References

[1] Google code search. Accessed Jan. 2012.

[2] Merobase. Accessed Jan. 2012.

[3] Sourcerer. Accessed Jan. 2012.

[4] *American Heritage Dictionary of the English Language*. Houghton Miflin Company, 4th edition, 2009.

[5] J.-R. Abrial, S. A. Schuman, and B. Meyer. Specification language. In R. M. McKeag and A. M. Macnaughten, editors, *On the Construction of Programs*, pages 343–410. Cambridge University Press, Cambridge, UK, November 1980.

[6] Jean-Raymond Abrial. *The B-book: Assigning Programs to Meanings*. Cambridge University Press, Cambridge, UK, October 1996.

[7] Christopher Ackermann, Mikael Lindvall, and Greg Dennis. Redesign for flexibility and maintainability: A case study. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 259–262, 2009.

[8] Charu C. Aggarwal and Haixun Wang. A survey of clustering algorithms for graph data. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, chapter 9, pages 275–301. Springer, 2010.

[9] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[10] Ademar Aguiar. A minimalist approach to framework documentation. In *Addendum to the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 143–144, 2000.

[11] Tero Aittokallio and Benno Schwikowski. Graph-based methods for analysing networks in cell biology. *Briefings in Bioinformatics*, 7(3):243–255, 2006.

[12] Samuel A. Ajila and Di Wu. Empirical study of the effects of open source adoption on software development economics. *Journal of Systems and Software*, 80(9):1517–1529, 2007.

[13] Jehad Al Dallal and Paul Sorenson. Reusing class-based test cases for testing object-oriented framework interface classes. *Journal of Software Maintenance and Evolution: Research and Practice*, 17(3):169–196, 2005.

[14] Réka Albert. Scale-free networks in cell biology. *Journal of Cell Science*, 118(21):4947–4957, 2005.

[15] Réka Albert and Albert-László Barabási. Dynamics of complex systems: Scaling laws for the period of boolean networks. *Physical Review Letters*, 84(24):5660–5663, 2000.

[16] Réka Albert and Albert-László Barabási. Topology of evolving networks: Local events and universality. *Physical Review Letters*, 85(24):5234–5237, 2000.

[17] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[18] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Diameter of the World Wide Web. *Nature*, 401:130–131, 1999.

[19] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.

[20] David Alderson, John C. Doyle, Lun Li, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005.

[21] Christopher Alexander. *Notes on the Synthesis of Form*. Harvard University Press, 1964.

[22] Edward B. Allen and Taghi M. Khoshgoftaar. Measuring coupling and cohesion: An information-theory approach. In *Proceedings of the IEEE International Symposium on Software Metrics*, pages 119–129, 1999.

[23] Mohammad Alshayeb and Wei Li. An empirical validation of object-oriented metrics in two different iterative software processes. *IEEE Transactions on Software Engineering*, 29(11):1043–1049, 2003.

[24] Klaus-Dieter Althoff, Andreas Birk, Christiane Gresse von Wangenheim, and Carsten Tautz. CBR for experimental software engineering. In Mario Lenz, Hans-Dieter Burkhard, Brigitte Bartsch-Spörl, and Stefan Wess, editors, *Case-Based Reasoning Technology: From Foundations to Applications*, volume 1400 of *Lecture Notes in Computer Science*, pages 235–254. Springer, 1998.

[25] L. A. N. Amaral, A. Scala, M. Barthélémy, and H. E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21):11149–11152, 2000.

[26] Vicenzo Ambriola and Alina Kmiecik. Transformations for architectural restructuring. *Informatica*, 28(2):117–128, 2004.

[27] Vincenzo Ambriola and Alina Kmiecik. Architectural transformations. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, pages 275–278, 2002.

[28] Juan-José Amor-Iglesias, Jesús M. González-Barahona, Gregorio Robles-Martinez, and Israel Herráiz-Tabernero. Measuring libre software using Debian 3.1 (Sarge) as a case study: Preliminary results. *CEPIS UPGRADE*, 6(3):13–16, 2005.

[29] Jesper Andersen and Julia L. Lawall. Generic patch inference. *Automated Software Engineering: An International Journal*, 17(2):119–148, June 2010.

[30] Bill Andreopoulos, Aijun An, Vassilios Tzerpos, and Xiaogang Wang. Multiple layer clustering of large software systems. In *Proceedings of the Working Conference on Reverse Engineering*, pages 79–88, 2005.

[31] Periklis Andritsos and Vassilios Tzerpos. Information-theoretic software clustering. *IEEE Transactions on Software Engineering*, 31(2):150–165, 2005.

[32] Nicolas Anquetil and Jannik Laval. Legacy software restructuring: Analyzing a concrete case. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 279–286, 2011.

[33] Giuliano Antoniol, Vincenzo Fabio Rollo, and Gabriele Venturi. Detecting groups of co-changing files in CVS repositories. In *Proceedings of the ACM International Workshop on Principles of Software Evolution*, pages 23–32, 2005.

[34] John Anvik, Lyndon Hiew, and Gail C. Murphy. Who should fix this bug? In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 361–370, 2006.

[35] Taweesup Apiwattanapong, Alessandro Orso, and Mary Jean Harrold. A differencing algorithm for object-oriented programs. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 2–13, 2004.

[36] Taweesup Apiwattanapong, Alessandro Orso, and Mary Jean Harrold. JDiff: A differencing technique and tool for object-oriented programs. *Automated Software Engineering: An International Journal*, 14(1):3–36, March 2007.

[37] Taweesup Apiwattanapong, Raul Santelices, Pavan Kumar Chittimalli, Alessandro Orso, and Mary Jean Harrold. MATRIX: Maintenance-oriented testing requirements identifier and examiner. In *Proceedings of Testing: Academic & Industrial Conference: Practice and Research Techniques*, pages 137–146, 2006.

[38] Jorge Aranda and Gina Venolia. The secret life of bugs: Going past the errors and omissions in software repositories. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 298–308, 2009.

[39] Erik Arisholm, Lionel C. Briand, and Audun Føyen. Dynamic coupling measurement for object-oriented software. *IEEE Transactions on Software Engineering*, 30(8):491–506, 2004.

[40] R. S. Arnold and S. A. Bohner. Impact analysis: Towards a framework for comparison. In *Proceedings of the Conference on Software Maintenance*, pages 292–301, 1993.

[41] Jai Asundi and Rajiv Jayant. Patch review processes in open source software development communities: A comparative case study. In *Proceedings of the Hawaii International Conference on Systems Science*, pages 166c/1–166c/7, 2007.

[42] D. C. Atkinson and W. G. Griswold. Effective pattern matching of source code using abstract syntax patterns. *Software: Practice & Experience*, 36(4):413–447, 2006.

[43] Anne Aula, Natalie Jhaveri, and Mika Käki. Information search and reaccess strategies of experienced web users. In *Proceedings of the International Conference on the World Wide Web*, pages 583–592, 2005.

[44] L. Aversano, L. Cerulo, and M. Di Penta. Relationship between design patterns defects and crosscutting concern scattering degree: An empirical study. *IET Software*, 3(5):395–409, 2009.

[45] K. Ayari, P. Meshkinfam, G. Antoniol, and M. Di Penta. Threats on building models from CVS and Bugzilla repositories: The Mozilla case study. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 215–228, 2007.

[46] Furqan Aziz, Richard C. Wilson, and Edwin R. Hancock. Graph characterization via backtrackless paths. In *Proceedings of the International Conference on Similarity-Based Pattern Recognition*, volume 7005 of *Lecture Notes in Computer Science*, pages 149–162, 2011.

[47] Franz Baader. Unification, weak unification, upper bound, lower bound, and generalization problems. In *Proceedings of the International Conference on Rewriting Techniques and Applications*, volume 448 of *Lecture Notes in Computer Science*, pages 86–97, 1991.

[48] B. S. Baker. On finding duplication and near-duplication in large software systems. In *Proceedings of the Working Conference on Reverse Engineering*, pages 86–95, 1995.

[49] Ittai Balaban, Frank Tip, and Robert Fuhrer. Refactoring support for class library migration. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 265–279, 2005.

[50] Zsolt Balanyi and Rudolf Ferenc. Mining design patterns from C++ source code. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 305–314, 2003.

[51] N. V. Balasubramanian. Object-oriented metrics. In *Proceedings of the Asia–Pacific Software Engineering Conference*, pages 30–34, 1996.

[52] Pierre F. Baldi, Cristina V. Lopes, Erik J. Linstead, and Sushil K. Bajracharya. A theory of aspects as latent topics. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 543–562, 2008.

[53] Carliss Y. Baldwin and Kim B. Clark. *Design Rules: The Power of Modularity*. MIT Press, 1999.

[54] Carliss Y. Baldwin and Kim B. Clark. Modularity in the design of complex engineering systems. In Dan Braha, Ali A. Minai, and Yaneer Bar-Yam, editors, *Complex Engineered Systems: Science Meets Technology*, pages 175–205. Springer, 2005.

[55] Elisa L. A. Baniassad, Gail C. Murphy, Christa Schwanninger, and Michael Kircher. Managing crosscutting concerns during software evolution tasks: An inquisitive study. In *Proc. Int. Conf. Aspect-Oriented Softw. Devel.*, pages 120–126, 2002.

[56] Rajiv D. Banker, Srikant M. Datar, Chris F. Kemerer, and Dani Zweig. Software complexity and maintenance costs. *Communications of the ACM*, 36(11):81–94, 1993.

[57] Shenghua Bao, Guirong Xue, Xiaoyuan Wu, Yong Yu, Ben Fei, and Zhong Su. Optimizing web search using social annotations. In *Proceedings of the International Conference on the World Wide Web*, pages 501–510, 2007.

[58] Albert-László Barabási. The origin of bursts and heavy tails in human dynamics. *Nature*, 435:207–211, 2005.

[59] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[60] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Mean-field theory for scale-free random networks. *Physica A: Statistical Mechanics and Its Applications*, 272(1–2):173–187, 1999.

[61] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: The topology of the world-wide web. *Physica A: Statistical Mechanics and Its Applications*, 281(1–4):69–77, 2000.

[62] Albert-László Barabási, Erzsébet Ravasz, and Tamás Vicsek. Deterministic scale-free networks. *Physica A: Statistical Mechanics and Its Applications*, 299(3–4):559–564, 2001.

[63] Olivier Barais, Laurence Duchien, and Anne-Francoise Le Meur. A framework to specify incremental software architecture transformations. In *Proceedings of the EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 62–69, 2005.

[64] Olivier Barais, Julia Lawall, Anne-Francoise Le Meur, and Laurence Duchien. Providing support for safe software architecture transformations. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture*, pages 201–202, 2005.

[65] Olivier Barais, Anne-Françoise Le Meur, Laurence Duchien, and Julia L. Lawall. Software architecture evolution. In Tom Mens and Serge Demeyer, editors, *Software Evolution*, chapter 10, pages 233–262. Springer, 2008.

[66] Bruce H. Barnes and Terry B. Bollinger. Making reuse cost-effective. *IEEE Software*, 8(1):13–24, 1991.

[67] A. Barrat and M. Weigt. On the properties of small-world networks. *European Physical Journal B*, 13(3):547–560, 2000.

[68] Stephen C. Barrett, Greg Butler, and Patrice Chalin. Mirador: A synthesis of model matching strategies. In *Proceedings of the International Workshop on Model Comparison in Practice*, pages 2–10, 2010.

[69] Marc Bartsch and Rachel Harrison. An exploratory study of the effect of aspect-oriented programming on maintainability. *Software Quality Journal*, 16(1):23–44, 2008.

[70] Victor R. Basili. Facts and myths affecting software reuse. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, page 269, 1994.

[71] Victor R. Basili, Lionel C. Briand, and Walcélio L. Melo. How reuse influences productivity in object-oriented systems. *Communications of the ACM*, 39(10):104–116, 1996.

[72] Victor R. Basili, Lionel C. Briand, and Walcélio L. Melo. A validation of object-oriented design metrics as quality indicators. *IEEE Transactions on Software Engineering*, 22(10):751–761, 1996.

[73] Victor R. Basili, Richard W. Selby, and David H. Hutchens. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, 12(7):733–743, 1986.

[74] Hamid Abdul Basit and Stan Jarzabek. A data mining approach for detecting higher-level clones in software. *IEEE Transactions on Software Engineering*, 35(4):497–514, 2009.

[75] Len Bass, Paul Clements, and Rick Kazman. *Software Architecture in Practice*. Addison-Wesley, 2nd edition, 2003.

[76] Paul Bassett. The case for frame-based software engineering. *IEEE Software*, 24(4):90–99, 2007.

[77] Paul G. Bassett. The theory and practice of adaptive reuse. *Proceedings of the ACM Symposium on Software Reusability*, 22(3):2–9, 1997.

[78] Marcia J. Bates. The design of browsing and berrypicking techniques for the online search interface. *Online Reviews*, 13(5):407–424, 1989.

[79] Gareth Baxter, Marcus Frean, James Noble, Mark Rickerby, Hayden Smith, Matt Visser, Hayden Melton, and Ewan Tempero. Understanding the shape of Java software. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 397–412, 2006.

[80] Ira D. Baxter. DMS: Program transformations for practical scalable software evolution. In *Proceedings of the ACM International Workshop on Principles of Software Evolution*, pages 48–51, 2002.

[81] Ira D. Baxter, Andrew Yahin, Leonardo M. De Moura, Marcelo Sant'Anna, and Lorraine Bier. Clone detection using abstract syntax trees. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 368–377, 1998.

[82] Kent Beck. *Test Driven Development: By Example*. Addison-Wesley Professional, 2002.

[83] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, 2nd edition, 2004.

[84] Andrew Begel, Nachiappan Nagappan, Christopher Poile, and Lucas Layman. Coordination in large-scale software teams. In *Proceedings of the International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 1–7, 2009.

[85] L. A. Belady and M. M. Lehman. A model of large program development. *IBM Systems Journal*, 15(3):225–252, 1976.

[86] Ahmed Belderrar, Segla Kpodjedo, Yann-Gaël Guéhéneuc, Giuliano Antoniol, and Philippe Galinier. Sub-graph mining: Identifying micro-architectures in evolving object-oriented software. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 171–180, 2011.

[87] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo. Comparison and evaluation of clone detection tools. *IEEE Transactions on Software Engineering*, 33(9):577–591, 2007.

[88] William Berg, Marshall Cline, and Mike Girou. Lessons learned from the OS/400 OO project. *Communications of the ACM*, 38(10):54–64, 1995.

[89] Fernando Berzal, Juan-Carlos Cubero, and Aída Jiménez. Hierarchical program representation for program element matching. In *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, pages 467–476, 2007.

[90] Dirk Beyer and Andreas Noack. Clustering software artifacts based on frequent common changes. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 259–268, 2005.

[91] Jean Bézivin. On the unification power of models. *Software and Systems Modeling*, 4(2):171–188, 2005.

[92] James M. Bieman and Byung-Kyoo Kang. Cohesion and reuse in an object-oriented system. In *Proceedings of the ACM Symposium on Software Reusability*, pages 259–262, 1995.

[93] T. Biggerstaff and C. Richter. Reusability framework, assessment, and directions. *IEEE Software*, 4(2):41–49, 1987.

[94] Ted J. Biggerstaff. The library scaling problem and the limits of concrete component reuse. In *Proceedings of the International Conference on Software Reuse*, pages 102–109, 1994.

[95] Ted J. Biggerstaff. A perspective of generative reuse. *Annals of Software Engineering*, 5(1):169–226, 1998.

[96] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas Webster. The concept assignment problem in program understanding. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 482–498, 1993.

[97] Ted J. Biggerstaff, Bharat G. Mitbander, and Dallas E. Webster. Program understanding and the concept assignment problem. *Communications of the ACM*, 37(5):72–82, 1994.

[98] Robert V. Binder. *Testing Object-Oriented Systems: Models, Patterns, and Tools*. Addison-Wesley, 1999.

[99] David Binkley. Using semantic differencing to reduce the cost of regression testing. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 41–50, 1992.

[100] Christian Bird, Alex Gourley, Prem Devanbu, Anand Swaminathan, and Greta Hsu. Open borders? immigration in open source projects. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 6/1–6/8, 2007.

[101] Rossella Bisio and Fabio Malabocchia. Cost estimation of software projects through case base reasoning. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 1010 of *Lecture Notes in Computer Science*, pages 11–22, 1995.

[102] Solveig Bjørnestad. Analogical reasoning for reuse of object-oriented specifications. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 2689 of *Lecture Notes in Computer Science*, pages 50–64, 2003.

[103] Marilyn Hughes Blackmon, Muneo Kitajima, and Peter G. Polson. Tool for accurately predicting website navigation problems, non-problems, problem severity, and effectiveness of repairs. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 31–40, 2005.

[104] Joshua Bloch. How to design a good API and why it matters. In *Companion to the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 506–507, 2006.

[105] Vincent D. Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM Review*, 46(4):647–666, April 2004.

[106] Barry Boehm. Managing software productivity and reuse. *Computer*, 32(9):111–113, 1999.

[107] S. A. Bohner. Software change impacts: An evolving perspective. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 263–272, 2002.

[108] Pia Borlund. The concept of relevance in IR. *Journal of the American Society for Information Science and Technology*, 54(10):913–925, 2003.

[109] Jan Bosch, Gert Florijn, Danny Greefhorst, Juha Kuusela, J. Henk Obbink, and Klaus Pohl. Variability issues in software product lines. In *Revised Papers of the International Workshop on Software Product-Family Engineering*, volume 2290 of *Lecture Notes in Computer Science*, pages 13–21, 2002.

[110] Jan Bosch and Peter Molin. Software architecture design: Evaluation and transformation. In *Proceedings of the IEEE Conference on Engineering Computer-Based Systems*, pages 4–10, 1999.

[111] Marat Boshernitsan and Susan L. Graham. iXj: Interactive source-to-source transformations for Java. In *Companion to the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 212–213, 2004.

[112] Marat Boshernitsan, Susan L. Graham, and Marti A. Hearst. Aligning development tools with the way programmers think about code changes. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 567–576, 2007.

[113] Salah Bouktif, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Extracting change-patterns from CVS repositories. In *Proceedings of the Working Conference on Reverse Engineering*, pages 221–230, 2006.

[114] S. R. Bourne. *An Introduction to the UNIX Shell*. Bell Laboratories, 1977.

[115] Robert W. Bowdidge and William G. Griswold. Supporting the restructuring of data abstractions through manipulation of a program visualization. *ACM Transactions on Software Engineering and Methodology*, 7(2):109–157, 1998.

[116] Joel Brandt, Mira Dontcheva, Marcos Weskamp, and Scott R. Klemmer. Example-centric programming: integrating web search into the development environment. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 513–522, 2010.

[117] Joel Brandt, Philip J. Guo, Joel Lewenstein, Mira Dontcheva, and Scott R. Klemmer. Two studies of opportunistic programming: Interleaving web foraging, learning, and writing code. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1589–1598, 2009.

[118] Hongyu Pei Breivold, Ivica Crnkovic, and Magnus Larsson. Software architecture evolution through evolvability analysis. *Journal of Systems and Software*, 85(11):2574–2592, November 2012.

[119] Hongyu Pei Breivold, Ivica Crnkovic, and Magnus Larsson. A systematic review of software architecture evolution research. *Information and Software Technology*, 54(1):16–40, January 2012.

[120] Pearl Brereton, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, 80(4):571–583, 2007.

[121] Silvia Breu, Rahul Premraj, Jonathan Sillito, and Thomas Zimmermann. Investigating information needs to improve cooperation between developers and bug reporters. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 301–310, 2010.

[122] Silvia Breu and Thomas Zimmermann. Mining aspects from version history. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 221–230, 2006.

[123] Silvia Breu, Thomas Zimmermann, and Christian Lindig. HAM: Crosscutting concerns in Eclipse. In *Proceedings of the Eclipse Technology eXchange*, pages 21–24, 2006.

[124] Lionel Briand, Prem Devanbu, and Walcelio Melo. An investigation into coupling measures for C++. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 412–421, 1997.

[125] Lionel C. Briand, John Daly, Victor Porter, and Jürgen Wüst. A comprehensive empirical validation of design measures for object-oriented systems. In *Proceedings of the IEEE International Symposium on Software Metrics*, pages 246–257, 1998.

[126] Lionel C. Briand, John W. Daly, and Jürgen Wüst. A unified framework for cohesion measurement in object-oriented systems. *Empirical Software Engineering*, 3(1):65–117, 1998.

[127] Lionel C. Briand, John W. Daly, and Jürgen K. Wüst. A unified framework for coupling measurement in object-oriented systems. *IEEE Transactions on Software Engineering*, 25(1):91–121, 1999.

[128] Lionel C. Briand, Sandro Morasca, and Victor R. Basili. Defining and validating measures for object-based high-level design. *IEEE Transactions on Software Engineering*, 25(5):722–743, 1999.

