mining to predict the likelihood of a class to be refactored in the next two months using machine learning techniques. In their prediction models, they do not distinguish different types of refactorings; they only assess the fact that developers try to improve the design. In addition, data extraction from development history/repository is very well covered. Research has been carried out to detect and interpret groups of software entities that change together. These co-change relationships have been used for different purposes. Zimmermann *et al.* [67] have used historical changes to point developers to possible places that need change. In addition historical common code changes are used to cluster software artifacts [70], to predict source code changes by mining change history [71], to identify hidden architectural dependencies [72], or to use them as change predictors [73]. The closest work to our contribution is the study proposed by Ouni *et al.* [74] to prioritize refactoring opportunities for software maintenance using multi-objective optimization.

Profiling is a dynamic program analysis methodology where different attributes of code, such as memory size, frequency of instruction or method calls, etc. Most commonly, profiling is used to optimize program execution. However, profiling has been used in a number of software engineering tasks. Reps *et al.* [75] and Chilimbi *et al.* [76] used profiling to detect and locate software defects. Lutz *et al.* [77] used profiling to detect when safety critical systems are likely to fail due inappropriate instruction calls, memory limitations, etc. Profiling can identify, in practice, which methods are called most frequently in common execution chains [78]. Using this information, we could get a set of dynamic program analysis metrics. These metrics would serve as another candidate feature set that could predict for documentation prioritization.

13 Conclusion

In this paper, we researched whether static source code attributes and textual information in source code could be used to automatically prioritize documentation effort. To the best of our knowledge, this is the first work to conduct a study in order to evaluate documentation effort prioritization. We conducted two user studies to determine what sections of source code are most important to document. We surprisingly found that common static source code attributes are not good predictors of documentation effort. However, we found that textual analysis of source code and existing documents can be effective predictors of documentation effort priority.

Acknowledgements

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1313583. This work was also supported by the National Science Foundation CAREER Award under Grant No. CCF-1452959. Any opinions, findings, and conclusions expressed herein are the authors', and do not necessarily reflect those of the sponsors. The authors would like to thank the participants of the user studies for their valuable feedback. Additionally, the authors would like to thank the employees at ABB Corporate Research who took time and great consideration in our closed-source study. This work is supported in part by the NSF CCF-1452959 and CNS-1510329 grants. Any opinions, findings, and conclusions expressed herein are the authors and do not necessarily reflect those of the sponsors.

References

[1] B. Fluri, M. Wursch, and H. C. Gall, "Do code and comments co-evolve? on the relation between source code and comment changes," in *Proceedings of the 14th Working Conference on Reverse Engineering*, ser. WCRE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 70–79. [Online]. Available: http://dx.doi.org/10.1109/WCRE.2007.21

[2] M. Kajko-Mattsson, "A survey of documentation practice within corrective maintenance," *Empirical Softw. Engg.*, vol. 10, no. 1, pp. 31–55, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1023/B:LIDA.0000048322.42751.ca

[3] A. Forward and T. C. Lethbridge, "The relevance of software documentation, tools and technologies: a survey," in *Proceedings of the 2002 ACM symposium on Document engineering*, ser. DocEng '02. New York, NY, USA: ACM, 2002, pp. 26–33. [Online]. Available: http://doi.acm.org/10.1145/585058.585065

[4] T. C. Lethbridge, J. Singer, and A. Forward, "How software engineers use documentation: The state of the practice," *IEEE Softw.*, vol. 20, no. 6, pp. 35–39, Nov. 2003. [Online]. Available: http://dx.doi.org/10.1109/MS.2003.1241364

[5] T. Roehm, R. Tiarks, R. Koschke, and W. Maalej, "How do professional developers comprehend software?" in *Proceedings of the 2012 International Conference on Software Engineering*, ser. ICSE 2012. Piscataway, NJ, USA: IEEE Press, 2012, pp. 255–265. [Online]. Available: http://dl.acm.org/citation.cfm?id=2337223.2337254

[6] I. R. Katz and J. R. Anderson, "Debugging: An analysis of bug-location strategies," *Hum.-Comput. Interact.*, vol. 3, no. 4, pp. 351–399, Dec. 1987. [Online]. Available: http://dx.doi.org/10.1207/s15327051hci0304_2

[7] S. Bugde, N. Nagappan, S. Rajamani, and G. Ramalingam, "Global software servicing: Observational experiences at microsoft," in *Proceedings of the 2008 IEEE International Conference on Global Software Engineering*, ser. ICGSE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 182–191. [Online]. Available: http://dx.doi.org/10.1109/ICGSE.2008.18

[8] G. C. Murphy, D. Notkin, and K. Sullivan, "Software reflexion models: Bridging the gap between source and high-level models," *ACM SIGSOFT Software Engineering Notes*, vol. 20, no. 4, pp. 18–28, 1995.

