elements. It was also found that certain recurring patterns of anomaly combinations or anomaly propagations are better indicators of architectural problems than individual anomaly occurrences. Therefore, developers should be warned about the harmful impact of these patterns and their existence in the source code in order to perform their early removal. However, these patterns usually cannot be specified or detected by existing techniques [21, 32], as they are intended to pick out individual anomaly occurrences.

9. Acknowledgements

This was sponsored by: I.Macia CNPq grant 579604/2008-0; A.Garcia FAPERJ grant E-26/102.211/2009, 111.152/2011 and CNPq grant 305526/2009-0; A.v.Staa CNPq grant 306802/2008-2; Projects: CNPq grants 483882/2009-7, 479344/2010-8 and 485348/2011-0. It was also sponsored by the US National Science Foundation under Grant number 1117593. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Aldrich, J. ArchJava: Connecting Software Architecture to Implementation. In Proc of the 24th ICSE, pp. 187-197, 2002.

[2] Alikacem, E.H and Sahraoui, H. Generic metric extraction framework. In Proc. of the 16th IWSM/MetriKon, 2006, pp. 383–390.

[3] Bieman, J.M. and Kang, B.K. Cohesion and Reuse in an Object Oriented System. In Proc of the ISSR, pp 259-262, 1995.

[4] Clements, P et al. Documenting Software Architectures: Views and Beyond. Addison-Wesley, 2nd Edition, 2010

[5] Code smells study: http://www.inf.puc-rio.br/~ibertran/aosd12.

[6] D'Ambros, M. et al. the Impact of Design Flaws on Software Defects. In Proc. of the 10th QSIC, pp. 23 - 31, 2010.

[7] Dhambri et al. Visual Detection of Design Anomalies. In Proc. of the 12th CSMR, pp. 279-283, 2008.

[8] Eichberg, M. et al. Defining and Continuous Checking of Structural Program Dependencies. In Proc. of the 30th ICSE, 2008.

[9] Emden, E. and Moonen, L. Java quality assurance by detecting code smells. In Proceedings of the 9th ICRE, 2002.

[10] FEAT tool, http://www.cs.mcgill.ca/~swevo/feat/

[11] Ferrari, F. et al. An exploratory study of error-proneness in evolving Aspect-Oriented Programs. In: Proc. of the 25th OOPSLA, USA, 2009.

[12] Figueiredo, E. et al. Evolving software product lines with aspects: An empirical study on design stability. In Proc of the 30th ICSE, 2008.

[13] Fowler, M. Refactoring: Improving the Design of Existing Code. Addison-Wesley, 1999.

[14] Garcia, J. et al. Identifying architectural bad smells. In Proc of the. 13th CSMR, pp 255–258, 2009.

[15] Greenwood, P. et al. On the impact of aspectual decompositions on design stability: An empirical study. In Proc. of the 21st ECOOP, 2007.

[16] Hochstein, L. and Lindvall, M. Combating architectural degeneration: A survey. Info. & Soft. Technology July, 2005.

[17] Hosmer, D. and Lemeshow, S. Applied Logistic Regression (2nd Edition). Wiley, 2000.

[18] Khomh, K. et al. An exploratory study of the impact of code smells on software change-proneness. In Proc of the 16th WCRE, 2009.

[19] Kiczales, G.,et al. Aspect-oriented programming. In Proc. of the 11th ECOOP. LNCS, vol. 1241. Springer, Heidelberg. pp. 220-242, 1997.

[20] Kitchenham, B. et al. Evaluating guidelines for empirical software engineering studies. ISESE pp 38-47, 2006

[21] Lanza, M. and Marinescu, R. Object-Oriented Metrics in Practice. Springer, 2006.

[22] Lippert, M. and Roock, S. Refactoring in Large Software Projects: Performing Complex Restructurings Successfully. Wiley. 2006.

[23] Macia, I. et al. A. An Exploratory Study of Code Smells in Evolving Aspect-Oriented Systems. In Proc of the 10th AOSD, 2011.

[24] Malek, S. et al. Reconceptualizing a family of heterogeneous embedded systems via explicit architectural support. In Proc. of the 29th ICSE. 2007.

[25] Mantyla, M.V. and Lassenius, C. Subjective evaluation of software evolvability using code smells: An empirical study. Empirical Software Enggineering, vol. 11, no. 3, pp. 395–431, 2006.

[26] Mara, L. et al. Hist-Inspect: A Tool for History-Sensitive Detection of Code Smells. In Proc. of the 10th AOSD, 2011

[27] Marinescu, R. Detection strategies: Metrics-based rules for detecting design flaws. In Proc. of the 20th ICSM, pp 350-359, 2004.

[28] Marinescu,R.; Ganea, G. and Veredi, I. inCode: Continuous Quality Assessment and Improvement. In Proc of the 14th CSMR, 2010.

[29] Martin, R. Agile Principles, Patterns, and Practices. Prentice Hall, 2002.

[30] McCabe, T.J. A Software Complexity Measure. IEEE Transactions on Software Engineering, 2 (4), pp 308-320, 1976.

[31] Meyer, B. Object-Oriented Software Construction. Prentice Hall Professional Technical 2nd edition, 2000.

[32] Moha, N. et al. DECOR: A Method for the Specification and Detection of Code and Design Smells. IEEE TSE, 2010.

[33] Munro, MJ. Product metrics for automatic identification of bad smell design problems in java source-code. In Proc of 11th METRICS, 2005

[34] MuLATo tool, http://sourceforge.net/projects/mulato/ (3/08/2009)

[35] Murphy, G.C., et al.. Software Reflexion Models: Bridging the Gap between Design and Implementation. IEEE TSE, pp 364–380, 2001.

[36] Murphy-Hill, E. Scalable, expressive, and context-sensitive code smell display. In Proc of the 23rd OPSLA, 2008.

[37] Olbrich, S.M. et al. Are all code smells harmful? A study of God Classes and Brain Classes in the evolution of three open source systems. In Proc of the 26th ICSM pp 1-10, 2010.

[38] Olbrich, S.M. et al. The evolution and impact of code smells: A case study of two open source systems. In Proc of the 3rd ESEM, 2009.

[39] Perry, D.E. and Wolf, A.L. Foundations for the study of software architecture, ACM Software. Eng. Notes 17 (4) pp 40–52, 1992.

[40] Ratiu, D. et al. Using History Information to Improve Design Flaws Detection. In Proc of the 8th CSMR, 2004.

[41] Ratzinger, J. et al. Improving evolvability through refactoring. In Proc of the 5th IEEE MSR, 2005.

[42] Sant'anna, C. et al. On the modularity of software architectures: A Concern-Driven measurement framework. In Proc. of ECSA, 2007.

[43] Sonar: http://docs.codehaus.org/display/SONAR/

[44] Srivisut, K. and Muenchaisri, P. Bad-smell Metrics for Aspect-Oriented Software. In Proc of the 6th ICIS, 2007.

[45] Together: http://www.borland.com/us/products/together/

[46] Tsantalis, N. and Chatzigeorgiou, A. Identification of move method refactoring opportunities. IEEE TSE, 35(3), pp 347–367, 2009.

[47] Understand: http://www.scitools.com/

[48] Wake, W.C. Refactoring Workbook. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[49] Wettel, R. and Lanza, M. Visually localizing design problems with disharmony maps. In Proc. of the 4th Softvis pp. 155–164, 2008.