Interactive scalable condensation of reverse engineered UML class diagrams for software comprehension

Osman, M.H.B.

Appendix A

Case Study Candidates

*In this Appendix, we list the candidate projects for our case studies (Table A.1). Item mark with * are the selected case studies.*

Table A.1: *List of Case Study Candidates*

No	Project	Source
1	**ArgoUML***	**http://sourceforge.net/projects/argouml/**
2	Bookszenbooks	http://code.google.com/p/bookszenbooks/
3	cmpt371t1	http://code.google.com/p/cmpt371t1/
4	Concurrentadt	http://code.google.com/p/concurrentadt/
5	CrimsonPortal	http://code.google.com/p/crimsonportal/
6	DBForms	http://jdbforms.sourceforge.net/UsersGuide/html /index.html
7	DocDoku	http://code.google.com/p/docdoku/
8	driving-bc	http://code.google.com/p/driving-bc/
9	DRMJ-Webshop	http://code.google.com/p/drmj-webshop/
10	EmpScheduler2008	http://code.google.com/p/empscheduler2008/
11	Epydoc	http://epydoc.sourceforge.net/api/epydoc-module.html
12	europa-ps	http://code.google.com/p/europa-pso/
13	Fipa-english-auction	http://code.google.com/p/fipa-english-auction-interaction-protocol/
14	Fuge	http://fuge.sourceforge.net/dev/index.php
15	gelsanalyzer	http://code.google.com/p/gelsanalyzer/downloads /detail?name=MVC%20Class%20Diagram %20Iteration2.png&can=2&q=

16	GNetWatch	http://gnetwatch.sourceforge.net/doc.html
17	Google-voice-java	http://code.google.com/p/google-voice-java/
18	Gotogate	http://gotogate.googlecode.com/svn/trunk/
19	gwavmerger	http://gwavmerger.sourceforge.net/gwm-design/design.html
20	**gwt-portlets***	**http://code.google.com/p/gwtportlets/**
21	gwtuml	http://code.google.com/p/gwtuml/
22	httpdbase4j	http://httpdbase4j.berlios.de/
23	Jalli	http://jalli.berlios.de/
24	Java Kohonen Neural Network Library	http://jknnl.sourceforge.net/
25	Javaassessment	http://code.google.com/p/javaassessment /downloads/list
26	**JavaClient***	**http://java-player.sourceforge.net/**
27	**JGAP***	**https://sourceforge.net/project/screenshots.php?group_id=11618**
28	jjack	http://jjack.berlios.de/
29	**jpmc***	**http://jpmc.sourceforge.net/diagrams.html**
30	Jvending	http://jvending.sourceforge.net/
31	Krank	http://code.google.com/p/krank/
32	Mandragora Project	http://mandragora.sourceforge.net/referenceguide/how-to-extend.html
33	**Mars Simulation***	**https://sourceforge.net/apps/mediawiki/mars-sim/index.php?title=UML_Diagrams**
34	**Maze-Solver***	**http://code.google.com/p/maze-solver/wiki/MazeModelDoc**
35	monopolj	http://code.google.com/p/monopolj/
36	MyDas	http://code.google.com/p/mydas/
37	Netbeams	http://code.google.com/p/netbeams/wiki/ DataSensorPlatform
38	network-keeper	http://code.google.com/p/network-keeper/wiki/Diagrams
39	**Neuroph***	**http://neuroph.sourceforge.net/**
40	nmt-cs326-g5	https://code.google.com/p/nmt-cs326-g5/
41	Novosoft Metadata Framework	http://nsuml.sourceforge.net/index.html
42	Nymp	http://code.google.com/p/nymph/
43	ObjectLabKit	http://objectlabkit.sourceforge.net/apidocs /net/objectlab/kit/datecalc/jdk/CalendarPeriodCountCalculator.html