[9] K. Inoue, R. Yokomori, H. Fujiwara, T. Yamamoto, M. Matsushita, and S. Kusumoto, "Component rank: relative significance rank for software component search," in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 14–24. [Online]. Available: http://dl.acm.org/citation.cfm?id=776816.776819

[10] M. Grechanik, C. Fu, Q. Xie, C. McMillan, D. Poshyvanyk, and C. Cumby, "A search engine for finding highly relevant applications," in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, vol. 1. IEEE, 2010, pp. 475–484.

[11] A. N. Langville and C. D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton, NJ, USA: Princeton University Press, 2006.

[12] J. Stylos and B. A. Myers, "Mica: A web-search tool for finding api components and examples," in *Proceedings of the Visual Languages and Human-Centric Computing*, ser. VLHCC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 195–202. [Online]. Available: http://dx.doi.org/10.1109/VLHCC.2006.32

[13] J. Stylos, B. A. Myers, and Z. Yang, "Jadeite: improving api documentation using usage information," in *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '09. New York, NY, USA: ACM, 2009, pp. 4429–4434. [Online]. Available: http://doi.acm.org/10.1145/1520340.1520678

[14] R. Holmes and G. C. Murphy, "Using structural context to recommend source code examples," in *Proceedings of the 27th international conference on Software engineering*, ser. ICSE '05. New York, NY, USA: ACM, 2005, pp. 117–125. [Online]. Available: http://doi.acm.org/10.1145/1062455.1062491

[15] J. Sillito, G. C. Murphy, and K. De Volder, "Asking and answering questions during a programming change task," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 434–451, Jul. 2008. [Online]. Available: http://dx.doi.org/10.1109/TSE.2008.26

[16] Y. Shmerlin, I. Hadar, D. Kliger, and H. Makabee, "To document or not to document? an exploratory study on developers motivation to document code," in *Advanced Information Systems Engineering Workshops*, ser. Lecture Notes in Business Information Processing, A. Persson and J. Stirna, Eds. Springer International Publishing, 2015, vol. 215, pp. 100–106. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-19243-7_10

[17] E. Tom, A. Aurum, and R. Vidgen, "An exploration of technical debt," *J. Syst. Softw.*, vol. 86, no. 6, pp. 1498–1516, Jun. 2013. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2012.12.052

[18] P. W. McBurney and C. McMillan, ""automatic source code summarization of context for java methods"," in *IEEE Transactions of Software Engineering (to appear)*, 2015.

[19] D. Binkley, "Source code analysis: A road map," in *2007 Future of Software Engineering*, ser. FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 104–119. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.27

[20] T. Menzies, J. Greenwald, and A. Frank, "Data mining static code attributes to learn defect predictors," *Software Engineering, IEEE Transactions on*, vol. 33, no. 1, pp. 2–13, Jan 2007.

[21] T. Menzies, Z. Milton, B. Turhan, B. Cukic, Y. Jiang, and A. Bener, "Defect prediction from static code features: current results, limitations, new approaches," *Automated Software Engineering*, vol. 17, no. 4, pp. 375–407, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10515-010-0069-5

[22] R. Subramanyam and M. Krishnan, "Empirical analysis of ck metrics for object-oriented design complexity: implications for software defects," *Software Engineering, IEEE Transactions on*, vol. 29, no. 4, pp. 297–310, April 2003.

[23] C. K. Roy, J. R. Cordy, and R. Koschke, "Comparison and evaluation of code clone detection techniques and tools: A qualitative approach," *Sci. Comput. Program.*, vol. 74, no. 7, pp. 470–495, May 2009. [Online]. Available: http://dx.doi.org/10.1016/j.scico.2009.02.007

[24] B. Dit, M. Revelle, M. Gethers, and D. Poshyvanyk, "Feature location in source code: a taxonomy and survey," *Journal of Software: Evolution and Process*, vol. 25, no. 1, pp. 53–95, 2013. [Online]. Available: http://dx.doi.org/10.1002/smr.567

[25] T. McCabe, "A complexity measure," *Software Engineering, IEEE Transactions on*, vol. SE-2, no. 4, pp. 308–320, Dec 1976.

