iTrace: Overcoming the Limitations of Short Code Examples in Eye Tracking Experiments

Bonita Sharif
Department of Computer Science and Information Systems
Youngstown State University
Youngstown, Ohio, USA 44555
Phone: +1 330-941-1769
Fax: + 1 330-941-2284
bsharif@ysu.edu

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent, Ohio, USA 44242
Phone: +1 330-672-9039
Fax: +1 330-941-2284
jmaletic@kent.edu

***Abstract*— Eye trackers are being used by software engineering researchers to study how developers work. In this technical briefing, we give an overview of eye tracking and how it can help researchers to conduct their own studies. Eye tracking studies are done on a single screen of text and there is no support for scrolling or switching between files. This scenario is impractical to study developers as they actually work on large software artifacts. To overcome this an Eclipse plugin, iTrace, is introduced that monitors developers eye movements even in the presence of scrolling and file switching within an IDE. In addition, it automatically maps the eye gaze to source code elements. Existing work using iTrace is presented followed by a scenario of how to setup and run an eye tracking study. Data filtering, data cleaning, and data analysis are also discussed.**

***Index Terms*— eye tracking, program comprehension.**

I. OBJECTIVE

The objective of this technical briefing is to introduce researchers and practitioners to a software tool, iTrace [1], that greatly expands the ability to conduct eye tracking studies on realistically sized source code examples. Currently, eye tracking studies on software are very limited in terms of the types of stimulus that can be presented to a study subject. Eye trackers and the related support software does not allow for scrolling or switching between windows. Thus, studies are designed using only short segments of code that can fit on the screen (i.e., approximately 30 lines of code or less on a standard 24 inch display).

This limitation makes it quite difficult to study how professional programmers read, understand, and examine the software systems that evolve and maintain on a daily basis. Any non-trivial software system involves hundreds of files and many thousands of lines of code. Hence, for researchers to understand how professional developers work, we must be able to conduct eye tracking studies in the same type of environment and on the same scale of software that they ply their trade.

The technical briefing outlines the types of new investigations that can be conducted using iTrace and how the technology can be leveraged.

iTrace[1] is an Eclipse plugin that incorporates implicit eye tracking on software artifacts such as source code, test cases, bug reports or stack overflow posts. iTrace is designed to alleviate the problems associated with conducting studies on small code snippets. It is now possible to conduct realistic studies consisting of hundreds of source files and not be limited to a single static view. The tool coordinates the eye tracker and supports the switching between files and scrolling within a file. The current implementation handles gazes at the statement level for Java source code, text files including HTML and XML files and various Eclipse UI elements. It is designed such that it is easy to write new handlers for different file types and other software artifacts to collect fine-grained data at the statement level.

The three main tasks of iTrace are as follows. First, it captures a developer's gaze from an eye tracker. Second, it determines which UI element that gaze falls on and finally, processes this information towards some functional goal. The goal is dependent on each researcher's purpose for using iTrace. iTrace was first used in our study [2, 3] to understand how developers try to fix bugs in an open source system.

II. PRESENTERS' EXPERIENCE

The authors started investigating how eye tracking technology can be applied to software engineering in 2006. They have been regularly publishing papers on this topic since 2007. Maletic co-organized a half day working session on eye tracking at ICPC 2009 [4]. Additionally, Maletic has organized a number of tutorials/technical briefings on srcML[2] (i.e., ICSE 2015, MUD 2015, and ICSME 2014, Shonan Japan 2016). Sharif presented a 30 minute invited talk [5] at SEmotion 2016 on how iTrace can be used to determine developer emotion.

The authors also recently published a short invited contribution in *IEEE Software* [6] outlining how iTrace can be used to support studies of actual programmers. We now outline other work published by the presenters related to eye tracking.

Kevic et al. conducted an eye tracking study on three bug fix tasks. They found that developers focus on small parts of methods that are often related to data flow. When it comes to switches between methods, they found that developers rarely follow call graph links and mostly switch to the elements in close proximity of the method within the class [2, 3] . This is

[1] See http://seresl.csis.ysu.edu/iTrace

[2] srcML is an infrastructure for the exploration, analysis, and manipulation of source code. See www.srcML.org.

the only previous eye-tracking study done in a realistic environment namely, iTrace.

Sharif et al. [7] replicated a study by Uwano et al. [8] showing that scan time (time taken to first read through the entire code snippet) plays an important role in defect detection time and visual effort required to review source code. Moreover, experts tend to focus on defects more, while novices watch lines in a more dispersed way.

Busjahn et al. [9] show that experts read code in a less linear fashion than novices. Eye tracking studies have also been done by the authors to gauge the effectiveness of identifier style [10, 11]. Results indicate a significant improvement in time and lower visual effort with the underscore style. However, expert programmers appear to not be impacted greatly by style.

III. OUTLINE

The technical briefing will be outlined as follows.

Time Slot	Description
5 minutes	Introduction to eye tracking and current limitations
10 minutes	Introduction to iTrace Studies using iTrace
10 minutes	How to setup an eye tracking study Data filtering, cleaning, analysis Short iTrace demo
5 minutes	Q & A

IV. AUDIO-VISUAL REQUIREMENTS

This briefing will require a projector and a screen. We will bring a laptop and a portable eye tracker to illustrate how iTrace works. The larger eye tracking equipment is somewhat difficult to move and set up for a short presentation.

