Can Social Screencasting Help Developers Learn New Tools?

Kaylee Lubick*, Titus Barik†, Emerson Murphy-Hill*
*North Carolina State University, Raleigh, North Carolina USA
†ABB Corporate Research, Raleigh, North Carolina, USA
kjlubick@ncsu.edu, titus.barik@us.abb.com, emerson@csc.ncsu.edu

***Abstract*—An effective way to learn about software development tools is by directly observing peers' workflows. However, these *tool knowledge transfer* events happen infrequently because developers must be both colocated and available. We explore an online social screencasting system that removes the dependencies of colocation and availability while maintaining the beneficial tool knowledge transfer of *peer observation*. Our results from a formative study indicate these online observations happen more frequently than in-person observations, but their effects are only temporary. We conclude that while peer observation facilitates online knowledge transfer, it is not the only component — other social factors may be involved.**

I. Introduction

Software developers obtain knowledge about their trade in a variety of ways. These include reading books, browsing sites like StackOverflow, and socially, by directly observing peers' workflows. This last way, known as *peer observation*, has been shown to be particularly effective at transferring knowledge of functionality within development applications, such as Eclipse or NetBeans [1], [2]. These functionalities, or more generally, *tools*, can be as mundane as Paste or as sophisticated as Introduce Parameter Object. For example, while pair programming, developer Jim sees his partner quickly jump to an unopened file using a keybinding. Astonished, Jim asks "How did you do that?" and learns the tool Open Resource.

Despite peer observation being effective at this *tool knowledge transfer*, such events might only happen once per month for a typical software developer [2]. Peer observation is typically impeded because it requires developers to be available in the same place at the same time to work together.

We hypothesized that it would be possible to overcome these availability and colocation requirements with an online, asynchronous environment that facilitates tool knowledge transfer like in-person peer observation. Our approach, *Continuous Social Screencasting* [3], extends traditional software development by continuously and automatically recording video clips, called *screencasts*, of interesting situations [4]. In our system, these screencasts are recorded anytime a developer uses a tool. The developers can share these screencasts with their peers to show them how they use the tool. Further, to take advantage of social connections, our system is intended to be used by groups of people who work with each other.

To evaluate our approach, we conducted a formative, seven-person study of a prototype social screencasting system. This paper describes a preliminary analysis of this study which explores the effect of social screencasts on tool knowledge transfer. The participants had tool usage habits where even experts could learn from their peers. We observed that five participants were able to learn about a new tool from their peer's screencasts. However, we were unable to identify significant changes in their tool usage behavior at the end of the study. We conclude that while social influences may facilitate tool knowledge transfer in an in-person setting, they do not guarantee transfer in an online environment.

II. Field Study Methodology

The seven participants in our field study were members of the authors' software engineering research lab which included the authors of this paper. Participants have been given pseudonyms in the following discussion. We installed a prototype social screencasting system on their work computers and told participants to work normally as we collected data for the following four-week period. Our system supported automatically generating screencasts for Gmail, Eclipse, and Excel, three applications the participants used frequently. Gmail is a particularly interesting case as it is a general purpose application, yet e-mail is used extensively by developers as a hub for software-related activities such as code reviews, bug reports, and feature discussions [5].

To identify any changes in tool-related behavior introduced by social screencasting, we disabled screencast sharing for the first week. After enabling sharing, we sent out an email to the participants along with a video demonstrating how to use the web interface to request and share screencasts with their peers. At the end of the study, we conducted a semi-structured interview with each participant to allow them to share their own experiences with the system.

III. Results

Unlike in-person peer observation, which occurs about once per month [2], we observed 149 actions, spread across three weeks, associated with requesting and viewing screencasts. This averages to about one interaction per participant per day, a considerable increase.

Five participants reported learning about one or more tools while using the social screencasting system. For example, Keith discovered the keyboard shortcut for Go to Inbox from Phil, and Phil learned the tool Archive Email and Go to Next from Bryant.

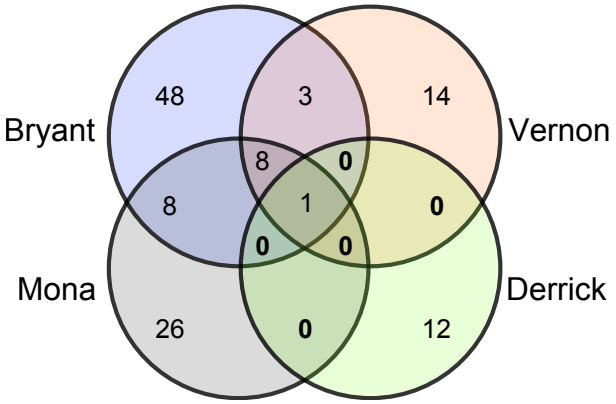

Fig. 1: A Venn diagram of four participants' use of Eclipse tools. A majority of the tools are not used by more than one person. For example, Mona uses 43 tools; she is the sole user of 26 of them. Only one tool is used by all four participants.

