On the Comprehension of Code Clone Visualizations: A Controlled Study using Eye Tracking

Md Sami Uddin, Varun Gaur, Carl Gutwin, and Chanchal K. Roy
Department of Computer Science
University of Saskatchewan
Saskatoon, Canada
{sami.uddin, varun.gaur, carl.gutwin, chanchal.roy}@usask.ca

***Abstract*—Code clone visualizations (CCVs) are graphical representations of clone detection results provided by various state-of-the-art command line and graphical analysis tools. In order to properly analyze and manipulate code clones within a target system, these visualizations must be easily and efficiently comprehensible. We conducted an eye-tracking study with 20 participants (expert, intermediate, and novice) to assess how well people can comprehend visualizations such as Scatter plots, Treemaps, and Hierarchical Dependency Graphs provided by *VisCad,* a recent clone visualization tool. The goals of the study were to find out what elements of the visualizations (e.g., colors, shapes, object positions) are most important for comprehension, and to identify common usage patterns for different groups. Our results help us understand how developers with different levels of expertise explore and navigate through the visualizations while performing specific tasks. Distinctive patterns of eye movements for different visualizations were found depending on the expertise of the participants. Color, shape and position information were found to play vital roles in comprehension of CCVs. Our results provide recommendations that can improve the implementation of visualization techniques in *VisCad* and other clone visualization systems.**

***Index Terms*—Clone visualization, comprehension, code clone, VisCad, eye tracking, human-computer interaction.**

I. Introduction

Identical or similar fragments of code in a software system are called code clones. Code clones are often unavoidable in software development, because of developers' frequent practice of copy-paste programming (where existing code fragments are reused rather than implementing from scratch). Separate studies have reported that in software systems the amount of cloning ranges from 5-15% [1] and can even be 50% of the code base [2]. Although, there is an ongoing debate over the positive and negative impacts of clones, it is unanimously accepted that there is a need to detect and analyze clones [31]. Hence, clone detection and analysis has become an integral part of software maintenance. Various clone detection tools are available: some are command-line based (such as *NiCad*, *SimCad*, and *Similarity Analyzer*) which provide results only in textual form; others provide analysis in graphical as well as textual form (such as *VisCad*, *Cyclone*, and *XIAO*) [31]. Generally, visual code clone representations have advantages compared to text-only tools, as the visual representations help the analyst to develop a mental model of the code clones and the system itself. Graphical representations of clone detection results provided by various state-of-the-art command line and graphical code clone analysis tools are known as code clone visualizations (CCVs). Efficient comprehension of these CCVs is a prerequisite for maintenance and software-evolution tasks with these tools. Studies have shown that up to 90% of a typical system's development cost is spent in the maintenance phase [3], which indicates the importance of efficient code clone comprehension. Clone detection and analysis tools provide different types of visualizations – but little is known about how well people comprehend those visualizations.

To investigate this issue, we conducted a controlled study that uses eye tracking technology to explore how people interpret various CCVs. Eye tracking is gaining popularity in many areas and is frequently used alongside more traditional usability assessment methods [5]. The data provided by eye tracking (such as gaze points, fixations and saccades) can help researchers comprehend the cognitive process of a subject in terms of processing visual data [6][5][7]. For our study, we used a GUI-based framework for large-scale clone analysis called *VisCad* [4], which helps users to analyze and identify code clones through a set of visualization techniques, and metrics which cover different clone relations and data filtering operations. The software provides analysis in the form of *Scatter plots*, *Treemaps* and *Hierarchical Dependency Graphs (HDG)* with which users can visualize and analyze large volume of raw cloning data in an interactive fashion.

Our study was focused on assessing the comprehension of the set of visualizations provided by *VisCad*. We used the eye-tracker to collect each participant's gaze points, fixations, saccades and scan paths in a non-obtrusive manner as they were interacting with different visualizations. These records helped us determine participants' common usage and navigation patterns for the different visualizations types. We also considered how participants processed the visual data provided in those visualizations. Another goal was to assess how well the visualizations were serving the purpose for which they were built. The last goal was to find out usage and navigation patterns of participants based on their expertise. We set out to answer the following questions:

- How do people comprehend clone visualizations?
- How do people navigate through clone visualizations?
- Does the use of color map, size and shape provide additional assistance?

- What items in the visualizations do people fixate on?
- What do people actually look at in clone visualizations?
- Is it possible to find out the relationships among subsystems of any subject through the visualization?
- Is there any difference among expert, intermediate and novice users' ways of interpreting the clone visualizations?
- How can we assist the transition from novice to expert?

Our work makes three main contributions. First, we determined how users comprehend different types of CCV techniques. We assessed the exploration, examination and navigations patterns of experts, intermediates and novices. Second, we highlighted drawbacks in the visualizations found by the participants in this study, which will contribute to the improvement of these clone visualization techniques. Third, we have introduced eye tracking technology for understanding the user's comprehension of CCVs. This is the first investigation, to our knowledge, that uses eye tracking technology to assess comprehension of CCVs.

This paper is organized as follows: we present background on clone visualization and eye tracking, and then describe the procedures of our study. Section IV covers the results and analysis of the experiment, and considers the recommendations which is followed by related work. Last, we discuss threats to the validity of our work.

II. Background

A. Clone Visualization

Recently, clone detection and analysis has become an integral part of the software maintenance process [31]. In a software system, identical or similar fragments of code are found which are called clones. Although clones can provide a number of benefits, they can be detrimental in many cases [31]. As a result, over the past few years many clone detection tools have been proposed, and many clone visualization tools have also been developed to analyze the results of those clone detection tools [1]. Clone visualization is important because it helps developers understand the characteristics of clones, and in the long run helps in software maintenance and evolution. *VisCad* is a clone visualization tool that supports almost all the clone detection tools that are currently available [10]. It provides Scatter plots, Treemaps, and HDGs to visually describe the clones in a system:

Scatter plot: A well-known and effective visualization technique [11] [1], Scatter plots are two-dimensional matrices where each cell represents the cloning status between a pair of files or directories (see Fig. 1: Left). Cells are labelled in the horizontal and vertical axes and the labels in the horizontal and vertical axes are the same, representing the same set of subsystems. The cloning status of cell is rendered using a color map.

Treemap: The Treemap view shows the cloning status of directories and files through rectangles while maintaining their hierarchical structures (see Fig. 1: Right). The size and color of a rectangle can specify different data about the directory or file that the rectangle represents. The Treemap view can be used to identify subsystems that contribute most to the total clone pairs of a system [4].

