# On the Use of Eye Tracking in Software Traceability

Bonita Sharif
Department of Electrical Engineering and Computer Science
Ohio University
Athens, Ohio 45701
sharif@ohio.edu

Huzefa Kagdi
Department of Computer Science
Winston-Salem State University
Winston Salem, NC 27110
kagdihh@wssu.edu

## ABSTRACT

The paper advocates for the induction of eye tracking technology in software traceability and takes a position that the use of eye tracking metrics can contribute to several software traceability tasks. The authors posit that the role of eye tracking is not simply restricted to an instrument for empirical studies, but also could extend to providing a foundation of a new software traceability methodology. Several scenarios where eye-tracking metrics could be meaningful are presented. The specific research directions include conducting empirical studies with eye-tracking metrics and replicating previously reported empirical studies, eye-tracking enabled traceability link recovery and management methodology, and visualization support.

## Categories and Subject Descriptors

D.2.7. **Software Engineering**: Distribution, Maintenance, and Enhancement – documentation, restructuring, reverse engineering and reengineering.

## General Terms

Measurement, Documentation, Experimentation, Human Factors.

## Keywords

Eye-tracking metrics, traceability studies, link recovery and evolution

## 1. INTRODUCTION

There have been a number of empirical studies in recent years evaluating software artifacts using eye-tracking equipment. Eye trackers have become accessible to researchers who are using them to gain additional insights into software development activities. Software traceability deals with several types of such artifacts such as UML diagrams and source code. This paper proposes that the use of eye tracking should extend to software traceability tasks. This approach may augment existing work in traceability such as link recovery. In addition, it provides a new direction of research to be investigated. There is potential for the use of eye tracking to provide some additional insight and add to the already existing body of traceability knowledge.


*TEFSE*'11, May 23, 2011, Waikiki, Honolulu, HI, USA


Modern eye trackers implicitly collect a subject's (e.g., developer) activity data in a non-obtrusive way while they are performing a given task. The equipment collects pertinent data including eye gazes on the visual display (stimulus) and an audio/video recording of the subject's session. This eye movement data could provide much valuable insight as to how and why subjects arrive at a certain solution. Therefore, we term the eye gaze measures collected from eye tracking as *white box* measures. We believe these measures can add a new additional dimension in supporting software traceability tasks. These measures could be grouped together to form an eye tracking metric for measurement.

In this paper, we highlight a few tasks in which eye tracking metrics may help in addressing some of the challenges put forth in the grand challenges document [1] published in 2007. We conjecture that these eye-tracking metrics can directly help with certain traceability tasks.

The rest of the paper is organized as follows. In Section 2, we describe some eye tracking terminology. The support for traceability tasks using eye tracking is discussed in Section 3, with concluding remarks in Section 4.

## 2. EYE TRACKING

We discuss basic eye tracking terminology and give examples of eye tracking done on source code and UML class diagrams.

### 2.1 Terminology

The underlying basis of an eye tracker is to capture various types of eye movements that occur while humans physically gaze at an object of interest. Fixations and saccades are the two types of eye movements. A *fixation* is the stabilization of eyes on an object of interest for a certain period of time. *Saccades* are quick movements that move the eyes from one location to the next (i.e., refixates). A *scan path* 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 [7]. The visual focus of the eyes on a particular location triggers certain mental processes in order to solve a given task [10].

### 2.2 Source Code and UML Design Examples

A visual description of eye gaze is given in **Figure 1** for a UML class diagram used in [15]. **Figure 2** shows eye gaze on source code used in [16]. The fixations are shown as circles on the diagram. The radius of the circle represents the duration of the fixation. The bigger the radius, the more time is spent looking

at that particular point on the diagram. Each fixation has a number displayed in the center of the circle, which indicates the order when the fixation occurred. A scan path is formed by connecting consecutive fixations.

