Conducting Eye Tracking Studies in Software Engineering - Methodology and Pipeline

Bonita Sharif
School of Computing
University of Nebraska - Lincoln
Lincoln, NE USA
bsharif@unl.edu

Andrew Begel
Software and Societal Systems Department
Carnegie Mellon University
Pittsburg, PA USA
abegel@cmu.edu

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent, OH USA
jmaletic@kent.edu

***Abstract*—This ICSE 2023 technical briefing is on state-of-the-art techniques to conduct eye tracking studies in software engineering. It is organized as a hands-on 180-minute briefing broken up into two 85-minute modules with a short break in between. The first module will teach participants the terminology and theories needed to understand eye tracking. The second will engage participants in hands-on groupwork to collect and analyze eye tracking data through a software pipeline. Our goal is to help participants one-on-one learn how to get starting using eye tracking to support their own research goals. Our team has been working with eye tracking for over 15 years. The briefing will be targeted towards researchers, practitioners, and educators. Our eye tracking software infrastructure, iTrace, will be demonstrated in person with several state-of-the-art eye trackers. Eye tracking is gaining a lot of traction in the community. We want to use the ICSE platform to communicate the current state-of-the-art (including limitations and workarounds) in a highly interactive setting starting from the theory, data collection, and processing pipeline. Sample data and scripts will be made available.**

***Index Terms*—eye-tracking, empirical studies, program comprehension**

I. Introduction and Relevance

Eye trackers are used by software engineering researchers to study how developers read and comprehend code. Research on mental models of program comprehension dates back to the 1980s [1]–[6]. Prior to the popularization of eye tracking, researchers used think-aloud, pre/post surveys, and interviews to collect data for studies. Since the cost of eye trackers has been dropping recently, many more researchers are taking advantage of eye tracking technology to study how people read source code [7]. Eye trackers are a vital research tool to really understand how developers read and comprehend visual stimuli [8], especially source code.

Accurate research-grade eye trackers operate only on fixed stimuli (i.e., an image or text) that fits on a single screen. Changes to the stimuli (e.g., a code editor) caused by scrolling or switching to another file are difficult to handle. Mapping eye positions (x, y) to their correct location in a stimulus (e.g., a 2,000-line source code file) is not simple. Fortunately, we have developed the iTrace infrastructure to deal with this problem [9]–[12]. iTrace (www.i-trace.org) allows a software engineering researcher to conduct eye tracking studies directly in an integrated development environment (IDE) such as Atom, Visual Studio or Eclipse. It supports the eye tracking in the presence of scrolling and context switching. iTrace links to the IDE via a plugin architecture and invokes application and system calls to map screen x/y-coordinates to lines and columns in the editor window in real time. Afterwards, the lines and columns are mapped to source code tokens. It has been shown in prior work that eye tracking studies on short code snippets [13] are not comparable to studies of real open-source systems [14]. It is crucial to ensure we are able to conduct studies as close as possible to the real work environment of developers in the field to keep our external validity high. iTrace recently added support for high-speed trackers for more advanced analysis [15].

iTrace is relevant to a broad range of stakeholders, including researchers, educators, and practitioners. Researchers can study developers in a real-world environment using large realistic software systems. Industry practitioners can use it to observe their work practice and monitor improvements. Educators can gain insights into how students read and debug code to better teach novices to program.

In this technical briefing, we give an overview of eye tracking and show researchers how to conduct their own eye tracking studies in software engineering. We present practical guidelines through hands-on activities. We strongly believe that interdisciplinary technical briefings such as this help researchers from other fields become more knowledgeable about cutting-edge research, which in turn help educate and advance our discipline.

II. iTrace – Commmunity Eye Tracking Infrastructure

iTrace is an eye tracking infrastructure [10], [15] package that enables research studies within multiple types of software development environments. It was designed and built to support the software engineering community in conducting eye tracking experiments seamlessly within realistic developer environments i.e., IDEs. Its design is modular and features three key components: iTrace Core, iTrace Plugins, and an offline post-processing application for gaze analysis called iTrace Toolkit. iTrace Core offers a unified interface for managing supported eye tracking devices. All data generated by the eye trackers is first received by iTrace Core, which then makes quick decisions based on validity indicators whether

the data is acceptable for use by other iTrace infrastructure applications (plugins). iTrace Core also provides socket and websocket servers to enable iTrace plugins to receive gaze data for additional processing.

III. Target Audience and Attendee Background

Our target audience is the software engineering researcher, practitioner, and educator 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 people is a plus, but not required.

