References

[Abbes *et al.*, 2011] ABBES, M.; F. KHOMH, F.; GUEHENEUC, Y-C.; ANTONIOL, G. **An Empirical Study of the Impact of Two Antipatterns Blob and Spaghetti Code on Program Comprehension**. In Proceedings of the 15th European Conference on Software Maintenance and Reengineering (CSMR), pp. 181-190, 2011.

[Aldrich, 2002] ALDRICH, J. **ArchJava: Connecting Software Architecture to Implementation**. In Proceedings of the 24th International Conference on Software Engineering (ICSE), pp. 187-197, 2002.

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

[Antoniol *et al.*, 1997] ANTONIOL, G.; FIUTEM, R.; LUTTERI, G.; TONELLA, P.; ZANFEI, S.; MERLO, E. **Program understanding and maintenance with the CANTO environment**. In Proceedings of the International Conference on Software Maintenance (ICSM), page 72, 1997.

[Anquetil *et al.*, 2011] ANQUETIL, N.; LAVAL, J. **Legacy Software Restructuring: Analyzing a Concrete Case**. In Proceedings of the 15th European Conference on Software Maintenance and Reengineering (CSMR), pp 279-286, 2011.

[Arcoverde *et al.*, 2011] ARCOVERDE, R.; GARCIA, A. and FIGUEIREDO, E. **Understanding the Longevity of Code Smells: Preliminary Results of an Explanatory Survey**. In Proceedings of the 4th International Workshop on Refactoring Tools (WRT 2011), held in conjunction with the 33rd International Conference on Software Engineering (ICSE), May 2011.

[Arcoverde *et al.*, 2012] ARCOVERDE, R.; MACIA, I.; GARCIA, A.; STAA, A. **Automatically Detecting Architecturally-Relevant Code Anomalies**. In Proceedings of the 3rd International Workshop on Recommendation Systems for Software Engineering, held in conjunction with the 34th International Conference on Software Engineering (ICSE). 2012.

[Arcoverde, 2012] ARCOVERDE, R. Prioritization of Code Anomalies Based on Architecture Sensitiveness. MSc Dissertation in Informatics, Pontifical Catholic University of Rio de Janeiro (PUC-Rio), pp. 89, 2012.

[Arisholm, 2002] ARISHOLM, E. **Dynamic Coupling Measures for object-oriented Software**. In Proceedings of the 8th International Symposium on Software Metrics (METRICS), pp. 33-42, 2002.

[Axivion, 2010] Axivion: http://www.axivion.com/

[Baldwin and Clark, 2000] BALDWIN, C.Y. and CLARK, K.B. Design Rules, Vol. 1: The Power of Modularity. MIT Press, 2000.

[Basili *et al.*, 1994] BASILI, V.R.; CALDIERA, G.; ROMBACH, H.D. 1994. **The goal question metric approach**. In Encyclopedia of Software Engineering. Wiley, 1994.

[Bass *et al.*, 2003] BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice** (Second Edition). Addison-Wesley Professional, 2003.

[Bergmans *et al.*, 1992] BERGMANS, L.; MEHMET, S.; BOSCH, J. **Composition filters: extended expressiveness for OOPLs**. Position paper for the object-oriented Programming, Systems, Languages, and Applications (OOPSLA): The Next Generation, Vancouver, 1992.

[Booch, 2007] BOOCH, G. **The economics of architecture**. IEEE Software, 24(5):18-20, 2007.

[Bouwers *et al.*, 2011] BOUWERS, E.; DEURSEN, E.; VISSER, J. **Dependency profiles for software architecture evaluations**. In Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM), pp. 540-543, 2011.

[Buschmann *et al.*, 2007] BUSCHMANN, F.; HENNEY, K.; SCHMIDT, D.C. **Pattern-Oriented Software Architecture: A Pattern Language for Distributed Computing**. In Wiley Software Patterns Series, John Wiley & Sons. 2007.

[Carneiro *et al.* 2010] CARNEIRO, G.; SILVA, M.; MARA, L.; FIGUEIREDO, E.; SANT'ANNA, C.; GARCIA, A.; MENDONCA, M. **Identifying code smells with multiple concern views**. In Proceedings of the Brazilian Symposium on Software Engineering (SBES), p. 128–137, 2010.

[Carneiro *et al.*, 2008a] CARNEIRO, G.; MAGNAVITA, R.; MENDONÇA, M. **Combining Software Visualization Paradigms to Support Software Comprehension Activities**. In Proceedings of the 4th ACM Symposium on Software Visualization (SoftViz), demo session, Ammersee, Germany: ACM, 2008, pp. 201-202.

[Carneiro *et al.*, 2008b] CARNEIRO, G.; MAGNAVITA, R.; SPINOLA, E.; SPINOLA, F; MENDONÇA, M. **Evaluating the Usefulness of Software Visualization in Supporting Software Comprehension Activities**. In Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), pp. 276-278, 2008.

[Chae *et al.*, 2000] CHAE H.S., KWON, Y.R.; HWAN BAE, D. **A Cohesion Measure for object-oriented Classes**. Journal Software Practice & Experience, Volume 30 Issue, pp. 1405-1431, 2000.

[Chase *et al.*, 1998] CHASE, M.P.; CHRISTEY, S.M.; HARRIS, D.R.; YEH, A.S. **Recovering software architecture from multiple source code analyses**. In Proceedings of the ACM SIGPLAN-SIGSOFT Workshop Program Analysis Software Tools and Enginerring, pp. 43 - 50 1998.

[Chidamber and Kemerer, 1994] CHIDAMBER, S.R. and KEMERER, C.F. **A Metrics Suite for Object Oriented Design**. IEEE Transactions on Software Engineering (TSE), vol. 20, 1994, pp. 476-493.

[Clements *et al.*, 2002] CLEMENTS, P.; BACHMANN, F.; BASS, L.; GARLAN, D.; IVERS, J.; LITTLE, R.; NORD, R.; STADORD, J. **Documenting Software Architectures: Views and Beyond**. Addison-Wesley Professional, 2002.

[Counsell *et al.*, 2010a] COUNSELL, S; HIERONS, R.M.; HAMZA, H.; BLACK, S.; DURRAND, M. **Exploring the Eradication of Code Smells: An Empirical and Theoretical Perspective**, Advances in Software Engineering, 2010.