[129] Lionel C. Briand, Jürgen Wüst, John W. Daly, and D. Victor Porter. Exploring the relationships between design measures and software quality in object-oriented systems. *Journal of Systems and Software*, 51(3):245–273, 2000.

[130] Reinder J. Bril, Loe M. G. Feijs, André Glas, René L. Krikhaar, and M. (Thijs) R. M. Winter. Maintaining a legacy: Towards support at the architectural level. *Journal of Software Maintenance: Research and Practice*, 12(3):143–170, 2000.

[131] Reinder J. Bril and André Postma. An architectural connectivity metric and its support for incremental re-architecting of large legacy systems. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 269–280, 2001.

[132] Reinder J. Bril, André Postma, and René L. Krikhaar. Embedding architectural support in industry. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 348–357, 2003.

[133] Fernando Brito e Abreu and Rogério Carapuça. Candidate metrics for object-oriented software within a taxonomy framework. *Journal of Systems and Software*, 26(1):87–96, 1994.

[134] Fernando Brito e Abreu, Miguel Goulão, and Rita Esteves. Toward the design quality evaluation of object-oriented software systems. In *Proceedings of the International Conference on Software Quality*, pages 43–57, 1995.

[135] Frederick P. Brooks, Jr. No silver bullet: Essence and accidents of software engineering. *Computer*, 20(4):10–19, 1987.

[136] Glenn J. Browne, Mitzi G. Pitts, and James C. Wetherbe. Cognitive stopping rules for terminating information search in online tasks. *MIS Quarterly*, 31(1):89–104, 2007.

[137] Marcel Bruch, Martin Monperrus, and Mira Mezini. Learning from examples to improve code completion systems. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 213–222, 2009.

[138] Marcel Bruch, Thorsten Schäfer, and Mira Mezini. On evaluating recommender systems for API usages. In *Proceedings of the International Workshop on Recommendation Systems for Softare Engineering*, pages 16–20, 2008.

[139] Julien Brunel, Damien Doligez, René Rydhof Hansen, Julia L. Lawall, and Gilles Muller. A foundation for flow-based program matching: Using temporal logic and model checking. In *Proceedings of the ACM SIGPLAN Conference on Principles of Programming Languages*, pages 114–126, 2009.

[140] Chris Buckley and Ellen M. Voorhees. Evaluating evaluation measure stability. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 33–40, 2000.

[141] R. Ian Bull, Andrew Trevors, Andrew J. Malton, and Michael W. Godfrey. Semantic grep: Regular expressions + relational abstraction. In *Proceedings of the Working Conference on Reverse Engineering*, pages 267–276, 2002.

[142] Peter Bulychev and Marius Minea. Duplicate code detection using anti-unification. In *Proceedings of the Spring Young Researchers Colloquium on Software Engineering*, pages 51–54, 2008.

[143] Peter Bulychev and Marius Minea. An evaluation of duplicate code detection using anti-unification. In *Proceedings of the International Workshop on Software Clones*, 2009. 6 pages.

[144] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, August 1997.

[145] Michael Burch, Stephan Diehl, and Peter Weißgerber. Visual data mining in software archives. In *Proceedings of the ACM Symposium on Software Visualization*, pages 37–46, 2005.

[146] Elizabeth Burd and Malcolm Munro. Investigating the maintenance implications of the replication of code. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 322–329, 1997.

[147] J. Burghardt. E-generalization using grammars. *Artificial Intelligence Journal*, 165(1):1–35, 2005.

[148] Jean-Marie Burkhardt and Françoise Détienne. An empirical study of software reuse by experts in object-oriented design. In *Proceedings of the IFIP TC13 International Conference on Human–Computer Interactaction*, pages 133–138, 1995.

[149] S. Burson, G. B. Kotik, and L. Z. Markosian. A program transformation approach to automating software re-engineering. In *Proceedings of the IEEE International Computer Software and Applications Conference*, pages 314–322, 1990.

[150] Frank Buschmann. Gardening your architecture, part 1: Refactoring. *IEEE Software*, 28(4):92–94, 2011.

[151] Frank Buschmann. Gardening your architecture, part 2: Reengineering and rewriting. *IEEE Software*, 28(5):21–23, 2011.

[152] Greg Butler and Pierre Dénommée. Documenting frameworks. In Mohamed E. Fayad, Douglas C. Schmidt, and Ralph E. Johnson, editors, *Building Application Frameworks: Object-Oriented Foundations of Framework Design*, chapter 21. John Wiley and Sons, 1999.

[153] Greg Butler, Rudolf K. Keller, and Hafedh Mili. A framework for framework documentation. *ACM Computing Surveys*, 32(1es):15/1–15/7, 2000.

[154] Gregory Butler. Architectural refactoring in framework evolution: A case study. In *Proceedings of the International Conference on Generative Programming and Component Engineering*, pages 128–139, 2002.

[155] Terry Caelli and Serhiy Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(4):515–519, April 2004.

[156] Dongxiang Cai and Miryung Kim. An empirical study of long-lived code clones. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, volume 6603 of *Lecture Notes in Computer Science*, pages 432–446, 2011.

[157] Yuanfang Cai and Kevin J. Sullivan. Modularity analysis of logical design models. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 91–102, 2006.

[158] Gianluigi Caldiera and Victor R. Basili. Identifying and qualifying reusable software components. *Computer*, 24(2):61–70, 1991.

[159] Gerardo Canfora and Luigi Cerulo. Impact analysis by mining software and change request repositories. In *Proceedings of the IEEE International Symposium on Software Metrics*, pages 29/1–29/9, 2005.

[160] Gerardo Canfora, Luigi Cerulo, and Massimiliano Di Penta. Tracking your changes: A language-independent approach. *IEEE Software*, 26(1):50–57, 2009.

[161] Gerardo Canfora, Aniello Cimitile, Andrea De Lucia, and Giuseppe A. Di Lucca. Software salvaging based on conditions. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 424–433, 1994.

[162] Dave Card and Ed Comer. Why do so many reuse programs fail? *IEEE Software*, 11(5):114–115, 1994.

[163] David N. Card, Gerald T. Page, and Frank E. McGarry. Criteria for software modularization. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 372–377, 1985.

[164] S. J. Carriere, S. Woods, and R. Kazman. Software architectural transformation. In *Proceedings of the Working Conference on Reverse Engineering*, pages 13–23, 1999.

[165] S. Castano and V. De Antonellis. A constructive approach to reuse of conceptual components. In *Selected Papers of the International Workshop on Software Reusability*, pages 19–28, 1993.

[166] Mariano Ceccato and Paolo Tonella. Measuring the effects of software aspectization. In *Proceedings of the Workshop on Aspects of Reverse Engineering*, 2004. 5 pages.

[167] Silvio Cesare and Yang Xiang. *Software Similarity and Classification*. Springer, 2012.

[168] Heung Seok Chae, Yong Rae Kwon, and Doo Hwan Bae. A cohesion measure for object-oriented classes. *Software: Practice & Experience*, 30(12):1405–1431, 2000.

[169] Heung Seok Chae, Yong Rae Kwon, and Doo Hwan Bae;. Improving cohesion metrics for classes by considering dependent instance variables. *IEEE Transactions on Software Engineering*, 30(11):826 – 832, 2004.

[170] Ian Chai. *Pedagogical Framework Documentation: How to Document Object-Oriented Frameworks an Empirical Study*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 2000.

[171] Pierre-Antoine Champin and Christine Solnon. Measuring the similarity of labeled graphs. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 2689 of *Lecture Notes in Computer Science*, pages 1066–1067, 2003.

[172] R. Chandrasekaran and M. Dawande. Structural analysis of a fractional matching problem. *Discrete Applied Mathematics*, 157(18):3708–3720, 28 November 2009.

[173] Ned Chapin, Joanne E. Hale, Khaled Md. Khan, Juan F. Ramil, and Wui-Gee Tan. Types of software evolution and software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice*, 13(1):3–30, 2001.

[174] Annie Chen, Eric Chou, Joshua Wong, Andrew Y. Yao, Qing Zhang, Shao Zhang, and Amir Michail. CVSSearch: Searching through source code using CVS comments. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 364–373, 2001.

[175] Kunrong Chen and Václav Rajlich. Case study of feature location using dependence graphs. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 241–247, 2000.

[176] Tao Chen, Qing Gu, Shusen Wang, Xiaoan Chen, and Daoxu Chen. Module-based large-scale software evolution based on complex networks. In *Proceedings of the IEEE International Conference on Computer and Information Technology*, pages 798–803, 2008.

[177] Yih-Farn Chen, Michael Y. Nishimoto, and C. V. Ramamoorthy. The C information abstraction system. *IEEE Transactions on Software Engineering*, 16(3):325–334, 1990.

[178] Yih-Farn R. Chen, Glenn S. Fowler, Eleftherios Koutsofios, and Ryan S. Wallach. Ciao: A graphical navigator for software and document repositories. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 66–75, 1995.

[179] Herman Chernoff and E. L. Lehmann. The use of maximum likelihood estimates in χ^2 tests for goodness-of-fit. *Annals of Mathematical Statistics*, 25(3):579–586, 1954.

[180] Ed H. Chi, Adam Rosien, Gesara Supattanasiri, Amanda Williams, Christiaan Royer, Celia Chow, Erica Robles, Brinda Dalal, Julie Chen, and Steve Cousins. The Bloodhound Project: Automating discovery of web usability issues using the InfoScent simulator. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 505–512, 2003.

[181] S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, 1994.

[182] Yves Chiricota, Fabien Jourdan, and Guy Melançon. Software components capture using graph clustering. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 217–226, 2003.

[183] Kingsum Chow and David Notkin. Semi-automatic update of applications in response to library changes. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 359–368, 1994.

[184] Michael Christensen, Christian Heide Damm, Klaus Marius Hansen, Elmer Sandvad, and Michael Thomsen. Design and evolution of software architecture in practice. In *Proceedings of the International Conference on Technology of Object-Oriented Languages and Systems*, pages 2–15, 1999.

[185] Michael Christensen, Christian Heide Damm, Klaus Marius Hansen, Elmer Sandvad, and Michael Thomsen. Software architectural evolution in the Dragon project. Technical report, University of Aarhus, Aarhus, Denmark, 2009.

[186] Steven Clarke. Measuring API usability. *Dr. Dobb's Journal*, pages S6–S9, 2004.

[187] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[188] Brendan Cleary, Chris Exton, Jim Buckley, and Michael English. An empirical analysis of information retrieval based concept location techniques in software comprehension. *Empirical Software Engineering*, 14(1):93–130, 2009.

[189] Wesley Coelho and Gail C. Murphy. Presenting crosscutting structure with active models. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 158–168, 2006.

[190] Michael L. Collard, Huzefa Kagdi, and Jonathan I. Maletic. Factoring changes for iterative change management. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 217–226, 2006.

[191] Giulio Concas, Michele Marchesi, Sandro Pinna, and Nicola Serra. Power-laws in a large object-oriented software system. *IEEE Transactions on Software Engineering*, 33(10):687–708, 2007.

[192] Charles Consel and Olivier Danvy. Tutorial notes on partial evaluation. In *Proceedings of the ACM SIGPLAN Conference on Principles of Programming Languages*, pages 493–501, 1993.

[193] Panos Constantopoulos, Matthias Jarke, John Mylopoulos, and Yannis Vassiliou. The software information base: A server for reuse. *VLDB Journal*, 4(1):1–43, 1995.

[194] Juliet Corbin and Anselm Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1):3–21, 1990.

[195] James R. Cordy. Comprehending reality: Practical barriers to industrial adoption of software maintenance automation. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 196–205, 2003.

[196] James R. Cordy. TXL: A language for programming language tools and applications. *Electronic Notes in Theoretical Computer Science*, 110:3–31, 2004.

[197] James R. Cordy. The TXL source transformation language. *Science of Computer Programming*, 61(3):190–210, 2006.

[198] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[199] Alberto Costa Neto, Marcio de Medeiros Ribeiro, Marcos Dosea, Rodrigo Bonifacio, Paulo Borba, and Sérgio Soares. Semantic dependencies and modularity of aspect-oriented software. In *Proceedings of the International Workshop on Assessment of Contemporary Modularization Techniques*, pages 11/1–11/3, 2007.

[200] Steve Counsell, Emilia Mendes, and Stephen Swift. Comprehension of object-oriented software cohesion: The empirical quagmire. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 33–42, 2002.

[201] Steve Counsell, Stephen Swift, Allan Tucker, and Emilia Mendes. Object-oriented cohesion subjectivity amongst experienced and novice developers: an empirical study. *SIGSOFT Software Engineering Notes*, 31(5):3/1–3/10, 2006.

[202] Patrick Cousot and Radhia Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of the ACM SIGPLAN Conference on Principles of Programming Languages*, pages 238–252, 1977.

[203] Anthony Cox and Charles Clarke. Syntactic approximation using iterative lexical analysis. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 154–164, 2003.

[204] D. R. Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society Series B (Methodology)*, 20(2):215–242, 1958.

[205] Harald Cramér. *Mathematical Models of Statistics*. Princeton University Press, 1946.

[206] Kevin Crowston, Hala Annabi, James Howison, and Chengetai Masango. Effective work practices for FLOSS development: A model and propositions. In *Proceedings of the Hawaii International Conference on Systems Science*, pages 197a/1–197a/9, 2005.

[207] Kevin Crowston and Barbara Scozzi. Coordination practices within FLOSS development teams: The bug fixing process. In *Proceedings of the International Workshop on Computer Supported Activity and Coordination*, pages 21–30, 2004.

[208] Kevin Crowston, Kangning Wei, Qing Li, and James Howison. Core and periphery in free/libre and open source software team communications. In *Proceedings of the Hawaii International Conference on Systems Science*, pages 118a/1–118a/7, 2006.

[209] Christoph Csallner and Yannis Smaragdakis. JCrasher: An automatic robustness tester for Java. *Software: Practice & Experience*, 34(11):1025–1050, 2004.

[210] Félix Cuadrado, Boni Garcia, Juan C. Dueñas, and Hugo A. Parada. A case study on software evolution towards service-oriented architecture. In *Workshop Proceedings of the International Conference on Advanced Information Networking and Applications*, pages 1399–1404, 2008.

[211] Davor Čubranić, Gail C. Murphy, Janice Singer, and Kellogg S. Booth. Hipikat: A project memory for software development. *IEEE Transactions on Software Engineering*, 31(6):446–465, June 2005.

[212] Bruno Carreiro da Silva, Eduardo Figueiredo, Alessandro Garcia, and Daltro Nunes. Refactoring of crosscutting concerns with metaphor-based heuristics. *Electronic Notes in Theoretical Computer Science*, 233:105–125, 2009.

[213] Barthélémy Dagenais and Laurie Hendren. Enabling static analysis for partial Java programs. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 313–328, 2008.

[214] Barthélémy Dagenais and Harold Ossher. Automatically locating framework extension examples. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 203–213, 2008.

[215] Barthélémy Dagenais and Martin Robillard. Creating and evolving developer documentation: Understanding the decisions of open source contributors. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 127–136, 2010.

[216] Barthélémy Dagenais and Martin Robillard. Recommending adaptive changes for framework evolution. *ACM Transactions on Software Engineering and Methodology*, 20(4):19/1–19/35, 2011.

[217] Ole-Johan Dahl and Kristen Nygaard. SIMULA: An ALGOL-based simulation language. *Communications of the ACM*, 9(9):671–678, 1966.

[218] Gerald E. Dallal. http://www.jerrydallal.com/LHSP/LHSP.htm.

[219] David P. Darcy, Chris F. Kemerer, Sandra Slaughter, and James E. Tomayko. The structural complexity of software: An experimental test. *IEEE Transactions on Software Engineering*, 31(11):982–995, 2005.

[220] Brian de Alwis and Gail C. Murphy. Using visual momentum to explain disorientation in the Eclipse IDE. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 51–54, 2006.

[221] Pedro de la Cámara, María del Mar Gallardo, and Pedro Merino. Abstract matching for software model checking. In *Proceedings of the International Conference on Model Checking Software*, pages 182–200, 2006.

[222] Lakshitha de Silva and Dharini Balasubramaniam. Controlling software architecture erosion: A survey. *Journal of Systems and Software*, 85(1):132–151, 2012.

[223] Tulio de Souza Alcantara, Jörg Denzinger, Jennifer Ferreia, and Frank Maurer. Learning gestures for interacting with low-fidelity prototypes. In *Proceedings of the International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, pages 32–36, 2012.

[224] Thomas R. Dean and James R. Cordy. A syntactic theory of software architecture. *IEEE Transactions on Software Engineering*, 21(4):302–313, April 1995.

[225] Uri Dekel and James D. Herbsleb. Reading the documentation of invoked API functions in program comprehension. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 168–177, 2009.

[226] Christian Del Rosso. Comprehend and analyze knowledge networks to improve software evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, 21(3):189–215, 2009.

[227] Serge Demeyer, Stéphane Ducasse, and Oscar Nierstrasz. Finding refactorings via change metrics. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 166–177, 2000.

[228] Jim des Rivières. Enable Eclipse to be used as a rich client platform. https://bugs.eclipse.org/bugs/show_bug.cgi?id=36967, April 2003. Eclipse Bug 36967.

[229] Françoise Détienne. *Software Design: Cognitive Aspects*. Springer, 2001.

[230] Harpal Dhama. Quantitative models of cohesion and coupling in software. *Journal of Systems and Software*, 29(1):65–74, 1995.

[231] Roberto Di Cosmo. Type isomorphisms in a type-assignment framework: From library searches using types to the completion of the ML type checker. In *Proceedings of the ACM SIGPLAN Conference on Principles of Programming Languages*, pages 200–210, 1992.

[232] Danny Dig, Can Comertoglu, Darko Marinov, and Ralph Johnson. Automated detection of refactorings in evolving components. In *Proceedings of the European Conference on Object-Oriented Programming*, volume 4067 of *Lecture Notes in Computer Science*, pages 404–428, 2006.

[233] Danny Dig and Ralph Johnson. The role of refactorings in API evolution. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 389– 398, 2005.

[234] Danny Dig and Ralph Johnson. How do APIs evolve? A story of refactoring. *Journal of Software Maintenance and Evolution: Research and Practice*, 18(2):83–107, 2006.

[235] Danny Dig and Ralph Johnson. How do APIs evolve? A story of refactoring. *Journal of Software Maintenance: Research and Practice*, 18(2):83–107, 2006.

[236] Danny Dig, Kashif Manzoor, Ralph E. Johnson, and Tien N. Nguyen. Effective software merging in the presence of object-oriented refactorings. *IEEE Transactions on Software Engineering*, 34(3):321–335, 2008.

[237] Danny Dig, Kashif Manzoor, Tien N. Nguyen, and Ralph Johnson. Refactoring-aware software merging and configuration management. *SIGSOFT Software Engineering Notes*, 31(6):16/1–16/2, 2006. 2 pages.

[238] Danny Dig, Stas Negara, Vibhu Mohindra, and Ralph Johnson. ReBA: Refactoring-aware binary adaptation of evolving libraries. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 441–450, 2008.

[239] E. W. Dijkstra. Structured programming. In J. N. Buxton and B. Randell, editors, *Software Engineering Techniques*, pages 84–87. NATO Scientific Affairs Division, 1970.

[240] Edsger W. Dijkstra. Notes on structured programming. In *Structured Programming*, chapter 1, pages 1–82. Academic Press Ltd., 1972.

[241] Dila: Dynamic load-time instrumentation library. http://wala.sourceforge.net/wiki/index.php/GettingStarted:wala.dila.

[242] Xinyi Dong and Michael W. Godfrey. Identifying architectural change patterns in object-oriented systems. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 33–42, 2008.

[243] Natalia Dragan. Emergent laws of method and class stereotypes in object oriented software. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 550–555, 2011.

[244] Natalia Dragan, Michael L. Collard, Maen Hammad, and Jonathan I. Maletic. Using stereotypes to help characterize commits. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 520–523, 2011.

[245] Thomas Drake. Measuring software quality: A case study. *Computer*, 29(11):78–87, 1996.

[246] Ekwa Duala-Ekoko and Martin P. Robillard. Clone region descriptors: Representing and tracking duplication in source code. *ACM Transactions on Software Engineering and Methodology*, 20(1):3/1–3/31, 2010.

[247] Ekwa Duala-Ekoko and Martin P. Robillard. Using structure-based recommendations to facilitate discoverability in APIs. In *Proceedings of the European Conference on Object-Oriented Programming*, pages 79–104, 2011.

[248] Stéphane Ducasse, Oscar Nierstrasz, and Matthias Rieger. On the effectiveness of clone detection by string matching. *Journal of Software Maintenance and Evolution: Research and Practice*, 18(1):37–58, 2006.

[249] Stéphane Ducasse, Matthias Rieger, and Serge Demeyer. A language independent approach for detecting duplicated code. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 109–118, 1999.

[250] Liesbeth Dusink and Jan van Katwijk. Reuse dimensions. In *Proceedings of the ACM Symposium on Software Reusability*, pages 137–149, 1995.