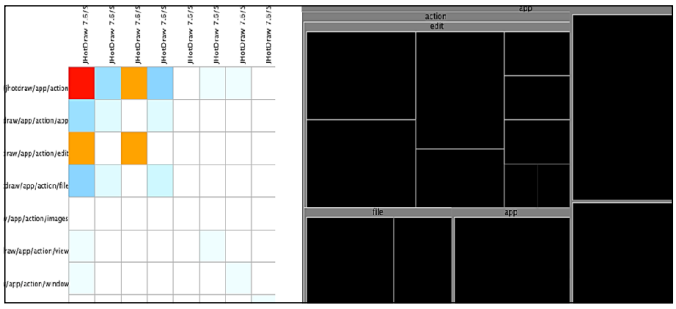

Fig. 1. *VisCad* visualizations. Left: Scatter plot, Right: Treemap.

Hierarchical Dependency Graph (HDG): This view represents the hierarchical organization of a system along with the distribution of clones (see Fig. 2). It helps to understand cloning relationship among a subject's subsystems. The graph consists of a set of nodes and edges. Nodes are represented as ellipses and edges with straight lines. While nodes can represent files or directories, edges represent containment relationships. The thickness of an edge represents the degree of cloning between two nodes. The width, height and fill color of an ellipse can be used to represent three different aspects of code cloning. The shape of a node and the thickness of an edge will change based on the cloning status they represent [4]. *VisCad* represents the graph using a radial layout where nodes are arranged in circles and the root of the hierarchical graph is placed at the center. Nodes that are at the same depth of the hierarchy, appear at the same distance from the center.

B. Eye Tracking

The physiology of the human visual system is the foundation of modern eye-tracking equipment [8] [9]. For tracking human eye movement, these systems normally use infra-red cameras: in our study, we used a *Tobii T60 XL* eye tracker (www.tobii.se) to capture different data related to eye movements and eye gaze. This equipment has two cameras, built into a 24 inch flat-panel screen, which are used to track the eye. This provided us an unobtrusive work environment which was essential for reliable measurements. The eye tracker operates at a sampling rate of 60 Hz, and has latency less than 33ms. It also provides us data with good accuracy (error rate of less than 0.5 degrees). The eye tracker system comes with a software system that records the XY screen coordinates of eye gazes and supports analysis of eye

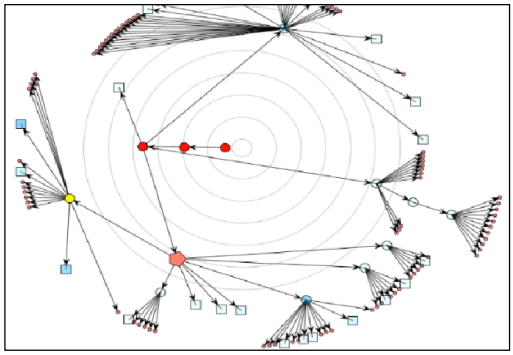

Fig. 2. *VisCad* Hierarchical Dependency Graph (HDG) visualization.

movements. The distance that this machine covers is 50-80cm. This device also supports audio/video capture option.

Eye trackers allow us to capture various types of eye movements that occur while humans gaze at an object of interest. Various types of information are provided. Fixations and saccades are two widely used eye movements in different studies. The stabilization of eyes on an object of interest for a period of time is termed as fixation. Saccades are quick movements of the eyes that move interest from one fixation to the next. The directed path formed by saccades between two fixations are known as a scan path. According to the eye tracking research community, the processing of visualized information occurs mainly during the fixations. On the other hand, this kind of processing is absent during saccades [9]. Sometimes, it helps users to find interesting parts in a visual scene which form the mental model.

III. Study

The main goal of the study was to determine how human participants interpret clone visualizations and how they use different types of information in clone visualizations to complete their tasks. Participants were given specific tasks to perform on different clone visualizations. An eye-tracker was used to capture their activities in terms of fixations, saccades, audio, and video.

A. Test System

We analyzed the open source system *JHotDraw* (version 7.6) [12] for our study. We used the clone detection results of *JHotDraw* generated by *Simian* from a previous study [4] and used these as input for *VisCad*. *JHotDraw* is a Java GUI framework for technical and structured graphics. We have used three types of clone visualizations mentioned in section II-B in our study. Additional related information about the test system is given in Table I.

B. Tasks

The tasks given to participants in our study consist of specific questions that can be answered by viewing three types of clone visualizations (Fig. 1 and 2). We prepared 25 questions for the experiment. These questions are related to general clone visualization techniques and high level concepts of clones. While there could be advanced questions for deeper level maintenance task such as bug fixing [31], we intentionally set the high level questions that represent sufficiently appropriate usages scenarios of the visualizations. Questions are given in Table II.

The set of Explorer questions are concerned with general *VisCad* interface information, but did not concentrate on visualizations. The set of questions for the Scatter plot visualization are mainly aimed to see how the participants utilize color information and textual information of the 2D matrix. In case of Treemap, the main target is to get an idea of clones while maintaining the hierarchical file structures. The last type is HDG, which also has shape and color information in it. The questions were set in such a way that the participants follow those specific visual cues: color and shape information. We tried to cover many aspect of clone visualizations with all the questions. Among all the files and clone classes shown in Table I, we manually selected a subset and produced the visualizations with *VisCad*. We included questions that were easy, medium and difficult in order to test participants' performance. We also produced visualizations with sufficient information so that participants could answer questions correctly in a reasonable time.

