Development Tools Usage Inside Out

Marko Gasparic(✉), Andrea Janes, and Francesco Ricci

Free University of Bozen-Bolzano, Dominikanerplatz 3, 39100 Bolzano, Italy
marko.gasparic@stud-inf.unibz.it, {andrea.janes,francesco.ricci}@unibz.it

Abstract. The software engineering community is continuously producing tools to tackle software construction problems. This paper presents a research study to identify which tools, artifacts, and commands developers use during task solving and how one can design software that can suggest and convince the developer to use specific software construction techniques. We want to understand under which conditions developers accept suggestions for a more efficient and effective usage of the available instruments, and if observed usage patterns correlate with observable improvements in the process or product. The expected results include detailed logs of how developers construct software during XP 2016, their preferences for software construction recommendations, and which effects accepted suggestions have on task execution and outcome.

Keywords: Tool usage · IDE command recommendation

1 Aim of Research and Research Questions

The aim of the proposed study is to observe in depth how developers solve their tasks, how developers accept different types of suggestions to support their work, and what are the effects of different behaviors.

Concretely, the research questions we want to answer with this empirical study are:

- RQ1: Which tools and artifacts developers use during task solving?
- RQ2: If a better way to solve a task exists, how can we design software that can persuade the developer to change his or her behavior?
- RQ3: Which effects on task execution and task outcome (i.e., the code) do different tools and command suggestions have?

To solve their daily tasks, software developers are using tools, such as integrated development environments (IDEs), web-browsers, communication tools, etc. The choice of tools and their usage have a strong impact on the productivity of developers. Understanding how developers work is therefore important to understand how to support their work.

We already performed a preliminary study, by analyzing interaction patterns within the IDE of eight developers, comparing the patterns in different contexts. In a conference setting, such as the XP 2016 coding sessions, we now have the opportunity to build on our experience and observe a bigger sample of skilled, focused developers, solving a predefined programming task. This is an experimental setting which is rare to find. Collecting such data from experienced programmers, all executing a similar task is difficult: companies are rarely willing to invest time to perform such experiments as they do not obtain a direct benefit from it.

We assume that in a conference setting developers will be less exposed to interruptions, which will make the results easier to interpret. This allows us to better understand how experienced developers are spending their time interacting with tools and how they are using the functionality provided by the IDE, solely for the purposes of programming.

To answer RQ1, we want to answer the following sub-questions:

- RQ1.1: Which tools (e.g., text editing, communication, source code management) are developers using to solve a particular task?
- RQ1.2: Which artifacts (e.g., websites, documents, source code files, text files) are they reading, writing, and modifying?
- RQ1.3: Which IDE commands are they invoking?

RQ2 addresses the question if and how we can write software that identifies and suggests to the developer more effective ways to solve a specific task. In this context, we want to focus on the tools developers are using, in particular the IDE. Many developers are not using even some basic features provided by their IDE, even if certain features are recognized as highly useful by the community [1].

To alleviate this problem, first researchers developed and validated IDE command recommendation algorithms [2]. These algorithms were either evaluated offline or by interviewing the study participants. We are not aware of any designed and tested user interface for IDE command recommendations. Consequently, we do not know how persuasive and effective such systems would be in practice and whether the developers would accept recommendations, even if the recommendations would be 100 % accurate.

The most precious resource that development tools require from the developer is the attention. Due to the low usage of tools that the scientific community developed in the last years, it is questionable if the developers are willing to accept them in practice at all [3]. We would like to investigate whether it even makes sense to start building new tools that would change the development process or will the developers rather stick to the current practices. To answer RQ2, we will answer the following sub-questions:

- RQ2.1: How do developers perceive the current integration of the various tools they use?
- RQ2.2: How will developers react to different types of user interfaces for persuasive and effective IDE command recommendations?
- RQ2.3: How efficient are the proposed user interfaces and how can they be improved?

Finally, RQ3 asks which effects on task execution and task outcome tool usage and command suggestions have. We will observe work patterns and interactions with tools and artifacts, as well as the effect on the source code itself.