[Counsell *et al.*, 2010b] COUNSELL, S.; HAMZA, H.; HIERONS, R.M. **An Empirical Investigation of Code Smell "Deception" and Research Contextualisation through Paul's Criteria**. Journal of Computing and Information Technology, pp. 333–340 2010.

[Coverity, 2011] Coverity: http://www.coverity.com/

[D'Ambros *et al.*, 2010] D'AMBROS, M.; BACCHELLI, A. and LANZA, M. **On the Impact of Design Flaws on Software Defects**. In Proceedings of the 10th International Conference on Quality Software (QSIC), pp. 23-31, 2010.

[Deligiannis *et al.*, 2003a] DELIGIANNIS, I.; M. SHEPPERD, M.; ROUMELIOTIS, M.; STAMELOS, I. **An Empirical Investigation of an object-oriented Design Heuristic for Maintainability**. Journal of Systems and Software, Volume 65 Issue 2, pp. 127-139, 2003.

[Deligiannis *et al.*, 2003b] DELIGIANNIS, I.; STAMELOS, I.; ANGELIS, L.; ROUMELIOTIS, M.; SHEPPERD, M. **A Controlled Experiment Investigation of an object-oriented Design Heuristic for Maintainability**. Journal of Systems and Software, Volume 72 Issue 2, pp. 129-143, 2004.

[DeMarco, 1979] DEMARCO, T. **Structured Analysis and System Specification**. Yourdon Press, 1979.

[Dig *et al.*, 2006] DIG, D.; COMERTOGLU, C.; MARINOV, D. and JOHNSON, R. **Automated Detection of Refactorings in Evolving Components**. In Proceedings of the 20th European conference on object-oriented Programming (ECOOP), pp 404-428, 2006.

[Dósea *et al.*, 2007] DÓSEA, M.; COSTA, A., BORBA, P.; SOARES, S. **Specifying design rules in aspect-oriented systems**. In I Latin American Workshop on aspect-oriented Software Development - LA-WASP, affiliated with SBES'07, 2007.

[Ducasse, 2006] DUCASSE, S. **Distribution Map**. In Proceedings of 22nd IEEE International Conference on Software Maintenance (ICSM), pp. 203–212, 2006.

[Eclipse, 2009] FOUNDATION, T. E.. Eclipse java development tools (jdt). http://www.eclipse.org/jdt/, 2009. Accessed in July.

[Eisenbarth *et al.*, 2003] EISENBARTH, T.; KOSCHKE, R.; SIMON, D. **Locating features in source code**. IEEE Transitions on Software Engineering (TSE), pp. 210–224, 2003.

[Eichberg *et al.*, 2008] EICHBERG, M.; KLOPPENBURG, S.; KLOSE, K. and MEZINI, M. **Defining and Continuous Checking of Structural Program Dependencies**. In Proceedings of the 30th International Conference on Software engineering (ICSE), 2008.

[Eick *et al.*, 2001] EICK, S.G. GRAVES, T.L.; KARR, A.F.; *et al*. **Does code decay? Assessing the evidence from change management data**. IEEE Transactions on Software Engineering , Volume 27 Issue 1, pp. 1–12, 2001.

[Eisenbarth *et al.*, 2003] EISENBARTH, T.; KOSCHKE, R.; SIMON, D. **Locating features in source code**. IEEE Transactions on Software Engineering (TSE), volume 29 issue 3, pp. 210–224, 2003.

[Emden and Moonen, 2002] EMDEN, E. and MOONEN, L. **Java quality assurance by detecting code smells**. In Proceedings of the 9th Working Conference on Reverse Engineering (WCRE), page 97, 2002.

[XP, 2010] Extreme Programming http://www.extremeprogramming.org/

[FEAT, 2009] FEAT tool, http://www.cs.mcgill.ca/~swevo/feat/

[Fenton and Pfleeger, 1996] FENTON, N.E. and PFLEEGER, S.L. **Software Metrics: A Rigorous and Practical Approach**, Pws Pub Co, 1996.

[Ferrari *et al.*, 2010] FERRARI, F.; BURROWS, R.; LEMOS, O.; GARCIA, A.; FIGUEIREDO, E.; CACHO, N.; LOPES, F.; TEMUDO, N.; SILVA, L.; SOARES, S.; RASHID, A.; MASIERO, P.; BATISTA, T. ; MALDONADO, J. **An exploratory study of fault-proneness in evolving aspect oriented programs**. In Proceedings of the International Conference on Software Engineering (ICSE), pp. 65–74, 2010. ACM.

[Figueiredo *et al.*, 2008] FIGUEIREDO, E.; CACHO, N.; SANT'ANNA, C.; MONTEIRO, M.; KULESZA, U.; GARCIA, A.; SOARES, S.; FERRARI, F.; KHAN, S.; CASTOR, F. ; DANTAS, F. **Evolving software product lines with aspects: An empirical study on design stability**. In Proceedings of the 30th International Conference on Software engineering (ICSE), pp. 261-270, 2008.

[Figueiredo *et al.*, 2009] FIGUEIREDO, E.; SILVA, B.; SANT'ANNA, C.; GARCIA, A.; WHITTLE, J.; NUNES, D. **Crosscutting patterns and design stability: An exploratory analysis**. In Proceedings of the IEEE 17th International Conference on Program Comprehension (ICPC), pp. 138-147, 2009.

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

[Gamma *et al.*, 1995] GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design Patterns: Elements of Reusable object-oriented Software**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 1995.

[Garcia *et al.*, 2009] GARCIA, J.; POPESCU, D.; EDWARDS, G. and MEDVIDOVIC, N. **Identifying architectural bad smells**. In Proceedings of the 13th European Conference on Software Maintenance and Reengineering (CSMR), pp 255–258, 2009.

[Garcia *et al.*, 2011] GARCIA, J.; POPESCU, D.; MATTMANN, C. *et al*. **Enhancing architectural recovery using concerns**. In Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 552-555, 2011.

[Gîrba *et al.*, 2004] GÎRBA, T.; DUCASSE, S.; MARINESCU, R. **Identifying Entities that Change Together**. In Proceedings of the 9th IEEE Workshop on Empirical Studies of Software Maintenance, 2004.

[Godfrey and Lee, 2000] GODFREY, M. and LEE, E. **Secrets from the monster: Extracting mozilla's software architecture**. In Proceedings of the 2nd International Symposium on Constructing Software Engineering Tools (CoSET), 2000.

[Gorton, 2006] GORTON, I. **Essential Software Architecture**. Springer-Verlag, 2006.