In the UML class diagram, we clearly see that the developer focuses their attention i.e., eyes on two major classes based on the task he/she is trying to solve. This activity suggests that there is some logical coupling (implicit link) between the two classes with respect to the task at hand. However, this link is within one model i.e., the class diagram. In this example, both these classes were Singletons and the developer was looking for Singletons in this particular module of *Qt*'s (an infrastructure for common graphical interface development) design.

In Figure 2, we see some areas of the code having a much higher density of fixations than others. One research question that is perhaps worthy of investigation is the following: *Given the associated source code and design for a certain feature, can we link the eye tracking metrics of the code with the eye tracking metrics of the design to generate traceability links or enhance existing methods of link retrieval?* We believe that these types of inter-artifact analysis tracking metrics provide opportunities for building probabilistic models for link establishment/recovery, and can be a worthwhile endeavor.

The current role of eye trackers in software engineering is mostly limited to empirical assessment and the status quo would be maintained for a while. This restriction is understandable due to the fact that eye-tracking equipment is not affordable to the level of common computer components (the cost difference is in the order of several magnitudes). It is perhaps not farfetched to say that in the future, eye-tracking technology would become more and more affordable; after all history is our guide in this aspect when it comes to any technology. When this happens, eye trackers would be a regular fixture on personal computers (similar to web cameras and other peripherals). It would be possible to do things with "the blink of an eye", similar to what we do today "by word of mouth or click of a button" in integrated development environments. Capturing eye gazes would be as simple as capturing screen shots or voice or videos or activity logs. Such an eye tracking enabled IDE would offer unique opportunities to software engineering research.

## 3. TRACEABILITY TASK SUPPORT WITH EYE TRACKING

Here, we describe four representative software traceability tasks that could benefit from the use of eye tracking metrics.

### 3.1 Empirical Studies

Traditionally, objective measures such as the accuracy/level of response and time needed are collected from traditional empirical studies. For example, human subjects are asked to report their final answers on the completion of a given task (e.g., to assess the validity of a tool-recommended traceability links) and their response time is recorded. We term such measures as *black box* measures as they only record the final outcome after a specific task conclusion. That is, no other data is collected, at least not implicitly, while a human subject is performing a given task. Explicit method such as "think aloud" are feasible, but they bring a potential side effect of distracting the subjects from the core task at hand. Additionally, black box measures 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.

