Bibliography

[1] Lehman MM. Laws of software evolution revisited. In: 5th European workshop on software process technology; 1996.

[2] Tsui F, Gharaat A, Duggins S, Jung E. Measuring levels of abstraction in software development. In: International conference on software engineering and knowledge engineering; 2011. p. 466–9.

[3] Jones C. Software quality in 2012: a survey of the state of the art, Available online at: http://sqgne.org/presentations/2012-13/Jones-Sep-2012.pdf.

[4] Samarthyam G, Suryanarayana G, Sharma T, Gupta S. MIDAS: a design quality assessment method for industrial software. In: International conference on software engineering, software engineering in practice track; 2013.

[5] Samarthyam G, Sharma T, Suryanarayana G. Towards a principle-based classification of structural design smells. J Object Technol 2013.

[6] Brooks Jr FP. The mythical man-month. Anniversary ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 1995.

[7] Fowler M. Refactoring: improving the design of existing code. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 1999.

[8] Demeyer S, Ducasse S, Nierstrasz O. Object-oriented reengineering patterns. Morgan Kaufmann; 2002.

[9] Beyer D, Lewerentz C. Crocopat: efficient pattern analysis in object-oriented programs. In: Proceedings of the 11th IEEE international workshop on program comprehension, IWPC'03, Washington, DC, USA; 2003. p. 294.

[10] Martin RC. Design principles and practices. Object Mentor; 2000. Available online at: http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf.

[11] Hunt A, Thomas D. The pragmatic programmer. Addison Wesley; 2000.

[12] Hoare T. The emperor's old clothes. In: The ACM turing award lecture; 1980.

[13] Opdyke WF. Refactoring object-oriented frameworks [Ph.D. thesis]. University of Illinois at Urbana-Champaign; 1992.

[14] Liskov B. Data abstraction and hierarchy. SIGPLAN Not January 1987;23:17–34.

[15] Martin RC. Single responsibility principle. Object Mentor; 2000. Available from: http://www.objectmentor.com/resources/articles/srp.pdf.

[16] Roy CK, Cordy JR. A survey on software clone detection research. Technical Report 2007-541. Canada: Queen's University; 2007.

[17] Evans E. Domain-driven design: tackling complexity in the heart of software. Addison Wesley; 2003.

[18] Page-Jones M. Fundamentals of object-oriented design in UML. Addison-Wesley Professional; 1999.

[19] Buschmann F, Henney K, Schmidt DC. Pattern-oriented software architecture: a pattern language for distributed computing, vol. 4. John Wiley & Sons; 2007.

[20] Trifu A. Towards automated restructuring of object oriented systems [Ph.D. thesis]. Universität Karlsruhe (TH), Fakultét fér Informatik; 2008.

[21] Lanza M, Marinescu R. Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems. Springer; 2006.

[22] Schader M, Korthaus A. The unified modeling language: technical aspects and applications. Physica; 1998.