TABLE I. Overview of Test System

Category	Details
Target System	JHotDraw (7.6)
Detection Tool	Similarity Analyser 2.3.32
Clone Classes	737
Processed Files	680
Duplicated Files	1414

TABLE II. Questions for the Study

No.	Questions
	Explorer
1	Which child directory under *Jhotdraw* directory has the second largest number of clones in it?
2	Name the clone class which has the largest number of clone fragments.
3	Name the tool used for clone detection.
4	How many duplicated files were found by the clone detection tool?
	Scatter plot
5	Which directory(s) has no code clone pairs compared to any directory (including itself)?
6	Which directory(s) has the smallest (but not zero) number of code clone pairs when compared to directory */palette*?
7	What is being shown on both axes of the *Scatter plot* - Folders or Files?
8	Which directory contains the largest number of clone pairs compared to both other directories and itself?
9	Is this *Scatter plot* comparing among the child and grand-child directories of the selected directory?
10	In this *Scatter plot*, at what level are the directories in the selected directory being compared?
11	Are the cells in this *Scatter plot* sorted by *Clones*?
12	Are the cells in this *Scatter plot* sorted by *Cloned Lines Of Code (CLOC)*?
	Treemap
13	Name the directories which contain the 3 fragments of the selected clone class *(CC-27)*?
14	What is the name of the directory that contains the 2 fragments of the selected clone class *(CC-16)*?
15	Do the fragments of the selected clone class belong to the same directory?
16	Name the parent directory of *"file"* directory?
17	Name the parent directory of *"tool"* directory?
18	How many child directories does the *"handle"* directory have?
19	How many child directories does the *"palette"* directory have
	Hierarchical Dependency Graph
20	What is the color of the selected directory in the graph?
21	How many child directories are there under the selected directory?
22	Which child directory under the selected directory contains the largest number of clone classes?
23	How many child directories are there under the selected directory which are not collapsed?
24	Name the directory which is located at the same level as the parent of *"app"* directory and has the 2nd largest number of clone classes?
25	Name the child directory under the selected directory having largest number of clone pairs?

C. Stimuli

In eye tracking terminology, an object that is viewed by a participant is known as the stimulus. We have merged a question

and the corresponding CCV into a single stimulus. Studies on the use of eye tracking for a variety of domains revealed that a human normally bias for the top-left corner [5], [13] and/or reading from the left to right [14], [15]. Therefore, in our stimulus, questions were placed at the top-left corner and the diagram and other relevant information occupied the remaining space. All the questions shown in Table II were presented to the participants in a predefined sequence. We rearranged the question sequence for the study in a manner that, no questions from one section come in sequence except for the Explorer. This was done to avoid any learning bias, which might occur if all the questions from one section were presented at a time. An example is given in Fig. 3. The four questions from the Explorer category were presented together at the beginning of the study in order to allow participants to familiarize themselves with the system.

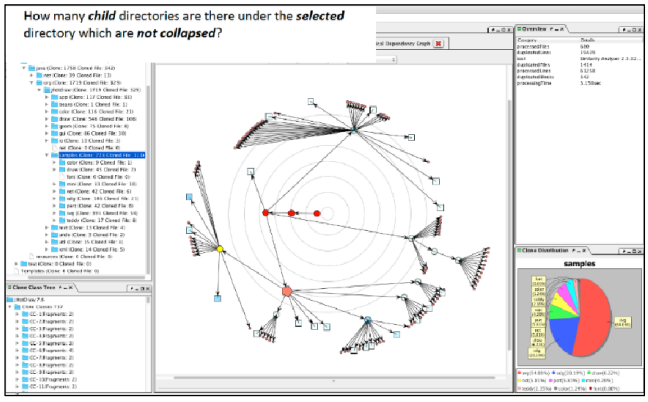

Fig. 3. Sample stimulus.

D. Participants

We recruited 20 participants (6 female) for our study, and they were compensated with $10. Participants ranged in age from 20-46 (mean 29.65) years. We wanted to have two types of participants: novices (who have little knowledge about clone visualizations) and experts (who have experience with clone visualizations). For this reason, we recruited 4 participants from outside of Computer Science, to see if there are any differences in their eye movements; the rest of the participants were from CS background. Eight people had Software Engineering as their major. Among the 20 participants, 3 were from software industry, 2 were undergraduate students, and rest were graduate level students.

Participants were asked to fill out a questionnaire about their knowledge of CCVs and clones. From their responses, we found that 30% participants had little or no previous knowledge of code clones or clone visualizations, and 70% had previous knowledge. However, 50% reported that they had very good knowledge of CCVs and other 20% had average knowledge. This was verified by analyzing their performance during and after the experiment. Additionally, almost all of the participants had prior knowledge of graph visualizations.

E. Procedure

We used the stimuli set discussed earlier (Section III-C) in the study. On the day of study, we trained the participants about our study system (for approximately 30 minutes about the code clone concept and clone visualization). We prepared a separate training manual for this purpose. Participants were also introduced to the eye tracking system.

During the study, we gave the participants the stimuli and asked them to answer questions verbally. There was no time limit to finish the task. We ran only one participant at a time and it took from 7 to 23 minutes to complete the experimental tasks. The participant was stationed in front of the eye tracker at a distance of 60 cm. Before running study, we calibrated the eye tracker for each user. The environment was a normal windows operating environment. The eye tracker automatically recorded audio, video and eye movements on the clone visualization.

The participants were asked to answer verbally, and the experimenters recorded all responses on paper. Participants were also asked to fill out a questionnaire consisting of a number of questions related to visualizations and their performance. They were also asked to write comments and suggestions after the session.

IV. Results and Analysis

We analyzed the data collected from our study to obtain understanding of participants' visual activities while answering questions about different types of clone visualization.

A. Participant and Question Categorization

We analyzed participants' performance after the study. We calculated accuracy and response time taken by participants to answer to the 25 stimuli questions using the audio and video recordings. Figure 4 shows the number of correctly answered questions for all the sections by all participants. The remainder of the questions were either wrongly answered or skipped. No one answered all 25 questions correctly.

Based on performance in answering the questions of all the sections, we categorized them into the following groups:

1) Novice: Participants who demonstrated very little knowledge of CCV. Four participants (participants 2, 7, 10 and 20) were identified as novices. These participants took approximately 7 to 13 minutes to complete the experiment. Their performance was not good compared to the other two categories mentioned below.

2) Intermediate: Participants who demonstrated average knowledge of CCV. Ten participants (participants 4, 5, 6, 8, 11, 12, 13, 14, 17 and 18) were identified as intermediates. These

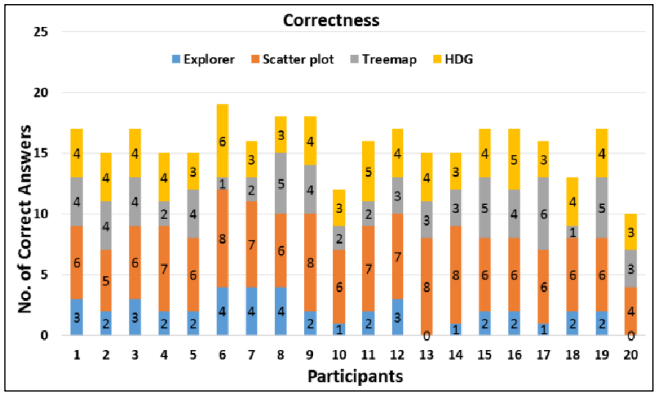

Fig. 4. Number of correct answers for all participants.

participants took between 7.14 minutes and 23.5 minutes to complete the experiment. Their performance was better than novices but below the experts.

