DFlow – Towards the Understanding of the Workflow of Developers

Roberto Minelli and Michele Lanza

REVEAL @ Faculty of Informatics — University of Lugano, Switzerland

Abstract. Developers interact with the Integrated Development Environment (IDE) to read, understand, and write source code; essentially *navigating* the system at hand. While doing so, they construct a mental model of the system using the tools and facilities provided by the IDE. It is unclear how developers navigate the system and whether the current facilities provided by the IDE appropriately support the navigation. Our goal is to investigate how, when, and why developers use certain tools to explore and modify the system, and to provide the means to retrospectively analyze and understand development sessions. We present our initial steps towards an approach to track the activities of developers while performing software engineering tasks, implemented as an extension of the Pharo Smalltalk IDE.

Introduction. Developers interact with the Integrated Development Environment (IDE) to read, understand, and write source code, essentially *navigating* the system at hand. While interacting with the tools and facilities provided by the IDE, developers construct a mental model of the system that eases the comprehension of the system itself. However, it remains to be investigated how developers navigate the system and whether the current tools and IDE extensions (e.g., plug-ins) conveniently support the navigation tasks.

Our Approach. To tackle this problem, we present our initial steps towards an approach to track the workflow of developers while performing software engineering tasks. In a nutshell, we want to record all the interactions that developers have with the IDE to enable retrospective analyses. Robbes *et al.* devised a similar approach and implemented SpyWare: A tool which tracks the changes that a developer performs on a program *as they happen* [5]. While their approach mostly focuses on fine-grained source code change operations, we are more concerned about concrete developer actions. In a similar vein, Singer *et al.* developed NavTracks: A tool that considers various kinds of relationships, such as control flow and inheritance relationships to help developers navigate the software space [7]. Our goal is not only to analyze the interactions, but use them as a means to suggest modifications to the Pharo Smalltalk IDE.

In short, we want to investigate how, when, and why developers use specific tools to explore and change the system in order to provide the means to retrospectively inspect and comprehend development sessions.

Present State. We use software visualization to understand the data recorded during the development sessions. For example, we provide a System Complexity view [4] containing all entities touched during a development session to get an overview of how developer activities are spread across the system [3]. Colors in the visualization map to the number of occurrences of each type of action. Another way to visualize the workflow of developers is using a tree map enriched with information about the particular kinds of activities performed during a development session. We are currently implementing our approach in DFlow: A tool for the Pharo Smalltalk IDE (see `http://www.pharo-project.org`). DFlow silently observes the workflow of developers inside the IDE while performing software engineering tasks, and records all the actions performed, e.g., *browsing the source code of a class*, *the senders of a method*, or *the steps in a debugger*.

Future work. Our approach is extensible. We plan to use our visualizations to compare and classify different developers sessions according to the activities performed, e.g., by super-imposing one tree-map over another. Moreover, by comprehending different workflows, we foresee to be able to identify some of the plagues of current IDEs which hamper the productivity and the efficiency of developers (e.g., the window plague [1]). DFlow can provide developers with suggestions or ease some of their recurring activities in the IDE. DFlow can also be used to discover how developers answers the different kinds of questions they pose themselves during software evolution tasks [6].

Conclusion. Developers interact with IDEs to read, understand, and write source code, essentially *navigating* the system at hand. We presented our initial steps towards an approach to understand the workflow of developers. We are implementing our approach in DFlow: A tool which silently monitors the IDE and records all the interactions of developers to understand how developers evolve software systems [2].

References

1. O. Nierstrasz D. Röthlisberger and S. Ducasse. Autumn leaves: Curing the window plague in ides. In *Proceedings of WCRE*, 2009.
2. T. Gîrba, A. Kuhn, M. Seeberger, and S. Ducasse. How developers drive software evolution. In *Proceedings of IWPSE*, 2005.
3. O. Greevy, T. Gîrba, and S. Ducasse. How developers develop features. In *Proceedings of CSMR*, 2007.
4. M. Lanza and S. Ducasse. Polymetric views — a lightweight visual approach to reverse engineering. In *Proceedings of TSE*, 2003.
5. R. Robbes and M. Lanza. Spyware: A change-aware development toolset. In *Proceedings of ICSE*, 2008.
6. J. Sillito, G. Murphy, and K. De Volder. Questions programmers ask during software evolution tasks. In *Proceedings of ACM SIGSOFT FSE*, 2006.
7. J. Singer, R. Elves, and M. Storey. Navtracks: supporting navigation in software maintenance. In *Proceedings of ICSM*, 2005.