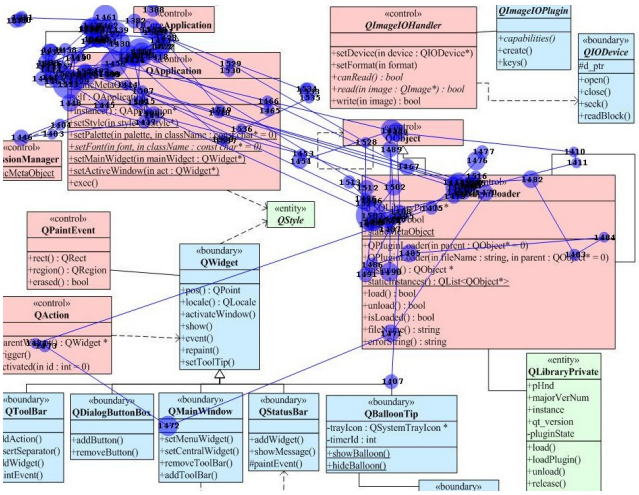


**Figure 1. A gaze plot of a UML class diagram while the developer is solving a particular software task.**

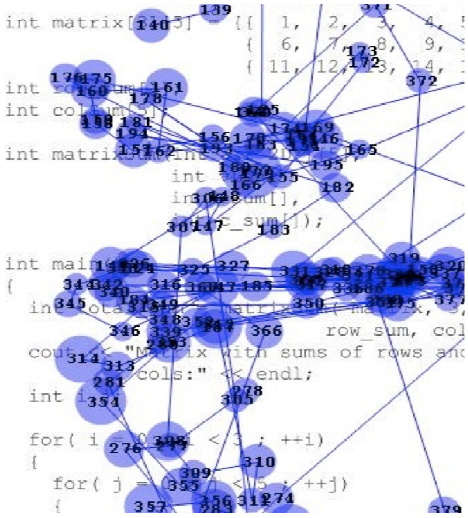

**Figure 2. A gaze plot of source code while the developer is trying to comprehend what it does**

The *white box* measures obtained from eye tracking provide an opportunity to empirically assess software traceability tools. The first immediate organic direction is to apply on tools that are visual in nature. Marcus et al. [13] presented a visualization environment for traceability links. White box measures can be used not only to assess the efficacy of different types of links and their representations employed by the developers for different tasks, but also ratiocinate the associated success and failure outcomes (e.g., from eye gaze patterns). We hope that such studies on the underlying visual metaphor, mapping, and

information and visual spaces would help provide insights that enable the sustained evolution of (needed) visualization tools.

Another direction is to use the added dimension of white box measures to replicate previously reported traceability studies. For example, work by Lucia, et al. [5] [6] on the tool *ADAMS*. Fixation-based effort metrics, such as the one presented in our previous work, could be used to note the amount of effort (in conjunction with the traditional time measure) needed in the traceability link assessment study. The eye gaze patterns could help in studying differences in strategies adopted by developers with varied expertise, domain knowledge, and skills [20].

Lastly, the eye tracking method could provide an avenue for cross validation with traditional methods. As the use of eye tracking gets more prevalent in empirical studies on various artifacts, it is not unreasonable to speculate that a large volume of eye gaze data to be generated. A relevant question here is how can this data be used in traceability studies?

## 3.2 Visualization Environments

We discuss how eye movement measures could aid in improving the visualization support for software traceability. Here, we focus on the common tasks of exploration, examination, and navigation support for traceability links.

An exploration activity deals with how subjects perform searches to locate or create traceability links of interest. The number and size of fixations could help identify areas of the source and sink nodes that smoothly assist or create bottlenecks in performing a task. Scan paths provide the order and directionality information in which the links were traversed. For example, were only the relevant links immediately visited and only once? Such information should help in designing effective visual layouts and organization of the visual space.

An examination activity deals with how subjects visualize, in detail, whole or parts of a specific class and relationships. From our experience, fixations can be recorded at the granularity of a specific line (e.g., source code class, attribute, and method names). Thus, fixations could be used to assess questions/tasks that are related to a specific node (e.g., class). Also, the durations of fixations give information about which parts of a specific class receive the most attention.

A navigation activity deals with how subjects move from one traceability link of interest to the next after their discovery. Once again fixations and saccades could be used to justify a traceability link layout strategy in supporting navigation (e.g., in providing a guided path).

The eye tracking data (if regularly collected) could be used to support other visualization tasks. For example, it can offer a new context to browsing history. One example would be to design a filter that uses the eye tracking metrics to show links that were most viewed first.

## 3.3 Link Recovery

We believe eye tracking metrics can be used to augment existing traceability link recovery techniques [18] as well as generate new ways of link recovery between software artifacts. In general, we observe two broad possibilities: linking eye tracking measures across models, and using eye tracking measures to augment existing link recovery techniques.

With respect to linking eye tracking measures across models, we could use fixations and saccades to determine which source code element for example, should be linked to a high level design element in a UML class diagram. Some data on how developers view UML class diagrams [9] [20] [17] exists. In order to perform such a linking mechanism, studies on source code for the corresponding systems would also be needed. One scenario is to collect eye tracking data while developers are working with different types of software artifacts in an IDE such as *Eclipse*. A link retrieval algorithm then analyzes the eye tracking data: fixations, their duration and saccades, to determine if a link between the software artifacts exists (e.g., source code and UML class diagrams). Another possibility is to compare eye tracking data of different developers and build a confidence based link recovery model. For example, if the same pattern is found in the eye gaze data of several developers, it's likely to be a link. Information about the fixation, saccades, and scan paths can also be used to figure out the order of the link i.e., link directionality, while the developer is viewing several software artifacts.