3) Expert: Participants who demonstrated good knowledge of CCV. Six participants (participants 1, 3, 9, 15, 16, and 19) were identified as experts. These participants took between 7.1 minutes and 16.16 minutes to complete the experiment. Their performance was better than both other groups.

However, it is important to mention that participants had very different reading speeds. Some were very fast readers while others read slowly and carefully. Moreover, the majority of our participants' first language was not English. Because of these reasons we could not compare performance based on the time to complete a particular task. Explorer results were excluded for categorizing the participants as it is not a visualization technique. Explorer related questions were added into the stimuli to make participants familiar with *VisCad*.

Our classification of participants shows that we have representatives with varying CCV comprehension 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 participants to find out the difficulty level in answering the questions.

We classified all the 25 questions based on the distribution of participants answering them correctly. Questions that were answered correctly by participants in the ranges [80%, 100%], [60%, 80%), [40%, 60%), and [0%, 40%) were classified as easy, intermediate, difficult, and challenging respectively which are shown in Table III.

TABLE III. CLASSIFICATION OF QUESTIONS BASED ON THE PERCENTAGES OF PARTICIPANTS' CORRECT ANSWERS. THE QUESTION NUMBERS CORRESPOND TO THE QUESTIONS IN TABLE II.

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

B. Exploration, Examination, and Navigation

We analyzed the study data to understand how participants use their eye movements for four main kinds of interaction with the visualizations. All the study data are available online [35].

Exploration of visual space: We determined the patterns of visual search performed by the participants to locate objects required for performing a given task.

Examination of visual objects: We assessed how participants comprehend the different types of visualizations provided by *VisCad* while accomplishing a given task.

Navigation: We assessed how participants move from one object of interest to the next while searching for information.

Area of Interest: We also defined specific areas of interest on the visualizations to find out participants' interest in specific objects or areas of the CCVs.

Gaze plots (example shown in Fig. 5) provide fixations, saccades and scan paths for the analysis. The findings are reported below:

1) The eye tracker provided the fixation points, saccades and fixation time. Using this information, we conclude that experts explored visualizations more efficiently than intermediates and novices. Experts and intermediates tend to explore different areas of a visualization when seeking particular information. In contrast, novices relied on a particular area of visualization for finding out information and did not exhibit a particular pattern of navigation. We conclude that experts confirm the information they gathered from one area by looking into another area. This behavior is also exhibited by intermediates to some extent but not as much as experts.

2) For experts and intermediates, most of the fixations were found on the visualizations and the *system navigation tree* (SNT); experts only rarely fixated on empty spaces, which is not The case for novices who fixated on irrelevant spaces in visualizations for considerable amounts of time.

3) For the *Scatter plot*, most participants traversed through the axis labels to find the required folders or files to compare, and then focused on the color of the cell represented by axes.

4) In case of the *Treemap*, most of the participants first focused on *Clone Class* area containing clone fragments and then fixated on clone fragments of a selected *Clone Class* represented in red. Then they moved to edges of rectangles representing folders to find out the directory structure.

5) For the *HDG* view, most participants started from the SNT area to find out the selected directory and then moved to the selected node in HDG which is represented as a diamond shape. Participants then followed the links between directories to reach the desired directory.

6) Overall, novices spent considerable time looking at irrelevant places and we found that they did not produce a consistent pattern of navigation for most of the questions. In contrast, most of the time experts and intermediates exhibited a consistent pattern of navigation for all types of visualizations.

C. Stereotype Usage

In this section, we discuss the use of explicit stereotype information that was provided in the form of textual annotations, color and use of pie chart, etc. in the visualizations.

1) For the standard Explorer view, the experts focused slightly more on *system navigation tree* (SNT) as compared to other areas. Whereas intermediates focused mainly on SNT and

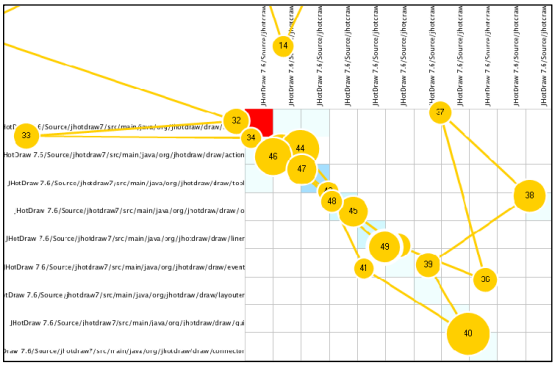

Fig. 5. Gaze plot on the portion of a stimulus.

novices, who mainly focused on the *clone distribution pie chart*, ignoring the SNT.

2) For the *Scatter plot,* all participants focused on the axes and coloring to find relevant information. Experts looked at the buttons or controls provided on the bar above the Scatter plot graph and then confirmed their answer by looking at the axes and color information provided by the cells of Scatter plot. Intermediates focused mainly on the axes and color information, and only rarely looked at the buttons or controls above the Scatter plot, but their exploration pattern was similar to experts. Novices, however, spent less time answering compared to the other groups, and did not use buttons or controls. Buttons and controls above the Scatter plot could be used as shortcut to find some of the information which otherwise takes more effort and time to figure out from the Scatter plot. Experts and intermediates were also better at comprehending color information compared to novices.

3) For the *Treemap*, the experts focused on edges or boundaries of the rectangles representing the folders to figure out the hierarchy of the folders, and also looked at the SNT to confirm the folder structure. Intermediates, however, did not focus on the SNT, but like experts they focused on the edges or boundaries of the rectangles representing the folders. There were no specific navigation patterns found for novices. Although the majority of their fixations were on the Treemap, novices did not focus on the edges or boundaries of the rectangles and the SNT.

4) For the *HDG* view, experts started from the HDG visualization and then moved to the SNT and sometimes *clone distribution pie chart* to validate the information. In contrast, intermediates and novices focused mainly on the HDG visualization. Experts and intermediates started from the selected directory, which is represented by a diamond-shaped node, and followed the links between directories, used color and shape information efficiently for answering the question. Novices also used color and shape information but their navigation pattern was more random – they did not follow the links between directories to navigate from one directory to another, and randomly searched for directories and clone-related information.

