Empirical Evaluation of the Effects of Experience on Code Quality and Programmer Productivity: An Exploratory Study

Journal-First Selected Article – Extended Abstract

Oscar Dieste
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, Spain
odieste@fi.upm.es

Alejandrina Aranda
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, Spain
alearanda@gmail.com

Fernando Uyaguari
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, Spain
fuyaguari01@gmail.com

Burak Turhan
Brunel University London
Uxbridge, Middlesex, UK
burak.turhan@brunel.ac.uk

Ayse Tosun
Istanbul Technical University
Maslak Istanbul, Turkey
tosunmisirli@itu.edu.tr

Davide Fucci
Universität Hamburg
Hamburg, Germany
fucci@informatik.uni-hamburg.de

Markku Oivo
University of Oulu
Oulu, Finland
Markku.Oivo@oulu.fi

Natalia Juristo
Universidad Politécnica de Madrid
Boadilla del Monte, Madrid, Spain
natalia@fi.upm.es

ABSTRACT

This extended abstract summarizes an article, which has been published in the Empirical Software Engineering Journal and was selected for the *Journal-First* presentations at the *International Conference on Software and System Process* (ICSSP 2018).

Full Article Reference. Oscar Dieste, Alejandrina M. Aranda, Fernando Uyaguari, Burak Turhan, Ayse Tosun, Davide Fucci, Markku Oivo, and Natalia Juristo. Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study. Empirical Software Engineering 22, 5 (01 Oct 2017), 2457–2542. https://doi.org/10.1007/s10664–016–9471–3

CCS CONCEPTS

• **General and reference** → **Empirical studies**; • **Social and professional topics** → **User characteristics**;

KEYWORDS

Experience, industry, academy, programming, iterative test-last development, external quality, productivity, performance

ACM Reference Format:
Oscar Dieste, Alejandrina Aranda, Fernando Uyaguari, Burak Turhan, Ayse Tosun, Davide Fucci, Markku Oivo, and Natalia Juristo. 2018. Empirical Evaluation of the Effects of Experience on Code Quality and Programmer Productivity: An Exploratory Study: Journal-First Selected Article – Extended Abstract. In *ICSSP '18: International Conference on the Software and Systems Process 2018 (ICSSP '18), May 26–27, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxx

ICSSP '18, May 26–27, 2018, Gothenburg, Sweden

ACM ISBN xxx-x-xxxx-xxxx-x/xx/xx.
https://doi.org/xx.xxxx/xxxxxxx.xxxxxx

1 SUMMARY

1.1 Context

Experience has been studied in almost all areas of SE, particularly in programming. Almost universally, experts outperform novices in a diversity of tasks and situations.

However, cases have been reported where experience does not have a positive influence. For instance, Sheppard et al. found that the years of programming experience do not correlate with performance in a program reconstruction task [7]. Müller and Padberg reported that there is no correlation between the experience level and the implementation time when subjects perform a coding task in Test-Driven Development [6]. Similar results have been obtained in other areas of SE [5, 8], and other sciences as well. Performance has even been found to drop as experience increases [2]. This should not come as a surprise. Surely everyone can think of someone that they know who has a lot of experience but is a poor performer.

There is a lot of uncertainty surrounding whether experience is associated with better performance. As regards programming, this uncertainty is especially worrying because: (1) programming, together with testing, are probably the most important activities in software development, and (2) experience is one of the key variables used by employers to hire programmers.

1.2 Research question

Is the performance of expert programmers (i.e., with longer periods of service) superior than that of novice programmers?

1.3 Method

We have conducted 10 quasi-experiments. The experimental task was to apply Iterative Test-Last Development on two experimental problems and then measure external code quality and programmer productivity. The measurement procedure involved executing test suites on the code written by the subjects. Subject information was obtained using a questionnaire. We did not characterize subjects

as "students" or "professionals"; in turn, we asked subjects about their programming and unit testing experience, as well as their academic credentials. Some students combined work with studying; to account for this situation, we collected the programming experience gained in industry and academia independently. We also gathered information about the setting, such as the type of site (industry, academia) and the IDE and unit testing framework used in the quasi-experiment.

A key characteristic of this research is that both dependent and independent variables are measured directly. Existing research usually presents some task(s) to "expert" and "novice" subjects, and some facet of the problem solving process (e.g., the top-down or bottom-up programming strategy) is associated to expert behaviour on theoretical grounds. However, expert behaviour does not equate to expert performance. Recent research [4] emphasizes the need of explicitly measuring expert performance, instead of relying on (apparent) expert behaviour. In fact, one of the two existing studies reporting negative results [6] uses direct measures.

We collected data at four companies and three universities from a total of 115 programmers with a range of experiences exceeding 10 years in a substantial number of cases. The inclusion of professional programmers currently working in industry is another key characteristic of this research, as many earlier studies were conducted without access to real programmers [9].