[23] Shalloway A, Trott JR. Design patterns explained: a new perspective on object-oriented design. Addison-Wesley; 2004.
[24] Meyer B. Object-oriented software construction. 2nd ed. Prentice Hall PTR; 2000.
[25] Ratzinger J, Fischer M, Gall H. Improving evolvability through refactoring. In: Proceedings of the 2005 international workshop on mining software repositories, MSR '05, New York, NY, USA; 2005. p. 1–5.
[26] Van Emden E, Moonen L. Java quality assurance by detecting code smells. In: Proceedings of the ninth working conference on reverse engineering (WCRE'02), Washington, DC, USA; 2002. p. 97.
[27] Garzas J, Piattini M. A catalog of design rules for OO micro-architecture. In: Garzas J, Piattini M, editors. Object-oriented design knowledge: principles, heuristics and best practices. Hershey: IGI Global; 2007. p. 307–47.
[28] Structural Investigation of Software Systems (SISSy) Tool. Available at: http://sissy.fzi.de/SISSy/CMS/index.html.
[29] Trifu A, Marinescu R. Diagnosing design problems in object oriented systems. In: Proceedings of the 12th working conference on reverse engineering, Washington, DC, USA; 2005. p. 155–64.
[30] Trifu A. Automated strategy based restructuring of object oriented code. In: Proceedings of the 7th German workshop on software-reengineering (WSR); 2005.
[31] Dudziak T, Wloka J. Tool-supported discovery and refactoring of structural weaknesses in code [Master's thesis]. Faculty of Computer Science, Technische Universität Berlin; February 2002.
[32] Linnaeus C. Systema naturae. B. De Graaf; 1735.
[33] Bass L, Clements P, Kazman R. Software architecture in practice. 3rd ed. Addison-Wesley; 2012.
[34] Hitz M, Montazeri B. Measuring product attributes of object-oriented systems. In: Proceedings of the 5th European software engineering conference. London, UK: Springer-Verlag; 1995. p. 124–36.
[35] Lippert M, Roock S. Refactoring in large software projects: performing complex restructurings successfully. John Wiley and Sons; 2006.
[36] Liskov BH, Wing JM. A behavioral notion of subtyping. ACM Trans Program Lang Syst November 1994;16:1811–41.
[37] Wake WC. Refactoring workbook. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 2003.
[38] Becker P. Common design mistakes – part II. C/C++ Users J February, 2000.
[39] Binder RV. Testing object-oriented systems: models, patterns, and tools. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.; 1999.
[40] Java Collections API Design FAQ. Available at: http://docs.oracle.com/javase/8/docs/technotes/guides/collections/designfaq.html.
[41] Parnas DL. On the criteria to be used in decomposing systems into modules. Commun ACM December 1972;15:1053–8.
[42] Meyer B. Touch of class: learning to program well with objects and contracts. Springer; 2009.
[43] Findbugs Tool for Java. Available at: http://findbugs.sourceforge.net/.
[44] Cunningham W. The WyCash portfolio management system. In: Addendum to the proceedings on object-oriented programming systems, languages, and applications. OOPSLA '92; 1992.
[45] Highsmith J. Zen and the art of software quality. In: Agile2009 conference; 2009.

[46] Kelion L. Why banks are likely to face more software glitches in 2013. Available at: http://www.bbc.co.uk/news/technology-21280943.
[47] Booch G, Maksimchuk R, Engle M, Young B, Conallen J, Houston K. Object-oriented analysis and design with applications. 3rd ed. Addison-Wesley Professional; 2007.
[48] Tsui F, Gharaat A, Duggins S, Jung E. Measuring levels of abstraction in software development. SEKE 2011:466–9.
[49] Bloch J. How to design a good API and why it matters. In: Companion to the 21st ACM SIGPLAN symposium on object-oriented programming systems, languages, and applications (OOPSLA '06); 2006.
[50] Kerievsky J. Refactoring to patterns. Pearson Higher Education; 2004.
[51] Llano MT, Pooley R. UML specification and correction of object-oriented anti-patterns. In: Proceedings of the 2009 fourth international conference on software engineering advances, ICSEA '09, Washington, DC, USA; 2009. p. 39–44.
[52] Rumbaugh J, Blaha M, Premerlani W, Eddy F, Lorensen W. Object-oriented modeling and design. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.; 1991.
[53] Succi G, Marchesi M. Extreme programming examined. Pearson Education; 2001.
[54] Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co.; 1995.
[55] Sonargraph-quality: A tool for assessing and monitoring technical quality. Available at: https://www.hello2morrow.com/products/sonargraph/quality.
[56] Sefika M, Sane A, Campbell RH. Monitoring compliance of a software system with its high-level design models. In: Proceedings of the 18th international conference on software engineering, ICSE '96, Washington, DC, USA; 1996. p. 387–96.
[57] Choinzon M, Ueda Y. Detecting defects in object oriented designs using design metrics. In: Proceeding of the 2006 conference on knowledge-based software engineering, Amsterdam, The Netherlands; 2006. p. 61–72.
[58] Moha N. DECOR: Détection et Correction des Défauts Dans les Systémes Orientés Objet [Ph.D. thesis]. Université de Montréal et Université de Lille; August 2008.
[59] Semmle Code Tool. Available at: http://semmle.com/semmlecode/; 2012.
[60] Johnson P, Rees C. Reusability through fine-grain inheritance. Softw Pract Exp December 1992;22:1049–68.
[61] Bloch J. Effective Java. 2nd ed. Addison-Wesley; 2008.
[62] Khomh F, Di Penta M, Guehéneuc Y-G, Antoniol G. An exploratory study of the impact of anti-patterns on software changeability. Technical report EPM-RT-2009-02. Iecole Polytechnique de Montreal; April 2009.
[63] Simon F, Seng O, Mohaupt T. Code quality management: Technische Qualität industrieller Softwaresysteme Transparent und Vergleichbar Gemacht. dpunkt-Verlag; 2006.
[64] Ford N. The productive programmer. O'Reilly; 2008.
[65] Budd T. An introduction to object-oriented programming. 3rd ed. Addison Wesley; 2001.
[66] Meyer B. The many faces of inheritance: a taxonomy of taxonomy. Computer May 1996;29(5):105–8.
[67] Ingalls DHH. Design principles behind smalltalk. BYTE Mag August 1981.
[68] Biehl M. APL – a language for automated anti-pattern analysis of OO-software. CS 846: Source Transformation Systems, Project Report. University of Waterloo; 2006.
[69] Arévalo G. High-level views in object-oriented systems using formal concept analysis [Ph.D. thesis]. The University of Bern; 2004.

