Their results showed that fully-spelled names were the easiest to understand but that there did not seem to be significant differences with abbreviated names in their comprehensibility level. While their work provides a useful motivation to study whether a shorter name is better or not, they did not discuss the fault-proneness of program.

Kawamoto and Mizuno [19] conducted an empirical study with two OSS products and reported that a class including a long identifier tends to be faulty. While their work is one of our most significant previous studies, our work focuses on a finer-grained artifact—local variable—and conducts a statistical analysis with taking into account of the scopes and the comments.

Binkley et al. [20] focused on the relationship between the length of identifier (including a variable's name, a method's name and a class's name) and the human short-term memory. They identified that identifiers with long names are related to a difficulty in program comprehension. They were concerned that a long chain, e.g., "`class.firstAssignment().name.trim()`," would cause a loss of the readability of the code. While the research viewpoint differs from our work, the fundamental concern about the length of name is common, and it seems to be well accorded with our results showing the compound names are not recommended for local variables.

Aman et al. [11] reported an empirical analysis showing that Java methods having local variables with long names are more likely to be fault-prone and change-prone than the other methods. That report is our significant previous work, and this paper focuses more detailed features of local variables, i.e., the composition of name and their scopes. While another work by Aman et al. [15], reporting that commented programs tend to be more fault-prone, is also our important previous work, we conduct a further analysis examining combinations of the local variable's name, the scope and the comments in this paper.

V. Conclusion

We have focused on programming artifacts which may vary among individuals: local variables' names and comments. Popular code conventions say that names of local variables should be shorter and simple, and it seems to have been a heuristic of programmers. We empirically evaluated the heuristic in terms of fault-proneness by checking the names of local variables, their scopes and the presence of comments. The empirical analysis for the data from nine popular OSS products showed the following three findings.

(1) Local variables with compound names can be signs of fault-prone methods.
(2) Methods having the representative local variables with non-compound and shorter names (≤ 4 letters) are less fault-prone, but their positive effects are decayed as their scopes get wider (around 10 or more lines).
(3) Methods having comments in their bodies are also more likely to be faulty.

These findings are expected to be useful guidelines for more efficient code reviews.

One of our significant future works is to conduct further analyses of local variables' names, which include an application of the natural language processing technologies to evaluate the meaning of local variables' names. A further analysis with products written in a programming language other than Java is also our future project in order to ensure the generality of the above findings.

Acknowledgment

This work was supported by JSPS KAKENHI #16K00099. The authors would like to thank anonymous reviewers for their helpful comments.

References

[1] G. J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*. N.J.: John Wiley & Sons, 2004.
[2] P. C. Rigby and C. Bird, "Convergent contemporary software peer review practices," in *Proc. 9th Joint Meeting on Foundations of Softw. Eng.*, Aug. 2013, pp. 202–212.
[3] P. L. Li, J. Herbsleb, M. Shaw, and B. Robinson, "Experiences and results from initiating field defect prediction and product test prioritization efforts at ABB Inc." in *Proc. 28th Int'l Conf. Softw. Eng.*, May 2006, pp. 413–422.
[4] S. Lessmann, B. Baesens, C. Mues, and S. Pietsch, "Benchmarking classification models for software defect prediction: A proposed framework and novel findings," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 485–496, July 2008.
[5] Y. Liu, T. Khoshgoftaar, and N. Seliya, "Evolutionary optimization of software quality modeling with multiple repositories," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 852–864, Nov 2010.
[6] F. Rahman and P. Devanbu, "How, and why, process metrics are better," in *Proc. 2013 Int'l Conf. Softw. Eng.*, May 2013, pp. 432–441.
[7] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "What's in a name? a study of identifiers," in *Proc. 14th Int'l Conf. Program Comprehension*, June 2006, pp. 3–12.
[8] Free Software Foundation, "Gnu coding standards," https://www.gnu.org/prep/standards/.
[9] Sun Microsystems, "Code conventions for the java programming language," http://www.oracle.com/technetwork/java/codeconvtoc-136057.html.
[10] B. W. Kernighan and R. Pike, *The practice of programming*. Boston, MA: Addison-Wesley Longman, 1999.
[11] H. Aman, S. Amasaki, T. Sasaki, and M. Kawahara, "Empirical analysis of change-proneness in methods having local variables with long names and comments," in *Proc. 2015 ACM/IEEE Int'l Symp. Empirical Softw. Eng. and Measurement*, Oct. 2015, pp. 50–53.
[12] M. J. Sousa and H. Moreira, "A survey on the software maintenance process," in *Proc. Int'l Conf. Softw. Maintenance*, Nov. 1998, pp. 265–274.
[13] R. P. Buse and W. R. Weimer, "A metric for software readability," in *Proc. 2008 Int'l Symp. Softw. Testing and Analysis*, 2008, pp. 121–130.
[14] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Boston, MA: Addison-Wesley Longman, 1999.
[15] H. Aman, S. Amasaki, T. Sasaki, and M. Kawahara, "Lines of comments as a noteworthy metric for analyzing fault-proneness in methods," *IEICE Trans. Inf. & Syst.*, vol. E98-D, no. 12, pp. 2218–2228, Dec. 2015.
[16] J. Śliwerski, T. Zimmermann, and A. Zeller, "When do changes induce fixes?" in *Proc. Int'l Workshop on Mining Softw, Repositories*, May 2005, pp. 1–5.
[17] H. Aman, "An empirical analysis of the impact of comment statements on fault-proneness of small-size module," in *Proc. 19th Asia-Pacific Softw. Eng. Conf.*, Dec. 2012, pp. 362–367.
[18] A. Agresti, *Categorical Data Analysis*, 2nd ed. N.J.: Wiley, 2002.
[19] K. Kawamoto and O. Mizuno, "Predicting fault-prone modules using the length of identifiers," in *Proc. 4th Int'l Workshop on Empirical Softw. Eng. in Practice*, Oct. 2012, pp. 30–34, Japan.
[20] D. Binkley, D. Lawrie, S. Maex, and C. Morrell, "Identifier length and limited programmer memory," *Science of Computer Programming*, vol. 74, no. 7, pp. 430–445, May 2009.