Guest Editors Introduction: Context for Software Developers

Kelly Blincoe[a,*], Daniela Damian[b], Giuseppe Valetto[c]

[a]*Department of Electrical and Computer Engineering, University of Auckland, Auckland, New Zealand, k.blincoe@auckland.ac.nz*
[b]*Department of Computer Science, University of Victoria, Victoria, BC, Canada, danielad@uvic.ca*
[c]*Fondazione Bruno Kessler, Trento, Italy, valetto@fbk.eu*

1. Introduction

Software Developers routinely work on complex development tasks that require a wealth of detailed support information, such as the change history of the source code, the software architecture, the task assignments of other team members, and so on. All of this information provides the context for the development task. Today, the context that software developers work with comes from many sources; as such, it is multi-faceted and increasingly large in scale. Having many sources of context causes a considerable amount of context switching during the developers' work activity and increases the cognitive load of the developers. Thus, recent research has begun to investigate ways to provide this contextual information to developers when it is needed in order to minimize the amount of time developers must spend looking for this information on their own (e.g. [1, 2, 3, 4]).

The context of a task can provide rich information that can also be used to gain insights on other aspects of software development. Another line of research on this topic is how to mine and leverage this contextual information as it is created to gain insights to support software developers. For example, recent research mined developers' activities within their IDEs to facilitate developer

*Corresponding author

coordination and collaboration [5].

This special issue is the culmination of two Context for Software Developers (CSD) workshops held at the International Conference of Software Engineering (ICSE) in 2015 [6] and the Foundations of Software Engineering (FSE) Conference in 2014 [7]. The previous research in this area has been mostly limited in considering only project artifacts, like the source code and development tasks, and developer interactions with these artifacts in the context of a task. However, the participants at these workshops defined a broader understanding of context for software developers as "information needed to implement a task" and "anything that might cause a developer to do something differently."

In fact, the context of a development task can expand significantly beyond the technical artifacts. The participants of the CSD workshops considered context to be technical-, people-, and process- related. The context of a task is constantly evolving and, in turn, it can cause the task itself to evolve. This high-level model of context for software developers is illustrated in Figure 1. Context can also be different for everyone depending on his or her awareness and perspective.

One take-away from the CSD workshops is that this is definitely an area rich of possible avenues of investigations. More research is needed to fully understand exactly what constitutes the context of a software development task and how contextual information can be leveraged to better support software developers.

2. In This Issue

This issue is a step towards that direction. A detailed context model, a first step towards understanding the types of contextual factors that are important for developers, is presented by Marko Gasparic, Gail Murphy, and Francesco Ricci in "A Context Model For IDE-Based Recommendation Systems." They describe 13 contextual factors that impact how a developer works with their IDE, like the day of the week and the current activity. We believe their model will serve as a foundation for further research in this area. Understanding the

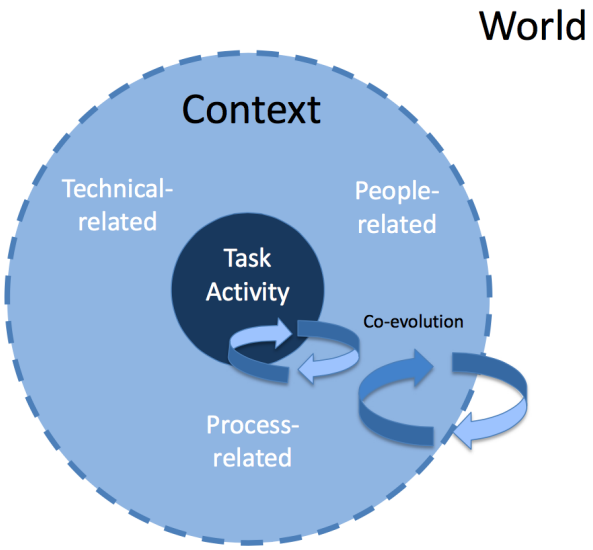

Figure 1: High-level model of context for software developers.

contextual factors that impact a developer in their IDE can, for example, enable improvements in the user interface of IDEs and help to improve the accuracy and timing of recommendation systems for software engineering (RSSEs).