44	ObjectListView	http://objectlistview.sourceforge.net/cs/index.html
45	OpenMeeting	http://code.google.com/p/openmeetings/
46	pacpounder	https://code.google.com/p/pacpounder/
47	Pendulim	http://code.google.com/p/pendulim/
48	Portions	http://portions.sourceforge.net/en/guide.html
49	Primitive Collections	http://pcj.sourceforge.net/docs/guide/pcj-guide.html
50	Qt OIC Container 3.5	http://qtioccontainer.sourceforge.net/uml.html
51	RandyLoops	http://randyloops.sourceforge.net/
52	rpcstruts	https://code.google.com/p/rcpstruts/
53	StarUML	http://staruml.sourceforge.net/docs/api-doc/Modeling%20Elements/UML%20Model %20Elements /Foundation/Core/package-summary.html
54	tiOPF	http://tiopf.sourceforge.net/Doc/Concepts /4_BuildingAnAbstractBOMWithTheComposite.shtml
55	Topology data scripting	https://sourceforge.net/apps/mediawiki/free-cad/index.php?title=Topological_data_scripting
56	**wro4J***	**http://code.google.com/p/wro4j/**
57	**Xuml-compiler***	**http://code.google.com/p/xuml-compiler/**

List of Figures

List of Tables

Bibliography

[1] IEEE standard for software maintenance. *IEEE Std 1219-1998*, pages i–, 1998. (cited on page 15).

[2] BerliOS. `http://www.berlios.de/`, 2012. (cited on page 33).

[3] Google search engine. `http://www.google.com`, 2012. (cited on page 33).

[4] GoogleCode. `http://code.google.com/`, 2012. (cited on page 33).

[5] Limeservice. `https://www.limeservice.com/en/`, 2012. (cited on pages 95 and 98).

[6] Object-oriented dataset. `http://www.liacs.nl/~hosman/DataSets.rar`, 2012. (cited on pages 6 and 123).

[7] SourceForge. `http://sourceforge.net/`, 2012. (cited on page 33).

[8] GitHub. `https://github.com/`, 2014. (cited on page 33).

[9] MagicDraw. `http://www.nomagic.com/`, 2014. (cited on pages 34, 121 and 139).

[10] PlantUML. `http://plantuml.sourceforge.net/`, 2014. (cited on page 165).

[11] RapidMiner. `http://rapidminer.com/`, 2014. (cited on pages 140 and 165).

[12] A. Abraham. Rule-based expert systems. *Handbook of measuring system design*, 2005. (cited on page 22).

[13] D. Akehurst, G. Howells, and K.M. Maier. Implementing associations: UML 2.0 to Java 5. *Software and Systems Modeling*, 6(1):3–35, March 2007. (cited on pages 49 and 64).

[14] J. Al Dallal and L.C. Briand. A precise method-method interaction-based cohesion metric for object-oriented classes. *ACM Trans. Softw. Eng. Methodol.*, 21(2):8:1–8:34, March 2012. (cited on page 28).

[15] S.A. Alvarez. An exact analytical relation among recall, precision, and classification accuracy in information retrieval. *Boston College, Boston, Technical Report BCCS-02-01*, pages 1–22, 2002. (cited on page 27).

[16] J.R. Anderson, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986. (cited on page 22).

[17] O. Andriyevska, N. Dragan, B. Simoes, and J.I. Maletic. Evaluating UML class diagram layout based on architectural importance. In *Proceedings of the 3rd IEEE International Workshop on Visualizing Software for Understanding and Analysis*, VISSOFT '05, pages 1–6, Washington, DC, USA, 2005. IEEE Computer Society. (cited on page 43).

[18] R.A. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999. (cited on page 139).

[19] T. Ball. The concept of dynamic analysis. *SIGSOFT Softw. Eng. Notes*, 24(6):216–234, October 1999. (cited on pages 17 and 198).

[20] V. R. Basili. Evolving and packaging reading technologies. *Journal of Systems and Software*, 38(1):3 – 12, 1997. (cited on page 11).

[21] S. Bassil and R.K. Keller. Software visualization tools: survey and analysis. In *Proceedings of 9th International Workshop on Program Comprehension (IWPC 2001)*, pages 7–17, 2001. (cited on page 71).

[22] B. Bellay and H. Gall. A comparison of four reverse engineering tools. In *Proceedings of the Fourth Working Conference on Reverse Engineering*, WCRE '97, pages 2–, Washington, DC, USA, 1997. IEEE Computer Society. (cited on pages 49, 70 and 118).

[23] K.H. Bennett and V.T. Rajlich. Software maintenance and evolution: A roadmap. In *Proceedings of the Conference on The Future of Software Engineering*, ICSE '00, pages 73–87, New York, NY, USA, 2000. ACM. (cited on page 11).