Overall, experts used the visual cues: color and shape information better than intermediates and novices. In most of the tasks, the required information was available in more than one area of the interface and experts tended to validate their answers by looking at various relevant sections. The intermediates also exhibited this behavior (although less often than experts). Novices tended to focus on one section of the visualization only and did not exhibit a particular pattern in any of the visualizations.

In addition, we also analyzed all the heat maps consisting of cumulative fixations of all the participants for a particular stimulus and compared this data with the results above. For example, the heat map in Fig. 6 for question 23 shows a large number of fixations on the visualization and some fixations on SNT, providing additional evidence that expert users tend to explore more than one area to find out required information.

D. Efficient Comprehension

For analyzing eye tracking data, there is a wide variety of eye tracking metrics [32]. Among them the most frequently used metric is the number of fixations. A large number of fixations is

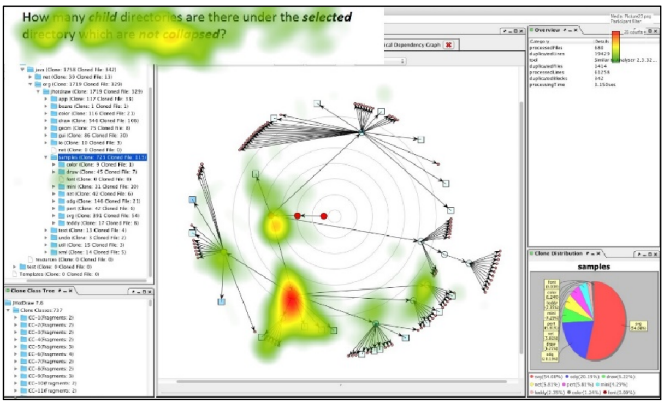

Fig. 6. Heat map showing the cumulative fixations of participants on a specific stimulus. The colors red, orange, yellow and green indicate the decrease in number of fixations from highest to lowest. Best viewed in color.

an indicator of poor arrangements of objects in a stimulus [28]. To find out the efficiency of a layout we need to count the total number of fixations on a stimulus. In our study, each stimulus corresponds to a snapshot with one of the three type of CCVs and explorer section of *VisCad*. Fewer total number of fixations on a stimulus means that the participant needed less effort to answer the associated question [13] [7] [29] [15].

The average number of fixations for a specific task was computed from the fixation points of all the participants on the associated stimulus. Table IV shows the average fixations for all the questions used in our study. To find out the relative effort required by the participants in answering the questions, four categories are formed from the analysis of the average number of fixation points. The median of all the average number of fixations of the stimuli is 101.7. The stimuli with average number of fixations in the range [0, 90), [90, 120), [120, 150), and [150, 270) are classified as low, medium, high, and extreme respectively. The previous classification from Table III is presented here to make it easier to compare between difficulty level and effort required for a stimuli.

Following the work of Yusuf et al. [28], in order to assess efficiency of comprehension for the three types of visualizations we compared the difficulty of the question and the effort needed to answer. In general, difficulty and effort should be directly related; when this is true, the question is an equal-effort task. Also, questions that require more effort than the corresponding baseline level are referred as more-effort, whereas questions require less effort than the corresponding baseline level are referred as less-effort.

We can map questions and stimuli using Table III and Table IV to the corresponding visualizations. The mapping is shown in Table V. As the number of questions were not equal in all sections, we calculated using percentages. HDG had the highest percentage of equal-effort questions whereas Scatter plot had the lowest percentage. As we are trying to find the efficiency of comprehending clone visualizations, we excluded the Explorer type from consideration.

As shown in Table V, Scatter plot was easy to comprehend: 37.50 % of the questions were of equal effort and overall most of the participants scored well for questions related to Scatter plot visualizations. It might seem from the table that Treemap

TABLE IV. CLASSIFICATION OF EFFORT REQUIRED TO ANSWER QUESTIONS BY EXPERT PARTICIPANTS BASED ON THE AVERAGE FIXATION POINTS.

Stimuli	Avg. Fixations	Effort	Levels
12	60	Low	Easy
7	63.45	Low	Easy
16	66.25	Low	Challenging
20	68.1	Low	Easy
11	68.5	Low	Intermediate
3	68.9	Low	Intermediate
8	69.55	Low	Easy
10	79.1	Low	Intermediate
17	81.05	Low	Difficult
2	87.45	Low	Challenging
14	87.85	Low	Easy
25	88.75	Low	Easy
1	101.7	Medium	Intermediate
5	107.2	Medium	Easy
19	109.05	Medium	Difficult
4	113	Medium	Difficult
13	116.25	Medium	Intermediate
18	119.8	Medium	Challenging
15	120.05	High	Difficult
23	122.6	High	Difficult
22	123.55	High	Intermediate
21	129.7	High	Challenging
9	130.1	High	Easy
6	161.9	Extreme	Difficult
24	269.6	Extreme	Intermediate

TABLE V. DISTRIBUTION OF QUESTIONS (%) BASED ON LEVEL AND EFFORT.

Type	Less Effort	Equal Effort	More Effort
Scatter plot	25.00	37.50	37.50
Treemap	57.10	42.90	0.00
HDG	16.67	50.00	33.33

(57.10 % less-effort questions) was the easiest to comprehend, but this is not the case. We found that participants spent less time for answering Treemap questions as they were not able to figure out the folder structure. If we compare the percentage of questions answered correctly by all the participants for Treemap as compared to other visualizations it was very low (see Table III). Thus, participant performed worst for Treemap as compared to other visualizations. Participants did well for HDG questions: 50% of the questions were equal-effort.

We also asked participants after the experiment to rate the comprehensibility of CCVs. Questions were 7-point Likert type (1 = strongly disagree, 7 = strongly agree). Participants were asked to rate whether color and shape information helped them along with the ease of using the different visualizations.

Results are shown in Fig. 7. From the participants' response it is evident that they found the Scatter plot easier to comprehend. Most of them experienced difficulties in comprehending the Treemap. HDG was relatively easy to comprehend which is reflected in Table V.

E. Participant Comments

Participant comments were evenly divided where some were in favor and same were against certain aspects of the different types of CCVs. Some of the comments are mentioned below for each visualization.