In parallel, we want to observe the acceptance of command recommendations generated specifically for the task at hand and delivered at the beginning of the coding session; also, we want to observe the effects of the recommendations. Thus, we want to perform an experiment according to the "one factor with two treatments" design type [4], where the treatment group will have access to the IDE command recommendation mockups.

Since the duration of the experiment is short, we plan to investigate RQ3 qualitatively through the following sub-questions:

- RQ3.1: How does the usage of different tools, commands, and artifacts affect the produced source code?
- RQ3.2: How do command recommendations change the interaction with the IDE?

2 Importance of Research

The software engineering community is continuously producing tools that help developers to tackle what Fred Brooks calls "essence and accidents" [5]. Currently, a particularly dynamic field is the field of recommendation systems for software engineering [6]. By obtaining the answer to RQ2, we would like to better understand under which conditions software developers accept the promotion of more efficient and effective usage of tools, by improving their accessibility (RQ2.1) and discovery (RQ2.2). This will pave the way to construct a recommender that can deliver useful recommendations in a real-life setting.

RQ3 is targeting the meaningfulness of the proposed tools and commands. We aim to better understand whether the suggestions to use additional tools, features, web-pages, etc. lead to observable improvements, i.e., cause a change in the data collected in RQ1. Knowing the effect of the usage of certain tools nurtures the motivation to develop new tools and facilitates the introduction of existing tools in practice. Some examples are: the diffusion of innovation within an organization, the training of newcomers, or the support for teaching.

3 Data Collection Methods

The majority of the data will be collected automatically by the following tools:

- A tool that logs the currently focused window, together with its process name and caption. The window caption often contains the path to the currently opened artifact, which can be used to infer the type of the artifact. The obtained log contributes to answer RQ1.1 and in part RQ1.2.

- Eclipse UDC[1] to collect command executions, user interface elements activations, and start and stop events of bundles. In addition, a modified version of Eclipse Mylyn[2] will be used to record the currently focused artifact within the IDE, together with the active perspective, including editors, and views. These Eclipse plugins contribute to answer RQ1.2 and RQ1.3.
- A tool to collect all the logged data to a central location.

We will provide the environment in the cloud and allow participant's to install the necessary tools on their own machines at the beginning of the session. If developers agree, we will use eye tracking devices to understand on what developers are looking during their work.

To investigate the motivations behind the manifested decisions following the display of an IDE command suggestion (RQ2), we will perform qualitative interviews (based on [7]) and an online survey, which will take less than 20 min.

4 Data Analysis and Data Usage

The collected data will be anonymized and studied using descriptive and inferential statistics, data mining techniques, and manual inspection. To study the results of the interviews, we will use quantitative and qualitative research methods. To study the impact of the recommendation on the code, we will use the data provided by code smell detection tools, e.g., FindBugs[3], but mainly manually study which effects the invocation of the suggested commands has on the code. The obtained data will be used to provide feedback to the participants, improve the understanding of development tools usage, and in the development of recommendation systems in software engineering.

References

1. Murphy-Hill, E.: Continuous social screencasting to facilitate software tool discovery. In: International Conference on Software Engineering (2012)

[1] http://www.eclipse.org/org/usagedata.
[2] http://www.eclipse.org/mylyn.
[3] http://findbugs.sourceforge.net.

2. Murphy-Hill, E., Jiresal, R., Murphy, G.C.: Improving software developers' fluency by recommending development environment commands. In: ACM SIGSOFT International Symposium on the Foundations of Software Engineering (2012)
3. Gasparic, M., Janes, A.: What recommendation systems for software engineering recommend: a systematic literature review. J. Syst. Softw. **113**, 101–113 (2016)
4. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Norwell (2000)
5. Brooks, F.P.: No silver bullet essence and accidents of software engineering. Computer **20**, 10–19 (1987)
6. Robillard, M.P., Maalej, W., Walker, R.J., Zimmermann, T.: Recommendation Systems in Software Engineering. Springer, Heidelberg (2014)
7. Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D.: User acceptance of information technology: toward a unified view. MIS Q. **27**, 425–478 (2003)