[24] J.M. Bieman, A.A. Andrews, and H.J. Yang. Understanding change-proneness in OO software through visualization. In *Proceedings of the 11th IEEE International Workshop on Program Comprehension*, pages 44–53, 2003. (cited on pages 158 and 196).

[25] T.J. Biggerstaff, B.G. Mitbander, and D.E. Webster. Program understanding and the concept assignment problem. *Communication ACM*, 37(5):72–82, May 1994. (cited on page 11).

[26] D. Billsus and M.J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54. Morgan Kaufmann Publishers Inc., 1998. (cited on page 27).

[27] A.B. Binkley and S.R. Schach. Validation of the coupling dependency metric as a predictor of run-time failures and maintenance measures. In *Proceedings of the 20th international conference on Software engineering*, pages 452–455. IEEE Computer Society, 1998. (cited on page 93).

[28] R.C. Bjork. ATM simulation system. `http://www.math-cs.gordon.edu/courses/cs211/ATMExample/`, 2002. (cited on pages 52, 96 and 175).

[29] M.G. Bocco, M. Piattini, and C. Calero. A survey of metrics for UML class diagrams. *Journal of Object Technology*, 4(9):59–92, 2005. (cited on pages 121 and 196).

[30] A. Boklund, S. MankeFors-Christiernin, C. Johansson, and H. Lindell. A comparative study of forward and reverse engineering in UML tools. IADIS International Conference Applied Computing 2007, 2007. (cited on page 49).

[31] G. Booch. *Object Solutions: Managing the Object-oriented Project*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1995. (cited on page 18).

[32] G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language user guide*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1999. (cited on page 31).

[33] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. (cited on page 26).

[34] L. Briand, P. Devanbu, and W. Melo. An investigation into coupling measures for C++. In *Proceedings of the 19th International Conference on Software Engineering*, ICSE '97, pages 412–421, New York, NY, USA, 1997. ACM. (cited on pages 122 and 162).

[35] L.C. Briand, J. Wüst, J.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. (cited on page 93).

[36] L.C. Briand, J. Wüst, S.V. Ikonomovski, and H. Lounis. Investigating quality factors in object-oriented designs: An industrial case study. In *Proceedings of the 21st International Conference on Software Engineering*, ICSE '99, pages 345–354, New York, NY, USA, 1999. ACM. (cited on pages 93, 121 and 196).

[37] R. Brooks. Towards a theory of the comprehension of computer programs. *International journal of man-machine studies*, 18(6):543–554, 1983. (cited on page 13).

[38] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning*. The MIT Press, 1st edition, 2010. (cited on page 23).

[39] N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. (cited on page 28).

[40] S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, 20(6):476–493, June 1994. (cited on pages 122 and 162).

[41] E.J. Chikofsky and J.H. Cross II. Reverse engineering and design recovery: A taxonomy. *IEEE Softw.*, 7(1):13–17, January 1990. (cited on pages 16, 17, 92, 118 and 205).

[42] E. Chung, C. Jensen, K. Yatani, V. Kuechler, and K.N. Truong. Sketching and drawing in the design of open source software. In *Proceedings of the 2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, VLHCC '10, pages 195–202, Washington, DC, USA, 2010. IEEE Computer Society. (cited on page 33).

[43] C.L. Corritore and S. Wiedenbeck. An exploratory study of program comprehension strategies of procedural and object-oriented programmers. *International Journal of Human-Computer Studies*, 54(1):1–23, 2001. (cited on pages 12 and 13).

[44] A. Craig, A. Dinardo, and R. Gillespie. Pacman game. `http://code.google.com/p/tb-pacman/`, 2009. (cited on pages 72, 97 and 176).

[45] P. Dayan, M. Sahani, and G. Deback. Unsupervised learning. In *The MIT Encyclopedia of the Cognitive Sciences*. The MIT Press, 1999. (cited on page 23).

[46] S. Demeyer. Research methods in computer science. In *Procedings of the International Conference of Software Maintenance (ICSM 2011)*, page 600, 2011. (cited on page 5).