The analysis was performed using Multiple Linear Regression (MLR). The dataset is big enough to detect medium effects with a statistical power close to 80%. In addition to the MLR, we used decision trees to explore nonlinear effects.

1.4 Results

The studies conducted in industry yield higher quality/productivity scores than those conducted in academia. Likewise, subjects with a degree in Computer Science achieve higher quality/productivity. However, programming experience gained in industry does not appear to have any effect whatsoever on quality and productivity. In other words: people working in industry achieve higher performance *independently* of their experience *in industry*.

However, the impact of the programming experience gained in academia is considerable. In terms of percentages, each training year adds around 4% increment in both quality and productivity, i.e., 3 years of programming experience gained by subjects during their bachelor (a reasonable assumption) implies that the code contains 12% less errors (in average). In terms of Cohen's effect size, these values represent a medium effect size for quality ($d = .59$) and a large effect size for productivity ($d = .84$).

Three other variables exhibit a clear influence on programmer performance: use of IDE, testing framework experience and unit testing experience, although only the first one was statistically significant. The results for IDE usage and testing framework experience have β-values of 19% and 12% for quality ($d = .41$ and $d = .29$) and 18% and 13% for productivity ($d = .46$ and $d = .36$). It is not at all surprising. It is reasonable to assume that the use of proper tools should improve programmer performance. It is remarkable, however, that these variables have such a noticeable effect.

With regard to unit testing experience, the MLR gives β-values of -11% and -5% for quality and productivity, respectively ($d = -.31$ and $d = -.18$). We did not expect to find that unit testing experience had negative effects. There are two more or less obvious interpretations of this result: 1) Subjects who are experienced in unit testing might pay more attention to quality and thus are less productive, and 2) testing and programming activities are performed by different subject profiles, i.e., testers do not make code and programmers do not code. However, the results of the MLR do not support these interpretations. The reason why unit testing experience leads to decreasing quality and productivity is unclear for us.

2 NEW INSIGHTS

Further research revealed that motivation may be influencing our results, at least partially. Using a different dataset, Dieste et al. [1] report that long-experienced professionals often perform marginal work on the experimental tasks during the experimental sessions. This could explain the absence of a relationship between experience in industry and performance.

We have also conducted exploratory research using Factor Analysis (EFA) on the original dataset. EFA yields three components: Unit testing experience is associated with the Component 1. Component 2 is characterized by a lengthy experience in unit testing frameworks. Component 3 refers to subjects with a specialized computer science background who currently use the IDE used in the experiment. We believe that these components point to three different generations of programmers, probably of different ages, with different outlooks and motivations.

3 CONCLUSION

Our results can be explained by the *intensity of practice*. As pointed out by Ericsson [4], subjects need to complete a period of intensive practice to improve their performance (i.e., achieving expertise). The mere practice of an activity (e.g., daily coding) can improve performance but not to the point of making a huge difference (as compared to other professionals) [3]. These results are connected to relevant topics, such as the design of lifelong learning programmes, or the suitability of students as experimental subjects.

REFERENCES

[1] O. Dieste, E. R. Fonseca C., G. Raura, and P. Rodríguez. 2017. Professionals Are Not Superman: Failures beyond Motivation in Software Experiments. In *5th International Workshop on Conducting Empirical Studies in Industry (CESI)*. 27–32.

[2] K. Anders Ericsson et al. 2006. The influence of experience and deliberate practice on the development of superior expert performance. *The Cambridge handbook of expertise and expert performance* 38 (2006), 685–705.

[3] K. Anders Ericsson and Neil Charness. 1994. Expert performance: Its structure and acquisition. *American psychologist* 49, 8 (1994), 725.

[4] K. Anders Ericsson, Neil Charness, Paul J. Feltovich, and Robert R. Hoffman. 2006. Introduction. In *The Cambridge handbook of expertise and expert performance*. Cambridge University Press, 3–19.

[5] George M. Marakas and Joyce J. Elam. 1998. Semantic structuring in analyst acquisition and representation of facts in requirements analysis. *Information Systems Research* 9, 1 (1998), 37–63.

[6] Matthias M. Muller and Frank Padberg. 2004. An empirical study about the feelgood factor in pair programming. In *10th International Symposium on Software Metrics*. IEEE, 151–158.

[7] Sylvia B. Sheppard, Bill Curtis, Phil Milliman, and Tom Love. 1979. Modern coding practices and programmer performance. *Computer* 12, 12 (1979), 41–49.

[8] Sabine Sonnentag. 1998. Expertise in professional software design: A process study. *Journal of applied psychology* 83, 5 (1998), 703.

[9] Lawrence G Votta. 1994. By the way, has anyone studied any real programmers, yet?. In *9th International Software Process Workshop*. IEEE, 93–95.