T., Adamczyk, P. D., Zheng, X. S., and Bailey, B. P., "Towards an index of opportunity: understanding changes in mental workload during task execution", in Proceedings of SIGCHI conference on Human factors in computing systems, Portland, Oregon, USA, 2005, pp. 311-320.

[17] Jacob, R. J. K., "What you look at is what you get: eye movement-based interaction techniques", in Proceedings of SIGCHI conference on Human factors in computing systems: Empowering people, Seattle, Washington, 1990, pp. 11-18.

[18] Khiat, A., Matsumoto, Y., and Ogasawara, T., "Task Specific Eye Movements Understanding for a Gaze-Sensitive Dictionary", in Proceedings of Conference on Intelligent User Interfaces, Funchal, Madeira, Portugal, 2004, pp. 265-267.

[19] 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.

[20] Nakamichi, N., Shima, K., Sakai, M., and KenIchi, M., "Detecting low usability web pages using quantitative data of users' behavior", in Proceedings of 28th International Conference on Software Engineering (ICSE'06), Shanghai, China, 2006, pp. 569-576.

[21] Pan, B., Hembrooke, H., Gay, G., Granka, L., Feusner, M., and Newman, J., "Determinants of web page viewing behavior: An eye-tracking study", in Proceedings of Eye tracking research & applications symposium on Eye tracking research & applications, San Antonio, Texas., 2004.

[22] Purchase, H. C., Allder, J.-A., and Carrington, D. A., "Graph Layout Aesthetics in UML Diagrams: User Preferences", J. Graph Algorithms Application, vol. 6, no. 3, 2002, pp. 255-279.

[23] Purchase, H. C., McGill, M., Colpoys, L., and Carrington, D., "Graph drawing aesthetics and the comprehension of uml class diagrams: an empirical study", in Proceedings of Australian Sym on Info. Vis., 2001, pp. 129-137.

[24] Staron, M., Kuzniarz, L., and Thurn, C., "An empirical assessment of using stereotypes to improve reading techniques in software inspections", in Proceedings of Proceedings of the Workshop on Software quality, St. Louis, MO, 2005, pp. 1-7.

[25] 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.

[26] Tilley, S. and Huang, S., "A qualitative assessment of the efficacy of UML diagrams as a form of graphical documentation in aiding program understanding", in Proceedings of 21st Conference on Documentation (SIGDOC), San Francisco, CA, USA, 2003, pp. 184-191.

[27] Uwano, H., Nakamura, M., Monden, A., and Matsumoto, K., "Analyzing individual performance of source code review using reviewers' eye movement", in Proceedings of 2006 symposium on Eye tracking research & applications (ETRA), San Diego, California, 2006, pp. 133-140.

[28] Whalen, T. and Inkpen, K. M., "Gathering evidence: use of visual security cues in web browsers", in Proceedings of Conf. on Graphics interface, Victoria, BC, 2005, pp. 137-144.