IV. Technical Briefing Format and Outline

The briefing is a hands-on, tool-supported session on how to conduct eye tracking studies in software engineering. We will cover both the methodology and the pipeline from start to finish. It will be held for 180 minutes in two modules of 85-minute sessions with a short break in between. The first module will teach theory and methods. The second will consist of two, hands-on exercises. The first exercise will collect eye tracking data for a sample study. The second will demonstrate how to process that data for further analysis. Several portable eye trackers will be available to illustrate how iTrace works. An outline for each of the sessions is shown in Table I.

TABLE I
Session Outline

Module 1 - 85 minutes	
Time	**Topics Covered**
10 mins	Introduction to eye tracking, terminology, and current limitations
20 mins	Attention, mental models, cognitive load and borrrowing from psychology
10 mins	Introduction to iTrace
15 mins	Equipment, Areas of Interest, Eye tracking Metrics, and Fixation Detection
15 mins	Studies Using iTrace
10 mins	Practical Guidelines
5 mins	Q&A

Module 2 - 85 minutes	
Time	**Topics Covered**
30 mins	Exercise 1 (data collection on a sample Java open source application)
30 mins	Exercise 2 (processing of data collected from Exercise 1 individually and/or in groups)
15 mins	Hands-on live demo on a separate station to try other plugins.
10 mins	Wrap-up and Conclusion

V. Acknowledgments

This work has been partly funded by the US NSF under Grant Numbers CNS 17-30181, CNS 18-55753, and CCF 18-55756

References

[1] R. Brooks, "Towards a theory of the comprehension of computer programs," *International Journal of Man-Machine Studies*, vol. 18, no. 6, pp. 543–554, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020737383800315

[2] S. Letovsky, "Cognitive processes in program comprehension," *Journal of Systems and Software*, vol. 7, no. 4, pp. 325–339, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/016412128790032X

[3] R. S. Rist, "Plans in programming: Definition, demonstration, and development," in *Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*. USA: Ablex Publishing Corp., 1986, p. 28–47.

[4] E. Soloway and K. Ehrlich, "Empirical studies of programming knowledge," *Software Engineering, IEEE Transactions on*, vol. SE-10, pp. 595 – 609, 10 1984.

[5] N. Pennington, "Stimulus structures and mental representations in expert comprehension of computer programs," *Cognitive Psychology*, vol. 19, no. 3, pp. 295–341, 1987. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0010028587900077

[6] A. Von Mayrhauser and A. Vans, "Program comprehension during software maintenance and evolution," *Computer*, vol. 28, no. 8, pp. 44–55, 1995.

[7] U. Obaidellah, M. Al Haek, and P. C.-H. Cheng, "A survey on the usage of eye-tracking in computer programming," *ACM Comput. Surv.*, vol. 51, no. 1, pp. 5:1–5:58, Jan. 2018. [Online]. Available: http://doi.acm.org/10.1145/3145904

[8] K. Rayner, "Eye movements in reading and information processing," *Psychological Bulletin*, vol. 85, no. 3, pp. 618–660, 1978.

[9] B. Sharif, C. Peterson, D. Guarnera, C. Bryant, Z. Buchanan, V. Zyrianov, and J. Maletic, "Practical eye tracking with itrace," in *2019 IEEE/ACM 6th International Workshop on Eye Movements in Programming (EMIP)*, 2019, pp. 41–42.

[10] D. T. Guarnera, C. A. Bryant, A. Mishra, J. I. Maletic, and B. Sharif, "itrace: eye tracking infrastructure for development environments," in *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*. ACM, 2018, p. 105.

[11] B. Sharif and J. I. Maletic, "itrace: Overcoming the limitations of short code examples in eye tracking experiments," in *2016 IEEE International Conference on Software Maintenance and Evolution, ICSME 2016, Raleigh, NC, USA, October 2-7, 2016*. IEEE Computer Society, 2016, p. 647. [Online]. Available: https://doi.org/10.1109/ICSME.2016.61

[12] B. Sharif, T. Shaffer, J. L. Wise, and J. I. Maletic, "Tracking developers' eyes in the IDE," *IEEE Softw.*, vol. 33, no. 3, pp. 105–108, 2016. [Online]. Available: https://doi.org/10.1109/MS.2016.84

[13] P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 390–401. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568247

[14] K. Kevic, B. M. Walters, T. R. Shaffer, B. Sharif, T. Fritz, and D. C. Shepherd, "Tracing software developers eyes and interactions for change tasks," *Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2015.

[15] V. Zyrianov, C. S. Peterson, D. T. Guarnera, J. Behler, P. Weston, B. Sharif, and J. I. Maletic, "Deja vu: semantics-aware recording and replay of high-speed eye tracking and interaction data to support cognitive studies of software engineering tasks - methodology and analyses," *Empir. Softw. Eng.*, vol. 27, no. 7, p. 168, 2022. [Online]. Available: https://doi.org/10.1007/s10664-022-10209-3