[251] Marc Eaddy, Alfred Aho, and Gail C. Murphy. Identifying, assigning, and quantifying crosscutting concerns. In *Proceedings of the International Workshop on Assessment of Contemporary Modularization Techniques*, pages 2/1–2/6, 2007.

[252] Marc Eaddy, Thomas Zimmermann, Kaitlin D. Sherwood, Vibhav Garg, Gail C. Murphy, Nachiappan Nagappan, and Alfred V. Aho. Do crosscutting concerns cause defects? *IEEE Transactions on Software Engineering*, 34(4):497–515, 2008.

[253] Eclipse Rich Client Platform UI. http://eclipse.org/rcp/generic_workbench_summary.html, December 2003.

[254] Johann Eder, Gerti Kappel, and Michael Schrefl. Coupling and cohesion in object-oriented systems. Technical report, Institut für Informationssysteme, University of Linz, Linz, Austria, 1995.

[255] Nick Edgar. Bug 36967: Enable Eclipse to be used as a rich client platform. https://bugs.eclipse.org/bugs/show_bug.cgi?id=36967, April 2003.

[256] Nick Edgar. Eclipse Rich Client Platform UI, December 2003.

[257] Nick Edgar. Need a trimmed-down org.eclipse.help. https://bugs.eclipse.org/bugs/show_bug.cgi?id=40050, July 2003. Eclipse Bug 40050.

[258] Frank Eichinger and Klemens Bohm. Software-bug localization with graph mining. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, chapter 17, pages 515–546. Springer, 2010.

[259] Stephen G. Eick, Todd L. Graves, Alan F. Karr, J. S. Marron, and Audris Mockus. Does code decay? assessing the evidence from change management data. *IEEE Transactions on Software Engineering*, 27(1):1–12, 2001.

[260] Thomas Eisenbarth, Rainer Koschke, and Daniel Simon. Locating features in source code. *IEEE Transactions on Software Engineering*, 29(3):210–224, 2003.

[261] Sebastian Elbaum, Hui Nee Chin, Matthew B. Dwyer, and Jonathan Dokulil. Carving differential unit test cases from system test cases. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 253–264, 2006.

[262] Sebastian Elbaum, Hui Nee Chin, Matthew B. Dwyer, and Matthew Jorde. Carving and replaying differential unit test cases from system test cases. *IEEE Transactions on Software Engineering*, 35(1):29–45, 2009.

[263] Brian Ellis, Jeffrey Stylos, and Brad Myers. The Factory pattern in API design: A usability evaluation. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 302–312, 2007.

[264] Nicole B. Ellison, Charles Steinfield, and Cliff Lampe. The benefits of facebook "friends": Social capital and college students' use of online social network sites. *Journal of Computer-Mediated Communication*, 12(4):1143–1168, 2007.

[265] David Emms, Richard C. Wilson, and Edwin R. Hancock. Graph matching using the interference of continuous-time quantum walks. *Pattern Recognition*, 42(5):985–1002, May 2009.

[266] David Emms, Richard C. Wilson, and Edwin R. Hancock. Graph matching using the interference of discrete-time quantum walks. *Image and Vision Computing*, 27(7):934–949, 4 June 2009.

[267] Murat Erder and Pierre Pureur. Transitional architectures for enterprise evolution. *IT Professional*, 8(3):10–17, 2006.

[268] Hakan Erdogmus. Representing architectural evolution. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 11/1–11/18, 1998.

[269] P. Erdős and A. Rényi. On the evolution of random graphs. *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, 5:17–61, 1960.

[270] Jacky Estublier and German Vega. Reuse and variability in large software applications. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 316–325, 2005.

[271] Letha H. Etzkorn, Sampson E. Gholston, Julie L. Fortune, Cara E. Stein, Dawn Utley, Phillip A. Farrington, and Glenn W. Cox. A comparison of cohesion metrics for object-oriented systems. *Information and Software Technology*, 46(10):677–687, 2004.

[272] Hoda Fahmy and Richard C. Holt. Software architecture transformations. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 88–96, 2000.

[273] Hoda Fahmy and Richard C. Holt. Using graph rewriting to specify software architectural transformations. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 187–196, 2000.

[274] Hoda M. Fahmy, Richard C. Holt, and James R. Cordy. Wins and losses of algebraic transformations of software architectures. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 51–62, 2001.

[275] Raimar Falke, Pierre Frenzel, and Rainer Koschke. Empirical evaluation of clone detection using syntax suffix trees. *Empirical Software Engineering*, 13(6):601–643, 2008.

[276] Wenfei Fan. Dependencies revisited for improving data quality. In *Proceedings of the ACM SIGMOD–SIGACT–SIGART Symposium on Principles of Database Systems*, pages 159–170, 2008.

[277] Martin S. Feather. Reuse in the context of a transformation-based methodology. In Ted J. Biggerstaff and Alan J. Perlis, editors, *Software Reusability*, volume 1: Concepts and Models, chapter 14, pages 337–359. Addison–Wesley, 1989.

[278] Martin Feilkas, Daniel Ratiu, and Elmar Jürgens. The loss of architectural knowledge during system evolution: An industrial case study. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 188–197, 2009.

[279] Norman E. Fenton and Niclas Ohlsson. Quantitative analysis of faults and failures in a complex software system. *IEEE Transactions on Software Engineering*, 26:797–814, 2000.

[280] M. Ferrer, E. Valveny, and F. Serratosa. Median graph: A new exact algorithm using a distance based on the maximum common subgraph. *Pattern Recognition Letters*, 30(5):579–588, April 2009.

[281] M. Ferrer, E. Valveny, and F. Serratosa. Median graphs: A genetic approach based on new theoretical properties. *Pattern Recognition*, 42(9):2003–2012, September 2009.

[282] C. J. Fidge. Contextual matching of software library components. In *Proceedings of the Asia–Pacific Software Engineering Conference*, pages 297–306, 2002.

[283] Eduardo Figueiredo, Nelio Cacho, Claudio Sant'Anna, Mario Monteiro, Uira Kulesza, Alessandro Garcia, Sérgio Soares, Fabiano Ferrari, Safoora Khan, Fernando Castor Filho, and Francisco Dantas. Evolving software product lines with aspects: An empirical study on design stability. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 261–270, 2008.

[284] Eduardo Figueiredo, Alessandro Garcia, Marcelo Maia, Gabriel Ferreira, Camila Nunes, and Jon Whittle. On the impact of crosscutting concern projection on code measurement. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 81–92, 2011.

[285] Eduardo Figueiredo, Claudio Sant'Anna, Alessandro Garcia, Thiago T. Bartolomei, Walter Cazzola, and Alessandro Marchetto. On the maintainability of aspect-oriented software: A concern-oriented measurement framework. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 183–192, 2008.

[286] Eduardo Figueiredo, Bruno Silva, Claudio Sant'Anna, Alessandro Garcia, Jon Whittle, and Daltro Nunes. Crosscutting patterns and design stability: An exploratory analysis. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 138–147, 2009.

[287] Fernando Castor Filho, Nelio Cacho, Raquel Maranhão, Eduardo Figueiredo, Alessandro Garcia, and Cecília Mary F. Rubira. Exceptions and aspects: The devil is in the details. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 152–162, 2006.

[288] G. R. Finnie, G. E. Wittig, and J.-M. Desharnais. Estimating software development effort with case-based reasoning. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 1266 of *Lecture Notes in Computer Science*, pages 13–22, 1997.

[289] Bernd Fischer. Specification-based browsing of software component libraries. *Automated Software Engineering: An International Journal*, 7(2):179–200, May 2000.

[290] G. Fischer. Cognitive view of reuse and redesign. *IEEE Software*, 4(4):60–72, 1987.

[291] Gerhard Fischer, Scott Henninger, and David Redmiles. Cognitive tools for locating and comprehending software objects for reuse. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 318–328, 1991.

[292] Gerhard Fischer and Brent Reeves. Beyond intelligent interfaces: Exploring, analyzing, and creating success models of cooperative problem solving. *Journal of Applied Intelligence*, 1(4):311–332, 1992.

[293] Marc Fisher, II, Dalai Jin, Gregg Rothermel, and Margaret Burnett. Test reuse in the spreadsheet paradigm. In *Proceedings of the International Symposium on Software Reliability Engineering*, pages 257–268, 2002.

[294] R. A. Fisher. XV.—the correlation between relatives on the supposition of Mendelian inheritance. *Transactions of the Royal Society of Edinburgh*, 52(2):399–433, January 1918.

[295] R. A. Fisher. On the interpretation of χ^2 from contingency tables, and the calculation of P. *Journal of the Royal Statistical Society*, 85(1):87–94, January 1922.

[296] Beat Fluri, Harald C. Gall, and Martin Pinzger. Fine-grained analysis of change couplings. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 66–74, 2005.

[297] Beat Fluri, Michael Wursch, Martin Pinzger, and Harald C. Gall. Change distilling: Tree differencing for fine-grained source code change extraction. *IEEE Transactions on Software Engineering*, 33(11):725–743, 2007.

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

[299] William B. Frakes and Christopher J. Fox. Sixteen questions about software reuse. *Communications of the ACM*, 38(6):75–88, 1995.

[300] William B. Frakes and Christopher J. Fox. Quality improvement using a software reuse failure modes model. *IEEE Transactions on Software Engineering*, 22(4):274–279, 1996.

[301] William B. Frakes and Kyo Kang. Software reuse research: Status and future. *IEEE Transactions on Software Engineering*, 31(7):529–536, 2005.

[302] William B. Frakes and Giancarlo Succi. An industrial study of reuse, quality, and productivity. *Journal of Systems and Software*, 57(2):99–106, 2001.

[303] Gary Froehlich, H. James Hoover, Ling Liu, and Paul Sorenson. Hooking into object-oriented application frameworks. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 491–501, 1997.

[304] Wai-Tat Fu, Thomas G. Kannampallil, and Ruogu Kang. Facilitating exploratory search by model-based navigational cues. In *Proceedings of the International Conference on Intelligence User Interfaces*, pages 199–208, 2010.

[305] Robert M. Fuhrer, Markus Keller, and Adam Kiezun. Advanced refactoring in the Eclipse JDT: Past, present, and future. In *Proceedings of the ACM Workshop on Refactoring Tools*, pages 30–31, 2007.

[306] G. W. Furnas, T. K. Landauer, L. M. Gomez, and S. T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30:964–971, 1987.

[307] Michael Furr and Jeffrey S. Foster. Polymorphic type inference for the JNI. In *Proceedings of the European Symposium on Programming*, pages 309–324, 2006.

[308] Mark Gabel and Zhendong Su. A study of the uniqueness of source code. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 147–156, 2010.

[309] J. E. Gaffney, Jr. and R. D. Cruickshank. A general economics model of software reuse. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 327–337, 1992.

[310] Harald Gall, Mehdi Jazayeri, René R. Klösch, and Georg Trausmuth. Software evolution observations based on product release history. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 160–166, 1997.

[311] Rosalva E. Gallardo-Valencia and Susan Elliott Sim. Internet-scale code search. In *Proceedings of the Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, pages 49–52, 2009.

[312] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[313] Dipayan Gangopadhyay and Subrata Mitra. Understanding frameworks by exploration of exemplars. In *Proceedings of the International Workshop on Computer-Aided Software Engineering*, pages 90–99, 1995.

[314] Dipayan Gangopadhyay and Subrata Mitra. Design by framework completion. *Automated Software Engineering: An International Journal*, 3(3/4):219–237, 1996.

[315] Yang Gao, Guoai Xu, Yixian Yang, Xinxin Niu, and Shize Guo. Empirical analysis of software coupling networks in object-oriented software systems. In *Proceedings of the IEEE International Conference on Software Engineering and Service Science*, pages 178–181, 2010.

[316] Alessandro Garcia, Cláudio Sant'Anna, Eduardo Figueiredo, Uirá Kulesza, Carlos Lucena, and Arndt von Staa. Modularizing design patterns with aspects: A quantitative study. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 3–14, 2005.

[317] David Garlan, Robert Allen, and John Ockerbloom. Architectural mismatch: Why reuse is so hard. *IEEE Software*, 12(6):17–26, 1995.

[318] David Garlan, Robert Allen, and John Ockerbloom. Architectural mismatch: Why reuse is still so hard. *IEEE Software*, 26(4):66–69, 2009.

[319] David Garlan, Jeffrey M. Barnes, Bradley R. Schmerl, and Orieta Celiku. Evolution styles: Foundations and tool support for software architecture evolution. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture/European Conference on Sofware Architecture*, pages 131–140, 2009.

[320] David Garlan and Bradley R. Schmerl. Ævol: A tool for defining and planning architecture evolution. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 591–594, 2009.

[321] E. S. Garnett and J. A. Mariani. Software reclamation. *Software Engineering Journal*, 5(3):185–191, 1990.

[322] Jaco Geldenhuys, Matthew B. Dwyer, and Willem Visser. Probabilistic symbolic execution. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 166–176, 2012.

[323] Generic workbench structure. http://eclipse.org/rcp/generic_workbench_structure.html, November 2003.

[324] Daniel German and Audris Mockus. Automating the measurement of open source projects. In *Proceedings of the Workshop on Open Source Software Engineering*, pages 63–67, 2003.

[325] Daniel M. German. An empirical study of fine-grained software modifications. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 316–325, 2004.

[326] Daniel M. German. Using software trails to reconstruct the evolution of software. *Journal of Software Maintenance and Evolution: Research and Practice*, 16(6):367–384, November/December 2004. Special Issue: Analyzing the Evolution of Large-Scale Software.

[327] Daniel M. German, Massimiliano Di Penta, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Code siblings: Technical and legal implications of copying code between applications. In *Proceedings of the International Working Conference on Mining Software Repositories*, pages 81–90, 2009.

[328] Daniel M. German, Yuki Manabe, and Katsuro Inoue. A sentence-matching method for automatic license identification of source code files. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 437–446, 2010.

[329] Celina Gibbs, Chunjan Robin Liu, and Yvonne Coady. Sustainable system infrastructure and big bang evolution: Can aspects keep pace? In *Proceedings of the European Conference on Object-Oriented Programming*, volume 3586 of *Lecture Notes in Computer Science*, pages 241–261, 2005.

[330] Tudor Gîrba and Stéphane Ducasse. Modeling history to analyze software evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, 18(3):207–236, 2006.

[331] Malcolm Gladwell. *The Tipping Point: How Little Things Can Make a Big Difference*. Little Brown, 2000.

[332] Barney G. Glaser and Anselm L. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Transaction, 1967.

[333] M. W Godfrey and D. M German. The past, present, and future of software evolution. In *Proceedings of the Frontiers of Software Maintenance*, 2008.

[334] Michael Godfrey and Qiang Tu. Tracking structural evolution using origin analysis. In *Proceedings of the ACM International Workshop on Principles of Software Evolution*, pages 117–119, 2002.

[335] Michael W. Godfrey and Qiang Tu. Evolution in open source software: A case study. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 131–142, 2000.

[336] Michael W. Godfrey and Lijie Zou. Using origin analysis to detect merging and splitting of source code entities. *IEEE Transactions on Software Engineering*, 31(2):166–181, 2005.

[337] Kwang-Il Goh, Eulsik Oh, Hawoong Jeong, Byungnam Kahng, and Doochul Kim. Classification of scale-free networks. *Proceedings of the National Academy of Sciences*, 99(20):12583–12588, 2002.

[338] Nicolas Gold and Andrew Mohan. A framework for understanding conceptual changes in evolving source code. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 431–439, 2003.

[339] P. Gomes, P. Gandola, and J. Cordeiro. Helping software engineers reusing UML class diagrams. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 4626 of *Lecture Notes in Computer Science*, pages 449–462, 2007.

[340] P. Gomes, F. C. Pereira, P. Carreiro, P. Paiva, N. Seco, J. L. Ferreira, and C. Bento. Solution verification in software design: A CBR approach. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 2689 of *Lecture Notes in Computer Science*, pages 171–185, 2003.

[341] Carsten Görg and Peter Weißgerber. Detecting and visualizing refactorings from software archives. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 205–214, 2005.

[342] Carsten Görg and Peter Weißgerber. Error detection by refactoring reconstruction. In *Proceedings of the International Workshop on Mining Software Repositories*, 2005. 5 pages.

[343] Martin Th. Görg and Jianjun Zhao. Identifying semantic differences in AspectJ programs. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 25–36, 2009.

[344] James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification*. Addison-Wesley, 3rd edition, 2005.

[345] James Gosling, Bill Joy, Guy Steele, Gilad Bracha, and Alex Buckley. *The Java Language Specification*. Addison-Wesley, java SE 7 edition, 2012. http://docs.oracle.com/javase/specs/jls/se7/html/index.html.

[346] Mohamed G. Gouda and Ted Herman. Adaptive programming. *IEEE Transactions on Software Engineering*, 17(9):911–921, 1991.

[347] Judith E. Grass. Cdiff: A syntax directed diff for C++ programs. In *Proceedings of the USENIX C++ Technical Conference*, pages 181–193, 1992. No apparent electronic version is available.

[348] Phil Greenwood, Thiago Bartolomei, Eduardo Figueiredo, Marcos Dosea, Alessandro Garcia, Nelio Cacho, Cláudio Sant'Anna, Sergio Soares, Paulo Borba, Uirá Kulesza, and Awais Rashid. On the impact of aspectual decompositions on design stability: An empirical study. In *Proceedings of the European Conference on Object-Oriented Programming*, volume 4609 of *Lecture Notes in Computer Science*, pages 176–200, 2007.

[349] Richard Gregory and Ladan Tahvildari. Architectural evolution: A case study of Apache. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.5886&rep=rep1&type=pdf, April 1999. Department of Electrical and Computer Engineering, University of Waterloo.

[350] Christiane Gresse von Wangenheim, Klaus-Dieter Althoff, and Ricardo M. Barcia. Goal-oriented and similarity-based retrieval of software engineering experienceware. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, volume 1756 of *Lecture Notes in Computer Science*, pages 118–141, 2000.

[351] Wolfgang Grieskamp. Multi-paradigmatic model-based testing. In *Revised Papers of the International Workshop on Formal Approaches to Software Testing and Runtime Verification*, volume 4262 of *Lecture Notes in Computer Science*, pages 1–19, 2006.

[352] R. Grimm. Better extensibility through modular syntax. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 38–51, 2006.

[353] William G. Griswold, Morison I. Chen, Robert W. Bowdidge, Jenny L. Cabaniss, Van B. Nguyen, and J. David Morgenthaler. Tool support for planning the restructuring of data abstractions in large systems. *IEEE Transactions on Software Engineering*, 24(7):534–558, 1998.

[354] William G. Griswold and David Notkin. Automated assistance for program restructuring. *ACM Transactions on Software Engineering and Methodology*, 2(3):228–269, 1993.

[355] Paul Gross and Caitlin Kelleher. Non-programmers identifying functionality in unfamiliar code: Strategies and barriers. *Journal of Visual Languages and Computing*, 21(5):263–276, 2010.

[356] Lars Grunske. Formalizing architectural refactorings as graph transformation systems. In *Proceedings of the International Conference on Software Engineering, Artificial Intelligence, Networks, and Parallel/Distributed Computing/ACIS International Workshop on Self-Assembling Wireless Networks*, pages 324–329, 2005.

[357] Yann-Gaël Guéhéneuc, Houari Sahraoui, and Farouk Zaidi. Fingerprinting design patterns. In *Proceedings of the Working Conference on Reverse Engineering*, pages 172–181, 2004.

[358] Yuepu Guo, Carolyn Seaman, Rebeka Gomes, Antonio Cavalcanti, Graziela Tonin, Fabio Q. B. Da Silva, Andre L. M. Santos, and Clauirton Siebra. Tracking technical debt: An exploratory case study. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 528–531, 2011.

[359] Tibor Gyimóthy, Rudolf Ferenc, and István Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering*, 31(10):897–910, 2005.

[360] Florian S. Gysin. Improved social trustability of code search results. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 513–514, 2010.

[361] Stefan Haefliger, Georg von Krogh, and Sebastian Spaeth. Code reuse in open source software. *Management Science*, 54(1):180–193, 2008.

[362] Sonia Haiduc, Jairo Aponte, and Andrian Marcus. Supporting program comprehension with source code summarization. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, volume 2, pages 223–226, 2010.

[363] Sonia Haiduc and Andrian Marcus. On the use of domain terms in source code. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 113–122, 2008.

[364] Raju Halder and Agostino Cortesi. Abstract program slicing on dependence condition graphs. *Science of Computer Programming*, 2012. In press.

[365] William G. J. Halfond, Alessandro Orso, and Panagiotis Manolios. Using positive tainting and syntax-aware evaluation to counter SQL injection attacks. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 175–185, 2006.

[366] Patrick A. V. Hall and Geoff R. Dowling. Approximate string matching. *ACM Computing Surveys*, 12(4):381–402, December 1980.