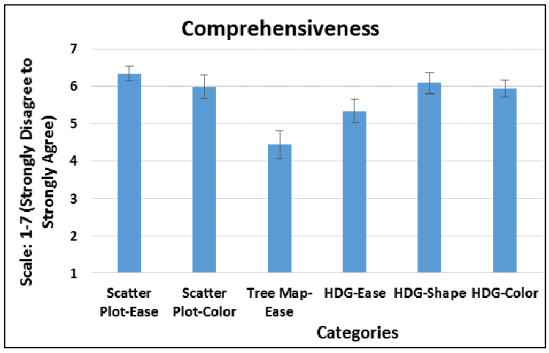

Fig. 7. Comprehensibility of visualizations.

1) Scatter Plot: One participant said, *"..Scatter plot was easily understood. Clone information was clearly shown by the colors and their intensity."* Another one mentioned, *"..color information [of Scatter plot] was very helpful….[but] it becomes difficult to differentiate between cells having white and very light blue color."* Other participant suggested, *"Colors [of Scatter plot] are sometime confusing [to understand] if the differences in clone pairs are not too high."* As evident from the above mentioned comments most of the participants did well with the Scatter plot. The only issue was that when two or more cells were represented by similar intensities of the same color it became tough to find out which cell has more clones.

2) Treemap: One participant said, *"..I found the graph [Treemap] a little confusing due to the vagueness of the border. I was not able to figure out the directory structure."* He also suggested, *"Make the colors and borders more discrete [for Treemap]."* Another participant commented, *"...Treemap view was confusing. Finding the folder hierarchy was tough...using different colors for different directories and source files..[should make it clear]."* However, one participant reported that, *"Identifying the distribution of clone fragments [in Treemap] was easy."* In case of *Treemap*, most of the participants mentioned about a common issue that is vagueness of the border and all directories being represented by the same color.

3) HDG: One participant said, *"…[in HDG] color and shape helped to figure out required information but a lot of lines [overlapping] between directories [sometimes] made the visualization clumsy."* Another one said, *"..different type of shapes and colors were really helpful to understand hierarchical dependencies [in HDGs]. Shapes of collapsed and non-collapsed directories were helpful too."* Another participant mentioned, *"..[sometimes] same intensity of a color made it tough to tell which directory has more clones. Using different colors instead of different intensities of same color will be more helpful."* Most of the participants were comfortable using HDG. The only two issues evident from comments are multiple directories being represented by the same intensity of a color and clumsiness of the HDG interface. However, clumsiness in HDG is not an issue as in actual *VisCad* system the user can move the objects and do zoom operation to make things clear.

F. Recommendations

Although this study used *VisCad*, our findings can be generally useful both for developers and end users of CCV tools who are concerned about efficient comprehension of CCVs. Although participants were able to comprehend the visualizations most of the time, there are some areas in each of the visualization techniques which were problematic. For example, in Scatter plot there might be a very small number of code clone pairs among two directories and hence, the color of the cell will be faint blue. Sometimes, participants reported that they faced difficulty distinguishing between the white and the faint blue cells. Based on the findings of the study we are providing some general recommendations:

1) For CCV Tool Developers:

a) Scatter Plot: In *VisCad*, different cell colors are used to represent the cloning status between two folders or files. As mentioned in the above example, participants got confused between the faint blue and white colors. To resolve this problem, we recommend using black for the cell representing no clone pairs instead of white. In addition, a legend explaining the meaning of visual cues such as color, shape, etc. should be used in CCVs. This reduces the cognitive load of users as they do not need to remember the mappings for the visual features. For example, currently in *VisCad*, red shading represents more clone pairs compared to blue shading, but this is not made explicit in the visualization.

b) Treemap: Most participants reported they could not figure out the directory structure clearly. The boundaries of the Treemap's rectangles were not clear. This problem becomes severe when the Treemap is large with many directories, because the borders become more difficult to distinguish. Our recommendation is to use different colors for representing different directories and the code clone fragments located in those directories. An alternative solution could be changing the color of the folder name or annotating it along with the locations of code fragments at run time. Both these solutions can help users identify directory structure and locations of code fragments.

c) HDG: A few participants reported that they faced problems understanding the directory structure in the HDG view as it was cluttered at times. However, in the actual *VisCad* system, users can move objects and perform zoom operations to overcome this problem. Like Scatter plot, in HDG different colors are used to represent different clone pair intensities. We recommend using legends as mentioned above.

Developers often provide zoom features for visualizations, and this feature is present in *VisCad*. However, when a user zooms in into a particular area of a visualization, other areas are hidden and hence, the user cannot see the whole visualization to compare different areas. One possible solution to this problem is a magnifier feature, which provides an in-place zoom of a limited area around the user's mouse cursor; this allows them to see a region in more detail without changing the whole view.

Our study found that experts often looked at control areas (e.g., buttons and labels above the Scatter plot) to find out required information – in some cases this was quick compared to traversing the whole visualization. We recommend that developers include such shortcuts to allow users to easily manipulate and quickly retrieve the required information.

2) For End Users: Based on the findings of our study, we have some recommendations for the end users especially for the novices and intermediates. In this study, we found that experts exhibited particular exploration and navigation behaviors, but others did not show consistent patterns. It is possible that novices and intermediates can improve their success with the tool by following the expert patterns (as mentioned in Section IV-C); these patterns could be tracked by the system, and suggestions could be given when users could engage in an expert behavior.

In general, efficient training in the organization of the visualizations and of the system's interface will result in better comprehension of the visualizations by users. In this study, we found that novices spent considerable time looking at irrelevant parts of the interface, and did not exhibit any consistent pattern of exploration and navigation. Experts produced a particular pattern of exploration and navigation as they knew the functionality and purpose of most of the objects located in visualizations. There were four non-computer-science participants without any idea of code clones and CCVs. Even without this background, they performed reasonably well after brief training on *VisCad*. Therefore, training should be such that users understands the functionality and purpose of all the objects making up a visualization.

3) For Researchers: From our study we noticed that for a given set of tasks, there were some issues with the comprehension of the visualizations (discussed in Sections IV-C, D, and E), which suggest that a given visualization technique might not work well for generalized tasks. To design the visualization techniques appropriately, it is also important to understand developers' intent [34]. Thus our findings suggest to conduct the user studies before implementing a particular visualization in order to find out whether the target task could indeed be achieved by that visualization and if yes to what extent. We recommend implementing use-case centric visualizations determined by the user studies. We also suggest further studies to figure out useful techniques for visualizing clone evolution from management perspective [31]: effective CCVs for clone management activities and their implications in the context of real world software development.

