Working Session: Using Eye-Tracking to Understand Program Comprehension

Yann-Gaël Guéhéneuc
Ptidej Team – DGIGL
École Polytechnique de Montréal
Montreal, Quebec, Canada
guehene@iro.umontreal.ca

Huzefa Kagdi
Department of Computer Science
Missouri University of Science and Technology
Rolla, MO 65401
kagdih@mst.edu

Jonathan I. Maletic
Department of Computer Science
Kent State University
Kent Ohio 44242
jmaletic@cs.kent.edu

Abstract

The working session focuses on the use of eye-tracking technology to assess, understand, and evaluate tools and techniques for program comprehension. An introduction to the technology and tools of eye-tracking will be presented. A discussion of how these tools augment existing evaluation mechanism in the context of program comprehension will follow. Research directions and open problems will be a main topic.

1. Introduction

Eye-tracking technology (*i.e.*, techniques and tools) has been successfully used for decades in psychology, neuropsychology, and cognitive science to study the human cognitive processes while performing various tasks such as reading, counting characters in an image, and solving puzzles [2, 3]. It has also been widely used to assess user interface and Web page designs.

However, not until very recently did eye-tracking surface in the software engineering community to unveil the developers' cognitive processes while performing program comprehension tasks, such as dealing with UML diagrams or reading source code.

Eye-tracking was recently used to evaluate and assess methods and tools in the context of program comprehension. For example, to show the impact of UML stereotypes [9], the (yet to be confirmed) lack of impact of binary class relationships [4]. Eye-trackers can also be used to assess visualization techniques, compare coloring/fonts choices, reduce the developers' efforts when using new development environment, and so on.

Eye tracking tools collect eye movement data to provide an insight into a subject's focus of attention, making it possible to draw conclusions about the underlying cognitive processes. These systems are based on the physiology of human visual capabilities and cognitive theories, like the theories on visual attention and visual perception [3].

The arrival of eye-tracking in software engineering can be attributed, in great part, to a number of recent advancements in eye-tracking technology: high quality, accurate, and user-friendly tools are available today. Most importantly, they have the capability to collect a subject's eye gazes in a non-obtrusive manner, as opposed to the clumsy head mounted devices. This accurate data can then be used for understanding the cognitive process involved in the processing of visual data [1, 2, 5].

Eye-tracing offers a unique prospective by enabling the measurement of various eye movements, which could provide a much valuable insight into how and why subjects arrive at a certain solution for a given task. These measures add a new additional dimension in assessing a tool's claim of supporting software comprehension tasks. These measures are different from traditionally prescribed measures such as the accuracy/level of the responses and time needed usually collected in program comprehension studies [7, 8].

These measures are used directly or indirectly to draw conclusions and/or meet other objectives of the performed study. A wide majority of these traditional measures are collected retrospectively. For example, human subjects are asked to report their final answers on the completion of a given task and their response time is recorded. This type of approach is subjected to some potential threats; *e.g.*, a subject may forget to report (or misreport) an observation after a lengthy task. Alternatively, subjects could be asked to note their observations while working towards their answers, albeit at the potential risk of obtrusiveness and distraction.

The working session will focus on using eye-tracking for understanding program comprehension. An introduction to the technology and tools will be made along with examples of current research. The main objective of the session is to inform researchers of the potential uses of eye-tracking technology to better understand how people comprehend software.

2. Eye-Tracking Technology

The fundamental design of eye-tracking equipment is based on the physiology of the human visual capability [3, 6]. These systems use cameras to track eye movements. There are a number of vendors that supply eye-tracker to capture eye movements and collect eye gaze data (e.g., www.sr-research.com, www.tobii.se, etc.). For example, in the equipment available from *Tobii*, two cameras used to track the eye are built into a flat-panel screen. No restraints such as wearing a headband or goggles are placed on the human subjects.

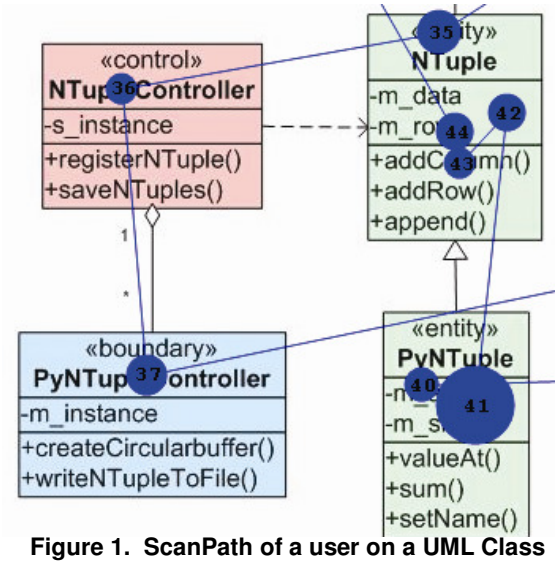

Figure 1. ScanPath of a user on a UML Class Diagram. Fixations are represented by the circles and saccades by the lines connecting the circles.

Moreover, this new equipment is very accurate and boasts error rates of less than 0.5 degrees and sampling rates of around 100Hz. Software that records the XY screen coordinates of eye gazes and supports analysis of eye movements is also provided along with the eye-tracker system. An audio/video recording is also made of each study session.

The underlying basis is to capture various types of eye movements that occur while humans physically gaze at an object of interest. Among these, fixations and saccades are the two most widely used eye movements in these types of studies. *Fixation* is the stabilization of eyes on an object of interest for a period of time. *Saccades* are quick movements of the eyes from one location to the next (i.e., refixates). *Scanpath* is a directed path formed by saccades between fixations.

The general consensus in the eye tracking research community is that the processing of visualized information occurs during fixations, whereas, no such processing occurs during saccades [6]. Humans use saccades to locate interesting parts in a visual scene to form a mental model.

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

3. References

[1] Bednarik, R. and Tukiainen, M., "An Eye-Tracking Methodology for Characterizing Program Comprehension Processes", in *Proc. Symposium on Eye tracking research & applications (ETRA)*, San Diego, 2006, pp. 125-132.

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

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

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

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

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

[7] Purchase, H., C., Colpoys, L., McGill, M., Carrington, D., and Britton, C., "UML class diagram syntax: an empirical study of comprehension", in *Proceedings 2001 Asia-Pacific symposium on Information visualisation - Volume 9*, Sydney, Australia, 2001, pp. 113-120.

[8] Ricca, F., Penta, M. D., Torchiano, M., Tonella, P., and Ceccato, M., "The Role of Experience and Ability in Comprehension Tasks Supported by UML Stereotypes", in *Proceedings 29th International Conference on Software Engineering (ICSE'01)*, 2007, pp. 375-384.

[9] Yusuf, S., Kagdi, H., and Maletic, J. I., "Assessing the Comprehension of UML Diagrams via Eye Tracking", in *Proc. 15th IEEE Int. Conference on Program Comprehension (ICPC'07)*, Banff Canada, June 26-29 2007, pp. 113-122.