[367] Christine A. Halverson, Jason B. Ellis, Catalina Danis, and Wendy A. Kellogg. Designing task visualizations to support the coordination of work in software development. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, pages 39–48, 2006.

[368] Edwin R. Hancock and Richard C. Wilson. Pattern analysis with graphs: Parallel work at Bern and York. *Pattern Recognition Letters*, 33(7):833–841, May 2012.

[369] Jan Hannemann, Gail C. Murphy, and Gregor Kiczales. Role-based refactoring of crosscutting concerns. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 135–146, 2005.

[370] Mehdi T. Harandi. The role of analogy in software reuse. In *Proceedings of the ACM SIGAPP Symposium on Applied Computing*, pages 40–47, 1993.

[371] R. Harrison, S. Counsell, and R. Nithi. An overview of object-oriented design metrics. In *Proceedings of the IEEE International Workshop on Software Technology and Practice*, pages 230–235, 1997.

[372] R. Harrison, S. Counsell, and R. Nithi. Coupling metrics for object-oriented design. In *Proceedings of the IEEE International Symposium on Software Metrics*, pages 150–157, 1998.

[373] Mary Jean Harrold and Ning Ci. Reuse-driven interprocedural slicing. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 74–83, 1998.

[374] Björn Hartmann, Sean Follmer, Antonio Ricciardi, Timothy Cardenas, and Scott R. Klemmer. d.note: Revising user interfaces through change tracking, annotations, and alternatives. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 493–502, 2010.

[375] Björn Hartmann, Daniel MacDougall, Joel Brandt, and Scott R. Klemmer. What would other programmers do: Suggesting solutions to error messages. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1019–1028, 2010.

[376] Les Hatton. Power-law distributions of component size in general software systems. *IEEE Transactions on Software Engineering*, 35(4):566–572, 2009.

[377] Lile P. Hattori and Michele Lanza. On the nature of commits. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 63–71, 2008.

[378] Paul Heckel. A technique for isolating differences between files. *Communications of the ACM*, 21(4):264–268, April 1978.

[379] Richard Helm, Ian M. Holland, and Dipayan Gangopadhyay. Contracts: Specifying behavioral compositions in object-oriented systems. In *Proceedings of the European Conference on Object-Oriented Programming/ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 169–180, 1990.

[380] Help API change for Rich Client Platform. http://eclipse.org/rcp/restructuring_help.html, September 2003.

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

[382] Johannes Henkel and Amer Diwan. CatchUp!: Capturing and replaying refactorings to support API evolution. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 274–283, 2005.

[383] Scott Henninger. Retrieving software objects in an example-based programming environment. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 251–260, 1991.

[384] Scott Henninger. Using iterative refinement to find reusable software. *IEEE Software*, 11(5):48–59, 1994.

[385] Scott Henninger. An evolutionary approach to constructing effective software reuse repositories. *ACM Transactions on Software Engineering and Methodology*, 6(2):111–140, 1997.

[386] James D. Herbsleb and Audris Mockus. Formulation and preliminary test of an empirical theory of coordination in software engineering. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 138–137, 2003.

[387] Emily Hill, Lori Pollock, and K. Vijay-Shanker. Automatically capturing source code context of NL-queries for software maintenance and reuse. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 232–242, 2009.

[388] Emily Hill, Lori Pollock, and K. Vijay-Shanker. Improving source code search with natural language phrasal representations of method signatures. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 524–527, 2011.

[389] Rosco Hill and Joe Rideout. Automatic method completion. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 228–235, 2004.

[390] Abram Hindle, Zhen Ming Jiang, Walid Koleilat, Michael W. Godfrey, and Richard C. Holt. YARN: Animating software evolution. In *Proceedings of the IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 129–136, 2007.

[391] Carl Hinsman, Neeraj Sangal, and Judith Stafford. Large scale refactoring through architecture visibility. Technical Report TR-2007-8, Tufts University, Medford, Massachusetts, USA, September 2007.

[392] Martin Hitz and Behzad Montazeri. Chidamber and Kemerer's metrics suite: A measurement theory perspective. *IEEE Transactions on Software Engineering*, 22(4):267–271, 1996.

[393] Lorin Hochstein and Mikael Lindvall. Combating architectural degeneration: A survey. *Information and Software Technology*, 47(10):643–656, 2005.

[394] Kevin J. Hoffman and Patrick Eugster. Towards reusable components with aspects: An empirical study on modularity and obliviousness. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 91–100, 2008.

[395] Ian M. Holland. Specifying reusable components using contracts. In *Proceedings of the European Conference on Object-Oriented Programming*, volume 615 of *Lecture Notes in Computer Science*, pages 287–308, 1992.

[396] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 1992.

[397] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65:026107/1–026107/4, 2002.

[398] Ric Holt and Jason Y. Pak. GASE: Visualizing software evolution-in-the-large. In *Proceedings of the Working Conference on Reverse Engineering*, pages 163–167, 1996.

[399] Hyoung Seok Hong, Insup Lee, and Oleg Sokolsky. Abstract slicing: A new approach to program slicing based on abstract interpretation and model checking. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 25–34, 2005.

[400] Kurt Hornik. The R FAQ. http://CRAN.R-project.org/doc/FAQ/R-FAQ.html, 2012.

[401] Susan Horwitz. Identifying the semantic and textual differences between two versions of a program. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 234–245, 1990.

[402] Daqing Hou. Investigating the effects of framework design knowledge in example-based framework learning. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 37–46, 2008.

[403] Daqing Hou and H. James Hoover. Towards specifying constraints for object-oriented frameworks. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 5/1–5/14, 2001.

[404] Xiaodi Huang and Wei Lai. Clustering graphs for visualization via node similarities. *Journal of Visual Languages and Computing*, 17(3):225–253, June 2006.

[405] Oliver Hummel. *Semantic Component Retrieval in Software Engineering*. PhD thesis, Universität Mannheim, Mannheim, Germany, 2008.

[406] Oliver Hummel. Facilitating the comparison of software retrieval systems through a reference reuse collection. In *Proceedings of the Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, pages 17–20, 2010.

[407] Oliver Hummel and Colin Atkinson. Extreme harvesting: Test driven discovery and reuse of software components. In *Proceedings of the IEEE International Conference on Information Reuse and Integration*, pages 66–72, 2004.

[408] Oliver Hummel and Colin Atkinson. Supporting agile reuse through extreme harvesting. In *Proceedings of the International Conference on Agile Processes in Software Engineering and Extreme Programming*, volume 4536 of *Lecture Notes in Computer Science*, pages 28–37, 2007.

[409] Oliver Hummel and Colin Atkinson. The Managed Adapter Pattern: Facilitating glue code generation for component reuse. In *Proceedings of the International Conference on Software Reuse*, volume 5791 of *Lecture Notes in Computer Science*, pages 211–224, 2009.

[410] Oliver Hummel, Werner Janjic, and Colin Atkinson. Evaluating the efficiency of retrieval methods for component repositories. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, pages 404–409, 2007.

[411] Oliver Hummel, Werner Janjic, and Colin Atkinson. Code Conjurer: Pulling reusable software out of thin air. *IEEE Software*, 25(5):45–52, 2008.

[412] Oliver Hummel, Werner Janjic, and Colin Atkinson. Proposing software design recommendations based on component interface intersecting. In *Proceedings of the International Workshop on Recommendation Systems for Softare Engineering*, pages 64–68, 2010.

[413] Andrew Hunt and David Thomas. *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley, 1999. Section 2.7: The Evils of Duplication.

[414] J. J Hunt and W. F Tichy. Extensible language-aware merging. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 511–520, 2002.

[415] J. W. Hunt and M. D. McIlroy. An algorithm for differential file comparison. Computer Science Technical Report 41, Bell Telephone Laboratories, 1976.

[416] Sunny Huynh and Yuanfang Cai. An evolutionary approach to software modularity analysis. In *Proceedings of the International Workshop on Assessment of Contemporary Modularization Techniques*, pages 6/1–6/6, 2007.

[417] David Hyland-Wood, David Carrington, and Simon Kaplan. Scale-free nature of Java software package, class and method collaboration graphs. Technical Report TR-MS1286, University of Maryland, College Park, 2006.

[418] Makoto Ichii, Makoto Matsushita, and Katsuro Inoue. An exploration of power-law in use-relation of Java software systems. In *Proceedings of the Australian Software Engineering Conference*, pages 422–431, 2008.

[419] Akinori Ihara, Masao Ohira, and Ken-ichi Matsumoto. An analysis method for improving a bug modification process in open source software development. In *Proceedings of the ACM International Workshop on Principles of Software Evolution/ERCIM Workshop on Software Evolution*, pages 135–144, 2009.

[420] Katsuro Inoue, Reishi Yokomori, Tetsuo Yamamoto, Makoto Matsushita, and Shinji Kusumoto. Ranking significance of software components based on use relations. *IEEE Transactions on Software Engineering*, 31(3):213–225, 2005.

[421] Paola Inverardi, Alexander L. Wolf, and Daniel Yankelevich. Static checking of system behaviors using derived component assumptions. *ACM Transactions on Software Engineering and Methodology*, 9(3):239–272, July 2000.

[422] Patricia Jablonski and Daqing Hou. CReN: A tool for tracking copy-and-paste code clones and renaming identifiers consistently in the IDE. In *Proceedings of the Eclipse Technology eXchange*, pages 16–20, 2007.

[423] Daniel Jackson and David A Ladd. Semantic Diff: A tool for summarizing the effects of modifications. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 243–252, 1994.

[424] Werner Janjic, Oliver Hummel, and Colin Atkinson. More archetypal usage scenarios for software search engines. In *Proceedings of the Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, pages 21–24, 2010.

[425] Anton Jansen and Jan Bosch. Evaluation of tool support for architectural evolution. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 375–378, 2004.

[426] Slinger Jansen, Sjaak Brinkkemper, Ivo Hunink, and Cetin Demir. Pragmatic and opportunistic reuse in innovative start-up companies. *IEEE Software*, 25(6):42–49, 2008.

[427] Doug Janzen and Kris De Volder. Navigating and querying code without getting lost. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 178–187, 2003.

[428] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002.

[429] Hojun Jaygarl, Sunghun Kim, Tao Xie, and Carl K. Chang. OCAT: Object capture-based automated testing. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 159–170, 2010.

[430] Jun-Jang Jeng and Betty H. C. Cheng. Specification matching for software reuse: A foundation. In *Proceedings of the ACM Symposium on Software Reusability*, pages 97–105, 1995.

[431] S. Jenkins and S. R. Kirk. Software architecture graphs as complex networks: A novel partitioning scheme to measure stability and evolution. *Information Sciences*, 177(12):2587–2601, 2007.

[432] H. Jeong, B. Tombor, R. Albert, Z. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

[433] Sae Young Jeong, Yingyu Xie, Jack Beaton, Brad A. Myers, Jeff Stylos, Ralf Ehret, Jan Karstens, Arkin Efeoglu, and Daniela K. Busse. Improving documentation for eSOA APIs through user studies. In *Proceedings of the International Symposium on End-User Development*, pages 86–105, 2009.

[434] Jeong-Hoon Ji, Soo-Hyun Park, Gyun Woo, and Hwan-Gue Cho. Source code similarity detection using adaptive local alignment of keywords. In *Proceedings of the International Conference on Parallel and Distributed Computing, Applications, and Technologies*, pages 179–180, 2007.

[435] Lingxiao Jiang, Ghassan Misherghi, Zhengdong Su, and Stephane Glondu. DECKARD: Scalable and accurate tree-based detection of code clones. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 96–105, 2007.

[436] Lingxiao Jiang and Zhendong Su. Automatic mining of functionally equivalent code fragments via random testing. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 81–92, 2009.

[437] X. Jiang and H. Bunke. Graph matching. In Petra Perner, editor, *Case-Based Reasoning on Images and Signals*, volume 73 of *Studies in Computational Intelligence*, chapter 5, pages 149–173. Springer, 2008.

[438] Liu Jing, He Keqing, Ma Yutao, and Peng Rong. Scale free in software metrics. In *Proceedings of the IEEE International Computer Software and Applications Conference*, pages 229–235, 2006.

[439] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 154–161, 2005.

[440] Bonnie John, Alonso Vera, Michael Matessa, Michael Freed, and Roger Remington. Automating CPM-GOMS. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 147–154, 2002.

[441] J. Howard Johnson. Identifying redundancy in source code using fingerprints. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, volume 1, pages 171–183, 1993.

[442] Ralph Johnson.

[443] Ralph E. Johnson. Documenting frameworks using patterns. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 63–76, 1992.

[444] Ralph E. Johnson and Brian Foote. Designing reuseable [*sic*] classes. *Journal of Object-Oriented Programming*, 1(2):22–35, 1988.

[445] Igor Jurisica. Similarity-based retrieval for diverse bookshelf software repository users. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 10/1–10/12, 1997.

[446] Huzefa Kagdi, Michael L. Collard, and Jonathan I. Maletic. A survey and taxonomy of approaches for mining software repositories in the context of software evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, 19(2):77–131, 2007.

[447] Huzefa Kagdi, Michael L. Collard, and Jonathan I. Maletic. A survey and taxonomy of approaches for mining software repositories in the context of software evolution. *Journal of Software Maintenance and Evolution: Research and Practice*, 19(2):77–131, 2007.

[448] Huzefa Kagdi and Jonathan I. Maletic. Mining evolutionary dependencies from web-localization repositories. *Journal of Software Maintenance and Evolution: Research and Practice*, 19(5):315–337, 2007.

[449] Huzefa Kagdi, Jonathan I. Maletic, and Bonita Sharif. Mining software repositories for traceability links. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 145–154, 2007.

[450] Cory Kapser and Michael W. Godfrey. 'Cloning considered harmful' considered harmful: Patterns of cloning in software. *Empirical Software Engineering*, 13(6):645–692, 2008.

[451] Christos Katsanos, Nikolaos Tselios, and Nikolaos Avouris. InfoScent Evaluator: A semi-automated tool to evaluate semantic appropriateness of hyperlinks in a web site. In *Proceedings of the Australian Conference on Computer–Human Interaction*, pages 373–376, 2006.

[452] Ishwinder Kaur and Anthony J. Hornof. A comparison of LSA, wordNet and PMI-IR for predicting user click behavior. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 51–60, 2005.

[453] Gabriella Kazai and Mounia Lalmas. eXtended Cumulated Gain measures for the evaluation of content-oriented XML retrieval. *ACM Transactions on Information Systems*, 24(4):503–542, 2006.

[454] Evelyn Fox Keller. Revisiting "scale-free" networks. *BioEssays*, 27(10):1060–1068, 2005.

[455] Rudolf K. Keller, Reinhard Schauer, and Alistair Cockburn. Object-oriented design quality. In *Addendum to the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 63–67, 1997.

[456] Diane Kelly. A study of design characteristics in evolving software using stability as a criterion. *IEEE Transactions on Software Engineering*, 32(5):315–329, 2006.

[457] Joshua Kerievsky. *Refactoring to Patterns*. Addison Wesley, 2004.

[458] Andruid Kerne, Eunyee Koh, Steven Smith, Hyun Choi, Ross Graeber, and Andrew Webb. Promoting emergence in information discovery by representing collections with composition. In *Proceedings of the ACM SIGCHI Conference on Creativity and Cognition*, pages 117–126, 2007.

[459] Reid Kerr and Wolfgang Stuerzlinger. Context-sensitive cut, copy, and paste. In *Proceedings of the Canadian Conference on Computer Science and Software Engineering*, pages 159–166, 2008.

[460] Mik Kersten and Gail C. Murphy. Mylar: A degree-of-interest model for IDEs. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 159–168, 2005.

[461] Mik Kersten and Gail C. Murphy. Using task context to improve programmer productivity. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 1–11, 2006.

[462] Taghi M. Khoshgoftaar, Lofton A. Bullard, and Kehan Gao. Detecting outliers using rule-based modeling for improving CBR-based software quality classification models. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 2689 of *Lecture Notes in Computer Science*, pages 216–230, 2003.

[463] Gregor Kiczales, Jim des Rivières, and Daniel Bobrow. *The Art of the Metaobject Protocol*. MIT Press, 1991.

[464] Gregor Kiczales, Erik Hilsdale, Jim Hugunin, Mik Kersten, Jeffrey Palm, and William G. Griswold. An overview of AspectJ. In *Proceedings of the European Conference on Object-Oriented Programming*, pages 327–353, 2001.

[465] Gregor Kiczales, John Lamping, Anurag Mendhekar, Chris Maeda, Cristina Videira Lopes, Jean-Marc Loingtier, and John Irwin. Aspect-oriented programming. In *Proceedings of the European Conference on Object-Oriented Programming*, volume 1241 of *Lecture Notes in Computer Science*, pages 220–242, 1997.

[466] Jörg Kienzle and Samuel Gélineau. AO challenge: Implementing the ACID properties for transactional objects. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 202–213, 2006.

[467] Beom Jun Kim, Chang No Yoon, Seung Kee Han, and Hawoong Jeong. Path finding strategies in scale-free networks. *Physical Review E*, 65(2):027103/1–027103/4, 2002.

[468] Miryung Kim, Lawrence Bergman, Tessa Lau, and David Notkin. An ethnographic study of copy and paste programming practices in OOPL. In *Proceedings of the International Symposium on Empirical Software Engineering*, pages 83–92, 2004.

[469] Miryung Kim, Matthew Gee, Alex Loh, and Napol Rachatasumrit. Ref-Finder: A refactoring reconstruction tool based on logic query templates. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 371–372, 2010.

[470] Miryung Kim and David Notkin. Program element matching for multi-version program analyses. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 58–64, 2006.

[471] Miryung Kim, David Notkin, and Dan Grossman. Automatic inference of structural changes for matching across program versions. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 333–343, 2007.

[472] Miryung Kim, Vibha Sazawal, David Notkin, and Gail Murphy. An empirical study of code clone genealogies. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 187–196, 2005.

[473] Sunghun Kim, Kai Pan, and E. James Whitehead, Jr. When functions change their names: Automatic detection of origin relationships. In *Proceedings of the Working Conference on Reverse Engineering*, pages 143–152, 2005.

[474] Sunghun Kim, E. James Whitehead, Jr., and Yi Zhang. Classifying software changes: Clean or buggy? *IEEE Transactions on Software Engineering*, 34(2):181–196, 2008.

[475] Taegyun Kim and Nacer Boudjlida. An experience report related to restructuring OODesigner: A CASE tool for OMT. In *Proceedings of the Asia–Pacific Software Engineering Conference*, pages 220–227, 1998.

[476] Barbara Kitchenham, David Ross Jeffery, and Colin Connaughton. Misleading metrics and unsound analyses. *IEEE Software*, 24(2):73–78, 2007.

[477] Konstantin Klemm and Victor M. Eguíluz. Highly clustered scale-free networks. *Physical Review E*, 65(3):036123/1–036123/5, 2002.

[478] Paul Klint, Ralf Lämmel, and Chris Verhoef. Toward an engineering discipline for grammarware. *ACM Transactions on Software Engineering and Methodology*, 14(3):331–380, 2005.

[479] A. Kmiecik and V. Ambriola. Transformations for software architecture model change. *Foundations of Computing and Decision Sciences*, 29(4):329–344, 2004.

[480] Claire Knight and Malcolm Munro. Program comprehension experiences with GXL: Comprehension for comprehension. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 147–156, 2002.

[481] David Knossow, Avinash Sharma, Diana Mateus, and Radu Horaud. Inexact matching of large and sparse graphs using Laplacian eigenvectors. In *Proceedings of the IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition*, pages 144–153, 2009.

[482] Andrew J. Ko, Robert DeLine, and Gina Venolia. Information needs in collocated software development teams. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 344–353, 2007.

[483] Andrew J. Ko and Brad A. Myers. Extracting and answering why and why not questions about Java program output. *ACM Transactions on Software Engineering and Methodology*, 20(2):4/1–4/36, 2010.

[484] Andrew J. Ko, Brad A. Myers, Michael J. Coblenz, and Htet Htet Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *IEEE Transactions on Software Engineering*, 32(12):971–987, 2006.

[485] Jürgen Koenemann and Scott P. Robertson. Expert problem solving strategies for program comprehension. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 125–130, 1991.

[486] Andrew Koenig. Patterns and antipatterns. *Journal of Object-Oriented Programming*, 8(1):46–48, 1995.

[487] Dimitrios S. Kolovos, Richard F. Paige, and Fiona A. C. Polack. Model comparison: A foundation for model composition and model transformation testing. In *Proceedings of the International Workshop on Global Integrated Model Management*, pages 13–20, 2006.

[488] Kostas Kontogiannis, Panos Linos, and Kenny Wong. Comprehension and maintenance of large-scale multi-language software applications. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 497–500, 2006.

[489] Rainer Koppler. A systematic approach to fuzzy parsing. *Software: Practice & Experience*, 27(6):637–649, 1997.

[490] A. Güneş Koru and Khaled El Emam. The theory of relative dependency: Higher coupling concentration in smaller modules. *IEEE Software*, 27(2):81–89, 2010.