V. TARGET AUDIENCE AND ATTENDEE BACKGROUND

The target audience is the software engineering researcher who is interested to understand how eye tracking works for software engineering studies. The typical attendee would be interested in how they could use eye tracking for their own purposes. This could also attract industry professionals interested in understanding how their developers actually use the IDE.

No prior knowledge of eye tracking is required. Some knowledge of conducting empirical studies with humans is a plus but not required.

VI. PRESENTERS' BIOGRAPHIES

Bonita Sharif, Ph.D. is an Assistant Professor in the Department of Computer Science at Youngstown State University, Youngstown, Ohio USA. Prior to this position, she was an adjunct Assistant Professor at Ohio University. She received her Ph.D. in 2010 and MS in 2003 in Computer Science from Kent State University, U.S.A and B.S. in Computer Science from Cyprus College, Nicosia Cyprus. Her research interests are in eye tracking related to software engineering, empirical software engineering, software traceability, and software visualization to support maintenance of large systems. She has authored over 30 refereed publications. She serves on numerous program committees including ICSME, VISSOFT, SANER and ICPC. She is serving as general chair of VISSOFT 2016 and ETRA 2018 and served as program chair for VISSOFT 2014 and OCWIC 2017. Sharif is also a recent recipient of the NSF CAREER award.

Jonathan I. Maletic, Ph.D. is Professor in the Department of Computer Science at Kent State University, Kent Ohio USA. He received the Ph.D. and M.S., both in Computer Science, from Wayne State University in 1995 and 1989 respectively. He received the B.S. in Computer Science in 1986 from The University of Michigan-Flint. His research interests are centered on software evolution, with a focus on the comprehension, analysis, manipulation, transformation, reverse engineering, traceability, and visualization of large-scale software systems. Prof. Maletic has authored over 115 refereed publications and two of his publications have received Most Influential Paper Awards. Prof. Maletic is regularly funded by the US National Science Foundation (NSF) and regularly serves on numerous program committees. He was program chair of ICPC 2016 and of ICSM 2012. Prof. Maletic has graduated 14 doctoral students, 12 of whom hold an academic position.

ACKNOWLEDGMENTS

This work is supported in part by a grant from the US National Science Foundation CCF 15-53573.

REFERENCES

[1] T. Shaffer, J. Wise, B. Walters, S. Müller, M. Falcone, and B. Sharif., "iTrace: Enabling Eye Tracking on Software Artifacts Within the IDE to Support Software Engineering Tasks," in *10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Tool Track, Bergamo, Italy, 2015, pp. 954-957.

[2] K. Kevic, B. Walters, T. Shaffer, B. Sharif, D. Shepherd, and T. Fritz, "Tracing Software Developers' Eyes and Interactions for Change Tasks," in *ESEC/FSE 2015*, Bergamo, Italy, 2015, pp. 202-213.

[3] K. Kevic, B. Walters, T. Shaffer, B. Sharif, D. Shepherd, and T. Fritz, "Eye Gaze and Interaction Context for Change Tasks - Observations and Potential," *Journal of Systems and Software,* p. in press, 2016.

[4] Y. G. Guéhéneuc, Kagdi, H., Maletic, J., "Working Session: Using Eye-Tracking to Understand Program Comprehension," in *6th IEEE International Conference on Program Comprehension (ICPC'09)*, Vancouver, BC, Canada, 2009, pp. 278-279.

[5] J. Wise, B. Prox, B. Clark, and B. Sharif, "Towards an emotionally aware development environment: invited talk," in *1st International Workshop on Emotion*

Awareness in Software Engineering (SEmotion), Austin, TX, 2016, pp. 26-27.

[6] B. Sharif, T. Shaffer, J. Wise, and J. I. Maletic. (2016, May-June 2016) Tracking Developers' Eyes in the IDE. *IEEE Software*. 105-108.

[7] B. Sharif, M. Falcone, and J. I. Maletic, "An eye-tracking study on the role of scan time in finding source code defects," in *the Symposium on Eye Tracking Research and Applications (ETRA)*, New York, NY, USA,, 2012, pp. 381-384.

[8] H. Uwano, M. Nakamura, A. Monden, and K.-i. Matsumoto, "Analyzing individual performance of source code review using reviewers' eye movement," in *the 2006 symposium on Eye tracking research & applications (ETRA)*, New York, NY, USA, 2006, pp. 133-140.

[9] T. Busjahn, R. Bednarik, A. Begel, M. E. Crosby, J. Paterson, C. Schulte, B. Sharif *et al.*, "Eye Movements in Code Reading: Relaxing the Linear Order," in *23rd International Conference on Program Comprehension (ICPC)*, Florence, Italy, 2015, pp. 255-265

[10] B. Sharif and J. I. Maletic, "An eye tracking study on camelcase and under score identifier styles.," in *the 2010 IEEE 18th International Conference on Program Comprehension (ICPC)*, Washington, DC, USA, 2010, pp. 196-205.

[11] D. Binkley, M. Davis, D. Lawrie, J. I. Maletic, C. Morrell, and B. Sharif, "The impact of identifier style on effort and comprehension," *Empir. Software Eng.,* vol. 18, pp. 219–276, 2013.