[Greenwood *et al.*, 2006] GREENWOOD, P. BARTOLOMEI, T. FIGUEIREDO, E.; *et al*. **On the impact of aspectual decompositions on design stability: An empirical study**. In Proceedings of the 21st European conference on object-oriented Programming (ECOOP), 2007.

[van Gurp and Bosch, 2002] GURP, J.V. and BOSCH, J. **Design erosion: Problems and causes**. Journal of Systems and Software, Volume 61 Issue 2, pp. 105-119, 2002.

[Han *et al.*, 2005] HAN, Y.; KNIESEL, G.; CREMERS, A.B. Towards Visual AspectJ by a Meta Model and Modeling Notation. In Proceedings of the Workshop on aspect-oriented Modeling, 2005.

[Hannemann and Kiczales, 2002] HANNEMANN, J. and KICZALES, G. **Design Pattern Implementation in Java and AspectJ**. In Proceedings of the 17th ACM SIGPLAN Conference on object-oriented Programming, Systems, Languages, and Applications (OOPSLA), pp. 161-173, 2002.

[Hannemann *et al.*, 2005] HANNEMANN, J., MURPHY, G. and KICZALES, G**. Role- based refactoring of crosscutting concerns**. In Proceedings of the 4th annual international conference on Aspect-oriented Software Development (AOSD), pp. 135-146, 2005.

[Harrison and Ossher, 1993] HARRISON, W.; OSSHER, H. **Subject-oriented programming: a critique of pure objects**. SIGPLAN Not. 28, 10, pp. 411-428, 1993.

[Hashimoto, 2008] HASHIMOTO, M. **Diff/TS: A Tool for Fine-Grained Structural Change Analysis**. In Proceedings of the 15th Working Conference on Reverse Engineering (WCRE'08), 2008.

[Health Watcher, Aspectual Watcher 2009] Health Watcher: http://www.comp.lancs.ac.uk/~greenwop/tao.

[Herrera *et al.*, 2012] HERRERA, J.; MACIA, I.; SALAS, P.; PINHO, R.; VARGAS, R.; GARCIA, A.; ARAUJO, J.; Breitman, K. **Revealing Crosscutting Concerns in Textual Requirements Documents: An Exploratory Study with Industry Systems**. In Proceedings of the ACM SIGSoft XXVI Brazilian Symposium on Software Engineering (SBES), 2012.

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

[Hopkins, 2002] HOPKINS, P. **Static and Dynamic Correlations in some Models of Fluids**. PhD Thesis, University of Bristol, 2002.

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

[iBATIS, 2009] iBATIS: Data Mapper - http://ibatis.apache.org/.

[IEEE1471, 2012] http://www.iso-architecture.org/ieee-1471/

[Intensional Views, 2011] Intensional Views: http://www.intensive.be/

[Iwamoto and Zhao, 2003] IWAMOTO, M. and ZHAO, J. **Refactoring aspect-oriented programs**. In Proceedings of the 2sd annual international conference on Aspect-oriented Software Development (AOSD), 2003.

[Jansen and Bosch, 2005] JANSEN, A. and BOSCH, J. **Software architecture as a set of architectural design decisions**. In Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA), pp. 109-120, 2005.

[Jürgens *et al.*, 2009] JÜRGENS, E.; DEISSENBOECK, F.; HUMMEL B.; WAGNER, S. **Do Code Clones Matter?** In Proceedings of the 31st International Conference on Software Engineering (ICSE), pp. 485-495, 2009.

[Kazman and Carriere, 1998] KAZMAN, R.; CARRIERE, S.J. **View extraction and view fusion in architectural understanding**. In Proceedings of the 5th International Conference on Software Reuse (ICSR), pp. 290–299, 1998.

[Khomh *et al.*, 2009] KHOMH, K.; DI PENTA, M. and GUEHENEUC. Y. **An exploratory study of the impact of code smells on software change-proneness**. In Proceedings of the 16th Working Conference on Reverse Engineering (WCRE), pp. 75- 84, 2009.

[Kiczales *et al.*, 1997] KICZALES, G.; LAMPING, J.; MENDHEKAR, A.; MAEDA, A.; LOPES, C.; LOINGTIER, J.; IRWIN, J. **Aspect-oriented programming**. In Proceedings of the 11th European conference on object-oriented Programming (ECOOP), pp. 220-242, 1997.

[Kim *et al.*, 2011] KIM, M.; CAI, D. and KIM, S. *et. al*. **An Empirical Investigation into the Role of API-Level Refactorings during Software Evolution**. In Proceedings of 33rd International Conference on Software engineering (ICSE), pp. 151-160, 2011.

[Kitchenham, *et al.*, 2006] KITCHENHAM, B.; AL-KHILIDAR, H.; BABAR, M.; BERRY, M.; COX, K.; KEUNG, J.; Kurniawati, F.; Staples, M.; Zhang, H.; ZHU, L. **Evaluating guidelines for empirical software engineering studies**. In Proceedings of the ACM/IEEE international symposium on Empirical software engineering (ISESE), pp 38-47, 2006.

[Klocwork, 2010] Klocwork: http://www.klocwork.com/

[Knodel *et al.*, 2008] KNODEL, J.; MUTHIG, D.; HAURY, U.; MEIER, G. **Architecture compliance checking - experiences from successful technology transfer to industry**. In Proceedings of the 12th European Conference on Software Maintenance and Reengineering (CSMR), pp. 43 52. 2008

[Koschke and Simon, 2003] Koschke, R.; Simon, D. **Hierarchical Reflexion Models**. In Proceedings of the 10th Working Conference on Reverse Engineering (WCRE) page 36, 2003.

[Lanza, 2004] LANZA, M. **CodeCrawler - Polymetric Views in Action**. In Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE), IEEE Computer Society, pp. 394-395, 2004.

[Lanza and Marinescu, 2006] LANZA, M. and MARINESCU, R. **object-oriented Metrics in Practice**. Springer, 2006.

[Lanza and Ducasse, 2003] LANZA, M; DUCASSE, S. **Polymetric Views - A Lightweight Visual Approach to Reverse Engineering**. IEEE Transactions on Software Engineering (TSE), vol. 29, pp. 782-795, 2003.

[LePUS3, 2011] LePUS3: http://www.lepus.org.uk/

[Li and Shatnawi, 2007] LI, W.; and SHATNAWI, R. **An Empirical Study of the Bad Smells and Class Error Probability in the Post-Release object-oriented System Evolution**. Journal of Systems and Software archive Volume 80 Issue 7, pp. 1120-1128, 2007.