The second possibility is using eye tracking measures as metadata to enhance existing traceability link recovery algorithms such as LSI and probabilistic approaches. As an example, in *Poirot Tracemaker* [12], we could use eye gaze information such as the number of fixations to determine the most viewed class or method in a class diagram and plug that into their traceability link finder (e.g., as meta data). The same can be done for the *ADAMS* [5] [6] traceability link recovery tool. This possibility uses existing eye gaze behavior of individuals to feed the link recovery process. Each of these venues calls for further investigation.

## 3.4 Link Maintenance and Evolution

Link maintenance deals with making sure the link model [14] between artifacts remains consistent after a change is made, such as adding a new feature. One possibility is to tag eye tracking data while developers are using traceability links to work on maintenance tasks. This data could then be used at some later point during the maintenance and evolution phase. Having access to the historical information of eye gazes can provide an avenue to explore new possibilities. For example, inferring the co-change relationships (or logical couplings) of links based on previously recorded gaze patterns. Such couplings may help in maintenance oriented tasks such as if a specific link changes, what other links may also need to be changed? Also, it can help in identification of links that are viewed most frequently and therefore their importance in being kept consistent (i.e., ranking links to be maintained according to their visual importance).

## 3.5 Related Work

We briefly discuss related work in the area of eye tracking used in the context of software engineering artifacts. Yusuf et al. [20] conducted a study to determine if different class diagram layouts with stereotype information help in solving certain tasks. They used the Tobii eye tracker that is unobtrusive with no equipment/gear needed to be mounted on the human subject (a big difference from previous generation eye trackers). Guehéneuc [8] investigated the comprehension of UML class diagrams. Jeanmart et al. [9] conducted a study on the effect of the *Visitor* design pattern on comprehension using an eye tracker. Sharif et al. [17] also conducted an eye tracking study

assessing the role layout has in the comprehension of design pattern roles. A statement advocating the use of eye tracking in assessing software visualizations is given in [11]. With respect to source code, one of the first studies done was by Crosby and Stelovksy [4]; they studied the eye gaze of novice and expert programmers. Uwano et al. [19] also study eye gaze patterns while they are detecting defects in source code. Bednarik et al. [2] study the comprehension of Java programs using an eye tracker. They also conduct an experiment in debugging strategies within an IDE setting using an eye tracker [3]. Sharif et al. [16] conduct an eye tracking study to analyze the impact of identifier style on code comprehension.

# 4. CONCLUDING REMARKS

The position this paper takes is to include eye tracking metrics in the field of software traceability. A number of different scenarios with respect to four main traceability tasks namely, empirical studies, link visualization, link recovery and link maintenance are provided. Each of the areas should be investigated in detail by developing hypotheses and testing them. There are some threats to validity when using eye trackers. One immediate threat is that what you see is not always what you need/want. Also, sometimes capturing data might not be very practical. Another issue is the noise in the eye tracking data itself due to calibration errors. Not all people are ideal candidates to use as eye tracking subjects. In this case, capturing data might not always be possible. However, acknowledging the threats and setting limits on what is realistically possible, we envision the future of software traceability to include eye-tracking measures, as eye trackers become more and more common.

# 5. REFERENCES


[1] Antoniol, G., Berenbach, B., Egyed, A., Ferguson, S., Maletic, J. I., and Zisman, A., "Problem Statements and Grand Challenges, Center of Excellence for Traceability", Lexington, KY September 10th 2006.

[2] Bednarik, R. and Tukiainen, M., "An Eye-tracking Methodology for Characterizing Program Comprehension Processes", in Proc. of Symp. on Eye tracking research & Applications (ETRA), San Diego, 2006, pp. 125-132.

[3] Bednarik, R. and Tukiainen, M., "Temporal Eye-tracking Data: Evolution of Debugging Strategies with Multiple Representations", in Proc. of Symposium on Eye Tracking Research & Applications (ETRA), Georgia,2008,pp. 99-102.

