Information and Software Technology 000 (2017) 1–20

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A graphical user interface for presenting integrated development environment command recommendations: Design, evaluation, and implementation

Marko Gasparic[a,*], Andrea Janes[a], Francesco Ricci[a], Gail C. Murphy[b], Tural Gurbanov[a]

[a] Free University of Bozen-Bolzano, Dominikanerplatz 3, 39100 Bolzano, Italy
[b] University of British Columbia, 201-2366 Main Mall, Vancouver, BC V6T 1Z4, Canada

ARTICLE INFO

Article history:
Received 12 December 2016
Revised 30 July 2017
Accepted 16 August 2017
Available online xxx

Keywords:
Integrated development environment
User interface
Command
Functionality
Recommender system
Software development

ABSTRACT

Context: A set of algorithms exist to generate integrated development environment (IDE) command recommendations. The recommendations are aimed at improving software developer's interaction with an IDE. Even though the interface is a critical element of every recommender system, we are not aware of any existing graphical user interface to present such recommendations.
Objective: This paper describes and evaluates a novel design of a graphical user interface to recommend commands within an IDE. The interface contains a description of the suggested command, an explanation of why the command is recommended, and a command usage example.
Method: The proposed design is based on the analysis of guidelines identified in the literature. Its acceptance and usability were evaluated through a user study with 36 software developers and semi-structured interviews with 11 software developers.
Results: The results indicate that the suggested interface is well accepted, but it can be further improved. Through the interviews and the implementation of the interface, we identified a series of requirements important for the development of future IDE command recommender systems.
Conclusions: This paper shows that a convenient graphical user interface is critical to achieve high acceptance of IDE command recommendations. Our work also illustrates steps useful for undertaking user studies related to IDE command recommendations in a practical setting without human intervention. A future step is to evaluate the interface within the business environment, where recommendations are generated and presented in an IDE used by practicing software developers as part of their normal workday.

1. Introduction

To serve the needs of a diverse user population, high-functionality applications provide a large set of functions that are potentially overwhelming for a user [1]. Although it is not expected that a generic user access all the provided functionality, for some applications—including integrated development environments (IDEs)—the proportion of the used functionality is surprisingly low. For example, average users of Eclipse IDE[1] use only 42 out of more than 1100 available commands [2]; where a command corresponds to a menu button or a shortcut that executes a function, such as paste or open resource, amongst many others.

One of the likely reasons for the low usage of functionality is the lack of the exact knowledge of which functionality is available and potentially useful [3,4]. As a consequence, many users do not exploit the full potential offered by an application. Conversely, a better knowledge of the target high-functionality application can help users to more effectively choose the precise functionality that is more useful in a specific situation. To improve such knowledge, the application of recommender systems (RSs) has been proposed. Recommender systems are personalized information search and filtering tools that direct the users to items that are estimated to be useful for them [5]. They are primarily conceived to support individuals who lack sufficient knowledge or time to evaluate the potentially overwhelming number of alternative options that may be available [6,7].

* Corresponding author.
E-mail addresses: marko.gasparic@stud-inf.unibz.it, m.gasparic@gmail.com (M. Gasparic).

[1] http://www.eclipse.org.

http://dx.doi.org/10.1016/j.infsof.2017.08.006
0950-5849/

In this paper, we focus on command recommendations in an IDE. Within software engineering, it is generally accepted that software development tools can affect the efficiency and quality of software construction [8–10] and that the knowledge of those tools directly impacts on the productivity of programmers [11]. One reason for this is that "tools allow repetitive, well-defined actions to be automated, reducing the cognitive load on the software engineer who is then free to concentrate on the creative aspects of the process" [12, pp. 368]. Thus, we conjecture that a better knowledge of the commands offered by an IDE would help programmers to more effectively choose the functionality that can help them in a specific situation.

We envision the following use case: while developers use an IDE, the command recommender system is observing their interaction with the application; if the recommender detects that a specific command would be useful, but it is never used, it invites the developer, at a convenient time and with a well suited graphical user interface (GUI), to learn and use this command in the future. In this scenario, it is important to design an appropriate GUI to present the recommended commands in a way that allows the user to evaluate and act upon them [13].

To date, a set of algorithms for selecting IDE command recommendations has been proposed by Murphy-Hill et al. [14], Zolaktaf and Murphy [15], and Gasparic et al. [16]; but we are not aware of any specifically designed and validated user interface to present these recommendations. Nevertheless, the user interface used to present recommendations is a critical element of a recommender system, as it can affect the user's trust and loyalty to the system, and it can influence the user's final decision to accept the recommendation [17].