[Lions *et al.*, 2002] LIONS, J.M.; SIMONEAU, D.; PITETTE, G.; MOUSSA, I. Extending OpenTool/UML Using Metamodeling: An Aspect Oriented Programming Case Study. In Proceedings of the 2nd Workshop on aspect-oriented Modeling with UML, 2002.

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

[Liu, 2011] LIU, H. **Schedule of Bad Smell Detection and Resolution: A New Way to Save Effort**. IEEE Transactions on Software Engineering (TSE), volume 8, issue 1, pp. 220-235 2011.

[Lozano and Wermelinger, 2008] LOZANO, A and WERMELINGER, M. **Assessing the Effect of Clones of Changeability.** In Proceedings of the 24th International Conference on Software Maintenance (ICSM), pp. 227-236, 2008.

[Lung, 1998] LUNG, CH. **Software architecture recovery and restructuring through clustering techniques**. In Proceedings of the 3rd International Workshop on Software Architecture, ACM Press, Orlando, FL, pp. 101–104, 1998

[Lungu and Lanza, 2007] LUNGU, M. and LANZA, M**. Exploring Inter-Module Relationships in Evolving Software Systems**. In Proceedings of the 11th European Conference on Software Maintenance and Reengineering (CSMR), pp. 91-102, 2007.

[Maccormack *et al.*, 2006] MACCORMACK, A.; RUSNAK, J.; BALDWIN, C.Y. **Exploring the structure of complex software designs: An empirical study of open source and proprietary code**. Management Science, Volume 52 Issue 7, 2006.

[Macia *et al.*, 2013] MACIA, I.; GARCIA, A.; CHAVEZ, C.; STAA, A. **Enhancing the Detection of Code Anomalies with Architecture-Sensitive Strategies**. In Proceedings of the 17th European Conference on Software Maintenance and Reengineering (CSMR), 2013 (to appear).

[Macia *et al.*, 2012a] MACIA, I.; GARCIA, J.; POPESCU, D.; GARCIA, A.; MEDVIDOVIC, N.; STAA, A. **Are Automatically-Detected Code Anomalies Relevant to Architecture Modularity? An Exploratory Analysis of Evolving Systems**. In Proceedings of the 11th annual international conference on Aspect-oriented Software Development (AOSD), pp. 167-178, 2012.

[Macia *et al.*, 2012b] MACIA, I.; ARCOVERDE, R.; GARCIA, A.; CHAVEZ, C.; STAA, A. **On the Relevance of Code Anomalies for Identifying Architecture Degradation Symptoms**. In Proceedings of the 16th European Conference on Software Maintenance and Reengineering (CSMR), pp. 277-286, 2012.

[Macia *et al.*, 2012c] MACIA, I.; ARCOVERDE, R.; CIRILO, E.; GARCIA, A.; STAA, A. **Supporting the Identification of Architecturally-Relevant Code Anomalies**. In Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM), 2012.

[Macia, 2011a] MACIA, I. **Revealing Architecturally-Relevant Flaws in Aspectual Decompositions**. In Proceedings of the 10th International Conference on aspect-oriented Software Development (AOSD), Fourth place at ACM Competition, pp. 85–86, 2011. ACM.

[Macia, 2011b] MACIA, I. **Detecting Architecturally-Relevant Code Smells in Evolving Software Systems**. In Proceedings of the 33rd International Conference on Software Engineering (ICSE) - Doctoral Symposium, Hawaii, USA, 2011.

[Macia *et al.*, 2011a] MACIA, I.; GARCIA, A. and STAA, A. **An Exploratory Study of Code Smells in Evolving aspect-oriented Systems**. In Proceedings of the 10th annual International Conference on Aspect-oriented Software Development (AOSD), pp. 203-214, 2011.

[Macia *et al.*, 2011b] MACIA, I.; GARCIA, A.; STAA, A.; GARCIA, J. and MEDVIDOVIC, N. **On the Impact of aspect-oriented Code Smells on Architecture Modularity: An Exploratory Study**. In Proceedings of the 5th Brazilian Symposium on Software Components, Architectures and Reuse (SBCARS), 2011.

[Macia *et al.*, 2010] MACIA, I.; GARCIA, A.; STAA, A. **Defining and Applying Detection Strategies for aspect-oriented Code Smells**. In: ACM SIGSoft XXIII Brazilian Symposium on Software Engineering (SBES), 2010, Salvador. Proceedings of the ACM SIGSoft XXIII Brazilian Symposium on Software Engineering (SBES), 2010.

[Malek *et al.*, 2007] MALEK, S.; SEO, C.; RAVULA, S.; *et al*. **Reconceptualizing a family of heterogeneous embedded systems via explicit architectural support**. In Proceedings of 29th International Conference on Software Engineering (ICSE), pp. 591-601, 2007.

[Mäntylä and Lassenius, 2006] MÄNTYLÄ, M.V. and LASSENIUS, C. **Subjective evaluation of software evolvability using code smells: An empirical study**. Empirical Software Engineering, Volume 11, no. 3, pp. 395–431, 2006.

[Mäntylä *et al.*, 2003] MÄNTYLÄ, M.; VANHANEN, J.; LASSENIUS, C. **A taxonomy and an initial empirical study of bad smells in code**. In Proceedings of the 19th International Conference on Software Maintenance (ICSM), pp. 381- 384, 2003.

[Maqbool and Babri, 2007] MAQBOOL, O. and BABRI, H. **Hierarchical clustering for software architecture recovery**. IEEE Transactions on Software Engineering (TSE), Volume 33 Issue 11, pp. 759–780, 2007.

[Mara *et al.*, 2011] MARA, L.; HONORATO, G.; DANTAS, F.; *et al*. **Hist-Inspect: A Tool for History-Sensitive Detection of Code Smells**. In Proceedings of the 10th annual International Conference on Aspect-oriented Software Development (AOSD), 2011.

[Marinescu, 2004] MARINESCU, R. **Detection strategies: metrics-based rules for detecting design flaws**. In Proceedings of 20th IEEE International Conference on Software Maintenance (ICSM), pp. 350-359, 2004.

[Marinescu *et al.*, 2004] MARINESCU, R.; DUCASSE, S.; GÎRBA, T. **Evolution-Enriched Detection of God Classes**. In Proceedings of the 2nd CAVIS. pp. 3-7. 2004.