[26] A. De Lucia, M. Di Penta, S. Stefanucci, and G. Ventuni, "Early effort estimation of massive maintenance processes," in *Software Maintenance, 2002. Proceedings. International Conference on*, 2002, pp. 234–237.

[27] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 452–461. [Online]. Available: http://doi.acm.org/10.1145/1134285.1134349

[28] N. Fenton and N. Ohlsson, "Quantitative analysis of faults and failures in a complex software system," *Software Engineering, IEEE Transactions on*, vol. 26, no. 8, pp. 797–814, Aug 2000.

[29] M. Shepperd and D. Ince, "A critique of three metrics," *Journal of Systems and Software*, vol. 26, no. 3, pp. 197 – 210, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0164121294900116

[30] S. R. Chidamber and C. F. Kemerer, "Towards a metrics suite for object oriented design," *SIGPLAN Not.*, vol. 26, no. 11, pp. 197–211, Nov. 1991. [Online]. Available: http://doi.acm.org/10.1145/118014.117970

[31] B. Henderson-Sellers, *Object-Oriented Metrics: Measures of Complexity*. Prentice Hall, 1996.

[32] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "Effective identifier names for comprehension and memory," *Innovations in Systems and Software Engineering*, vol. 3, no. 4, pp. 303–318, 2007. [Online]. Available: http://dx.doi.org/10.1007/s11334-007-0031-2

[33] A. A. Takang, P. A. Grubb, and R. D. Macredie, "The Effects of Comments and Identifier Names on Program Comprehensibility: An Experimental Study," *Journal of Programming Languages*, vol. 4, no. 3, pp. 143–167, 1996.

[34] P. McBurney and C. McMillan, "An empirical study of the textual similarity between source code and source code summaries," *Empirical Software Engineering*, pp. 1–26, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10664-014-9344-6

[35] Y. Li, D. McLean, Z. A. Bandar, J. D. O'Shea, and K. Crockett, "Sentence similarity based on semantic nets and corpus statistics," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 8, pp. 1138–1150, 2006.

[36] G. A. Miller, "Wordnet: A lexical database for english," *Commun. ACM*, vol. 38, no. 11, pp. 39–41, Nov. 1995. [Online]. Available: http://doi.acm.org/10.1145/219717.219748

[37] G. Salton, A. Wong, and C. S. Yang, "A vector space model for automatic indexing," *Commun. ACM*, vol. 18, no. 11, pp. 613–620, Nov. 1975. [Online]. Available: http://doi.acm.org/10.1145/361219.361220

[38] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, MA, USA: PWS Publishing Co., 1996.

[39] G. R. Finnie, G. E. Wittig, and J.-M. Desharnais, "A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models," *J. Syst. Softw.*, vol. 39, no. 3, pp. 281–289, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1016/S0164-1212(97)00055-1

[40] C. López-Martín and A. Abran, "Neural networks for predicting the duration of new software projects," *J. Syst. Softw.*, vol. 101, no. C, pp. 127–135, Mar. 2015. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2014.12.002

[41] A. Arcuri and G. Fraser, "Parameter tuning or default values? an empirical investigation in search-based software engineering," *Empirical Software Engineering*, vol. 18, no. 3, pp. 594–623, 2013. [Online]. Available: http://dx.doi.org/10.1007/s10664-013-9249-9

[42] F. Wilcoxon, *Individual Comparisons by Ranking Methods*, ser. Bobbs-Merrill Reprint Series in the Social Sciences, S541. Bobbs-Merrill, College Division.

[43] F. Deissenbock and M. Pizka, "Concise and consistent naming [software system identifier naming]," in *Program Comprehension, 2005. IWPC 2005. Proceedings. 13th International Workshop on*, May 2005, pp. 97–106.

[44] D. Lawrie, C. Morrell, H. Feild, and D. Binkley, "What's in a name? a study of identifiers," in *In 14th International Conference on Program Comprehension*. IEEE Computer Society, 2006, pp. 3–12.