[47] B. Dobing and J. Parsons. Current practices in the use of UML. In *ER (Workshops)*, volume 3770 of *Lecture Notes in Computer Science*, pages 2–11. Springer, 2005. (cited on pages 32 and 33).

[48] B. Dobing and J. Parsons. How UML is used. *Commun. ACM*, 49(5):109–113, May 2006. (cited on pages 32 and 33).

[49] S. Ducasse and D. Pollet. Software architecture reconstruction: A process-oriented taxonomy. *IEEE Trans. Software Eng.*, 35(4):573–591, 2009. (cited on page 13).

[50] B.P. Eddy, J.A Robinson, N.A Kraft, and J.C. Carver. Evaluating source code summarization techniques: Replication and expansion. In *Proceedings of the IEEE 21st International Conference on Program Comprehension (ICPC)*, pages 13–22, May 2013. (cited on pages 137 and 197).

[51] A. Egyed. Semantic abstraction rules for class diagrams. In *Automated Software Engineering, 2000. Proceedings ASE 2000. The Fifteenth IEEE International Conference on*, pages 301–304. IEEE, 2000. (cited on page 93).

[52] A. Egyed. Automated abstraction of class diagrams. *ACM Trans. Softw. Eng. Methodol.*, 11(4):449–491, October 2002. (cited on pages 93 and 131).

[53] K. El Emam, W. Melo, and J.C. Machado. The prediction of faulty classes using object-oriented design metrics. *Journal of Systems and Software*, 56(1):63–75, 2001. (cited on page 93).

[54] J. Erickson and K. Siau. Theoretical and practical complexity of modeling methods. *Commun. ACM*, 50(8):46–51, August 2007. (cited on page 33).

[55] H.-E. Eriksson, M. Penker, and D. Fado. *UML 2 Toolkit*. John Wiley & Sons, Inc., New York, NY, USA, 2003. (cited on pages 73 and 96).

[56] J.-R. Falleri, M. Huchard, and C. Nebut. A generic approach for class model normalization. In *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*, ASE '08, pages 431–434, Washington, DC, USA, 2008. IEEE Computer Society. (cited on page 93).

[57] J.-M. Favre. GSEE: a generic software exploration environment. In *Proceedings of the 9th International Workshop on Program Comprehension (IWPC 2001)*, pages 233–244, 2001. (cited on page 13).

[58] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006. (cited on pages 25, 27 and 28).

[59] A.M. Fernández-Sáez, M.R.V. Chaudron, M. Genero, and I. Ramos. Are forward designed or reverse-engineered UML diagrams more helpful for code maintenance?: A controlled experiment. In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, EASE '13, pages 60–71, New York, NY, USA, 2013. ACM. (cited on pages 4, 92, 118 and 156).

[60] A.M. Fernández-Sáez, M. Genero, and M.R.V. Chaudron. Does the level of detail of UML models affect the maintainability of source code? In *Proceedings of the 2011th International Conference on Models in Software Engineering*, MODELS'11, pages 134–148, Berlin, Heidelberg, 2012. Springer-Verlag. (cited on page 36).

[61] A. Field. *Discovering Statistics Using SPSS*. SAGE Publications, 2005. (cited on page 26).

[62] G. Forman and M. Scholz. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010. (cited on pages 28 and 126).

[63] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3 edition, 2003. (cited on pages 37, 38 and 52).

[64] A.K. Gahalut and P. Khandnor. Reverse engineering : An essence for software re-engineering and program analysis. *International Journal of Engineering and Technology*, 2:2296–2303, 2010. (cited on page 49).

[65] M. Grechanik, K.S. McKinley, and D.E. Perry. Recovering and using Use-case-diagram-to-source-code traceability links. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering*, ESEC-FSE '07, pages 95–104, New York, NY, USA, 2007. ACM. (cited on page 37).

[66] T.J. Grose, G.C. Doney, and S.A. Brodsky. *Mastering XMI: Java Programming with XMI, XML and UML*. John Wiley & Sons, Inc., New York, NY, USA, 2001. (cited on page 21).

[67] M. Grossman, J.E. Aronson, and R.V. McCarthy. Does UML make the grade? Insights from the software development community. *Inf. Softw. Technol.*, 47(6):383–397, April 2005. (cited on page 33).

