S. Becker, I. Bogicevic, G. Herzwurm, S. Wagner (Hrsg.): SE/SWM 2019,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 85

Understanding How Programmers Forget

Jacob Krüger[1,2], Jens Wiemann[2], Wolfram Fenske[2], Gunter Saake[2], Thomas Leich[1,3]

Abstract: This extended abstract is based on our paper "Do You Remember This Source Code?", published at the International Conference on Software Engineering 2018 [Kr18]. We summarize and discuss our results on programmers' memory and forgetting. To this end, we focus on reverse engineering of software, which was the primary context in which we conducted this work.

Keywords: Familiarity; forgetting; empirical study; maintenance; program comprehension

During software development, programmers rely on their knowledge about a system. More precisely, software familiarity comprises knowledge about the system's source code, design, architecture, and usage. Consequently, being familiar with a system is essential for various aspects of software development, such as task performance, program comprehension, bug fixing, maintenance, and re-engineering. Thus, researchers and practitioners consider familiarity as an important factor in techniques for cost modeling, expertise identification, and task assignments. Still, empirical analyses of software familiarity are sparse and these techniques usually rely on indirect heuristics or user estimates to consider forgetting.

Within our previous paper [Kr18], we report an empirical study with 60 open-source developers. We investigated whether an established forgetting model applies to software familiarity, analyzed three factors that may influence the memory, and computed a memory strength for our participants. In Figure 1, we display the familiarity of our participants, relating the times since their last commit to their subjective familiarity. Each circle represents a single developer and its size the number of commits the developer performed on the selected file. The solid blue line represents the average familiarity the developers stated, while the red dashed line represents the average only considering the data for single commits.

We can see that, besides the time, repeated commits to a file seem to significantly influence the familiarity of a developer. This is indicated by the peak of the average that appears at around 120 days after the last commit. In order to test this observation and other hypotheses, we performed two significance tests for four factors: The code size of the file; for which we tested that it does not have any impact, which we assumed based on the used forgetting curve; the number of repeated commits, the ratio of own code, and the tracking behavior of a developer. Our results indicate that *repetitions as well as the ratio of own code are significantly positively correlated to familiarity*. Furthermore, we computed the average

[1] Harz University of Applied Sciences Wernigerode; tleich@hs-harz.de
[2] Otto-von-Guericke-University Magdeburg; jkrueger@ovgu.de; wfenske@ovgu.de; saake@ovgu.de
[3] METOP GmbH Magdeburg

doi:10.18420/se2019-23

memory strengths to compare the used forgetting curve to our participants' responses. The dashed red line that we show in Figure 1 resembles this curve and matches with our results: We found that the curve is only applicable if a developer edited a file only once, as this mitigates the effects of repetitions and changes that others may perform. Thus, we conclude that *we require adapted forgetting curves for software development* that consider the identified factors that correlate to forgetting. These curves and factors can then be considered in software engineering techniques—based on further analyses of forgetting.

We conducted this research in the context of re-engineering software artifacts and estimating the efforts to perform this task. During re-engineering, software developers have to comprehend the code they aim to refactor and migrate. As program comprehension is one of the main and most costly activities in software development, the effort of re-engineering is heavily dependent on the remaining familiarity a developer may have. The question arises, whether it is more suitable to assign the original developer who implemented most code (ratio of own code) or the developer how did most (repetitions) or more recent (time) changes? Answering this and related questions is challenging and often based on educated guesses. In our research, we are concerned with understanding the impact of forgetting especially on re-engineering tasks, supporting decision making, improving cost estimations, and facilitating expert identification. Thus, our results help researchers to scope new techniques and practitioners in making reasonable decisions. We aim to conduct more studies to verify and extent our findings in future work, focusing on the impact of familiarity on development activities and utilizing implicit and explicit knowledge, for example, in version control systems, to support developers.

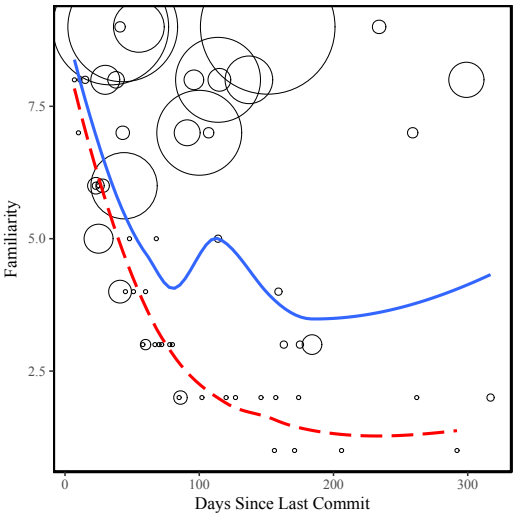

Fig. 1: Participants' familiarity related to the time since the last commit. The size of a circle represents the number of commits, the solid blue line average values, and the dotted red line the average of responses with only one commit.

Acknowledgments: Supported by DFG grants LE 3382/2-1 and SA 465/49-1.

References

[Kr18] Krüger, Jacob; Wiemann, Jens; Fenske, Wolfram; Saake, Gunter; Leich, Thomas: Do You Remember This Source Code? In: International Conference on Software Engineering. ICSE. ACM, pp. 764–775, 2018.