Because the participants were from the same lab, and had similar responsibilities, we expected them to use mostly the same tools. Surprisingly, this was not the case, even across applications. For example, consider the Venn diagram of the four participants who used Eclipse over the course of the study, shown in Figure 1. Each circle represents the set of tools used by an individual; intersections represent tools used by more than one person.[1] Only a handful of common tools (e.g. SAVE, PASTE) were used by more than one person, the rest seem to be more complicated or potentially obscure — indicating there are potentially many opportunities for tool transfer, even among experts.

The data we captured showed five participants tried out new tools at least once. However, the new tools were used only a few times before returning to the old habits. After Phil investigated ARCHIVE EMAIL AND GO TO NEXT by requesting Bryant's screencast, he used it once on the fifteenth day of the study and then never again. Keith and Joey showed similar behaviors, when they used their newly learned keyboard shortcuts a few times, but then returning to the mouse-based invocation after a few days.

IV. CHALLENGES

We just saw that participants did not radically change their behavior as a result of using our social screencasting system. To explain why, consider Phil's post-study interview, where he said, "It's not that I don't want to be brought to the next email, I just don't use [ARCHIVE EMAIL AND GO TO NEXT]. I have pretty good muscle memory for [my old pattern], which is hard to change." Mona and Joey had similar remarks. These three experiences are consistent with the *active user paradox*, which states that even expert computer users are likely to continue with their familiar, inefficient tool habits, even if there is a demonstrably more efficient way to do the same job [6].

[1] A constraint in this visual representation is that it elides two intersections: between Mona and Vernon (0), and Bryant and Derrick (1). Excel ($n = 2$) and Gmail ($n = 7$) had similar usage characteristics to Eclipse (not shown).

We know developers learn tools from their coworkers and eventually incorporate them into their work habits [1], [7], but why did we not observe this to be the case when participants used social screencasting? It appears our implemented social influences are not enough to overcome the active user paradox, and other factors may be at play.

Perhaps the participants simply forgot the keyboard shortcuts they learned. Indeed, Krisler and Alterman found some success by constantly reminding users of new keyboard shortcuts with HotKeyCoach [8]. However, our evidence suggests only two participants would have benefited from this approach. For the other three participants, the tools they learned were more complex than keyboard shortcuts.

Another explanation is that there is some social pressure that comes with peer observation that positively influences the tool knowledge transfer, which is not present in our system. For example, Deutsch and Gerard showed that tasks can be influenced by merely having another person in the room [9]. If this is the case, it is not obvious how to translate these social influences to a workplace software setting in order to more effectively facilitate tool knowledge transfer.

V. CONCLUSION

We hypothesized that social screencasting would facilitate the transfer of tool usages in a way that was as effective as in-person peer observation, without the need for colocation and availability. Though we saw a lot of potential for learning new tools and observed several cases of tool knowledge transfer, the effects ended up being temporary. We suspect this to be due to social nuances of the tool knowledge transfer process. We are optimistic that these nuances can be identified and emulated to fully realize the potential of social screencasting.

VI. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under grant number 1252995.

REFERENCES

[1] M. B. Twidale, "Over the shoulder learning: Supporting brief informal learning," *CSCW 2005*, vol. 14, no. 6, pp. 505–547, 2005.

[2] E. Murphy-Hill and G. C. Murphy, "Peer interaction effectively, yet infrequently, enables programmers to discover new tools," in *CSCW 2011*. ACM, 2011, pp. 405–414.

[3] E. Murphy-Hill, "Continuous social screencasting to facilitate software tool discovery," in *ICSE 2012*. IEEE, 2012, pp. 1317–1320.

[4] M. Blum, A. Pentland, and G. Troster, "Insense: Interest-based life logging," *IEEE MultiMedia*, vol. 13, no. 4, pp. 40–48, 2006.

[5] A. Guzzi, A. Bacchelli, M. Lanza, M. Pinzger, and A. v. Deursen, "Communication in open source software development mailing lists," in *MSR 2013*. IEEE, 2013, pp. 277–286.

[6] W.-T. Fu and W. D. Gray, "Resolving the paradox of the active user: Stable suboptimal performance in interactive tasks," *Cognitive Science*, vol. 28, no. 6, pp. 901–935, 2004.

[7] D. L. Jones and S. D. Fleming, "What use is a backseat driver? A qualitative investigation of pair programming," in *VL/HCC 2013*. IEEE, 2013, pp. 103–110.

[8] B. Krisler and R. Alterman, "Training towards mastery: overcoming the active user paradox," in *NordiCHI 2008*. ACM, 2008, pp. 239–248.

[9] M. Deutsch and H. B. Gerard, "A study of normative and informational social influences upon individual judgment." *The Journal of Abnormal and Social Psychology*, vol. 51, no. 3, p. 629, 1955.