[Marinescu *et al.*, 2010] MARINESCU, R.; GANEA, G.; VEREDI, I. **InCode: Continuous Quality Assessment and Improvement**. In Proceedings of the 14th European Conference on Software Maintenance and Reengineering (CSMR), pp. 274-275, 2010.

[Martin, 2003] MARTIN, R. **Agile Software Development: Principles, Patterns, and Practices**. Prentice Hall, 2003.

[Marwan and Aldrich, 2009] MARWAN A. and ALDRICH, J. **Static Extraction and Conformance Analysis of Hierarchical Runtime Architectural Structure using Annotations**. In Proceedings of the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications (OOPSLA), pp. 321-340, 2009.

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

[Metha *et al.*, 2000] MEHTA, N.R.; MEDVIDOVIC, N.; PHADKE, S. **Towards a taxonomy of software connectors**. In Proceedings of the 22nd International Conference on Software Engineering (ICSE), 2000.

[Meyer, 1997] MEYER, B. **object-oriented Software Construction**. Prentice-Hall first edition, 1997.

[Meyer, 1994] MEYER, B. **An object-oriented Environment: Principles and Application**. Prentice-Hall, 1994.

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

[Monteiro and Fernandez, 2005] MONTEIRO, M.P.; FERNANDEZ, J.M. **Towards a catalog of aspect-oriented refactorings**. In Proceedings of the 4th International Conference on Aspect-oriented software development (AOSD), pp. 111-122, 2005.

[Morgan, 2007] MORGAN, C. **A static aspect language of checking design rules**. In Proceedings of the 6th annual International Conference on Aspect-oriented Software Development (AOSD), pp. 63-72, 2007.

[MuLATo, 2009] MuLATo: http://sourceforge.net/projects/mulato/

[Muller *et al.*, 1993] MULLER, H.A.; ORGUN, M.A.; TILLEY, S.R.; UHL, J.S. **A Reverse Engineering Approach to Subsystem Structure Identification**. Journal Software Maintenance 5 (4), pp. 181–204, 1993.

[Munro, 2005] MUNRO, MJ. **Product Metrics for Automatic Identification of "Bad Smell" Design Problems in Java Source-Code**. In Proceedings of the 11th International Symposium on Software Metrics (METRICS), page 15, 2005

[Murphy *et al.*, 2001] MURPHY, G.C., NOTKIN, D.; SULLIVAN, K. **Software Reflexion Models: Bridging the Gap between Design and Implementation**. IEEE Transactions on Software Engineering (TSE), volume 27 Issue 4, pp 364–380, 2001.

[Murphy-Hill *et al.*, 2009] MURPHY-HILL, E., PARNIN, C., and BLACK A. **How we refactor, and how we know it**. In Proceedings of 31st International Conference on Software Engineering (ICSE), pp. 287-297, 2009.

[Murphy-Hill, 2008] MURPHY-HILL, E. **Scalable, expressive, and context-sensitive code smell display**. In Proceedings of the 23rd ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications (OPSLA), pp. 771-772, 2008.

[Myers and Well, 2003] MYERS, J.L.; WELL, A.D. Research Design and Statistical Analysis (2nd Edition), Lawrence Erlbaum, 2003.

[Nguyen *et al.*, 2011] NGUYEN, T.; NGUYEN, H.; NGUYEN, H.; NGUYEN, T. **Aspect Recommendation for Evolving Software**. In Proceedings of the 33rd ACM/IEEE International Conference on Software Engineering (ACM/IEEE ICSE 2011), 2011.

[Novais *et al.*, 2012] NOVAIS, R.; NUNES, C.; LIMA, C.; CIRILO, E.; DANTAS, F.; GARCIA, A.; MENDONCA, M. **On the proactive and interactive visualization for feature evolution comprehension: An industrial investigation**. In Proceedings of the International Conference on Software Engineering (ICSE), Software Engineering In Practice, pp. 1044–1053, 2012.

[Novais *et al.*, 2011] NOVAIS, R.; CARNEIRO, G.; SIMÕES, P.; MENDONÇA, M. **On the use of software visualization to analyze software evolution - an interactive differential approach**. In Proceedings of the International Conference on Enterprise Information Systems (ICEIS), pp. 15–24, 2011.

[Nunes *et al.*, 2011] NUNES, C.; GARCIA, A.; FIGUEIREDO, E. ; LUCENA, C. **Revealing mistakes in concern mapping tasks: An experimental evaluation**. In: Proceedings of the European Conference on Software Maintenance and Reengineering (CSMR), p. 101–110, 2011.

[Olbrich *et al.*, 2010] OLBRICH, S.M.; CRUZES, D.S.; SJOBERG, D.I.K. Are all code smells harmful? **A study of God Classes and Brain Classes in the evolution of three open source systems**. In Proceedings of 26th IEEE International Conference on Software Maintenance (ICSM), pp. 1-10, 2010.

[Olbrich *et al.*, 2009] OLBRICH, S.M. CRUZES, D.S.; Basili, V.; Zazworka, N. **The Evolution and Impact of Code Smells: A Case Study of two Open Source Systems**. In Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 390-400, 2009.

[Oliveira, 2011] OLIVEIRA, M. **PREViA: An Approach for Visualizing the Evolution of Software Models**. Master Thesis. COPPE/UFRJ, 185p. 2011.

[Parnas 1994] PARNAS, D.L. **Software aging**. In Proceedings of the IEEE International Conference on Software Maintenance (ICSM), pp. 279–287, Los Alamitos, CA, USA, 1994. IEEE Computer Society Press.

[PDL, 2011] PDL: http://pdl.perl.org/.

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

[Piveta, *et al.*, 2006] PIVETA, E.K.; HECHT, M.; PIMENTA, M.S.; PRICE, P.T. Detecting bad smells in AspectJ. Journal of Universal Computer Science, vol. 12, 2006.

[Prehofer, 1997] PREHOFER, C. 1997. Feature-oriented programming: A fresh look at objects. In Proceedings of the 11th European conference on object-oriented Programming (ECOOP), pp. 419-443, 1997.

[Rahman *et al.*, 2010] Rahman, F.; Bird, C.; Devanbu, P. **Clones: What is that Smell?** In Proceedings of the 7th IEEE International Workshop on Mining software Repositories (MSR), pp. 72-81, 2010.