[491] Rainer Koschke, Raimar Falke, and Pierre Frenzel. Clone detection using abstract syntax suffix trees. In *Proceedings of the Working Conference on Reverse Engineering*, pages 253–262, 2006.

[492] E. V. Kostylev and V. A. Zakharov. On the complexity of the anti-unification problem. *Discrete Mathematics and Applications*, 18(1):85–98, 2008.

[493] Wojtek Kozaczynski, Jim Ning, and Andre Engberts. Program concept recognition and transformation. *IEEE Transactions on Software Engineering*, 18(12):1065–1075, 1992.

[494] Segla Kpodjedo. Approximate graph matching in software engineering. In *Proceedings of the Working Conference on Reverse Engineering*, pages 295–298, 2009.

[495] Segla Kpodjedo, Philippe Galinier, and Giulio Antoniol. Enhancing a tabu algorithm for approximate graph matching by using similarity measures. In *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 119–130, 2010.

[496] Segla Kpodjedo, Filippo Ricca, Philippe Galinier, and Giuliano Antoniol. Error correcting graph matching application to software evolution. In *Proceedings of the Working Conference on Reverse Engineering*, pages 289–293, 2008.

[497] R. Krikhaar, A. Postma, A. Sellink, M. Stroucken, and C. Verhoef. A two-phase process for software architecture improvement. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 371–380, 1999.

[498] Jens Krinke. Statement-level cohesion metrics and their visualization. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 37–46, 2007.

[499] Jens Krinke, Nicolas Gold, Yue Jia, and David Binkley. Cloning and copying between GNOME projects. In *Proceedings of the International Working Conference on Mining Software Repositories*, pages 98–101, 2010.

[500] Evgeny B. Krissinel and Kim Henrick. Common subgraph isomorphism detection by backtracking search. *Software: Practice & Experience*, 34(6):591–607, May 2004.

[501] Klaus Krogmann. *Reconstruction of Software Component Architectures and Behaviour Models Using Static and Dynamic Analysis*. KIT Scientific Publishing, 2012.

[502] Phillippe B. Kruchten. The 4+1 view model of architecture. *IEEE Software*, 12(6):42–50, 1995.

[503] Charles W. Krueger. Software reuse. *ACM Computing Surveys*, 24(2):131–183, 1992.

[504] Charles W. Krueger. Software product line reuse in practice. In *Proceedings of the IEEE Symposium on Application-Specific Systems and Software Engineering & Technology*, pages 117–118, 2000.

[505] Charles W. Krueger. Towards a taxonomy for software product lines. In *Revised Papers of the International Workshop on Software Product-Family Engineering*, volume 3014 of *Lecture Notes in Computer Science*, pages 323–331, 2004.

[506] Ulf Krumnack, Angela Schwering, Helmar Gust, and Kai-Uwe Kühnberger. Restricted higher-order anti-unification for analogy making. In *Proceedings of the Australian Joint Conference on Artificial Intelligence*, pages 273–282, 2007.

[507] William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[508] M. V. Ksenzov. Architectural refactoring of corporate program systems. *Programming and Computing Software*, 32(1):31–43, 2006.

[509] Adrian Kuhn, Stéphane Ducasse, and Tudor Gîrba. Semantic clustering: Identifying topics in source code. *Information and Software Technology*, 49(3):230–243, 2007.

[510] M. Raveendra Kumar and R. Hari Kumar. Architectural refactoring of a mission critical integration application: A case study. In *Proceedings of the India Software Engineering Conference*, pages 77–83, 2011.

[511] Philipp Kumar and Thomas Baar. Using AOP for discovering and defining executable test cases. In *Proceedings of the International Conference on Perspectives of Systems Informatics*, volume 5947 of *Lecture Notes in Computer Science*, pages 269–281, 2010.

[512] Juha Kuusela. Architectural evolution. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture*, pages 471–478, 1999.

[513] Patricia Lago and Hans van Vliet. Explicit assumptions enrich architectural models. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 206–214, 2005.

[514] Albert Lai and Gail C. Murphy. The structure of features in Java code: An exploratory investigation. In *Proceedings of the Workshop on Multi-dimensional Separation of Concerns*, 1999. 6 pages.

[515] Matthew J. LaMantia, Yuanfang Cai, Alan D. MacCormack, and John Rusnak. Analyzing the evolution of large-scale software systems using design structure matrices and design rule theory: Two exploratory cases. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture*, pages 83–92, 2008.

[516] W. Landi. Undecidability of static analysis. *ACM Letters in Programming Languages and Systems*, 1(4):323–337, 1992.

[517] Beth M. Lange and Thomas G. Moher. Some strategies of reuse in an object-oriented programming environment. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 69–73, 1989.

[518] Filippo Lanubile and Giuseppe Visaggio. Extracting reusable functions by flow graph-based program slicing. *IEEE Transactions on Software Engineering*, 23(4):246–259, 1997.

[519] C. Larman. Protected variation: The importance of being closed. *IEEE Software*, 18(3):89–91, 2001.

[520] Thomas D. LaToza, David Garlan, James D. Herbsleb, and Brad A. Myers. Program comprehension as fact finding. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 361–370, 2007.

[521] Thomas D. LaToza, Gina Venolia, and Robert DeLine. Maintaining mental models: A study of developer work habits. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 492–501, 2006.

[522] Kung-Kiu Lau and Zheng Wang. Software component models. *IEEE Transactions on Software Engineering*, 33(10):709–724, October 2007.

[523] Joseph Lawrance, Rachel Bellamy, Margaret Burnett, and Kyle Rector. Can information foraging pick the fix?: A field study. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 57–64, 2008.

[524] Joseph Lawrance, Rachel Bellamy, Margaret Burnett, and Kyle Rector. Using information scent to model the dynamic foraging behavior of programmers in maintenance tasks. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1323–1332, 2008.

[525] J. L. Lawrence. Why is software always late? *SIGSOFT Software Engineering Notes*, 10(1):19–30, 1985.

[526] Vianney le Clément, Yves Deville, and Christine Solnon. Constraint-based graph matching. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, volume 5732 of *Lecture Notes in Computer Science*, pages 274–288, 2009.

[527] Olivier Le Goaer, Dalila Tamzalit, and Mourad Oussalah. Evolution styles to capitalize evolution expertise within software architectures. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, pages 159–164, 2010.

[528] Olivier Le Goaer, Dalila Tamzalit, Mourad Oussalah, and Abdelhak-Djamel Seriai. Evolution shelf: Reusing evolution expertise within component-based software architectures. In *Proceedings of the IEEE International Computer Software and Applications Conference*, pages 311–318, 2008.

[529] Olivier Le Goaer, Dalila Tamzalit, Mourad Chabane Oussalah, and Abdelhak-Djamel Seriai. Evolution styles to the rescue of architectural evolution knowledge. In *Proceedings of the International Workshop on Sharing and Reusing Architectural Knowledge*, pages 31–36, 2008.

[530] Bongshin Lee, Cynthia S. Parr, Catherine Plaisant, Benjamin B. Bederson, Vladislav D. Veksler, Wayne D. Gray, and Christopher Kotfila. TreePlus: Interactive exploration of networks with enhanced tree layouts. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1414–1426, 2006.

[531] Wan-Jui Lee and Robert P. Duin. An inexact graph comparison approach in joint eigenspace. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, pages 35–44, 2008.

[532] Y.-S. Lee, B.-S. Liang, S.-F. Wu, and F.-J. Wang. Measuring the coupling and cohesion of an object-oriented program based on information flow. In *Proceedings of the International Conference on Software Quality*, pages 81–90, 1995.

[533] M. M. Lehman. Programs, life cycles, and laws of software evolution. *Proceedings of the IEEE*, 68(9):1060–1076, 1980.

[534] M. M. Lehman. Laws of software evolution revisited. In *Proceedings of the European Workshop on Software Process Technology*, volume 1149 of *Lecture Notes in Computer Science*, 1996.

[535] M. M. Lehman and L. A. Belady. *Program Evolution: Processes of Software Change*. Academic Press, 1985.

[536] M. M. Lehman, J. F. Ramil, P. D. Wernick, D. E. Perry, and W. M. Turski. Metrics and laws of software evolution: The nineties view. In *Proceedings of the IEEE International Symposium on Software Metrics*, pages 20–32, 1997.

[537] K. Rustan M. Leino. Efficient weakest preconditions. *Information Processing Letters*, 93(6):281–288, 2005.

[538] Andreas Leitner, Ilinca Ciupa, Manuel Oriol, Bertrand Meyer, and Arno Fiva. Contract driven development = test driven development - writing test cases. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 425–434, 2007.

[539] Otávio Augusto Lazzarini Lemos, Sushil Bajracharya, Joel Ossher, Paulo Cesar Masiero, and Cristina Lopes. Applying test-driven code search to the reuse of auxiliary functionality. In *Proceedings of the ACM SIGAPP Symposium on Applied Computing*, pages 476–482, 2009.

[540] Otávio Augusto Lazzarini Lemos, Sushil Bajracharya, Joel Ossher, Paulo Cesar Masiero, and Cristina Lopes. A test-driven approach to code search and its application to the reuse of auxiliary functionality. *Information and Software Technology*, 53(4):294–306, 2011.

[541] Otávio Augusto Lazzarini Lemos, Sushil Krishna Bajracharya, and Joel Ossher. CodeGenie: A tool for test-driven source code search. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 917–918, 2007.

[542] Otávio Augusto Lazzarini Lemos, Sushil Krishna Bajracharya, Joel Ossher, Ricardo Santos Morla, Paulo Cesar Masiero, Pierre Baldi, and Cristina Videira Lopes. CodeGenie: Using test-cases to search and reuse source code. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 525–526, 2007.

[543] Timothy C. Lethbridge and Robert Laganière. *Object-Oriented Software Engineering: Practical Software Development using UML and Java*. McGraw-Hill, 2nd edition, 2005.

[544] Deyi Li, Yanni Han, and Jun Hu. Complex network thinking in software engineering. In *Proceedings of the International Conference on Computer Science and Software Engineering*, pages 264–268, 2008.

[545] Li Li and A. Jefferson Offutt. Algorithmic analysis of the impact of changes to object-oriented software. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 171–184, 1996.

[546] Wei Li and Sallie Henry. Object-oriented metrics that predict maintainability. *Journal of Systems and Software*, 23(2):111–122, 1993.

[547] Karl J. Lieberherr, Ignacio Silva-Lepe, and Cun Xiao. Adaptive object-oriented programming using graph-based customization. *Communications of the ACM*, 37(5):94–101, 1994.

[548] Eckhard Limpert, Werner A. Stahel, and Markus Abbt. Log-normal distributions across the sciences: Keys and clues. *BioScience*, 51(5):341–352, 2001.

[549] Joshua Lindsay, James Noble, and Ewan Tempero. Does size matter?: A preliminary investigation of the consequences of powerlaws in software. In *Proceedings of the Workshop on Emerging Trends in Software Metrics*, pages 16–23, 2010.

[550] Mikael Lindvall and Kristian Sandahl. How well do experienced software developers predict software change? *Journal of Systems and Software*, 43(1):19–27, 1998.

[551] Erik Linstead, Sushil Bajracharya, Trung Ngo, Paul Rigor, Cristina Lopes, and Pierre Baldi. Sourcerer: Mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery*, 18(2):300–336, 2009.

[552] Jacques-Louis Lions, Lennart Lübeck, Jean-Luc Fauquembergue, Gilles Kahn, Wolfgang Kubbat, Stefan Levedag, Leonardo Mazzini, Didier Merle, and Colin O'Halloran. Ariane 5: Flight 501 failure. Technical report, Ariane 501 Inquiry Board, 1996.

[553] Ernst Lippe and Norbert van Oosterom. Operation-based merging. *SIGSOFT Software Engineering Notes*, 17(5):78–87, 1992.

[554] Martin Lippert. Towards a proper integration of large refactorings in agile software development. In *Proceedings of the International Conference on Agile Processes in Software Engineering and Extreme Programming*, pages 113–122, 2004.

[555] Martin Lippert and Cristina Videira Lopes. A study on exception detection and handling using aspect-oriented programming. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 418–427, 2000.

[556] Martin Lippert and Stefan Roock. *Refactorings in Large Software Projects: How to Successfully Execute Complex Restructurings*. Wiley, 2006.

[557] Greg Little and Robert Miller. Keyword programming in Java. *Automated Software Engineering: An International Journal*, 16(1):37–71, 2009.

[558] David Littman, Jeannine Pinto, Stan Letovsky, and Elliot Soloway. Mental models and software maintenance. *Journal of Systems and Software*, 7(4):341–355, 1987.

[559] Jing Liu, Huaikou Miao, and Xiaolei Gao. A specification-based software construction framework for reuse. In *Proceedings of the International Conference on Formal Engineering Methods*, volume 2495 of *Lecture Notes in Computer Science*, pages 69–79, 2002.

[560] Qichao Liu, Marjan Mernik, and Barrett R. Bryant. MMDiff: A modeling tool for metamodel comparison. In *Proceedings of the ACM Southeast Regional Conference*, pages 118–123, 2012.

[561] David Lo, Siau-Cheng Khoo, and Chao Liu. Mining temporal rules for software maintenance. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(4):227–247, 2008.

[562] Alex Loh and Miryung Kim. LSdiff: A program differencing tool to identify systematic structural differences. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, volume 2, pages 263–266, 2010.

[563] John B. Lohse and Stuart H. Zweben. Experimental evaluation of software design principles: An investigation into the effect of module coupling on system modifiability. *Journal of Systems and Software*, 4(4):301–308, 1984.

[564] Cristina Videira Lopes and Sushil Krishna Bajracharya. An analysis of modularity in aspect oriented design. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 15–26, 2005.

[565] Roberto E. Lopez-Herrejon and Sven Apel. Measuring and characterizing crosscutting in aspect-based programs: Basic metrics and case studies. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, volume 4422 of *Lecture Notes in Computer Science*, pages 423–437, 2007.

[566] Panagiotis Louridas, Diomidis Spinellis, and Vasileios Vlachos. Power laws in software. *ACM Transactions on Software Engineering and Methodology*, 18(1):2/1–2/26, 2008.

[567] Chris Lüer and David S. Rosenblum. Wren: An environment for component-based development. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 207–217, 2001.

[568] Chung-Horng Lung, Xia Xu, Marzia Zaman, and Anand Srinivasan. Program restructuring through clustering techniques. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 75–84, 2004.

[569] Bin Luo, Richard C.Wilson, and Edwin R. Hancock. A spectral approach to learning structural variations in graphs. *Pattern Recognition*, 39(6):1188–1198, June 2006.

[570] Luqi. A graph model for software evolution. *IEEE Transactions on Software Engineering*, 16(8):917–927, 1990.

[571] Alessandro Maccari and Claudio Riva. Architectural evolution of legacy product families. In *Revised Papers of the International Workshop on Software Product-Family Engineering*, volume 2290 of *Lecture Notes in Computer Science*, pages 64–69, 2001.

[572] Alan MacCormack, John Rusnak, and Carliss Y. Baldwin. Exploring the structure of complex software designs: An empirical study of open source and proprietary code. *Management Science*, 52(7):1015–1030, 2006.

[573] Monica Maceli and Michael E. Atwood. From human crafters to human factors to human actors and back again: Bridging the design time–use time divide. In *Proceedings of the International Conference on End-User Development*, pages 76–91, 2011.

[574] Jeff Magee and Jeff Kramer. Dynamic structure in software architectures. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 3–13, 1996.

[575] N. Maiden. Analogy as a paradigm for specification reuse. *Software Engineering Journal*, 6(1):3–15, 1991.

[576] Neil Maiden and Alistair Sutcliffe. Exploiting reusable specifications through analogy. *Communications of the ACM*, 35(4):55–64, 1992.

[577] Jonathan I. Maletic and Naveen Valluri. Automatic software clustering via latent semantic analysis. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 251–254, 1999.

[578] David Mandelin, Lin Xu, Rastislav Bodík, and Doug Kimelman. Jungloid mining: Helping to navigate the API jungle. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 48–61, 2005.

[579] Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[580] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[581] Mika V. Mantyla and Casper Lassenius. What types of defects are really discovered in code reviews? *IEEE Transactions on Software Engineering*, 35(3):430–448, 2009.

[582] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. ADDiff: Semantic differencing for activity diagrams. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 179–189, 2011.

[583] Shahar Maoz, Jan Oliver Ringert, and Bernhard Rumpe. CDDiff: Semantic differencing for class diagrams. In *Proceedings of the European Conference on Object-Oriented Programming*, volume 6813 of *Lecture Notes in Computer Science*, pages 230–254, 2011.

[584] Onaiza Maqbool and Haroon Babri. Hierarchical clustering for software architecture recovery. *IEEE Transactions on Software Engineering*, 33(11):759–780, 2007.

[585] Michele Marchesi, Sandro Pinna, Nicola Serra, and Stefano Tuveri. Power laws in Smalltalk. In *Proceedings of the European Smalltalk User Group*, 2004. 17 pages.

[586] Andrian Marcus and Jonathan I. Maletic. Identification of high-level concept clones in source code. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 107–114, 2001.

[587] Andrian Marcus and Jonathan I. Maletic. Recovering documentation-to-source-code traceability links using latent semantic indexing. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 125–135, 2003.

[588] Andrian Marcus and Denys Poshyvanyk. The conceptual cohesion of classes. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 133–142, 2005.

[589] Andrian Marcus, Andrey Sergeyev, Václav Rajlich, and Jonathan I. Maletic. An information retrieval approach to concept location in source code. In *Proceedings of the Working Conference on Reverse Engineering*, pages 214–223, 2004.

[590] Marius Marin, Leon Moonen, and Arie van Deursen. An approach to aspect refactoring based on crosscutting concern types. In *Proceedings of the Workshop on the Modelling and Analysis of Concerns in Software*, 2005. 5 pages.

[591] Marius Marin, Leon Moonen, and Arie van Deursen. A classification of crosscutting concerns. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 673–676, 2005.

[592] Marius Marin, Leon Moonen, and Arie van Deursen. Documenting typical crosscutting concerns. In *Proceedings of the Working Conference on Reverse Engineering*, pages 31–40, 2007.

[593] Evan Martin and Tao Xie. Understanding software application interfaces via string analysis. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 901–904, 2006.

[594] Robert Martin. OO design quality metrics: An analysis of dependencies. In *Proceedings of the Workshop on Pragmatic and Theoretical Directions in Object-Oriented Software Metrics*, 1994. 8 pages.

[595] Girish Maskeri, Santonu Sarkar, and Kenneth Heafield. Mining business topics in source code using latent Dirichlet allocation. In *Proceedings of the India Software Engineering Conference*, pages 113–120, 2008.

[596] Tobias Mayer and Tracy Hall. A critical analysis of current OO design metrics. *Software Quality Journal*, 8(2):97–110, 1999.

[597] Steve McConnell. *Code Complete*. Microsoft Press, 2nd edition, 2004.

[598] S. Scott McCrickard and Colleen M. Kehoe. Visualizing search results using SQWID. In *Proceedings of the International Conference on the World Wide Web*, 1997.

[599] James J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software: Practice & Experience*, 12(1):23–34, January 1982.

[600] M. D. McIlroy. Mass-produced software components. In *Software Engineering: Report on a Conference by the NATO Science Committee*, pages 138–155, 1968.

[601] Andrew McVeigh, Jeff Kramer, and Jeff Magee. Evolve: Tool support for architecture evolution. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 1040–1042, 2011.

[602] Akhil Mehra, John Grundy, and John Hosking. A generic approach to supporting diagram differencing and merging for collaborative design. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 204–213, 2005.

[603] Thilo Mende, Rainer Koschke, and Felix Beckwermert. An evaluation of code similarity identification for the grow-and-prune model. *Journal of Software Maintenance and Evolution: Research and Practice*, 21(2):143–169, 2009.

[604] Na Meng, Miryung Kim, and Kathryn S. McKinley. Sydit: creating and applying a program transformation from an example. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 440–443, 2011.

[605] Na Meng, Miryung Kim, and Kathryn S. McKinley. Systematic editing: Generating program transformations from an example. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 329–342, 2011.

[606] T. Mens, G. Kniesel, and O. Runge. Transformation dependency analysis: A comparison of two approaches. In *Actes des journées Langages et Modèles à Objets*, pages 167–182, 2006.

[607] Tom Mens. Transformational software evolution by assertions. In *Proceedings of the Workshop on Formal Foundations of Software Evolution*, 2001. 8 pages.

[608] Tom Mens. A state-of-the-art survey on software merging. *IEEE Transactions on Software Engineering*, 28(5):449–462, 2002.

[609] Tom Mens and Michele Lanza. A graph-based metamodel for object-oriented software metrics. *Electronic Notes in Theoretical Computer Science*, 72(2):57–68, 2002.

[610] Tom Mens and Tom Tourwé. A survey of software refactoring. *IEEE Transactions on Software Engineering*, 30(2):126–139, 2004.