[70] Miller BK. Object-oriented architecture measures. In: Proceedings of the thirty-second annual Hawaii international conference on system sciences, vol. 8. HICSS '99; 1999. p. 8069.
[71] Page-Jones M. The practical guide to structured systems design. 2nd ed. Prentice Hall; 1988.
[72] SDMetrics. A UML design quality metrics tool; 2012. Available at: http://www.sdmetrics.com.
[73] Marquardt K. Dependency structures – architectural diagnoses and therapies. In: Proc. of EuroPLoP; 2001.
[74] Stal M. Software architecture refactoring. Tutorial. In: The international conference on object oriented programming, systems, languages and applications (OOPSLA); 2007.
[75] Hannemann J, Kiczales G. Design pattern implementation in Java and AspectJ. In: Proceedings of the 17th ACM SIGPLAN conference on object-oriented programming, systems, languages, and applications, OOPSLA '02, New York, NY, USA; 2002. p. 161–73.
[76] InFusion Hydrogen. Design flaw detection tool; 2012. Available at: http://www.intooitus.com/products/infusion.
[77] Structural Analysis for Java Tool (Stan4J). Available at: http://stan4j.com/.
[78] Gil JY, Maman I. Micro patterns in Java code. SIGPLAN Not October 2005;40:97–116.
[79] Martin RC. Agile software development, principles, patterns, and practices. Addison-Wesley; 2003.
[80] Bouwers E, Visser J, Lilienthal C, Deursen A. A cognitive model for software architecture complexity. In: Proceedings of the IEEE 18th international conference on program comprehension (ICPC '10), IEEE computer society, Washington, DC, USA; 2010. p. 152–5.
[81] Buschmann F, Henney K, Schmidt. Pattern oriented software architecture: on patterns and pattern languages, vol. 5. Wiley; 2007.
[82] Yu L, Schach SR, Chen K. Maintaining Linux: The Role of current. In: Proceedings of the Fourth International Symposium on Empirical Software Engineering, Noosa Heads, Queensland, Australia; November 2005. p. 44–52.
[83] Riel AJ. Object-Oriented Design Heuristics. Addison-Wesley Professional, 1996.
[84] Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley Professional; 2002.