[Rajan and Sullivan, 2005] RAJAN, H.; SULLIVAN, K.J. **Classpects: unifying aspect- and object-oriented language design**. In Proceedings of the 26th International Conference on Software Engineering (ICSE), pp. 59-68, 2011.

[Ratiu *et al.*, 2004] RATIU, D.; DUCASSE, S.; GÎRBA, T.; Marinescu, R. **Using History Information to Improve Design Flaws Detection**. In Proceedings of the 8th European Conference on Software Maintenance and Reengineering (CSMR), pp. 223, 2004.

[Ratzinger, 2005] RATZINGER, J.; FISCHER, M.; GALL, H. Improving evolvability through refactoring. In Proceedings of the 5th international workshop on Mining Software Repositories (MSR), pp. 1-5, 2005.

[Raza *et al.*, 2006] Raza , A.; Vogel, G.; Plödereder, E. **Bauhaus – A Tool Suite for Program Analysis and Reverse Engineering**. In Proceedings of the 11th Ada-Europe international conference on Reliable Software Technologies pp. 71-82, 2006.

[Riel, 1996] RIEL, A.J. **object-oriented Design Heuristics**. Addison-Wesley, 1996.

[Sangal *et al.*, 2005] SANGAL, N.; JORDAN, E.; SINHA, V.; JACKSON, D. **Using dependency models to manage complex software architecture**. In Proceedings of the Proceedings of the 20th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA), pp. 167-176, 2005.

[Sangal and Waldman, 2005] SANGAL, N. and WALDMAN, F. **Dependency models to manage software architecture**. CrossTalk, 18(11), 2005.

[Sarkar *et al.*, 2009] SARKAR, S.; RAMACHANDRAN, S.; KUMAR, G. S.; IYENGAR, M. K.; RANGARAJAN, K. & SIVAGNANAM, S. **Modularization of a large-scale business application: A case study**. IEEE Transactions on Software Engineering, 26:28 35. 2009

[Sant'Anna *et al.*, 2007] SANT'ANNA, C.; FIGUEIREDO, E.; GARCIA, A.; and LUCENA, C.J.P. **On the modularity of software architectures: A Concern-Driven measurement framework**. In Proceedings of the 5th IEEE/IFIP Conference on Software Architecture (ECSA), 2007.

[Sheskin, 2007] SHESKIN, D. **Handbook of Parametric and Nonparametric Statistical Procedures** (fourth edition). Chapman & All, 2007.

[Semmle Code, 2012] SEMMLE CODE. http://semmle.com/semmlecode/. 2012.

[Sjobert *et al.*, 2013] SJOBERT, D.; YAMASHITA, A.; ANDA, B.; MOCKUS, A.; DYBA, T. **Quantifying the Effect of Code Smells on Maintenance Effort**. IEEE Transactions on Software Engineering (TSE), 2013.

[Stasko *et al.*, 1998] STASKO, J.T.; DOMINGUEZ, J.B.; BROWN, M.H.; PRICE, B.A.; FOLEY, J. **Software Visualization**. The MIT Press, 1998.

[Soares *et al.*, 2002] SOARES, S.; LAUREANO, E.; BORBA, P. **Implementing distribution and persistence aspects with AspectJ**. In Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (OOPSLA), pp. 174-190, 2002.

[Sonar, 2009] Sonar: http://docs.codehaus.org/display/SONAR/

[Sonarsource, 2010] Sonarsource: nemo.sonarsource.org/

[Srivisut and Muenchaisri, 2007] SRIVIST, K.; MUENCHAISRI, P. **Bad-smell metrics for aspect-oriented software**. In Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS), pp. 1060 - 1065, Canada, 2007.

STAL, M (2011): http://stal.blogspot.com.

[Structure101, 2009] Structure101: http://structure101.com/

[Taylor *et al.*, 2009] TAYLOR, R., MEDVIDOVIC, N. and DASHOFY, E. **Software Architecture: Foundations, Theory, and Practice**. Wiley Publishing. 2009.

[Tsantalis and Chatzigeorgiou, 2009] TSANTALIS, N. and CHATZIGEORGIOU, A. **Identification of move method refactoring opportunities**. IEEE Transactions on Software Engineering (TSE), 35(3), pp 347–367, 2009.

[Tsantalis *et al.*, 2008] TSANTALIS, N.; CHAIKALIS, T.; CHATZIGEORGIOU, A. JDeodorant: Identification and removal of type-checking bad smells. In Proceedings of the 18th European Conference on Software Maintenance and Reengineering (CSMR), pp; 329–331, 2008.

[Terra and Valente, 2009] TERRA, R. and VALENTE, T. **A Dependency Constraint Language to Manage object-oriented Software Architectures**. In Software: Practice and Experience, Volume 32, Issue 12, John Wiley & Sons, pp. 1073-1094, 2009.

[Together, 2009] Together: http://www.borland.com/us/products/together/

[Tsantalis and Chatzigeorgiou, 2009] TSANTALIS, N. and CHATZIGEORGIOU, A. **Identification of move method refactoring opportunities**. IEEE Transactions on Software Engineering (TSE), 35(3), pp. 347–367, 2009.

[Ubayashi *et al.*, 2010] UBAYASHI, N.; NOMURA, J. and TAMAI, T. **Archfase: A Contract Place Where Architectural Design and Code Meet Together**. In Proceedings of the 32sd International Conference on Software Engineering (ICSE) pp. 75-84, 2010.

[Understand, 2009] Understand: http://www.scitools.com/

[Vidal and Marcos, 2012] VIDAL, S. and MARCOS, C.A. **Building an Expert System to Assist System Refactorization**. In Expert Systems with Applications: An International Journal, Vol 39 Issue 3, February 2012.

[Wake, 2003] WAKE, W.C. **Refactoring workbook**. AddisonWesley Longman Publishing Co., Inc., 2003.

[Webster, 1995] WEBSTER, B.F. Pitfalls of Object Oriented Development, 1st ed. M & T Books, February 1995.

[Weißgerber and Diehl, 2006] WEIßGERBER, P. and DIEHL, S. **Are refactorings less error-prone than other changes?** In Proceedings of the International Workshop on Mining software Repositories (MSR) pp. 112-118, 2006.

Wettel *et al.*, 2011] WETTEL, R.; LANZA, M.; ROBBES, R. **Software Systems as Cities: A Controlled Experiment**. In Proceedings of the 33rd International Conference on Software Engineering (ICSE), pp. 551-560, 2011.