V. Related Work

A. Clone Visualizations

Over the years, a great many studies have been conducted to detect clones and then visualize the large amount of textual data returned by the detectors as results. A detailed list of them can be found in Roy and Cordy's technical report [1]. Johnson [16] applied Hasse diagrams, a graph based visualization, to represent textual similarity between files. Later, he proposed exploring the files and clone classes via hyperlinked web pages [16]. A set of polymetric views to identify clones within a subject system was proposed by Rieger et al. [2]. Scatter plots are the most popular approach among them for clone visualization, which helps visualizing both inter- and intra-system clones. It was later enhanced by Higo et al. [17]. After that, Livieri et al. [11] introduced a different variant of Scatter plot: color heat map. There were also

some other visualization tools- for example, *Gemini* [18] developed by Ueda et al. which worked with *CCFinder* [19] for clone detection. These techniques were limited either by clone detection tool or by visualization method. Harder et al.'s *Cyclone* multi-perspective clone evolution inspection tool which supports five views to inspect clone data [33]. However, *VisCad* supports Scatter plot, Treemap, and HDG and can analyze clone detection results of a large set of detectors including *CCFinder*, *Cyclone* and others which are recently available [4] [10].

Jiang and Hasan [20] presented a visualization called the clone system hierarchical (CSH) which was basically a tree layout. Kapser and Godfrey [21] developed a tool *CLICS* [22], which can display the cloning relationships between subsystems with a hierarchical containment graph and the visualization is performed through a program called *LSEdit*. *VisCad* also provide same information with radial layout utilizing the display area more efficiently. *ConQAT* [18] is a quality assessment toolkit which offers Treemap, clone visualizer and family visualizer views for analyzing clones. *VisCad* allows users to analyze the cloning dependency among the subsystems and supports both CC and clone system metric sets, many of which are absent in *ConQAT* and other visualization tools. Another project was conducted in this area by Alalfi et al. [23] where they developed *SIMGraph* to help their industrial partners to visualize and understand clones. *Visual Studio 2012* has also integrated a code clone detection and analysis tool called *XIAO* [30] which reports clone results in line by coloring the cloned blocks or snippets.

We have seen that there are a large number of clone visualization techniques and tools for clone visualization and analysis. But, they are focused on different directions to solve different problems related to clone visualizations. Moreover, most of them are dependent on a particular clone detection tool and only a few of them are publicly available for use. Considering all these facts, we selected *VisCad* for our study, which supports different clone detection tools and covers almost all the aspects of visualization techniques offered by others. Even though we used *VisCad*, our findings are generally applicable to all CCVs.

B. Eye Tracking and Comprehension

In *Human-Computer Interaction (HCI)*, there are some factors and technical considerations for using eye tracking. Some of them are discussed by Jacob [9]. Eye tracking technology has recently been used in several comprehension studies which were related to code, webpage and UML. *WebGazeAnalyzer* is a tool developed by Beymer et al. [24] to record and analyze eye gaze during web browsing sessions. A study conducted by Whalen et al. [25] investigated elements in web browsers that are viewed and sometimes ignored by users, and how noticeable are different elements. Many studies have been also been done to understand programmers' behavior. For example, Crosby et al. [14] analyzed the eye gaze of novice and expert programmers; Bednarik et al. [6] applied eye tracking to study comprehension of Java programs; and Iqbal et al. [7] investigated the mental workload demanded by computer-based tasks.

Sharif et al. colleagues [26] used eye tracking to capture quantitative data to investigate the effect of identifier-naming conventions on code comprehension. According to them, the use of eye tracking was a better alternative to traditional means which were used in a similar kind of previous work [27]. In that study, they replicated their previous work and used an eye tracker to extend the results and assess the effects of layout on the detection of roles in design patterns. To assess how well participants comprehend UML class diagrams, Yusuf et al. used eye tracking [28], and assessed the most effective characteristics of UML class diagrams that support software tasks both for novices and experts. Another study [29] was conducted to understand the software visualization by using eye tracking. Considerable work has been done with eye tracking techniques to comprehend various types of visualization, code pattern and UML diagram understanding. However, ours is the first use of eye tracking to investigate clone visualizations, one of the promising vehicles of software maintenance and evolution.

VI. Threats to the Validity

One of the major threats of such a study is the possible bias because of the selection of the participants. In order to mitigate this issue, we recruited participants not only people from software engineering background but also from other backgrounds. Overall, people were engaged and motivated during the tasks – this reduces the threat that they may not have been motivated enough to perform to their full potential and interest. Also, most of them had never used the *VisCad* system and were trained on *VisCad* for only a short time. Thus, the learning ability of the participants in such a short time might have affected the results. To counteract this effect we included participants who had good knowledge of CCVs.

The small number of participants, who had a variety of knowledge about CCVs, might be another threat to validity. However, our total of 20 participants is comparatively high compared to other studies that also used eye tracking [6] [27] [29]. We showed users the still images of the visualizations provided by the *VisCad* system. It would have been better if we could let the users interact with *VisCad* directly since they could utilize the visualizations themselves. But, due to the limitations of eye tracking device we could not do so. However, we were careful in choosing the images. We created all possible use-cases which a user can perform with these visualizations while interacting with *VisCad*. Thus, we believe that our study is not really affected by the choice of images for this study.

One could argue on the visualization techniques chosen since there are many others [1]. We mitigate this threat by choosing the most popular clone visualization techniques [31]. One might also question about the visualization tool chosen. While other tools could have been used, we opted to use *VisCad* as this is one of the latest visualization tools available that covers most of the visualization techniques well and that we are familiar with this tool well. While the choice of the tools may have an impact on the study, our findings and recommendations seem to be generic and tool independent. One might also argue about the stimuli questions for the user study since there could be advanced usage scenarios for software maintenance and evolution [31]. On this issue we should note that we focus on the popular visualization techniques and not those related to clone evolution. We chose questions that could sufficiently represent the chosen visualiza-

tions and thus we believe that our study findings and recommendations are of great value to the tool developers, end users and researchers.

VII. Conclusions

We used eye-tracking to assess how people comprehend different types of Code Clone Visualizations provided by the *VisCad* system. Our findings showed that experts tend to look at different areas and objects in the visualizations to validate the information they have gathered. This behavior was also exhibited by intermediates though it was not as great as experts. Whereas the novices focus only one specific area/object to find out the information and does not look around to verify the information. Additionally, experts and intermediates tend to follow a particular navigation pattern (for example, in HDG they followed the links to find out the directories/files whereas the novices did not exhibit any particular pattern). We also made an observation that, even if participants could not answer the question correctly; they got very close to the answer by using stereotype and color information. Our findings will help designers and developers of CCVs improve their systems, which will ultimately benefit users in comprehending CCVs.