We propose and validate a GUI designed for an IDE command recommender system, which is based on design guidelines identified in the literature. The proposed GUI consist of three parts: command name and description, explanation of recommendation, and command usage example. The GUI is not bound to a specific type of IDE commands nor to a particular recommendation algorithm, hence, it is general and can be supported by a variety of command recommender systems. The specific research goals addressed in this paper are:

- design a novel GUI to recommend IDE commands based on approaches and guidelines identified in the literature; and
- evaluate the acceptance and usability of the proposed GUI.

To design the GUI, we followed Design Science research guidelines [18]. We evaluated the perceived usability and acceptance of our GUI by performing a user study with 36 software developers and semi-structured interviews with 11 software developers. The results show that most of the participants would like to use an IDE command recommender system if it is augmented with a convenient GUI, such as the one proposed in this paper. The results also indicate that the acceptance of the GUI can be improved further by adapting the GUI to the recommended command and the recipient's profile. In particular, the developers would like to be in control of the information that is included in the presentation of the command, some would like to block certain types of recommendations, and some would like to be able to customize the types of data that are shared with the recommender system. Finally, we discovered that users find the description of the suggested command and its usage example more valuable than the explanation of the reasons for the recommendation.

Based on the evaluation results, we updated the GUI and implemented a prototype that can automatically generate personalized recommendation explanations. Additionally, we describe in this paper a potential approach to automate the generation of other parts of the GUI content, such as command descriptions and usage examples. The full automation of the generation of the content for presenting IDE commands will enable the set of recommendable commands to grow together with the IDE.

The main contributions of our work are: the development of a new GUI design for an IDE command recommender system based on guidelines identified in the literature, its evaluation in a form of a user study, and a road-map towards a fully automated IDE command recommendation presentation. A brief explanation of the GUI and initial survey results are also reported in a short paper published at the International Conference on Intelligent User Interfaces [19].

The reminder of the paper is structured as follows: the research method is discussed in detail in Section 2; the proposed GUI is presented in Section 3; the guidelines identified in the literature to design GUIs of recommender systems are presented in Section 4.1; the list of existing command recommender systems and their assessment in respect to the main guidelines are in Section 4.2; the results of the evaluation are in Section 5; the discussion of the results is in Section 6; the future work required for putting the GUI into practice, including a road-map towards a fully automated generation of the GUI content, is presented in Section 7; and the conclusions are in Section 8.

2. Research method

In this section, we describe the Design Science paradigm and explain how we followed the guidelines devised by Hevner et al. [18].

In general, engineers in various fields are devising artifacts that have desired properties to attain specific goals [20]. Unlike traditional natural sciences, which are focused on understanding the reality, the focus of Design Science is the creation of artifacts that serve human purpose and are assessed according to their value or utility [21]. Fundamentally, Design Science is a problem-solving paradigm, which combines existing theories with experience, creativity, and intuition of the researcher, to solve problems arising from the organization of people and technology [18].

Design Science consists of two basic activities: building and evaluating; hence, it is particularly applicable to computer science research, which is mainly concerned with artificial phenomena that can be both created and studied [21]. Moreover, due to the complexity of the artifact design and often insufficient existing theories, a Design Science approach is proactive, in the sense that a new artifact is first created with theories focused on its application impact following after [18]. According to the literature review and typology of Offermann et al. [22], who studied 62 research papers published in MIS Quarterly journal and in the proceedings of the international conference on Global Perspectives on Design Science Research, there are eight basic types of artifacts that emerged during the application of Design Science paradigm, namely: *system design, method, language/notation, algorithm, guideline, requirements, pattern*, and *metric*. The overall goal of this paper is to design and evaluate a GUI to recommend useful commands within an IDE, thus, the type of the artifact described in this paper is *system design*.

Hevner et al. [18] devised guidelines to conduct Design Science, which state that Design Science research requires the creation of an innovative, purposeful artifact (guideline 1) for a specified problem domain (guideline 2). To understand if the artifact helps to solve the specified problem, a thorough evaluation of the artifact is crucial (guideline 3). Moreover, the artifact must solve a so far unsolved problem or solve a known problem in a more effective or efficient way (guideline 4); the novelty requirement is what differentiates Design Science from design. The artifact itself must be rigorously defined, formally represented, coherent, and internally consistent (guideline 5). The process by which the artifact is created defines the problem space and describes the mechanism by

Please cite this article as: M. Gasparic et al., A graphical user interface for presenting integrated development environment command recommendations: Design, evaluation, and implementation, Information and Software Technology (2017), http://dx.doi.org/10.1016/j.infsof.2017.08.006