[611] D. Merkl, A. M. Tjoa, and G. Kappel. Learning the semantic similarity of reusable software components. In *Proceedings of the International Conference on Software Reuse*, pages 33–41, 1994.

[612] E. Merlo, M. Dagenais, P. Bachand, J. S. Sormani, S. Gradara, and G. Antoniol. Investigating large software system evolution: The Linux kernel. In *Proceedings of the IEEE International Computer Software and Applications Conference*, pages 421–426, 2002.

[613] Bertrand Meyer. On to components. *Computer*, 32(1):139–140, 1999.

[614] Bertrand Meyer. *Object-Oriented Software Construction*. Prentice Hall, 2nd edition, 2000.

[615] Timothy M. Meyers and David Binkley. An empirical study of slice-based cohesion and coupling metrics. *ACM Transactions on Software Engineering and Methodology*, 17(1):2/1–2/27, 2008.

[616] Mira Mezini and Klaus Ostermann. Integrating independent components with on-demand remodularization. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 52–67, 2002.

[617] Amir Michail. Data mining library reuse patterns using generalized association rules. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 167–176, 2000.

[618] Amir Michail. Code Web: Data mining library reuse patterns. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 827–828, 2001.

[619] Amir Michail and David Notkin. Assessing software libraries by browsing similar classes, functions and relationships. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 463–472, 1999.

[620] Tommi Mikkonen and Peeter Pruuden. Practical perspectives on software architectures, high-level design, and evolution. In *Proceedings of the ACM International Workshop on Principles of Software Evolution*, pages 122–125, 2001.

[621] Matthew B. Miles and Michael Huberman. *Qualitative Data Analysis: An Expanded Sourcebook*. Sage Publications, 2nd edition, 1994.

[622] A. Mili, R. Mili, and R. Mittermeir. Storing and retrieving software components: A refinement system. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 91–100, 1994.

[623] A. Mili, R. Mili, and R. T. Mittermeir. A survey of software reuse libraries. *Annals of Software Engineering*, 5(1):349–414, 1998.

[624] R. Mili, A. Mili, and R. Mittermeir. Storing and retrieving software components: A refinement based system. *IEEE Transactions on Software Engineering*, 23(7):445–460, July 1997.

[625] Webb Miller and Eugene W. Myers. A file comparison program. *Software: Practice & Experience*, 15(11):1025–1040, November 1985.

[626] A. V. Miranskyy, M. Davison, R. M. Reesor, and S. S. Murtaza. Using entropy measures for comparison of software traces. *Information Sciences*, 203:59–72, 25 October 2012.

[627] Vojislav B. Mišić. Cohesion is structural, coherence is functional: Different views, different measures. In *Proceedings of the IEEE International Symposium on Software Metrics*, pages 135–144, 2001.

[628] Brian Mitchell and Spiros Mancoridis. On the evaluation of the Bunch search-based software modularization algorithm. *Soft Computing*, 12(1):77–93, 2008.

[629] Brian S. Mitchell. A heuristic approach to solving the software clustering problem. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 285–288, 2003.

[630] Brian S. Mitchell and S. Mancoridis. On the automatic modularization of software systems using the Bunch tool. *IEEE Transactions on Software Engineering*, 32(3):193–208, 2006.

[631] Brian S. Mitchell and Spiros Mancoridis. Comparing the decompositions produced by software clustering algorithms using similarity measurements. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 744–753, 2001.

[632] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, March 1982.

[633] Michael Mitzenmarcher. A brief history of generative models for power law and lognormal distribution. *Internet Mathematics*, 1(2):226–251, 2004.

[634] Audris Mockus, Roy T. Fielding, and James D. Herbsleb. Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, 11(3):309–346, 2002.

[635] Parastoo Mohagheghi, Reidar Conradi, Ole M. Killi, and Henrik Schwarz. An empirical study of software reuse vs. defect-density and stability. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 282–292, 2004.

[636] D. L. Moise and K. Wong. Extracting and representing cross-language dependencies in diverse software systems. In *Proceedings of the Working Conference on Reverse Engineering*, pages 209–218, 2005.

[637] Leon Moonen. Generating robust parsers using island grammars. In *Proceedings of the Working Conference on Reverse Engineering*, pages 13–24, 2001.

[638] Leon Moonen. Lightweight impact analysis using island grammars. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 219–228, 2002.

[639] Brandon Morel and Perry Alexander. SPARTACAS: Automating component reuse and adaptation. *IEEE Transactions on Software Engineering*, 30(9):587–600, 2004.

[640] M. Moriconi and T. C. Winkler. Approximate reasoning about the semantic effects of program changes. *IEEE Transactions on Software Engineering*, 16(9):980–992, 1990.

[641] M. Morisio, M. Ezran, and C. Tully. Success and failure factors in software reuse. *IEEE Transactions on Software Engineering*, 28(4):340–357, 2002.

[642] Michał Moskal, Jakub Łopuszański, and Joseph R. Kiniry. E-matching for fun and profit. *Electronic Notes in Theoretical Computer Science*, 198(2):19–35, May 2008.

[643] Tridas Mukhopadhyay, Steven S. Vicinanza, and Michael J. Prietula. Examining the feasibility of a case-based reasoning model for software effort estimation. *MIS Quarterly*, 16(2):155–171, 1992.

[644] Frank Mulder and Andy Zaidman. Identifying cross-cutting concerns using software repository mining. In *Proceedings of the ACM International Workshop on Principles of Software Evolution/ERCIM Workshop on Software Evolution*, pages 23–32, 2010.

[645] H. A. Müller and K. Klashinsky. Rigi: A system for programming-in-the-large. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 80–86, 1988.

[646] Arnoldo José Müller Molina and Takeshi Shinohara. On approximate matching of programs for protecting libre software. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 21/1–21/15, 2006.

[647] Arnoldo José Müller Molina and Takeshi Shinohara. Fast approximate matching of programs for protecting libre/open source software by using spatial indexes. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 111–122, 2007.

[648] Arnoldo José Müller-Molina and Takeshi Shinohara. On the configuration of the similarity search data structure d-index for high dimensional objects. In *Proceedings of the International Conference on Computational Science and Its Applications*, volume 6018 of *Lecture Notes in Computer Science*, pages 443–457, 2010.

[649] Gail C. Murphy, Mik Kersten, and Leah Findlater. How are Java software developers using the Eclipse IDE? *IEEE Software*, 23(4):76–83, 2006.

[650] Gail C. Murphy and David Notkin. Lightweight lexical source model extraction. *ACM Transactions on Software Engineering and Methodology*, 5(3):262–292, 1996.

[651] Emerson Murphy-Hill. A model of refactoring tool use. In *Proceedings of the ACM Workshop on Refactoring Tools*, 2009. 4 pages.

[652] Emerson Murphy-Hill and Andrew P. Black. Breaking the barriers to successful refactoring: Observations and tools for Extract Method. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 421–430, 2008.

[653] Christopher R. Myers. Software systems as complex networks: Structure, function, and evolvability of software collaboration graphs. *Physical Review E*, 68:046116/1–046116/15, 2003.

[654] Eugene W. Myers. An $o(nd)$ difference algorithm and its variations. *Algorithmica*, 1(1–4):251–266, November 1986.

[655] Jerome L. Myers and Arnold D. Well. *Research Design & Statistical Analysis*. Routledge, 2nd edition, 2002.

[656] Volker Nannen. The paradox of overfitting. Master's thesis, Rijksuniversiteit Groningen, Groningen, The Netherlands, April 2003.

[657] Gonzalo Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, 33(1):31–88, March 2001.

[658] Cornelius Ncube, Patricia Oberndorf, and Anatol W. Kark. Opportunistic software systems development: Making systems from what's available. *IEEE Software*, 25(6):38–41, 2008.

[659] L. R. Neal. A system for example-based programming. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 63–68, 1989.

[660] Iulian Neamtiu, Jeffrey S. Foster, and Michael Hicks. Understanding source code evolution using abstract syntax tree matching. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 1–5, 2005.

[661] J. M. Neighbors. Draco: A method for engineering reusable software systems. In Ted J. Biggerstaff and Alan J. Perlis, editors, *Software Reusability*, volume 1: Concepts and Models, pages 295–319. Addison–Wesley, 1989.

[662] James M. Neighbors. An assessment of reuse technology after ten years. In *Proceedings of the International Conference on Software Reuse*, pages 6–13, 1994.

[663] M. E. J. Newman. Models of the small world: A review. *Journal of Statistical Physics*, 101:819–841, 2000.

[664] M. E. J. Newman. Assortative mixing in networks. *Physical Review Letters*, 89(20):208701/1–208701/4, 2002.

[665] M. E. J. Newman. Ego-centered networks and the ripple effect: Or, why all your friends are weird. *Social Networks*, 25:83–95, 2003.

[666] M. E. J. Newman. Power laws, Pareto distributions and Zipf's law. *Contemporary Physics*, 46(5):323–351, 2005.

[667] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[668] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):026118/1–026118/17, 2001.

[669] Mark Newman, Albert-László Barabási, and Duncan J. Watts. *The Structure and Dynamics of Networks*. Princeton University Press, 2006.

[670] Hoan Anh Nguyen, Tung Thanh Nguyen, Hung Viet Nguyen, and Tien N. Nguyen. iDiff: Interaction-based program differencing tool. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 572–575, 2011.

[671] Tien N. Nguyen. Managing software architectural evolution at multiple levels of abstraction. *Journal of Software*, 3(3):60–70, 2008.

[672] Tien N. Nguyen, Ethan V. Munson, John T. Boyland, and Cheng Thao. Architectural software configuration management in Molhado. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 296–305, 2004.

[673] Oscar Nierstrasz and Dennis Tsichritzis, editors. *Object-Oriented Software Composition*. Prentice Hall, 1995.

[674] The economic impacts of inadequate infrastructure for software testing. Planning report 02-3, National Institute of Standards & Technology, May 2002.

[675] Eugen C. Nistor, Justin R. Erenkrantz, Scott A. Hendrickson, and André van der Hoek. ArchEvol: Versioning architectural-implementation relationships. In *Proceedings of the International Workshop on Software Configuration Managment*, pages 99–111, 2005.

[676] Marius Nita and David Notkin. Using twinning to adapt programs to alternative APIs. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 205–214, 2010.

[677] Joost Noppen and Dalila Tamzalit. ETAK: Tailoring architectural evolution by (re-)using architectural knowledge. In *Proceedings of the International Workshop on Sharing and Reusing Architectural Knowledge*, pages 21–28, 2010.

[678] Mehrdad Nurolahzade, Seyed Mehdi Nasehi, Shahedul Huq Khandkar, and Shreya Rawal. The role of patch review in software evolution: An analysis of the mozilla firefox. In *Proceedings of the ACM International Workshop on Principles of Software Evolution/ERCIM Workshop on Software Evolution*, pages 9–18, 2009.

[679] Cosmin Oancea, Clare So, and Stephen M. Watt. Generalization in Maple. In *Proceedings of the Maple Conference*, pages 277–382, 2005.

[680] A. Offutt, M. Harrold, and P. Kolte. A software metric system for module coupling. *Journal of Systems and Software*, 20(3):295–308, 1993.

[681] Asako Ohno and Hajime Murao. Measuring source code similarity using reference vectors. In *Proceedings of the International Conference on Innovative Computing, Information, and Control*, volume 2, pages 92–95, 2006.

[682] Dirk Ohst, Michael Welle, and Udo Kelter. Differences between versions of UML diagrams. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 227–236, 2003.

[683] Mark O'Keeffe and Mel Ó Cinnéide. Search-based refactoring: An empirical study. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(5):345–364, 2008.

[684] Rocco Oliveto, Giuliano Antoniol, Andrian Marcus, and Jane Huffman Hayes. Software artefact traceability: The never-ending challenge. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 485–488, 2007.

[685] Santiago Ontañón and Enric Plaza. Similarity measures over refinement graphs. *Machine Learning*, 87(1):57–92, April 2012.

[686] William F. Opdyke. *Refactoring Object-Oriented Frameworks*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois, USA, 1992.

[687] Ciaran O'Reilly, Philip Morrow, and David Bustard. Lightweight prevention of architectural erosion. In *Proceedings of the ACM International Workshop on Principles of Software Evolution*, pages 59–64, 2003.

[688] Alessandro Orso and Bryan Kennedy. Selective capture and replay of program executions. In *Proceedings of the International Workshop on Dynamic Analysis*, pages 1–7, 2005.

[689] Harold Ossher and Peri Tarr. Hyper/J: Multi-dimensional separation of concerns for Java. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 729–730, 2001.

[690] Joel Ossher, Sushil Bajracharya, and Cristina Lopes. Automated dependency resolution for open source software. In *Proceedings of the International Working Conference on Mining Software Repositories*, pages 130–140, 2010.

[691] Eduardo Ostertag, James Hendler, Rubén Prieto-Díaz, and Christine Braun. Computing similarity in a reuse library system: An AI-based approach. *ACM Transactions on Software Engineering and Methodology*, 1(3):205–228, 1992.

[692] Thomas J. Ostrand and Elaine J. Weyuker. The distribution of faults in a large industrial software system. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 55–64, 2002.

[693] Tansel Özyer, Keivan Kianmehr, and Mehmet Tan, editors. *Recent Trends in Information Reuse and Integration*. Springer, 2012.

[694] Carlos Pacheco and Michael D. Ernst. Eclat: Automatic generation and classification of test inputs. In *Proceedings of the European Conference on Object-Oriented Programming*, pages 504–527, 2005.

[695] Carlos Pacheco, Shuvendu K. Lahiri, Michael D. Ernst, and Thomas Ball. Feedback-directed random test generation. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 75–84, 2007.

[696] Claus Pahl. An ontology for software component matching. *International Journal on Software Tools for Technology Transfer*, 9(2):169–178, March 2007.

[697] Nicolas Palix, Julia Lawall, and Gilles Muller. Tracking code patterns over multiple software versions with Herodotos. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 169–180, 2010.

[698] Ying Pan, Lei Wang, Lu Zhang, Bing Xie, and Fuqing Yang. Relevancy based semantic interoperation of reuse repositories. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 211–220, 2004.

[699] Juyong Park and Albert-László Barabási. Distribution of node characteristics in complex networks. *Proceedings of the National Academy of Sciences*, 104(46):17016–17020, 2007.

[700] D. L. Parnas. On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, 15(12):1053–1058, 1972.

[701] D. L. Parnas. On the design and development of program families. *IEEE Transactions on Software Engineering*, 2(1):1–9, 1976.

[702] D. L. Parnas and D. P. Siewiorek. Use of the concept of transparency in the design of hierarchically structured systems. *Communications of the ACM*, 18(7):401–408, July 1975.

[703] David L. Parnas. Designing software for ease of extension and contraction. *IEEE Transactions on Software Engineering*, 5(2):128–138, 1979.

[704] David Lorge Parnas. Software aging. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 279–287, 1994.

[705] Jeffrey Parsons and Chad Saunders. Cognitive heuristics in software engineering: Applying and extending anchoring and adjustment to artifact reuse. *IEEE Transactions on Software Engineering*, 30(12):873–888, 2004.

[706] Sukesh Patel, William Chu, and Rich Baxter. A measure for composite module cohesion. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 38–48, 1992.

[707] Santanu Paul. SCRUPLE: A reengineer's tool for source code search. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 329–346, 1992.

[708] Theodore P Pavlic and Kevin M Passino. Generalizing foraging theory for analysis and design. *International Journal of Robotics Research*, 30(5):505–523, 2011.

[709] Karl Pearson. I.mathematical contributions to the theory of evolution—VII. on the correlation of characters not quantitatively measurable. *Philosophical Transactions of the Royal Society of London, Series A*, 195:1–47 & 405, 1900.

[710] John Penix and Perry Alexander. Efficient specification-based component retrieval. *Automated Software Engineering: An International Journal*, 6(2):139–170, April 1999.

[711] Jeff Perkins. Automatically generating refactorings to support API evolution. In *Proceedings of the ACM SIGPLAN–SIGSOFT Workshop on Program Analysis and Software Tools Engineering*, pages 111–114, 2005.

[712] Paul J. Perrone, Venkata S. R. R. Chaganti, and Tom Schwenk. *J2EE Developer's Handbook*. Sams Publishing, 2003.

[713] Shari Lawrence Pfleeger. Design and analysis in software engineering—Part 1: The language of case studies and formal experiments. *SIGSOFT Software Engineering Notes*, 19(4):16–20, 1994.

[714] Shari Lawrence Pfleeger. Experimental design and analysis in software engineering. *Annals of Software Engineering*, 1(1):219–253, 1995.

[715] Nam H. Pham, Tung Thanh Nguyen, Hoan Anh Nguyen, Xinying Wang, Anh Tuan Nguyen, and Tien N. Nguyen. Detecting recurring and similar software vulnerabilities. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, volume 2, pages 227–230, 2010.

[716] Peter Pirolli and Wai-Tat Fu. SNIF-ACT: A model of information foraging on the World Wide Web. In *Proceedings of the International Conference on User Modeling*, pages 45–54, 2003.

[717] R. L. Plackett. Karl Pearson and the chi-squared test. *International Statistical Review*, 51(1):59–72, 1983.

[718] G. D. Plotkin. A further note on inductive generalization. *Machine Intelligence*, 6:101–124, 1971.

[719] Gordon D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.

[720] Andy Podgurski and Lori A. Clarke. A formal model of program dependences and its implications for software testing, debugging, and maintenance. *IEEE Transactions on Software Engineering*, 16(9):965–979, 1990.

[721] Andy Podgurski and Lynn Pierce. Behavior sampling: A technique for automated retrieval of reusable components. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 349–361, 1992.

[722] Denys Poshyvanyk, Yann-Gaël Guéhéneuc, Andrian Marcus, Giuliano Antoniol, and Václav Rajlich. Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. *IEEE Transactions on Software Engineering*, 33(6):420–432, 2007.

[723] Denys Poshyvanyk and Andrian Marcus. Combining formal concept analysis with information retrieval for concept location in source code. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 37–48, 2007.

[724] Denys Poshyvanyk, Andrian Marcus, and Yubo Dong. JIRiSS: An Eclipse plug-in for source code exploration. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 252–255, 2006.

[725] Denys Poshyvanyk, Andrian Marcus, Václav Rajlich, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. Combining probabilistic ranking and latent semantic indexing for feature identification. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 137–148, 2006.

[726] A. Postma, P. America, and J. G. Wijnstra. Component replacement in a long-living architecture: The 3RDBA approach. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture*, pages 89–98, 2004.

[727] Alex Potanin, James Noble, Marcus Frean, and Robert Biddle. Scale-free geometry in object-oriented programs. *Communications of the ACM*, 48(5):99–103, 2005.

[728] Jeffrey S. Poulin, Joseph M. Caruso, and Debera R. Hancock. The business case for software reuse. *IBM Systems Journal*, 32(4):567–594, 1993.

[729] Lutz Prechelt. An empirical comparison of seven programming languages. *Computer*, 33(10):23–29, 2000.

[730] Wolfgang Pree and Kai Koskimies. Rearchitecting legacy system: Concepts and case study. In *Proceedings of the IEEE/IFIP Working Conference on Software Architecture*, pages 51–64, 1999.

[731] Kyle Prete, Napol Rachatasumrit, Nikita Sudan, and Miryung Kim. Template-based reconstruction of complex refactorings. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 1–10, 2010.

[732] Rubén Prieto-Díaz. Status report: Software reusability. *IEEE Software*, 10(3):61–66, 1993.

[733] Michael J. Prietula, Steven S. Vicinanza, and Tridas Mukhopadhyay. Software-effort estimation with a case-based reasoner. *Journal of Experimental and Theoretical Artificial Intelligence*, 8(3–4):341–363, 1996.

[734] Adam Przybyłek. An empirical assessment of the impact of aspect-oriented programming on software modularity. In *Proceedings of the International Conference on Evaluating Novel Approaches to Software Engineering*, pages 139–148, 2010.

[735] Adam Przybyłek. Impact of aspect-oriented programming on software modularity. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 369–372, 2011.

[736] Adam Przybyłek. Where the truth lies: AOP and its impact on software modularity. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, volume 6603 of *Lecture Notes in Computer Science*, pages 447–461, 2011.

[737] Huaijun Qiu and Edwin R. Hancock. Graph matching and clustering using spectral partitions. *Pattern Recognition*, 39(1):22–34, January 2006.

[738] Kok-Seng Quah. Case study on re-architecting of established enterprise software product: Major challenges encountered and SDM prescriptions from lessons learned. Sm thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2005.

[739] Philip Quinn, Andy Cockburn, Indratmo, and Carl Gutwin. An investigation of dynamic landmarking functions. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 322–325, 2008.

[740] Dmitri A. Rachkovskij and Serge V. Slipchenko. Similarity-based retrieval with structure-sensitive sparse binary distributed representations. *Computational Intelligence*, 28(1):106–129, February 2012.