[68] Object Management Group. Unified Modeling Language (UML), Superstructure, Version 2.4.1, August 2011. (cited on pages 18, 39, 52 and 205).

[69] Object Management Group. ISO/IEC 19509:2014 Information technology – Object management group XML metadata interchange (XMI). `http://www.iso.org/iso/catalogue_detail.htm?csnumber=61845`, 2014. (cited on page 21).

[70] Y.-G. Guéhéneuc. A systematic study of UML class diagram constituents for their abstract and precise recovery. In *Proceedings of the 11th Asia-Pacific Software Engineering Conference*, APSEC '04, pages 265–274, Washington, DC, USA, 2004. IEEE Computer Society. (cited on page 92).

[71] Y.-G Guéhéneuc. Taupe: towards understanding program comprehension. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, page 1. IBM Corp., 2006. (cited on page 21).

[72] Y.-G. Guéhéneuc and H. Albin-Amiot. Recovering binary class relationships: Putting icing on the UML cake. In *Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '04, pages 301–314, New York, NY, USA, 2004. ACM. (cited on pages 48, 54, 57 and 59).

[73] T. Gyímothy, R. Ferenc, and I. Siket. Empirical validation of object-oriented metrics on open source software for fault prediction. *IEEE Transactions on Software Engineering*, 31(10):897–910, 2005. (cited on pages 93, 121 and 196).

[74] S. Haiduc, J. Aponte, and A Marcus. Supporting program comprehension with source code summarization. In *Proceedings of the ACM/IEEE 32nd International Conference on Software Engineering (ICSE)*, volume 2, pages 223–226, May 2010. (cited on pages 136 and 197).

[75] S. Haiduc, J. Aponte, L. Moreno, and A Marcus. On the use of automated text summarization techniques for summarizing source code. In *Proceedings of the 17th Working Conference on Reverse Engineering (WCRE)*, pages 35–44, Oct 2010. (cited on pages 137 and 197).

[76] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: An update, 2009. (cited on pages 26, 127, 146, 147 and 165).

[77] M. Hammad, M.L. Collard, and J.I. Maletic. Measuring class importance in the context of design evolution. In *Proceedings of the IEEE 18th International Conference on Program Comprehension (ICPC 2010)*, pages 148–151, 2010. (cited on pages 120, 133, 157 and 196).

[78] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011. (cited on page 154).

[79] J.A. Hanley and B.J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, April 1982. (cited on pages 28 and 147).

[80] T. Hettel, M. Lawley, and K. Raymond. Model synchronisation: Definitions for round-trip engineering. In *Theory and Practice of Model Transformations*, pages 31–45. Springer, 2008. (cited on page 48).

[81] J. Hutchinson, J. Whittle, M. Rouncefield, and S. Kristoffersen. Empirical assessment of MDE in industry. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 471–480, New York, NY, USA, 2011. ACM. (cited on page 33).

[82] M. Ichii, T. Myojin, Y. Nakagawa, M. Chikahisa, and H. Ogawa. A Rule-based automated approach for extracting models from source code. In *Proceedings of the 19th Working Conference on Reverse Engineering*, WCRE '12, pages 308–317, Washington, DC, USA, 2012. IEEE Computer Society. (cited on page 43).

[83] I. Jacobson. *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley Professional, 1 edition, June 1992. (cited on page 18).

[84] G. Jalloul. *UML by Example*. Cambridge University Press, New York, NY, USA, 2004. (cited on page 19).

[85] S. Jarzabek. *Effective software maintenance and evolution: A reuse-based approach*. CRC Press, 2007. (cited on page 17).

[86] R. Kazman and S.J. Carriere. View extraction and view fusion in architectural understanding. In *Proceedings of the Fifth International Conference on Software Reuse*, pages 290–299, Jun 1998. (cited on page 13).

[87] S. Kearney and J.F. Power. REM4j - A framework for measuring the reverse engineering capability of UML CASE tools. In *SEKE*, pages 209–214. Knowledge Systems Institute Graduate School, 2007. (cited on page 50).

[88] A. Kent and J.G. Williams. *Encyclopedia of Computer Science and Technology: Volume 35 - Supplement 20: Acquiring Task-Based Knowledge and Specifications to Seek Time Evaluation*. Encyclopedia of Computer Science Series. Taylor & Francis, 1996. (cited on page 12).