Acknowledgment

We thank Mr. Muhammad Asaduzzaman for his assistance in the use of *VisCad*. This work was supported by NSERC.

References

[1] C. K. Roy, and J. R. Cordy. A survey on software clone detection research. Technical Report-541, SC, Queen's University, 2007.

[2] M. Rieger, S. Ducasse, and M. Lanza. Insights into system-wide code duplication. Proc. WCRE '04, pp. 100 – 109, Nov. 2004.

[3] F. P. Brooks. The mythical man-month: Essays on software engineering. Addison-Wesley Professional, 20th anniv. ed, 1995.

[4] M. Asaduzzaman, C. K. Roy, and K. A. Schneider. VisCad: flexible code clone analysis support for NiCad. In IWSC, pp. 77–78, 2011.

[5] A. Bojko, "Eye tracking in user experience testing: How to make the most of it". Proc. CUPA, Montréal, Canada, 2005.

[6] R. Bednarik, and M. Tukiainen , "An eye-tracking methodology for characterizing program comprehension processes", Proc. SETRA, San Diego, CA, 2006, pp. 125-132.

[7] S. T. Iqbal, P. D. Adamczyk, X. S. Zheng, and B. P. Bailey, "Towards an index of opportunity: Understanding changes in mental workload during task execution", Proc. SIGCHI conf. on HFCS, Portland, Oregon, USA, 2005, pp. 311-320.

[8] A. T. Duchowski, Eye tracking methodology: Theory and practice, London, Springer-Verlag, 2003.

[9] R. J. K. Jacob, "What you look at is what you get: Eye movement-based interaction techniques", Proc. SIGCHI-CHFCS: Empowering people, Washington, 1990, pp. 11-18.

[10] VisCad: http://www.cs.usask.ca/~croy/viscad (Dec 2011).

[11] S. Livieri, Y. Higo, M. Matushita, & K. Inoue. Very-large scale code clone analysis & visualization of open source programs using distributed ccfinder: D-CCFinder. ICSE, pp. 106-115, '07.

[12] The JHotDraw: http://www.jhotdraw.org/ (Dec 2011).

[13] J. H. Goldberg, M. J. Stimson, M. Lewenstein, N. Scott, and A. M. Wichansky, "Eye tracking in web search tasks: Design implications", Proc. SETRA, Louisiana, 2002, pp. 51-58.

[14] M. E. Crosby, and J. Stelovsky, "How do we read algorithms? A case study", IEEE Computer, vol. 23, no. 1, 1990, pp. 24-35.

[15] A. Khiat, Y. Matsumoto, and T. Ogasawara, "Task specific eye movements understanding for a gaze-sensitive dictionary", Proc. CIUI, Madeira, Portugal, 2004, pp. 265-267.

[16] J. H. Johnson. Visualizing textual redundancy in legacy source. Proc. CASCON '94, pp. 32–41. IBM Press, 1994.

[17] Y. Higo, T. Kamiya, S. Kusumoto, and K. Inoue. Code clone analysis methods for efficient software maintenance. Tech. Report of SEL in Osaka University, (SEL-Mar-29-2004), 2004.

[18] Y. Ueda, T. Kamiya, S. Kusumoto, and K. Inoue, "Gemini: Maintenance support environment based on code clone analysis". Proc. METRICS '02, pp. 67–76, Washington, DC, USA, 2002.

[19] T. Kamiya, S. Kusumoto, and K. Inoue, "CCFinder: a multilinguistic token-based code clone detection system for large scale source code". IEEE Transactions on SE, 28:654–670, 2002.

[20] Z. M. Jiang, and A. E. Hassan. "A framework for studying clones in large software systems". Proc. SCAM '07, pp. 203–212, 2007.

[21] C. Kapser, and M. W. Godfrey. "Aiding comprehension of cloning through categorization". Proc. IWPSE, pp. 85–94, 2004.

[22] The CLICS: http://www.swag.uwaterloo.ca/clics/ (Dec 2011).

[23] M. H. Alalfi, J. R. Cordy, and T. R. Dean, "Analysis and clustering of model clones: An automotive industrial experience," CSMR-WCRE '14 - SEW, pp.375-378, 3-6 Feb. 2014.

[24] D. Beymer, and D. M. Russell, "WebGazeAnalyzer: a system for capturing and analyzing web reading behavior using eye gaze", Proc. CHI '05, Portland, OR, USA, 2005, pp. 1913-1916.

[25] T. Whalen, and K. M. Inkpen, "Gathering evidence: use of visual security cues in web browsers", Proc. CGI, pp. 137-144, 2005.

[26] B. Sharif, and J. Maletic, "An eye tracking study on the effects of layout in understanding the role of design patterns". Proc. ICSM, pp. 1–10, Sept 2010.

[27] D. Binkley, M. Davis, D. Lawrie, and C. Morrell, "To camelCase or under_score". In IEEE 17th ICPC, pp. 158–167, May 2009.

[28] S. Yusuf, H. Kagdi, and J. Maletic, "Assessing the comprehension of UML class diagrams via eye tracking". In ICPC, pp. 113–122, 2007.

[29] H. Kagdi, S. Yusuf, and J. I. Maletic, "On using eye tracking in empirical assessment of software visualizations", Proc. WEASELT, Atlanta, GA, Nov 5, 2007, pp. 21-22.

[30] Y. Dang, D. Zhang, S. Ge, C.Chu, Y. Qiu, and T. Xie, XIAO: Tuning code clones at hands of engineers in practice, ACSAC '12.

[31] C. K. Roy, M. F. Zibran, and R. Koschke, "The vision of software clone management: past, present and future", Proc. CSMR-18/WCRE-21 - SEW'14, 16 pp., Belgium, Feb. 2014.

[32] J. Hyona, R. Radach, and H. Deubel, The mind's eye: Cognitive and applied aspects of eye movement research, 0-444-51020-6 ed., 2003.

[33] J. Harder and N. Göde. Efficiently handling Clone Data: RCF and Cyclone. Proc. IWSC, pp. 81–82. ACM, 2011.

[34] D. Chatterji, J. C. Carver, and N. A. Kraft. Cloning: The need to understand developer intent. In IWSC, pp. 14–15, 2013.

[35] Clone Visualization Eye Tracking Data: http://www.cs.usask.ca/~croy/SCAM2015EyeData.zip.