[741] Václav Rajlich. Intensions are a key to program comprehension. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 1–9, 2009.

[742] Václav Rajlich and Norman Wilde. The role of concepts in program comprehension. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 271–280, 2002.

[743] Alexander Ran. Software isn't built from Lego blocks. In *Proceedings of the ACM Symposium on Software Reusability*, pages 164–169, 1999.

[744] Ghulam Rasool and Patrick Mäder. Flexible design pattern detection based on feature types. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 243–252, 2011.

[745] Ghulam Rasool and Ilka Philippow. Software artifacts extraction for program comprehension. In Tarek Sobh and Khaled Elleithy, editors, *Innovations in Computing Sciences and Software Engineering*, pages 443–447. Springer Netherlands, 2010.

[746] Jacek Ratzinger, Thomas Sigmund, Peter Vorburger, and Harald Gall. Mining software evolution to predict refactoring. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 354–363, 2007.

[747] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67:026112/1–026112/7, 2003.

[748] Romain Raveaux, Jean-Christophe Burie, and Jean-Marc Ogier. A graph matching method and a graph matching distance based on subgraph assignments. *Pattern Recognition Letters*, 31(5):394–406, April 2010.

[749] Thiagarajan Ravichandran and Marcus A. Rothenberger. Software reuse strategies and component markets. *Communications of the ACM*, 46(8):109–114, 2003.

[750] Eric Raymond. The cathedral and the bazaar. *Knowledge and Technology Policy*, 12(3):23–49, 1999.

[751] David F. Redmiles. Reducing the variability of programmers' performance through explained examples. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 67–73, 1993.

[752] David Francis Redmiles. *From programming tasks to solutions: Bridging the gap through the explanation of examples.* PhD thesis, University of Colorado at Boulder, Boulder, Colorado, USA, 1992.

[753] Christian Robottom Reis and Renata Pontin de Mattos Fortes. An overview of the software engineering process and tools in the mozilla project. In *Proceedings of the Workshop on Open Source Software Development*, pages 155–175, 2002.

[754] Steven P. Reiss. Software tools and environments. *ACM Computing Surveys*, 28(1):281–284, 1996.

[755] Steven P. Reiss. The paradox of software visualization. In *Proceedings of the IEEE International Workshop on Visualizing Software for Understanding and Analysis*, pages 19/1–19/5, 2005.

[756] Steven P. Reiss. Semantics-based code search. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 243–253, 2009.

[757] Steven P. Reiss. Specifying what to search for. In *Proceedings of the Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, pages 41–44, 2009.

[758] Xiaoxia Ren, Fenil Shah, Frank Tip, Barbara G. Ryder, and Ophelia Chesley. Chianti: A tool for change impact analysis of Java programs. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 432–448, 2004.

[759] Meghan Revelle, Tiffany Broadbent, and David Coppit. Understanding concerns in software: Insights gained from two case studies. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 23–32, 2005.

[760] John C. Reynolds. Transformational systems and the algebraic structure of atomic formulas. *Machine Intelligence*, 5(1):135–151, 1970.

[761] Charles Rich and Richard C. Waters. The Programmer's Apprentice project: A research overview. Technical Report AIM-1004, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, November 1987.

[762] Kaspar Riesen, Xiaoyi Jiang, and Horst Bunke. Exact and inexact graph matching: Methodology and applications. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, chapter 7, pages 217–247. Springer, 2010.

[763] Peter C. Rigby and Daniel M. German. A preliminary examination of code review processes in open source projects. Technical Report DCS-305-IR, University of Victoria, January 2006.

[764] Peter C. Rigby, Daniel M. German, and Margaret-Anne Storey. Open source software peer review practices: A case study of the Apache server. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 541–550, 2008.

[765] Ran Rinat, Menachem Magidor, and Scott F. Smith. Correspondence polymorphism for object-oriented languages. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 167–186, 1999.

[766] Edwina L. Rissland. Examples and learning systems. In Oliver G. Selfridge, Edwina L. Rissland, and Michael A. Arbib, editors, *Adaptive Control of Ill-Defined Systems*. Plenum, 1984.

[767] Romain Robbes. On the evaluation of recommender systems with recorded interactions. In *Proceedings of the Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, pages 45–48, 2009.

[768] Romain Robbes and Michele Lanza. A change-based approach to software evolution. *Electronic Notes in Theoretical Computer Science*, 166:93–109, 2007.

[769] Romain Robbes and Michele Lanza. Characterizing and understanding development sessions. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 155–166, 2007.

[770] Romain Robbes and Michele Lanza. Example-based program transformation. In *Proceedings of the International Conference on Model-Driven Engineering of Languages and Systems*, volume 5301 of *Lecture Notes in Computer Science*, pages 174–188, 2008.

[771] Romain Robbes and Michele Lanza. SpyWare: A change-aware development toolset. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 847–850, 2008.

[772] Romain Robbes and Michele Lanza. Improving code completion with program history. *Automated Software Engineering: An International Journal*, 17(2):181–212, 2010.

[773] Romain Robbes, Michele Lanza, and Mircea Lungu. An approach to software evolution based on semantic change. In *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, pages 27–41, 2007.

[774] Romain Robbes and Mircea Lungu. A study of ripple effects in software ecosystems. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 904–907, 2011.

[775] Romain Robbes, Damien Pollet, and Michele Lanza. Logical coupling based on fine-grained change information. In *Proceedings of the Working Conference on Reverse Engineering*, pages 42–46, 2008.

[776] Martin Robillard and Robert DeLine. A field study of API learning obstacles. *Empirical Software Engineering*, 16(6):703–732, 2011.

[777] Martin P. Robillard. Topology analysis of software dependencies. *ACM Transactions on Software Engineering and Methodology*, 17(4):18/1–18/36, 2008.

[778] Martin P. Robillard, Wesley Coelho, and Gail C. Murphy. How effective developers investigate source code: An exploratory study. *IEEE Transactions on Software Engineering*, 30(12):889–903, 2004.

[779] Martin P. Robillard and Barthélémy Dagenais. Recommending change clusters to support software investigation: an empirical study. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(3):143–164, 2010.

[780] Martin P. Robillard and Gail C. Murphy. Representing concerns in source code. *ACM Transactions on Software Engineering and Methodology*, 16(1):3/1–3/38, 2007.

[781] Martin P. Robillard and Frédéric Weigand-Warr. ConcernMapper: Simple view-based separation of scattered concerns. In *Proceedings of the Eclipse Technology eXchange*, pages 65–69, 2005.

[782] Gregorio Robles and Jesus Gonzalez-Barahona. Contributor turnover in libre software projects. In *Proceedings of the IFIP International Conference on Open Source Systems*, volume 203 of *IFIP International Federation for Information Processing*, pages 273–286. Springer, 2006.

[783] Antonio Robles-Kelly and Edwin R. Hancock. String edit distance, random walks and graph matching. In *Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 104–112, 2002.

[784] Roshanak Roshandel, André van der Hoek, Marija Mikic-Rakic, and Nenad Medvidovic. Mae: A system model and environment for managing architectural evolution. *ACM Transactions on Software Engineering and Methodology*, 13(2):240–276, 2004.

[785] Mary Beth Rosson and John M. Carroll. Active programming strategies in reuse. In *Proceedings of the European Conference on Object-Oriented Programming*, volume 707 of *Lecture Notes in Computer Science*, pages 4–20, 1993.

[786] Mary Beth Rosson and John M. Carroll. The reuse of uses in Smalltalk programming. *ACM Transactions on Computer–Human Interaction*, 3(3):219–253, 1996.

[787] Marcus A. Rothenberger, Kevin J. Dooley, Uday R. Kulkarni, and Nader Nada. Strategies for software reuse: A principal component analysis of reuse practices. *IEEE Transactions on Software Engineering*, 29(9):825–837, 2003.

[788] Gregg Rothermel, Sebastian Elbaum, Alexey G. Malishevsky, Praveen Kallakuri, and Xuemei Qiu. On test suite composition and cost-effective regression testing. *ACM Transactions on Software Engineering and Methodology*, 13(3):277–331, 2004.

[789] Jean-François Rouet, Catherine Deleuze-Dordron, and André Bisseret. Documentation as part of design: Exploratory field studies. In *Proceedings of the IFIP TC13 International Conference on Human–Computer Interactaction*, pages 213–216, 1995.

[790] Chanchal K. Roy, James R. Cordy, and Rainer Koschke. Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, 74(7):470–495, 2009.

[791] Alejandro F. Rozenfeld, Reuven Cohen, Daniel ben Avraham, and Shlomo Havlin. Scale-free networks on lattices. *Physical Review Letters*, 89(21):218701/1–218701/4, 2002.

[792] David L. Sackett. Bias in analytic research. *Journal of Chronic Diseases*, 32(1–2):51–63, 1979.

[793] Chris Sadler and Barbara Ann Kitchenham. Evaluating software engineering methods and tools—Part 4: The influence of human factors. *SIGSOFT Software Engineering Notes*, 21(5):11–13, 1996.

[794] Nassima Sadou, Dalila Tamzalit, and Mourad Oussalah. A unified approach for software architecture evolution at different abstraction levels. In *Proceedings of the ACM International Workshop on Principles of Software Evolution*, pages 65–70, 2005.

[795] David Saff, Shay Artzi, Jeff H. Perkins, and Michael D. Ernst. Automatic test factoring for Java. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 114–123, 2005.

[796] Tobias Sager, Abraham Bernstein, Martin Pinzger, and Christoph Kiefer. Detecting similar Java classes using tree algorithms. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 65–71, 2006.

[797] Naiyana Sahavechaphan and Kajal T. Claypool. XSnippet: Mining for sample code. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 413–430, 2006.

[798] Robert J. Sandusky and Les Gasser. Negotiation and the coordination of information and activity in distributed software problem management. In *Proceedings of the ACM SIGGROUP International Conference on Supporting Group Work*, pages 187–196, 2005.

[799] Santonu Sarkar, Girish Maskeri Rama, and Avinash C. Kak. API-based and information-theoretic metrics for measuring the quality of software modularization. *IEEE Transactions on Software Engineering*, 33(1):14–32, 2007.

[800] Ilie Şavga and Michael Rudolf. Refactoring-based support for binary compatibility in evolving frameworks. In *Proceedings of the International Conference on Generative Programming and Component Engineering*, pages 175–184, 2007.

[801] Ilie Şavga, Michael Rudolf, Sebastian Götz, and Uwe Aßmann. Practical refactoring-based framework upgrade. In *Proceedings of the International Conference on Generative Programming and Component Engineering*, pages 171–180, 2008.

[802] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, August 2007.

[803] Thorsten Schafer, Michael Eichberg, Michael Haupt, and Mira Mezini. The SEXTANT software exploration tool. *IEEE Transactions on Software Engineering*, 32(9):753–768, 2006.

[804] Thorsten Schäfer, Jan Jonas, and Mira Mezini. Mining framework usage changes from instantiation code. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 471–480, 2008.

[805] Thorsten Schäfer, Jan Jonas, and Mira Mezini. Mining framework usage changes from instantiation code. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 471–480, 2008.

[806] D. F. Schenker and T. M. Khoshgoftaar. The application of fuzzy enhanced case-based reasoning for identifying fault-prone modules. In *Proceedings of the IEEE International High-Assurance Systems Engineering Symposium*, pages 90–97, 1998.

[807] Kjeld Schmidt and Carla Simone. Coordination mechanisms: towards a conceptual foundation of cscw systems design. *Computer Supported Cooperative Work*, 5(2-3):155–200, 1996.

[808] Curtis Schofield, Brendan Tansey, Zhenchang Xing, and Eleni Stroulia. Digging the development dust for refactorings. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 23–34, 2006.

[809] Lajos Schrettner, Peter Hegedus, Rudolf Ferenc, Lajos Jeno Fulop, and Tibor Bakota. Development of a methodology, software-suite and service for supporting software architecture reconstruction. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 190–193, 2010.

[810] Adrian Schröter, Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. If your bug database could talk... In *Proceedings of the International Symposium on Empirical Software Engineering*, volume 2, pages 18–20, 2006.

[811] Ken Schwaber and Mike Beedle. *Agile Software Development with Scrum.* Prentice Hall, 2001.

[812] Pascal Schweitzer. Isomorphism of (mis)labeled graphs. In *Proceedings of the European Symposium on Algorithms*, volume 6942 of *Lecture Notes in Computer Science*, pages 370–381, 2011.

[813] Richard W. Selby. Enabling reuse-based software development of large-scale systems. *IEEE Transactions on Software Engineering*, 31(6):495–510, 2005.

[814] Arun Sen. The role of opportunism in the software design reuse process. *IEEE Transactions on Software Engineering*, 23(7):418–436, 1997.

[815] Ross Shannon, Aaron Quigley, and Paddy Nixon. Deep Diffs: Visually exploring the history of a document. In *Proceedings of the International Conferece on Advanced Visual Interfaces*, pages 361–364, 2010.

[816] Mary Shaw, Robert DeLine, Daniel V. Klein, Theodore L. Ross, David M. Young, and Gregory Zelesnik. Abstractions for software architecture and tools to support them. *IEEE Transactions on Software Engineering*, 21(4):314–335, 1995.

[817] Martin Shepperd. Case-based reasoning and software engineering. In Aybüke Aurum, Ross Jeffery, Claes Wohlin, and Meliha Handzic, editors, *Managing Software Engineering Knowledge*, chapter 9. Springer, 2003.

[818] Martin Shepperd and Chris Schofield. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering*, 23(11):736–743, 1997.

[819] Jonathan Sillito, Gail C. Murphy, and Kris De Volder. Asking and answering questions during a programming change task. *IEEE Transactions on Software Engineering*, 34(4):434–451, 2008.

[820] Susan Elliott Sim and Margaret-Anne D. Storey. A structured demonstration of program comprehension tools. In *Proceedings of the Working Conference on Reverse Engineering*, pages 184–194, 2000.

[821] Herbert A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69(1):99–118, 1955.

[822] Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482, 12 December 1962.

[823] Herbert A. Simon. How big is a chunk?: By combining data from several experiments, a basic human memory unit can be identified and measured. *Science*, 183(4124):482–488, 8 February 1974.

[824] Renuka Sindhgatta. Using an information retrieval system to retrieve source code samples. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 905–908, 2006.

[825] Janice Singer. Practices of software maintenance. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 139–145, 1998.

[826] Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. An examination of software engineering work practices. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 21/1–21/15, 1997.

[827] Murali Sitaraman, Steven Atkinson, Gregory Kulczycki, Bruce W. Weide, Timothy J. Long, Paolo Bucci, Wayne Heym, Scott Pike, and Joseph E. Hollingsworth. Reasoning about software-component behavior. In *Proceedings of the International Conference on Software Reuse*, volume 1844 of *Lecture Notes in Computer Science*, pages 266–283, 2000.

[828] Maxym Sjachyn and Ljerka Beus-Dukic. Semantic component selection: SemaCS. In *Proceedings of the IEEE International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems*, pages 83–89, 2006.

[829] Odd Petter N. Slyngstad, Reidar Conradi, M. Ali Babar, Viktor Clerc, and Hans van Vliet. Risks and risk management in software architecture evolution: An industrial survey. In *Proceedings of the Asia–Pacific Software Engineering Conference*, pages 101–108, 2008.

[830] Harry M. Sneed. Encapsulation of legacy software: A technique for reusing legacy software components. *Annals of Software Engineering*, 9(1–4):293–313, 2000.

[831] Gregor Snelting. Reengineering of configurations based on mathematical concept analysis. *ACM Transactions on Software Engineering and Methodology*, 5(2):146–189, 1996.

[832] Alan Snyder. Encapsulation and inheritance in object-oriented programming languages. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 38–45, 1986.

[833] Christine Solnon. *AllDifferent*-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12–13):850–864, August 2010.

[834] Sébastien Sorlin and Christine Solnon. A global constraint for graph isomorphism problems. In *Proceedings of the International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 3011 of *Lecture Notes in Computer Science*, pages 287–302, 2004.

[835] Sébastien Sorlin and Christine Solnon. Reactive tabu search for measuring graph similarity. In *Proceedings of the IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition*, pages 172–182, 2005.

[836] Sébastien Sorlin, Christine Solnon, and Jean-Michel Jolion. A generic graph distance measure based on multivalent matchings. In Abraham Kandel, Horst Bunke, and Mark Last, editors, *Applied Graph Theory in Computer Vision and Pattern Recognition*, volume 52 of *Studies in Computational Intelligence*, pages 151–181. Springer, 2007.

[837] C. Spearman. The proof and measurement of association between two things. *American Journal of Psychology*, 15(1):72–101, January 1904.

[838] Giriprasad Sridhara, Emily Hill, Lori Pollock, and K. Vijay-Shanker. Identifying word relations in software: A comparative study of semantic similarity tools. In *Proceedings of the IEEE International Conference on Program Comprehenension*, pages 123–132, 2008.

[839] M. Srinivas and L. M. Patnaik. Genetic search: Analysis using fitness moments. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):120–133, 1996.

[840] Michael Stal. Good is not good enough: Evaluating and improving software architecture. In *Proceedings of the ACM SIGSOFT Conference on the Quality of Software Architectures/International Symposium on Architecing Critical Systems*, pages 73–74, 2011.

[841] Thomas A. Standish. An essay on software reuse. *IEEE Transactions on Software Engineering*, 10(5):494–497, 1984.

[842] Jennifer Stapleton. *DSDM: A Framework for Business-Centered Development*. Addison-Wesley Professional, 2nd edition, 2003.

[843] Jamie Starke, Chris Luce, and Jonathan Sillito. Working with search results. In *Proceedings of the Workshop on Search-Driven Development: Users, Infrastructure, Tools, and Evaluation*, pages 53–56, 2009.

[844] Stefan Stattelmann, Oliver Bringmann, and Wolfgang Rosenstiel. Dominator homomorphism based code matching for source-level simulation of embedded software. In *Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, pages 305–314, 2011.

[845] S. S. Stevens. On the theory of scales of measurement. *Science*, 103(2684):677–680, 7 June 1946.

[846] W. P. Stevens, G. J. Myers, and L. L. Constantine. Structured design. *IBM Systems Journal*, 13(2):115–139, 1974.

[847] Patrick Steyaert, Carine Lucas, Kim Mens, and Theo D'Hondt. Reuse contracts: Managing the evolution of reusable assets. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 268–285, 1996.

[848] M.-A. D. Storey, K. Wong, and H. A. Müller. How do program understanding tools affect how programmers understand programs? *Science of Computer Programming*, 36(2–3):183–207, 2000.

[849] Margaret-Anne Storey. Theories, methods and tools in program comprehension: Past, present and future. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 181–191, 2005.

[850] Heiner Stuckenschmidt. A semantic similarity measure for ontology-based information. In *Proceedings of the International Conference on Flexible Query Answering Systems*, volume 5822 of *Lecture Notes in Computer Science*, pages 406–417, 2009.

[851] Michael P. H. Stumpf, Carsten Wiuf, and Robert M. May. Subnets of scale-free networks are not scale-free: Sampling properties of networks. *Proceedings of the National Academy of Sciences*, 102(12):4221–4224, 2005.

[852] Ramanath Subramanyam and M. S. Krishnan. Empirical analysis of CK metrics for object-oriented design complexity: Implications for software defects. *IEEE Transactions on Software Engineering*, 29(4):297–310, 2003.

[853] Giancarlo Succi, Luigi Benedicenti, and Tullio Vernazza. Analysis of the effects of software reuse on customer satisfaction in an RPG environment. *IEEE Transactions on Software Engineering*, 27(5):473–479, 2001.

[854] Kevin J. Sullivan, William G. Griswold, Yuanfang Cai, and Ben Hallen. The structure and value of modularity in software design. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 99–108, 2001.

[855] Mikael Svahnberg. *Supporting Software Architecture Evolution.* PhD thesis, Blekinge Institute of Technology, Blekinge, Sweden, October 2003.

[856] Nikita Synytskyy, James R. Cordy, and Thomas R. Dean. Robust multilingual parsing using island grammars. In *Proceedings of the IBM Centre for Advanced Studies Conference on Collaborative Research*, pages 266–278, 2003.

[857] Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming.* ACM Press, 2002.

[858] Ladan Tahvildari and Ajit Singh. Categorization of object-oriented software metrics. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, volume 1, pages 235–239, 2000.

[859] Dalila Tamzalit and Tom Mens. Guiding architectural restructuring through architectural styles. In *Proceedings of the IEEE Conference on Engineering Computer-Based Systems*, pages 69–78, 2010.

[860] Dalila Tamzalit, Nassima Sadou, and Mourad Oussalah. Evolution problem within component-based software architecture. In *Proceedings of the International Conference on Software Engineering and Knowledge Engineering*, pages 296–301, 2006.

[861] Dalila Tamzalit, Nassima Sadou, and Mourad Oussalah. Connectors conveying software architecture evolution. In *Proceedings of the IEEE International Computer Software and Applications Conference*, volume 1, pages 391–396, 2007.

