Towards Self-Adaptive IDEs

Roberto Minelli
REVEAL @ Faculty of Informatics — University of Lugano, Switzerland

Integrated Development Environments (IDEs) provide developers with tools and facilities to support development activities. Developers use IDEs to read, understand, and write source code. Research showed that they spend more time reading source code than writing it. Reading code is the foundation of program understanding, which has been estimated to occupy 50% of developers' work time and to be one of the most challenging tasks performed by developers. To read and comprehend code, developers have also to navigate the system at hand. During this process developers build their mental models of it. In general, all software engineering activities happening inside the IDE (*e.g.,* reading, writing, understanding, and navigating source code) generate a large amount of events that we call *"IDE interaction data"*. Examples include opening a code browser on a method, inspecting an object at run-time, editing a line of a method, popping up a refactoring menu, *etc.*

Unfortunately, current IDEs neglect this information, but researchers have pointed up its importance. Murphy *et al.* showed that it can be used (1) to evolve the development environment according to user needs, (2) to provide means to evaluate new tools and application programming interfaces, and (3) to prevent feature bloat, *i.e.,* the tendency to add unnecessary features to a software system [1]. Frey *et al.* believe that future program investigation tools need to track the way developers navigate code to support software engineering activities [2]. Interaction data includes, but is not limited to, IDE interactions. Our goal is to also leverage interactions with different information sources outside the IDE, such as bug-tracking systems and versioning control systems, to evolve current development environments.

We envision *self-adaptive IDEs*: IDEs that collect, process, and leverage the interactions of developers with different information sources to better support the workflow of developers. Collecting interaction data means that the environment monitors the actions that a developer is carrying on inside the IDE itself and persists them. Then IDEs mine this data by transforming and structuring it for further use. Finally, the most important and challenging task is to leverage such data, either retrospectively or at run-time. Retrospective analyses serve as a means to understand, characterize, and classify development sessions. As an initial contribution, we devised a catalogue of software visualizations to present interaction data. Views support the analysis of development sessions, for example, to seize how different development activities are distributed, to understand which program entities are involved in a development session, or how the developer interacted with the UI of the IDE [3], [4], [5]. Leveraging the data at run-time means exploiting the potential of interaction data to directly influence, modify, and reshape the working environment of developers to support different software engineering activities, such as comprehending, browsing, navigating, and editing source code. We envision three main research directions: live/adaptive visualizations, interaction-based recommender systems, and adaptive user interfaces (UI). Live visualizations in sync with the activities of developers help them to visually follow their sessions and act as immediate navigation means. We believe that such views can as a "visual memory" for developers: Developers unconsciously associate parts of the visualization with relevant source code fragments. Later they can click on a part of the view and quickly jump to the desired code snipper. Interactions-based recommender systems leverage previous interactions to suggest developers how to quickly accomplish activities such as navigation or debugging. The IDE, for example, can support better browsing through software by providing targeted suggestions on which program entities are more likely to be navigated at a certain moment in time. Adaptive UIs are elements of the IDE that reshape themselves to increase the efficiency of developers, for example by rearranging frequently used UI components, such as menus and system browsers.

The expected contributions of our Ph.D. research can be summarized as: (a) The development of a *self-adaptive IDE* that features novel elements, such as interaction-based recommender systems and adaptive user interfaces; (b) DFLOW: A tool to record the workflow of developers; (c) a catalogue of live and adaptive visualizations to support developers; and a series of (d) user studies and (e) empirical investigations to evaluate the usefulness of our approach and better understand the behavior of developers.

At this stage we developed DFLOW, a tool that silently captures the interactions happening inside the IDE and enables visual analyses [3] and storytelling [4], [5]. We used the recorded data as subject for an empirical investigation that shows, for example, how program comprehension was underestimated by previous research [6].

REFERENCES

[1] G. C. Murphy, M. Kersten, and L. Findlater, "How are java software developers using the eclipse IDE?" *IEEE Software*, vol. 23, no. 4, pp. 76–83, 2006.

[2] T. Frey, M. Gelhausen, and G. Saake, "Categorization of concerns: A categorical program comprehension model," in *Proceedings of PLATEAU 2011 (3rd Workshop on Evaluation and Usability of Programming Languages and Tools)*. ACM, 2011, pp. 73–82.

[3] R. Minelli and M. Lanza, "Visualizing the workflow of developers," in *Proceedings of VISSOFT 2013 (1st Working Conference on Software Visualization)*. IEEE, 2013, pp. 1–4.

[4] R. Minelli, A. Mocci, M. Lanza, and L. Baracchi, "Visualizing developer interactions," in *Proceedings of VISSOFT 2014 (2nd Working Conference on Software Visualization)*. IEEE, 2014, p. to appear.

[5] R. Minelli, L. Baracchi, A. Mocci, and M. Lanza, "Visual storytelling of development sessions," in *Proceedings of ICSME 2014 (30th International Conference on Software Maintenance and Evolution)*, 2014, p. to appear.

[6] R. Minelli, A. Mocci, M. Lanza, and T. Kobayashi, "Quantifying program comprehension with interaction data," in *Proceedings of QSIC 2014 (14th International Conference on Quality Software)*, 2014, p. to appear.