[45] S. Butler, M. Wermelinger, Y. Yu, and H. Sharp, "Relating identifier naming flaws and code quality: An empirical study," in *Reverse Engineering, 2009. WCRE '09. 16th Working Conference on*, Oct 2009, pp. 31–35.

[46] ——, "Exploring the influence of identifier names on code quality: An empirical study," in *Software Maintenance and Reengineering (CSMR), 2010 14th European Conference on*, March 2010, pp. 156–165.

[47] G. Norman, "Likert scales, levels of measurement and the "laws" of statistics," *Advances in Health Sciences Education*, vol. 15, no. 5, pp. 625–632, 2010. [Online]. Available: http://dx.doi.org/10.1007/s10459-010-9222-y

[48] G. M. Sullivan and A. R. Artino Jr, "Analyzing and interpreting data from likert-type scales," *Journal of graduate medical education*, vol. 5, no. 4, pp. 541–542, 2013.

[49] B. Lantz, "Equidistance of likert-type scales and validation of inferential methods using experiments and simulations," *The Electronic Journal of Business Research Methods*, vol. 11, no. 1, pp. 16–28, 2013.

[50] M. H. Halstead, *Elements of Software Science (Operating and Programming Systems Series)*. New York, NY, USA: Elsevier Science Inc., 1977.

[51] N. Dragan, M. L. Collard, and J. I. Maletic, "Reverse engineering method stereotypes," in *2006 22nd IEEE International Conference on Software Maintenance*, Sept 2006, pp. 24–34.

[52] ——, "Automatic identification of class stereotypes," in *Software Maintenance (ICSM), 2010 IEEE International Conference on*, Sept 2010, pp. 1–10.

[53] L. Moreno and A. Marcus, "Jstereocode: Automatically identifying method and class stereotypes in java code," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE 2012. New York, NY, USA: ACM, 2012, pp. 358–361.

[54] J. Y. Gil and I. Maman, "Micro patterns in java code," in *Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, ser. OOPSLA '05. New York, NY, USA: ACM, 2005, pp. 97–116.

[55] P. W. McBurney and C. McMillan, "Automatic documentation generation via source code summarization of method context,"

in *Proceedings of the 22Nd International Conference on Program Comprehension*, ser. ICPC 2014. New York, NY, USA: ACM, 2014, pp. 279–290. [Online]. Available: http://doi.acm.org/10.1145/2597008.2597149

[56] S. Haiduc, J. Aponte, and A. Marcus, "Supporting program comprehension with source code summarization," in *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, vol. 2, May 2010, pp. 223–226.

[57] S. Haiduc, J. Aponte, L. Moreno, and A. Marcus, "On the use of automated text summarization techniques for summarizing source code," in *Proceedings of the 2010 17th Working Conference on Reverse Engineering*, ser. WCRE '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 35–44. [Online]. Available: http://dx.doi.org/10.1109/WCRE.2010.13

[58] T. K. Landauer, P. W. Foltz, and D. Laham, "An introduction to latent semantic analysis," *Discourse processes*, vol. 25, no. 2-3, pp. 259–284, 1998.

[59] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Using ir methods for labeling source code artifacts: Is it worthwhile?" in *Program Comprehension (ICPC), 2012 IEEE 20th International Conference on*, June 2012, pp. 193–202.

[60] B. Eddy, J. Robinson, N. Kraft, and J. Carver, "Evaluating source code summarization techniques: Replication and expansion," in *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, May 2013, pp. 13–22.

[61] W. Li and A. McCallum, "Pachinko allocation: Dag-structured mixture models of topic correlations," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 577–584. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143917

[62] P. Rodeghero, C. McMillan, P. W. McBurney, N. Bosch, and S. D'Mello, "Improving automated source code summarization via an eye-tracking study of programmers," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 390–401. [Online]. Available: http://doi.acm.org/10.1145/2568225.2568247

[63] L. Moreno, J. Aponte, S. Giriprasad, A. Marcus, L. Pollock, and K. Vijay-Shanker, "Automatic generation of natural language summaries for java classes," in *Proceedings of the 21st International Conference on Program Comprehension*, ser. ICPC '13, 2013.

