Modeling Programmer's Expertise Based on Software Metrics

Pavol ZBELL*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
pavol.zbell@gmail.com

Knowledge of programmers expertise in a software house environment is usually utilized to effective task resolving (by identifying experts suitable for specific tasks), better forming of teams, effective communication between programmers, personalized recommendation or search in source code, and thus indirectly improves overall software quality. The process of modeling programmer's expertise (building the knowledge base) usually expects on its input some information about programmer's activities during software development such as interactions with source code (typically fine grained actions performed in an IDE – integrated development environment), interactions with issue tracking systems and revision control systems, activities on the Web or any other interaction with external documents.

In our research, we focus on modeling programmer's expertise using software metrics such as source code complexity and authorship. We assume that programmer's expertise is related to complexity of the source code she is interacting with as well as to a degree of authorship of that code. By considering both metrics we intend to find answers for these questions:

- What is the programmer's familiarity of a software component? Is it affected more by programmer's degree of authorship or her readonly interactions (like trying to understand desired code, hence complexity) with it?
- Who works on common library APIs and who just uses them, i.e. are we able to distinguish software architects from business logic programmers?

There are several approaches to model programmer's expertise of which most are based on a simple heuristic – Line 10 Rule [1]. On the other hand, more sophisticated models exist such as Expertise profile [1]. Expertise profile is designed to distinguish between programmers who create methods and who call them, it is a composition of Implementation expertise (i.e. programmer's authorship degree) and Usage expertise (i.e. programmer knows which method to call and how to call it). Degree of knowledge

* Supervisor: Eduard Kuric, Institute of Informatics and Software Engineering

Spring 2014 PeWe Workshop, March 21, 2014, pp. 81–82.

model [2] is a similar solution which takes degree of authorship and degree of interest into account. The degree of authorship is a combination of "first authorships" (first emergence of code by original author), deliveries (changes to the code by the original author) and acceptances (changes by others) and represents long term knowledge of the component. The degree of interest reflects component selections and edits, and represents short term knowledge. In case of source code complexity we have not fully explored the possibilities of its utilization and measurement. We believe that approaches based just on LOC (lines of code) metric or its variations can be further improved, e.g. by static analysis of the source code. Our idea is to explore alternative approaches like weighting AST (abstract syntax tree) nodes or call graph based metrics.

In comparison to the existing approaches we intend to focus more on modeling programmer's knowledge from her interactions in an IDE. An extended analysis [3] shows us that programmers interact with source code in many different ways and hence we may improve our expertise model by taking interaction types or patterns into account (e.g. who uses advanced refactoring tools on some code probably has significant knowledge of it). However, the foundation of our expertise model will still be a combination of source code complexity and authorship degree primarily derived from programmer's interactions in the IDE.

We are continually implementing our solution as an extension to Eclipse IDE since the Eclipse platform is easily extensible and has rich possibilities for user interaction tracking. We plan to evaluate our research on data from academic environment or (preferably) real software house environment.

Acknowledgement. This contribution is the partial result of the Research & Development Operational Programme for the project Research of methods for acquisition, analysis and personalized conveying of information and knowledge, ITMS 26240220039, co-funded by the ERDF.

References

[1] Schuler, D., Zimmermann, T.: Mining usage expertise from version archives. In *Proceedings of the 2008 international workshop on Mining software repositories*. 2008. pp. 121.

[2] Fritz, T., Ou J., Murphy, G., C., Murphy-Hill, E.: A degree-of-knowledge model to capture source code familiarity. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1*. 2010. pp. 385-394.

[3] Murphy, G. C., Kersten, M., Findlater, L.: How Are Java Software Developers Using the Eclipse IDE? In *IEEE Software*. 2006. Vol. 23, no. 4, pp. 76-83.