[862] Mei-Huei Tang, Ming-Hung Kao, and Mei-Hwa Chen. An empirical study on object-oriented metrics. In *Proceedings of the IEEE International Symposium on Software Metrics*, pages 242–249, 1999.

[863] Wesley Tansey and Eli Tilevich. Annotation refactoring: Inferring upgrade transformations for legacy applications. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 295–312, 2008.

[864] Peri L. Tarr, Harold Ossher, William H. Harrison, and Stanley M. Sutton, Jr. *N* degrees of separation: Multi-dimensional separation of concerns. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 107–119, 1999.

[865] Carsten Tautz and Klaus-Dieter Althoff. Using case-based reasoning for reusing software knowledge. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 1266 of *Lecture Notes in Computer Science*, pages 156–165, 1997.

[866] John R. Taylor. *An Introduction to Error Analysis: The Study of Uncertainties in Physical Measurements*. University Science Books, 2nd edition, 1996.

[867] Ewan Tempero, Craig Anslow, Jens Dietrich, Ted Han, Jing Li, Markus Lumpe, Hayden Melton, and James Noble. The Qualitas Corpus: A curated collection of Java code for empirical studies. In *Proceedings of the Asia–Pacific Software Engineering Conference*, pages 336–345, 2010.

[868] Suresh Thummalapenta and Tao Xie. PARSEWeb: A programmer assistant for reusing open source code on the web. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 204–213, 2007.

[869] Frank Tip, Adam Kiezun, and Dirk Bäumer. Refactoring for generalization using type constraints. In *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 13–26, 2003.

[870] Michael Toomim, Andrew Begel, and Susan L. Graham. Managing duplicated code with linked editing. In *Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 173–180, 2004.

[871] Richard Torkar and Stefan Mankefors. A survey on testing and reuse. In *Proceedings of the IEEE International Conference on Software Sciience, Technology, and Engineering*, pages 164–173, 2003.

[872] Cristen Torrey, Elizabeth F. Churchill, and David W. McDonald. Learning how: The search for craft knowledge on the internet. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 1371–1380, 2009.

[873] Will Tracz. Where does reuse start? *SIGSOFT Software Engineering Notes*, 15(2):42–46, 1990.

[874] Will Tracz. Confessions of a used-program salesman: Lessons learned. In *Proceedings of the ACM Symposium on Software Reusability*, pages 11–13, 1995.

[875] John B. Tran, Michael W. Godfrey, Eric H. S. Lee, and Richard C. Holt. Architectural repair of open source software. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 48–59, 2000.

[876] Shiu Lun Tsang, Siobhán Clarke, and Elisa Baniassad. An evaluation of aspect-oriented programming for Java-based real-time systems development. In *Proceedings of the International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 291–300, 2004.

[877] Nikolaos Tsantalis, Alexander Chatzigeorgiou, George Stephanides, and Spyros T. Halkidis.

[878] Koji Tsuda and Hiroto Saigo. Graph classification. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, chapter 11, pages 337–363. Springer, 2010.

[879] Qiang Tu and Michael W. Godfrey. An integrated approach for studying architectural evolution. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 127–136, 2002.

[880] Jaroslav Tulach. *Practical API Design: Confessions of a Java Framework Architect*. Apress, 2008.

[881] Esko Ukkonen. Algorithms for approximate string matching. *Information and Control*, 64(1–3):100–118, January–March 1985.

[882] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM*, 23(1):31–42, January 1976.

[883] Shinji Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, September 1988.

[884] S. Valverde, R. Ferrer Cancho, and R. V. Solé. Scale-free networks from optimal design. *Europhysics Letters*, 60(4):512–517, 2002.

[885] Sergi Valverde and Ricard V. Solé. Logarithmic growth dynamics in software networks. *Europhysics Letters*, 72(5):858–864, 2005.

[886] André van der Hoek, Marija Mikic-Rakic, Roshanak Roshandel, and Nenad Medvidovic. Taming architectural evolution. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 1–10, 2001.

[887] J. van Gurp and J. Bosch. Design, implementation and evolution of object oriented frameworks: Concepts and guidelines. *Software: Practice & Experience*, 31(3):277–300, March 2001.

[888] Jilles van Gurp and Jan Bosch. Design erosion: Problems and causes. *Journal of Systems and Software*, 61(2):105–119, 2002.

[889] Rob van Ommering. Software reuse in product populations. *IEEE Transactions on Software Engineering*, 31(7):537–550, 2005.

[890] Rajesh Vasa, Markus Lumpe, Philip Branch, and Oscar Nierstrasz. Comparative analysis of evolving software systems using the Gini coefficient. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 179–188, 2009.

[891] Mathieu Verbaere, Ran Ettinger, and Oege de Moor. JunGL: A scripting language for refactoring. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 172–181, 2006.

[892] Eelco Visser. Program transformation with Stratego/XT: Rules, strategies, tools, and systems in Stratego/XT 0.9. In Christian Lengauer, Don Batory, Charles Consel, and Martin Odersky, editors, *Domain-Specific Program Generation*, volume 3016, pages 216–238. Springer Berlin Heidelberg, 2004.

[893] Lars Vogel. Rich Client Platform. http://wiki.eclipse.org/index.php/Rich_Client_Platform, January 2012.

[894] Anneliese von Mayrhauser, Richard Mraz, Jeff Walls, and Pete Ocken. Domain based testing: Increasing test case reuse. In *Proceedings of the IEEE International Conference on Computer Design*, pages 484–491, 1994.

[895] Anneliese von Mayrhauser and A. Marie Vans. Program comprehension during software maintenance and evolution. *Computer*, 28(8):44–55, 1995.

[896] Ulrich Wagner. Combinatorically restricted higher order anti-unification: An application to programming by analogy. Master's thesis, Technische Universität Berlin, Berlin, Germany, April 2002.

[897] Vera Wahler, Dietmar Seipel, Jürgen Wolff von Gudenberg, and Gregor Fischer. Clone detection in source code by frequent itemset techniques. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 128–135, 2004.

[898] Haixun Wang and Charu C. Aggarwal. A survey of algorithms for keyword search on graph data. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, chapter 8, pages 249–273. Springer, 2010.

[899] Shuanglin Wang, Stephen R. Schach, and Gillian Z. Heller. A case study in repeated maintenance. *Journal of Software Maintenance and Evolution: Research and Practice*, 13(2):127–141, 2001.

[900] Yingxu Wang, Graham King, Mohamed Fayad, Dilip Patel, Ian Court, Geoff Staples, and Margaret Ross. On built-in test reuse in object-oriented framework design. *ACM Computing Surveys*, 32(1es):7–12, 2000.

[901] Michael Wasmund. Reuse facts and myths. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, page 273, 1994.

[902] D. J. Watts and S. H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(6684):409–410, 1998.

[903] Duncan J. Watts. Networks, dynamics, and the small-world phenomenon. *American Journal of Sociology*, 105(2):493–527, 1999.

[904] Duncan J. Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.

[905] Duncan J. Watts. Connections: A twenty-first century science. *Nature*, 445:489, 2007.

[906] Duncan J. Watts, Roby Muhamad, Daniel C. Medina, and Peter S. Dodds. Multiscale, resurgent epidemics in a hierarchical metapopulation model. *Proceedings of the National Academy of Sciences*, 102(32):11157–11162, 2005.

[907] Markus Weber, Christoph Langenhan, Thomas Roth-Berghofer, Marcus Liwicki, Andreas Dengel, and Frank Petzold. a.SCatch: Semantic structure for architectural floor plan retrieval. In *Proceedings of the International Conference on Case-Based Reasoning*, pages 510–524, 2010.

[908] Markus Weber, Christoph Langenhan, Thomas Roth-Berghofer, Marcus Liwicki, Andreas Dengel, and Frank Petzold. Fast subgraph isomorphism detection for graph-based retrieval. In *Proceedings of the International Conference on Case-Based Reasoning*, pages 319–333, 2011.

[909] Markus Weber, Marcus Liwicki, and Andreas Dengel. Indexing with well-founded total order for faster subgraph isomorphism detection. In *Proceedings of the IAPR-TC-15 International Workshop on Graph-Based Representations in Pattern Recognition*, pages 185–194, 2011.

[910] Rosina Weber, Michael Waller, June Verner, and William Evanco. Predicting software development project outcomes. In *Proceedings of the International Conference on Case-Based Reasoning*, volume 2689 of *Lecture Notes in Computer Science*, pages 595–609, 2003.

[911] Bruce W. Weide, William F. Ogden, and Stuart H. Zweben. Reusable software components. *Advances in Computers*, 33(2):1–65, 1991.

[912] Karsten Weihe. A software engineering perspective on algorithmics. *ACM Computing Surveys*, 33(1):89–134, 2001.

[913] Gerald M. Weisenberg. *The Psychology of Computer Programming*. Dorset House, 1971.

[914] Mark Weiser. Program slicing. *IEEE Transactions on Software Engineering*, 10(4):352–357, 1984.

[915] Peter Weißgerber and Stephan Diehl. Are refactorings less error-prone than other changes? In *Proceedings of the International Workshop on Mining Software Repositories*, pages 112–118, 2006.

[916] Peter Weissgerber and Stephan Diehl. Identifying refactorings from source-code changes. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 231–240, 2006.

[917] Peter Weißgerber, Stephan Diehl, and Carsten Görg. An interactive visualization of refactorings retrieved from software archives. In *Companion to the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 176–177, 2005.

[918] Peter Weißgerber, Stephan Diehl, and Carsten Görg. Mining refactorings in ArgoUML. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 175–176, 2006.

[919] Zhihua Wen and Vassilios Tzerpos. Evaluating similarity measures for software decompositions. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 368–377, 2004.

[920] Michel Wermelinger and Yijun Yu. Analyzing the evolution of Eclipse plugins. In *Proceedings of the International Working Conference on Mining Software Repositories*, pages 133–136, 2008.

[921] Robert West and Jure Leskovec. Human wayfinding in information networks. In *Proceedings of the International Conference on the World Wide Web*, pages 619–628, 2012.

[922] Richard Wettel and Michele Lanza. Visual exploration of large-scale system evolution. In *Proceedings of the Working Conference on Reverse Engineering*, pages 219–228, 2008.

[923] Elaine J. Weyuker. Testing component-based software: A cautionary tale. *IEEE Software*, 15(5):54–59, 1998.

[924] Richard Wheeldon and Steve Counsell. Power law distributions in class relationships. In *Proceedings of the IEEE International Workshop on Source Code Analysis and Manipulation*, pages 45–54, 2001.

[925] Ryen W. White and Steven M. Drucker. Investigating behavioral variability in web search. In *Proceedings of the International Conference on the World Wide Web*, pages 21–30, 2007.

[926] Andreas Wierda, Eric Dortmans, and Lou Lou Somers. Using version information in architectural clustering: A case study. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 214–228, 2006.

[927] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, December 1945.

[928] F. G. Wilkie and B. A. Kitchenham. Coupling measures and change ripples in C++ application software. *Journal of Systems and Software*, 52(2–3):157–164, 2000.

[929] Dennis M. Wilkinson. Strong regularities in online peer production. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 302–309, 2008.

[930] Byron J. Williams and Jeffrey C. Carver. Characterizing software architecture changes: A systematic review. *Information and Software Technology*, 52(1):31–51, 2010.

[931] Chadd C. Williams and Jaime W. Spacco. Branching and merging in the repository. In *Proceedings of the International Working Conference on Mining Software Repositories*, pages 19–22, 2008.

[932] Richard C. Wilson and Ping Zhu. A study of graph spectra for comparing graphs and trees. *Pattern Recognition*, 41(9):2833–2841, September 2008.

[933] Niklaus Wirth. Program development by stepwise refinement. *Communications of the ACM*, 14(4):221–227, 1971.

[934] Michael J. Wise. YAP3: Improved detection of similarities in computer program and other texts. In *Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education*, pages 130–134, 1996.

[935] S. N. Woodfield, H. E. Dunsmore, and V. Y. Shen. The effect of modularization and comments on program comprehension. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 215–223, 1981.

[936] Jingwei Wu, Ahmed E. Hassan, and Richard C. Holt. Comparison of clustering algorithms in the context of software evolution. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 525–535, 2005.

[937] Sun Wu and Udi Manber. Agrep: A fast approximate pattern-matching tool. In *Proceedings of the USENIX Winter Technical Conference*, pages 153–162, 1992.

[938] Wei Wu, Yann-Gaël Guéhéneuc, Giuliano Antoniol, and Miryung Kim. AURA: A hybrid approach to identify framework evolution. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, volume 1, pages 325–334, 2010.

[939] Bai Xiao, Edwin R. Hancock, and Richard C. Wilson. A generative model for graph matching and embedding. *Computer Vision and Image Understanding*, 113(7):777–789, July 2009.

[940] Chenchen Xiao and Vassilios Tzerpos. Software clustering based on dynamic dependencies. In *Proceedings of the European Conference on Software Maintenance and Reengineering*, pages 124–133, 2005.

[941] Tao Xie and Jian Pei. MAPO: Mining API usages from open source repositories. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 54–57, 2006.

[942] Zhenchang Xing and Eleni Stroulia. Design mentoring based on design evolution analysis. In *Proceedings of the Eclipse Technology eXchange*, pages 83–87, 2004.

[943] Zhenchang Xing and Eleni Stroulia. Understanding class evolution in object-oriented software. In *Proceedings of the IEEE International Workshop on Program Comprehenension*, pages 34–43, 2004.

[944] Zhenchang Xing and Eleni Stroulia. Analyzing the evolutionary history of the logical design of object-oriented software. *IEEE Transactions on Software Engineering*, 31(10):850–868, 2005.

[945] Zhenchang Xing and Eleni Stroulia. UMLDiff: An algorithm for object-oriented design differencing. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 54–65, 2005.

[946] Zhenchang Xing and Eleni Stroulia. Refactoring detection based on UMLDiff change-facts queries. In *Proceedings of the Working Conference on Reverse Engineering*, pages 263–274, 2006.

[947] Zhenchang Xing and Eleni Stroulia. Refactoring practice: How it is and how it should be supported—an eclipse case study. In *Proceedings of the IEEE International Conference on Software Maintenance*, pages 458–468, 2006.

[948] Zhenchang Xing and Eleni Stroulia. Differencing logical UML models. *Automated Software Engineering: An International Journal*, 14(2):215–259, 2007.

[949] Zhengchang Xing and Eleni Stroulia. API-evolution support with Diff-CatchUp. *IEEE Transactions on Software Engineering*, 33(12):818–836, 2007.

[950] Lugang Xu. *Cascaded refactoring for framework development and evolution*. PhD thesis, Concordia University, Montréal, Canada, 2006.

[951] Yinxing Xue, Zhenchang Xing, and Stan Jarzabek. Understanding feature evolution in a family of product variants. In *Proceedings of the Working Conference on Reverse Engineering*, pages 109–118, 2010.

[952] Tetsuo Yamamoto, Makoto Matsushita, Toshihiro Kamiya, and Katsuro Inoue. Measuring similarity of large software systems based on source code correspondence. In *Proceedings of the International Conference on Product-Focused Software Process Improvement*, pages 530–544, 2005.

[953] Xifeng Yan and Jiawei Han. Graph indexing. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, chapter 5, pages 161–180. Springer, 2010.

[954] Wuu Yang. Identifying syntactic differences between two programs. *Software: Practice & Experience*, 21(7):739–755, July 1991.

[955] Stephen S. Yau and James S. Collofello. Some stability measures for software maintenance. *IEEE Transactions on Software Engineering*, 6(6):545–552, 1980.

[956] Stephen S. Yau and James S. Collofello. Design stability measures for software maintenance. *IEEE Transactions on Software Engineering*, 11(9):849–856, 1985.

[957] Yunwen Ye and Gerhard Fischer. Supporting reuse by delivering task-relevant and personalized information. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 513–523, 2002.

[958] Yunwen Ye, Gerhard Fischer, and Brent Reeves. Integrating active information delivery and reuse repository systems. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 60–68, 2000.

[959] Daniel M. Yellin and Robert E. Strom. Protocol specifications and component adaptors. *ACM Transactions on Programming Languages and Systems*, 19(2):292–333, 1997.

[960] Annie T. T. Ying, Gail C. Murphy, Raymond Ng, and Mark C. Chu-Carroll. Predicting source code changes by mining change history. *IEEE Transactions on Software Engineering*, 30(9):574–586, 2004.

[961] Reishi Yokomori, Harvey Siy, Norihiro Yoshida, Masami Noro, and Katsuro Inoue. Measuring the effects of aspect-oriented refactoring on component relationships: Two case studies. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 215–226, 2011.

[962] Lin Yu and Daniel J. Rosenkrantz. A linear-time scheme for version reconstruction. *ACM Transactions on Programming Languages and Systems*, 16(3):775–797, May 1994.

[963] Yijun Yu, Thein Than Tun, and Bashar Nuseibeh. Specifying and detecting meaningful changes in programs. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, pages 273–282, 2011.

[964] Yuhanis Yusof and Omer F. Rana. Template mining in source-code digital libraries. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 122–126, 2004.

[965] Andy Zaidman and Serge Demeyer. Automatic identification of key classes in a software system using webmining techniques. *Journal of Software Maintenance and Evolution: Research and Practice*, 20(6):387–417, 2008.

[966] Amy Moormann Zaremski and Jeannette M. Wing. Signature matching: A key to reuse. In *Proceedings of the ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 182–190, 1993.

[967] Amy Moormann Zaremski and Jeannette M. Wing. Signature matching: A tool for using software libraries. *ACM Transactions on Software Engineering and Methodology*, 4(2):146–170, April 1995.

[968] Amy Moormann Zaremski and Jeannette M. Wing. Specification matching of software components. *ACM Transactions on Software Engineering and Methodology*, 6(4):333–369, October 1997.

[969] Mikhail Zaslavskiy, Francis Bach, and Jean-Philippe Vert. A path following algorithm for the graph matching problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2227–2242, December 2009.

[970] Andreas Zeller. The future of programming environments: Integration, synergy, and assistance. In *Proceedings of the Future of Software Engineering*, pages 316–325, 2007.

[971] Andreas Zeller. Where do bugs come from? *Electronic Notes in Theoretical Computer Science*, 174(4):55–59, 2007.

[972] Charles Zhang and Hans-Arno Jacobsen. Quantifying aspects in middleware platforms. In *Proceedings of the International Conference on Aspect-Oriented Software Deveopment*, pages 130–139, 2003.

[973] Hongyu Zhang. Discovering power laws in computer programs. *Information Processing and Management*, 45(4):477–483, 2009.

[974] Jian Zhang. A survey on streaming algorithms for massive graphs. In Charu C. Aggarwal and Haixun Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, chapter 13, pages 393–420. Springer, 2010.

[975] Xiangyu Zhang and Rajiv Gupta. Matching execution histories of program versions. In *Proceedings of the European Software Engineering Conference/ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 197–206, 2005.

[976] Zhi Zhang, Haoyang Che, Pengfei Shi, Yong Sun, and Jun Gu. Multi-labeled graph matching: An algorithm model for schema matching. In *Proceedings of the Asian Computing Science Conference*, volume 3818 of *Lecture Notes in Computer Science*, pages 90–103, 2005.

[977] Yuming Zhou, Jiangtao Lu, Hongmin Lu, and Baowen Xu. A comparative study of graph theory-based class cohesion measures. *SIGSOFT Software Engineering Notes*, 29(2):13/1–13/6, 2004.

[978] Linhong Zhu, Wee Keong Ng, and James Cheng. Structure and attribute index for approximate graph matching in large graphs. *Information Systems*, 36(6):958–972, September 2011.

[979] Linhong Zhu, Wee Keong Ng, and Shuguo Han. Classifying graphs using theoretical metrics: a study of feasibility. In *Proceedings of the International Conference on Database Systems for Advanced Applications*, volume 6637 of *Lecture Notes in Computer Science*, pages 53–64, 2011.

[980] Yuanyuan Zhu, Lu Qin, Jeffrey Xu Yu, Yiping Ke, and Xuemin Lin. High efficiency and quality: Large graphs matching. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 1755–1764, 2011.

[981] Thomas Zimmermann. Knowledge collaboration by mining software repositories. In *Proceedings of the International Workshop on Supporting Knowledge Collaboration in Software Development*, pages 64–65, 2006.

[982] Thomas Zimmermann and Peter Weißgerber. Preprocessing CVS data for fine-grained analysis. In *Proceedings of the International Workshop on Mining Software Repositories*, pages 2–6, 2004.

[983] Thomas Zimmermann, Peter Weißgerber, Stephan Diehl, and Andreas Zeller. Mining version histories to guide software changes. In *Proceedings of the ACM/IEEE International Conference on Software Engineering*, pages 563–572, 2004.

[984] Lijie Zou and Michael W. Godfrey. Detecting merging and splitting using origin analysis. In *Proceedings of the Working Conference on Reverse Engineering*, pages 146–154, 2003.