An Empirical Study on Novice Programmer’s Behaviors with Analysis of Keystrokes

Dapeng Liu, GradientX, Santa Monica, CA, USA
Shaochun Xu, Department of Computer Science, Algoma University, Sault Ste Marie, ON, Canada
Huafu Liu, Department of Computer Science, Changsha University, Changsha, China

ABSTRACT

This paper presents a series of two experiments in which programming behaviors were observed and analyzed when they were programming with pressure and without pressure. There were eleven and twenty-four subjects respectively. In both experiments, the authors used a software tool to record the keystroke frequency, designed criteria to evaluate program quality, and conducted a survey after the experiment. The experiment results show that there is no direct relation between the numbers of keystrokes and programmer’s performance when programmers are working without pressure or with pressure. The first experiment results demonstrate while novice programmers are diverse in terms of programming styles, ones with more experiences tend to control code execution in finer granularity. Source code format can be an indicator of programming performance. The second experiment results demonstrate that programmers with higher performance likely have higher keystroke productivity. Programmers are also more productive under pressure in terms of keystrokes.

Keywords: Cognitive Activity, Empirical Study, Keystroke Logging, Novice Programmer, Programmer Performance

1. INTRODUCTION

Software development is an intelligence-intensive activity and studies of cognitive process inside it can shed light on many software engineering problems (Robillard, Kruchten, & D’Astous, 2002). It might be also able to provide some useful hints on training programmers once we understand the mental differences between novice and expert programmers.

DOI: 10.4018/ijsi.2013010106

There have been a lot of researches with regards to various cognitive activities during software engineering process. However, most of them focused on high level activities such as programmer’s behavior and mental activities (Davies, 1993; Rajlich & Xu, 2003; Xu & Rajlich, 2005; Buchoux & Clark, n.d.). There was little research on low level activities, such as keystrokes and mouse clicks.

Keystroke has rarely been used as a device of investigating programmer’s coding behavior. Thomas et al. (Thomas, Karahasanovic, & Ken-
nedy, 2005) was probably the only one who has studied the correlation between keystroke speed and programming performance. However, their work was primitive because they only evaluated the quality of the programs based on the completeness of the program.

We (Liu & Xu, 2011) think that keystrokes might be an essential indicator of the programmer’s brain activities, both consciously and subconsciously. By observing the keystroke patterns of programmers during their work, it might help us to better understand the cognitive process of programmers. Therefore, more empirical study seems necessary.

In this study, we design two experiments to study programmer’s behaviors and performance with help of recorded keystroke frequency. During the first experiment, programmers are voluntary; while programmers in second experiment are programming under pressure. The pressure is indispensable in daily programming jobs in industrial settings. We would like to learn whether there is any correlation between keystroke frequency and the programmer’s performance when they are programming under pressure or without pressure. We also would like to study whether there are any differences between programming performance without and with pressure in terms of other factors.

The rest of the paper is organized as follows: Section 2 describes related work. The case study setting of two experiments is described in Section 3. Section 4 discusses the experiment results in detail. The conclusions and the future work are presented in Section 5.

2. RELATED WORK

There are a lot of research activities regarding the cognitive activities during software engineering process. Davies did a systematical analysis on the programming strategy (Davies, 1993), and suggested to study the explanation of programming skill and to integrate ideas about knowledge representation with a strategic model, which might enable us to make predictions about how changes in knowledge representation might give rise to particular strategies and to the strategy changes associated with developing expertise.

Visser (1987) conducted experiments on professional programmers to study the strategies that were used during programming. He found that programmers memorized a variety of data sources and sample program listings, so programmers may recall that a solution exists in a listing, find the listing, and then use the coded solution as an approach for the current problem. The programming knowledge was classified by Ye and Salvendy (Ye & Salvendy, 1996) into a five level abstractions. They also found that experts have better knowledge at an abstract level, and the novices tend to have more concrete knowledge.

Coding activities by experts and novices were studied by Davies (Davies, 1993), in term of information externalization strategies. Additionally Davies (Davies, 1994) found that experts tend to rely much more upon the use of external memory sources. He stated that novices tend to focus on key words in the problem statement rather the deep structure of the problem. Petre and Blackwell (Petre & Blackwell, 1997) studied the mental imagery of experts during software design. They discovered that there are some common elements or principles all the experts applied. Hofer (Hofer, 2011) conducted experiment to compare novice and expert pair programming and he found experts wrote tests with a higher instruction, line, and method coverage, but were slower than novice pairs.

There are a lot of recent researches on understanding how undergraduate students are doing their programming and how to improve undergraduate teaching. Ginat (2003) studied the programming pattern for students and found that they might be misled by explicit keyword association. M. Rodrigo et al. (2009) observed the programming behaviors and found that students show a lot of confusion when expected to implement object-oriented constructs such as constructors and object interaction. Ma et al. (Ma, Ferguson, Roper, Ross, & Wood, 2009) conducted a serious experiment by asking
Related Content

Industry Software Reviews Survey Results and Findings
[www.igi-global.com/chapter/industry-software-reviews-survey-results/26905?camid=4v1a](www.igi-global.com/chapter/industry-software-reviews-survey-results/26905?camid=4v1a)

Java Integrated Development Environments' Support for Reuse-Oriented Software Development
[www.igi-global.com/chapter/java-integrated-development-environments-support/29976?camid=4v1a](www.igi-global.com/chapter/java-integrated-development-environments-support/29976?camid=4v1a)
Modeling Trust Relationships for Developing Trustworthy Information Systems
[www.igi-global.com/article/modeling-trust-relationships-for-developing-trustworthy-information-systems/106933?camid=4v1a](www.igi-global.com/article/modeling-trust-relationships-for-developing-trustworthy-information-systems/106933?camid=4v1a)

Model-Driven Applications: Using a Model-Driven Mechanism to Bridge the Gap between Business and IT
Tong-Ying Yu (2014). *Advances and Applications in Model-Driven Engineering* (pp. 53-72).
[www.igi-global.com/chapter/model-driven-applications/78610?camid=4v1a](www.igi-global.com/chapter/model-driven-applications/78610?camid=4v1a)