[Wettel and Lanza, 2007] WETTEL, R.; LANZA, M. **Visualizing Software Systems as Cities**. In Proceedings of the 4th IEEE International Workshop on Visualizing Software For Understanding and Analysis, pp. 92 - 99, 2007.

[Wohlin *et al.* 2000] WOHLIN, C.; RUNESON, P.; HOST, M.; OHLSSON, M.; REGNELL, B. ; WESSLEN, A. **Experimentation in Software Engineering - An Introduction**. Kluwer Academic Publishers, illustrated edition, 2000.

[Wong *et al.*, 2011] WONG, S.; CAI, Y.; KIM, M.; DALTON, M. **Detecting Software Modularity Violations**. In Proceedings of the 33rd International Conference on Software Engineering (ICSE), pp. 411-420, 2011.

[Xi *et al.*, 2012] XI, G.; DUBOSE, Q.L.; MURPHY-HILL, E. **Reconciling Manual and Automatic Refactoring**. Proceedings of the 34th International Conference on Software Engineering (ICSE), Switzerland, 2012.

[Xing and Stroulia, 2006] XING, Z. and STROULIA, E. **Refactoring practice: How it is and how it should be supported**. In Proceedings of 22sd IEEE International Conference on Software Maintenance (ICSM), pp. 458-468, 2006.

[Yamashita and Moonen, 2013] YAMASHITA, A.; MOONEN, E. **Exploring the Impact of Inter-smell Relations on Software Maintainability: An Empirical Study**. In Proceedings of the 35h International Conference on Software engineering (ICSE), 2013 (to appear).

[ydiff, 2010] ydiff: http://yinwang0.wordpress.com/

[Zazworka *et al.*, 2011] ZAZWORKA, N.; SHAW, M.A.; SHULL, F.; SEAMAN, C. **Investigating the Impact of Design Debt on Software Quality**. In Proceedings of the 2nd Workshop on Managing Technical Debt (MTD) at 33rd International Conference on Software Engineering (ICSE), pp. 17-23, 2011.

Appendix A:
System Characteristics

This appendix presents the main characteristics of the target systems considered in the empirical studies. In order to present these characteristics, the following acronyms are used:

HW = Health Watcher;
AW = Aspectual Watcher;
MM=MobileMedia;
AM = Aspectual Media;
CE = Code Elements (classes and aspects);
AE = Architectural Elements (components and interfaces);
CA = Code Anomalies;
PC = Page Controller;
AE = Architecture Elements (modules and interfaces);
AD= Aspect Design.

Target System	AM	AW	iBATIS
System Type	Software Product Line	Web	Persistence Framework
Programming Language	Java & AspectJ	Java & Aspect	Java and AspectJ
Architecture Design	MVC & AD	MVC & AD	Client Server
# of Versions	8	10	4
# of Selected Versions	8	10	4
Avg. # of CE	94	113	
Avg. #of AE	21	13	
KLOC	48	41	
Avg # of CA	126	188	

Target System	MIDAS	MM	HW	PDP
System Type	Middleware	Software Product Line	Web	Web
Programming Language	C++	Java	Java	C#
Architecture Design	Layers	MVC	Layers	PC & Layers
# of Versions	2	8	10	8
# of Selected Versions	2	8	10	2
Avg. # of CE	392	60	85	97
Avg. #of AE	19	14	7	15
KLOC	76	54	49	22
Avg # of CA	111	170	252	175

Target Systems	S1	S2	S3
System Type		Desktop Application	
Programming Language	Java	Java	Java
Architecture Design		Client - Server	
Selected Version	2.0	8.0	4.0
Avg. # of CE	446	530	443
Avg. #of AE	39	44	45
KLOC	122	118	93
Avg # of AR Code Anomalies	248	299	197

Appendix B:
Detection Strategies and Thresholds

This appendix details the definitions of the conventional strategies applied in our studies as well as the used thresholds (Table B-1). The appendix also presents the thresholds used in the architecture-sensitive detection strategies for both conventional (Table B-2) and architecture-sensitive metrics (Table B-3).

Conventional Detection Strategies

Shotgun Surgery = CC > LOW and CM > MANY

CM (*Changing Method*): counts the number of distinct methods that call the measured method.

CC (*Changing Class*): counts the number of classes in with the method that call the measured method are defined.

Feature Envy = ATFD > FEW and LAA > ONE THIRD and FDP < FEW

ATFD (*Access to Foreign Data*): counts the number of distinct attributes the measured element accesses.

LAA (*Locality of Attribute Accesses*): the relative number of attributes that a method accesses on its class.

FDP (*Foreign Data Providers*): the number of classes where the accessed attributes belong to.

Long Method = LOC > VERY_HIGH and CYCLO > HIGH

LOC (*Lines Of Code*) counts the number of Lines Of Code.

CC (*Cyclomatic Complexity*): represents McCabe's Cyclomatic Complexity.

Disperse Coupling = CINT > FEW and CDISP > FEW

CINT (*Coupling Intensity*): The number of distinct methods called by the measured method.

CDISP (*Coupling Dispersion*): The number of classes in which the called methods are defined divided by CINT.

Intensive Coupling = CINT > FEW and CDISP < FEW

CINT (*Coupling Intensity*): The number of distinct methods called by the measured method.

CDISP (*Coupling Dispersion*): The number of classes in which the called methods are defined divided by CINT.

God Class = ATFD > LOW and WMC > VERY_HIGH and TCC < THIRD

ATFD (*Access To Foreign Data*): counts the number of attributes from unrelated classes that are accessed directly or by invoking accessor methods

TCC (*Tight Class Cohesion*): counts the relative number of method pairs of a class that access in common at least one attribute of the measured class.

WMC (*Weighted Method Count*): represents the sum of the statical complexity of all methods of a class.

Misplaced Class = CL > THIRD and NOED > HIGH and DD > LOW

NOED (*Number Of External Dependencies*): counts the number of classes from other packages on which the measured class depends on.

CL (*Class Locality*): counts the relative number of dependencies that a class has on its own package.

DD (*Dependency Dispersion*): counts the number of other packages on which the class depends.

Table B-1: Thresholds used in the conventional strategies.