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

[5] De Lucia, A., Fasano, F., Oliveto, R., and Tortora, G., "ADAMS: ADvanced Artefact Management System", in Proc. of 10th Eur. Conf on Soft. Maint. and Reengineering (CSMR'06) Italy, Mar 22-24 2006,pp.349-350.

[6] De Lucia, A., Oliveto, R., and Tortora, G., "Assessing IR-based traceability recovery tools through controlled experiments", *Empirical Software Engineering*, vol. 14, no. 1, 2009, pp. 57-92.

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

[8] Guéhéneuc, Y.-G., "TAUPE: towards understanding program comprehension", in Proc. of 16th IBM Centers for Advanced Studies on Collaborative research (CASCON), Canada, Oct 2006, pp. 1-13.

[9] Jeanmart, S., Guéhéneuc, Y.-G., Sahraoui, H., and Habra, N., "Impact of the Visitor Pattern on Program Comprehension and Maintenance", in Proc. of 3rd International Symposium on Empirical Software Engineering and Measurement, Lake Buena Vista, Florida, Oct 15-16 2009, pp. 69-78.

[10] Just, M. and Carpenter, P., "A Theory of Reading: From Eye Fixations to Comprehension", *Psychological Review*, vol. 87, 1980, pp. 329-354.

[11] Kagdi, H., Yusuf, S., and Maletic, J. I., "On Using Eye Tracking in Empirical Assessment of Software Visualizations", in Proc. of ACM Workshop on Empirical Assessment of Software Engineering Languages and Technologies, Atlanta, GA, November 5, 2007, pp. 21-22.

[12] Lin, J., Lin, C. C., Cleland-Huang, J., Settimi, R., Amaya, J., Bedford, G., Berenbach, B., Khadra, O. B., Duan, C., and Zou, X., "Poirot: A Distributed Tool Supporting Enterprise-Wide Traceability", in Proc. of IEEE International Conference on Requirements Engineering (RE'06) - Tool Demo, Sept 2006.

[13] Marcus, A., Xie, X., and Poshyvanyk, D., "When and How to Visualize Traceability Links?" in Proc. of 3rd ACM International Workshop on Traceability in Emerging Forms Of Software Engineering, California, USA, 2005, pp. 56-61.

[14] Sharif, B. and Maletic, J. I., "Using Fine-Grained Differencing to Evolve Traceability Links", in Proc. of ACM International Symposium on Grand Challenges in Traceability (GCT/TEFSE), Lexington, Kentucky, March 22-23 2007, pp. 76-81.

[15] Sharif, B. and Maletic, J. I., "The Effects of Layout on Detecting the Role of Design Patterns", in Proc. of 23rd IEEE-CS International Conference on Software Engineering Education and Training (CSEE&T 2010), Carnegie Mellon University, Pittsburgh, USA, March 9-12 2010, pp. 41-48.

[16] Sharif, B. and Maletic, J. I., "An Eye tracking Study on CamelCase and Under_score Identifier Styles", in Proc. of 18th IEEE Intl. Conf. on Prog. Comp. (ICPC'10), Portugal, Jun 30-Jul 2 2010, pp. 196-205.

[17] Sharif, B. and Maletic, J. I., "An Eye tracking Study on the Effects of Layout in Understanding the Role of Design Patterns", in Proc. of 26th IEEE International Conference on Software Maintenance (ICSM'10), Timisoara, Romania, Sept 12-18 2010, pp. 1-10.

[18] Spanoudakis, G. and Zisman, A., "Software Traceability: A Roadmap", in *Handbook of Software Engineering and Knowledge Engineering*, Chang, S. K., Ed. World Scientific Publishing Co, 2005, pp. 395-428.

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

[20] Yusuf, S., Kagdi, H., and Maletic, J. I., "Assessing the Comprehension of UML Class Diagrams via Eye Tracking", in Proc. of IEEE Intl. Conf. on Prog. Comp. (ICPC'07), Banff, Jun 26-29, 2007, pp. 113-122.