The four other articles presented here leverage context in interesting ways, demonstrating that contextual information can be extremely useful to support the work of both individual developers and teams. In "Using Contextual Information to Predict co-changes", Igor Scaliante Wiese, Reginaldo Re, Igor Steinmacher, Rodrigo Kuroda, Gustavo Oliva, Christoph Treude, and Marco Aurelio Gerosa present a new technique to predict which artifacts will change together, called co-change prediction. Co-change predictions can be used to give developers useful advice when performing changes in source code. Their approach is different from existing approaches because they rely on detailed contextual information of software changes collected from issues, developers' communication, and commit metadata to make more accurate predictions.

In "On the use of Developers' Context for Automatic Refactoring of Soft-

ware Anti-patterns", Rodrigo Morales, Zephyrin Soh, Foutse Khomh, Giuliano Antoniol, and Francisco Chicano use developer context in an innovative way to recommend refactoring strategies to remove anti-patterns, poorly designed code, from the source code. Their recommendations present refactoring solutions that affect only entities in the developer's current working set. This approach can remove more anti-patterns using less resources than the other existing approaches by minimizing the context switches a developer must make.

In "Eye Gaze and Interaction Context for Change Tasks - Observations and Potential", Katja Kevic, Braden M Walters, Timothy R Shaffer, Bonita Sharif, David C Shepherd, and Thomas Fritz examine developer code navigation by examining IDE interaction context data together with fine-grained eye-tracking technology, which offers richer data about developer navigations. The results reveal a number of interesting insights, such that developers don't always read method signatures and that often only small parts of methods receive focus. Additionally, they use this data to predict future navigations and task difficulty.

In "Using Contexts Similarity to Predict Relationships between Tasks", Walid Maalej, Mathias Ellmann, and Romain Robbes present a new approach that assesses the similarity of tasks based on the similarity of the task contexts (artifacts) associated with the tasks. The models they developed can be used to recommend similar tasks to developers. This could make task switches more efficient and limit context switches, improving developer productivity.

While demonstrating the usefulness of leveraging the right context data, the articles in this special issue also uncover many open research questions. Much more work remains to be done to understand what constitutes the context of a development task, how that contextual information can be captured, and how it can be leveraged to support better and more efficient software engineering work. This special issue is an effort in that direction. We hope that the readers of the Journal of Systems and Software will find this special issue useful and inspiring in moving forward the research on context for software developers.

Acknowledgement

We thank the authors of the papers in this special issue for their contributions. We are also grateful to the reviewers for the time they dedicated to ensuring a high quality issue. We would also like to thank the participants of the CSD workshops for contributing their insights on this topic. Finally, we thank JSS for including this special issue and the editor-in-chief for his support.

References

[1] M. Kersten, G. C. Murphy, Using task context to improve programmer productivity, in: Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering, ACM, 2006, pp. 1–11.

[2] D. Čubranić, G. C. Murphy, Hipikat: Recommending pertinent software development artifacts, in: Proceedings of the 25th international Conference on Software Engineering, IEEE Computer Society, 2003, pp. 408–418.

[3] L. Ponzanelli, G. Bavota, M. Di Penta, R. Oliveto, M. Lanza, Prompter, Empirical Software Engineering 21 (5) (2016) 2190–2231.

[4] O. Baysal, R. Holmes, M. W. Godfrey, Situational awareness: personalizing issue tracking systems, in: Proceedings of the 2013 International Conference on Software Engineering, IEEE Press, 2013, pp. 1185–1188.

[5] K. Blincoe, G. Valetto, D. Damian, Facilitating coordination between software developers: A study and techniques for timely and efficient recommendations, IEEE Transactions on Software Engineering 41 (10) (2015) 969–985.

[6] K. Blincoe, D. Damian, G. Valetto, J. D. Herbsleb, 2nd international workshop on context for software development (csd 2015), in: Proceedings of the 37th International Conference on Software Engineering-Volume 2, IEEE Press, 2015, pp. 973–974.

[7] K. Blincoe, D. Damian, G. Valetto, G. Murphy, 1st international workshop on context in software development (csd 2014), https://fse22.gatech.edu/workshops.