	ATFD	CC	CDISP	CINT	CM	CYCLO	DD	FDP	LOC	NOED	WMC
Health Watcher											
LOW	4	-	-	-	-	-	2	-	-	-	-
FEW	-	4	5	4	-	-	-	4	-	-	-
HIGH	-	-	-	-	-	4	-	-	-	8	-
V_HIGH	-	-	-	-	-	-	-	-	20	-	50
MANY	-	-	-	-	7	-	-	-	-	-	-
Mobile Media											
LOW	4	-	-	-	-	-	3	-	-	-	-
FEW	-	3	3	4	-	-	-	4	-	-	-
HIGH	-	-	-	-	-	5	-	-	-	8	-
V_HIGH	-	-	-	-	-	-	-	-	30	-	40
MANY	-	-	-	-	8	-	-	-	-	-	-
Aspectual Watcher											
LOW	3	-	-	-	-	-	2	-	-	-	-
FEW	-	2	3	4	-	-	-	4	-	-	-
HIGH	-	-	-	-	-	4	-	-	-	6	-
V_HIGH	-	-	-	-	-	-	-	-	15	-	40
MANY	-	-	-	-	7	-	-	-	-	-	-
Aspectual Media											
LOW	3	-	-	-	-	-	3	-	-	-	-
FEW	-	3	3	4	-	-	-	4	-	-	-
HIGH	-	-	-	-	-	5	-	-	-	6	-

V_HIGH	-	-	-	-	-	-	-	-	20	-	45
MANY	-	-	-	-	8	-	-	-	-	-	-
MIDAS											
LOW	5	-	-	-	-	-	4	-	-	-	-
FEW	-	7	6	6	-	-	-	5	-	-	-
HIGH	-	-	-	-	-	5	-	-	-	8	-
V_HIGH	-	-	-	-	-	-	-	-	35	-	65
MANY	-	-	-	-	10	-	-	-	-	-	-
PDP											
LOW	4	-	-	-	-	-	4	-	-	-	-
FEW	-	2	3	6	-	-	-	4	-	-	-
HIGH	-	-	-	-	-	3	-	-	-	6	-
V_HIGH	-	-	-	-	-	-	-	-	25	-	35
MANY	-	-	-	-	8	-	-	-	-	-	-
S1, S2 and S3											
LOW	4	-	-	-	-	-	3	-	-	-	-
FEW	6	2	3	4	-	-	-	4	-	-	-
HIGH	8	-	-	-	-	4	-	-	-	6	-
V_HIGH	8	-	-	-	-	-	-	-	30	-	100
MANY	12	-	-	-	7	-	-	-	-	-	-

Table B-2: Thresholds used for the conventional metrics in the architecture-sensitive strategies.

	ATFD	CC	CDISP	CINT	CM	CYCLO	DD	FDP	LOC	NOED	WMC
Health Watcher											
LOW	4	-	-	-	-	-	4	-	-	-	-
FEW	6	6	6	5	-	-	-	5	-	-	-
HIGH	8	-	-	-	-	4	-	-	-	8	-
V_HIGH	10	-	-	-	-	-	-	-	25	-	60
MANY	12	-	-	-	12	-	-	-	-	-	-
Mobile Media											
LOW	4	-	-	-	-	-	5	-	-	-	-
FEW	7	7	6	6	-	-	-	4	-	-	-
HIGH	10	-	-	-	-	5	-	-	-	7	-
V_HIGH	12	-	-	-	-	-	-	-	30	-	55
MANY	14	-	-	-	10	-	-	-	-	-	-
S1, S2 and S3											
LOW	4	-	-	-	-	-	3	-	-	-	-
FEW	6	2	3	4	-	-	-	4	-	-	-
HIGH	8	-	-	-	-	4	-	-	-	8	-
V_HIGH	10	-	-	-	-	-	-	-	45	-	60
MANY	12	-	-	-	7	-	-	-	-	-	-

Table B-3: Thresholds used for the architecture-sensitive metrics in the architecture-sensitive strategies.

	NEE	EFO	EFI	ACL	NAC	CoL	CoC
Health Watcher							
LOW	-	1	1	0.25	0.25	0.25	-
FEW	5	2	-	-	-	-	-
MANY	8	-	-	-	-	-	-
Mobile Media							
LOW	-	2	2	0.33	0.25	0.25	-
FEW	6	3	-	-	-	-	-
MANY	10	-	-	-	-	-	-
S1, S2 and S3							
LOW	-	3	2	0.33	0.25	0.25	-
FEW	7	4	-	-	-	-	-
MANY	12	-	-	-	-	-	-

Appendix C:
BNF SCOOP Grammar

The BNF description uses the bold font to display terminal symbols and the first characters of non-terminal symbols are shown in upper-case format. The symbols '[A]', '(A)+' and ()A'*' respectively impose the cardinalities: optional (0 or 1); at least one; and zero or more to a the symbol A.

DetectionStrategy	::=	constraints += Constraint*
Constraint	::=	**code anomaly <**ElementType**>** ID **=** LogicExpression **;**
ElementType	::=	**package** | **class** | **method**
LogicExpression	::=	NumExpression (LogicOperator LogicExpression)?
NumExpression	::=	**(** LogicExpression **)** | CompExpression | NotExpression | ConstraintExpression
ConstraintExpression	::=	[Constraint]
NotExpression	::=	**not** NumExpression
CompExpression	::=	Metric CompOperator INT;
CompOperator	::=	**<** | **>** | **=** | **≤** | **≥**
Metric	::=	**NEE** | **EFO** | **EFI** | **ACL** | **NAC** | **CoL** | **CoC** | **LOC** | **LCON** | **CC** | **CM** | **CO**
LogicOperator	::=	**or** | **and**

Appendix D:
SCOOP Rules File

By editing the *Rules.ds* file, engineers can modify or create the detection strategies that will be applied in SCOOP. In the following, we present a fragment example of this file.

code anomaly <**method**> DisperseCoupling **=** CINT > 4 **and** CDISP > 5
code anomaly<**method**> IntensiveCoupling **=** CINT > 4 **and** CDISP < 5
code anomaly<**method**> LongMethod **=** LOC > 30 **and**
CYCLO > 3
code anomaly<**method**> ShotgunSurgery **=** CC > 4 **and** CM > 7
code anomaly<**method**> FeatureEnvy **=** (ATFD > 4 **or** NEE > 3) **and**
(LAA > 0.33 **or** ACL < 0.33) **and**
(FDP < 4 **or** EFO < 3)
code anomaly<**class**> God Class **=** (WMC > 150 **and** NAC > 1 **and**
CoC < 0.5)
code anomaly<**class**> MisplacedClass **=** (CoL > 0.5) **and** (NAC > 1)