[64] L. Moreno, A. Marcus, L. Pollock, and K. Vijay-Shanker, "Jsummarizer: An automatic generator of natural language summaries for java classes," in *Program Comprehension (ICPC), 2013 IEEE 21st International Conference on*, May 2013, pp. 230–232.

[65] G. Sridhara, E. Hill, D. Muppaneni, L. Pollock, and K. Vijay-Shanker, "Towards automatically generating summary comments for java methods," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ser. ASE '10. New York, NY, USA: ACM, 2010, pp. 43–52. [Online]. Available: http://doi.acm.org/10.1145/1858996.1859006

[66] A. Ying, G. Murphy, R. Ng, and M. Chu-Carroll, "Predicting source code changes by mining change history," *Software Engineering, IEEE Transactions on*, vol. 30, no. 9, pp. 574–586, Sept 2004.

[67] T. Zimmermann, A. Zeller, P. Weissgerber, and S. Diehl, "Mining version histories to guide software changes," *Software Engineering, IEEE Transactions on*, vol. 31, no. 6, pp. 429–445, June 2005.

[68] Q. Soetens, J. Perez, and S. Demeyer, "An initial investigation into change-based reconstruction of floss-refactorings," in *Software Maintenance (ICSM), 2013 29th IEEE International Conference on*, Sept 2013, pp. 384–387.

[69] J. Ratzinger, T. Sigmund, P. Vorburger, and H. Gall, "Mining software evolution to predict refactoring," in *Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on*, Sept 2007, pp. 354–363.

[70] H. Gall, K. Hajek, and M. Jazayeri, "Detection of logical coupling based on product release history," in *Proceedings of the International Conference on Software Maintenance*, ser. ICSM '98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 190–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850947.853338

[71] T. Gîrba, S. Ducasse, A. Kuhn, R. Marinescu, and R. Daniel, "Using concept analysis to detect co-change patterns," in *Ninth International Workshop on Principles of Software Evolution: In Conjunction with the 6th ESEC/FSE Joint Meeting*, ser. IWPSE '07. New York, NY, USA: ACM, 2007, pp. 83–89. [Online]. Available: http://doi.acm.org/10.1145/1294948.1294970

[72] A. E. Hassan and R. C. Holt, "Predicting change propagation in software systems," in *Proceedings of the 20th IEEE International Conference on Software Maintenance*, ser. ICSM '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 284–293. [Online]. Available: http://dl.acm.org/citation.cfm?id=1018431.1021436

[73] D. Beyer and A. Noack, "Clustering software artifacts based on frequent common changes," in *Proceedings of the 13th International Workshop on Program Comprehension*, ser. IWPC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 259–268. [Online]. Available: http://dx.doi.org/10.1109/WPC.2005.12

[74] A. Ouni, M. Kessentini, S. Bechikh, and H. Sahraoui, "Prioritizing code-smells correction tasks using chemical reaction optimization," *Software Quality Control*, vol. 23, no. 2, pp. 323–361, Jun. 2015. [Online]. Available: http://dx.doi.org/10.1007/s11219-014-9233-7

[75] T. Reps, T. Ball, M. Das, and J. Larus, *Software Engineering — ESEC/FSE'97: 6th European Software Engineering Conference Held Jointly with the 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering Zurich, Switzerland, September 22–25, 1997 Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, ch. The use of program profiling for software maintenance with applications to the year 2000 problem, pp. 432–449. [Online]. Available: http://dx.doi.org/10.1007/3-540-63531-9_29

[76] T. M. Chilimbi, B. Liblit, K. Mehra, A. V. Nori, and K. Vaswani, "Holmes: Effective statistical debugging via efficient path profiling," in *Proceedings of the 31st International Conference on Software Engineering*, ser. ICSE '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 34–44. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2009.5070506

[77] R. R. Lutz, "Software engineering for safety: A roadmap," in *Proceedings of the Conference on The Future of Software Engineering*, ser. ICSE '00. New York, NY, USA: ACM, 2000, pp. 213–226. [Online]. Available: http://doi.acm.org/10.1145/336512.336556

[78] S. L. Graham, P. B. Kessler, and M. K. Mckusick, "Gprof: A call graph execution profiler," in *Proceedings of the 1982 SIGPLAN Symposium on Compiler Construction*, ser. SIGPLAN '82. New York, NY, USA: ACM, 1982, pp. 120–126. [Online]. Available: http://doi.acm.org/10.1145/800230.806987