[89] A. Kent and J.G. Williams. *Encyclopedia of Microcomputers: Volume 25 - Supplement 4*. Microcomputers Encyclopedia. Taylor & Francis, 2000. (cited on page 12).

[90] B. Klatt, M. Küster, K. Krogmann, and O. Burkhardt. A change impact analysis case study: Replacing the input data model of SoMoX. *Recall*, 71:99–58. (cited on page 158).

[91] P. Klint, D. Landman, and J. Vinju. Exploring the limits of domain model recovery. In *Proceedings of the 29th IEEE International Conference on Software Maintenance (ICSM2013)*, pages 120–129, Sept 2013. (cited on page 197).

[92] A.J. Ko and B.A. Myers. A framework and methodology for studying the causes of software errors in programming systems. *Journal of Visual Languages & Computing*, 16(1):41–84, 2005. (cited on page 4).

[93] A.J. Ko, B.A Myers, M.J. Coblenz, and H.H. Aung. An exploratory study of how developers seek, relate, and collect relevant information during software maintenance tasks. *Software Engineering, IEEE Transactions on*, 32(12):971–987, Dec 2006. (cited on pages 4 and 195).

[94] R. Kollman, P. Selonen, E. Stroulia, T. Systä, and A. Zundorf. A study on the current state of the art in tool-supported UML-based static reverse engineering. In *Proceedings of the Ninth Working Conference on Reverse Engineering (WCRE'02)*, WCRE '02, pages 22–, Washington, DC, USA, 2002. IEEE Computer Society. (cited on pages 49 and 64).

[95] R. Koschke. Software visualization in software maintenance, reverse engineering, and re-engineering: A research survey. *Journal of Software Maintenance*, 15(2):87–109, March 2003. (cited on page 71).

[96] R. Koschke. Architecture reconstruction. In Andrea De Lucia and Filomena Ferrucci, editors, *Software Engineering*, volume 5413 of *Lecture Notes in Computer Science*, pages 140–173. Springer Berlin Heidelberg, 2009. (cited on page 195).

[97] J. Koskinen and T. Lehmonen. Analysis of ten reverse engineering tools. In K. Elleithy, editor, *Advanced Techniques in Computing Sciences and Software Engineering*, pages 389–394. Springer Netherlands, 2010. (cited on pages 48 and 49).

[98] S.B. Kotsiantis. Supervised machine learning: A review of classification techniques. In *Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pages 3–24, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press. (cited on pages 23, 24 and 205).

[99] A. Lake and C.R. Cook. Use of factor analysis to develop OOP software complexity metrics. Technical report, Oregon State University, Corvallis, OR, USA, 1994. (cited on pages 122 and 162).

[100] M. Lanza and S. Ducasse. Polymetric views - A lightweight visual approach to reverse engineering. *IEEE Transactions on Software Engineering*, 29(9):782–795, Sept 2003. (cited on page 13).

[101] J.-F Le Gall. Random trees and applications. *Probab. Surv*, 2(245-311):15–43, 2005. (cited on page 26).

[102] F. Leemhuis. Devnology community. `http://devnology.nl/`, 2012. (cited on page 74).

[103] T.C. Lethbridge, J. Singer, and A. Forward. How software engineers use documentation: The state of the practice. *IEEE Softw.*, 20(6):35–39, November 2003. (cited on pages 3 and 48).

[104] W. Li and S. Henry. Object-oriented metrics that predict maintainability. *Journal of systems and software*, 23(2):111–122, 1993. (cited on page 93).

[105] D. Lin and P. Pantel. Induction of semantic classes from natural language text. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 317–322, New York, NY, USA, 2001. ACM. (cited on page 138).

[106] J.B. Lovins. Development of a Stemming Algorithm. June 1968. (cited on page 141).

[107] W. Maalej, R. Tiarks, T. Roehm, and R. Koschke. On the comprehension of program comprehension. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 23(4):31, 2014. (cited on page 12).

[108] S. Mancoridis, B.S. Mitchell, C. Rorres, Y. Chen, and E.R. Gansner. Using automatic clustering to produce high-level system organizations of source code. *International Conference on Program Comprehension*, 1998. (cited on page 158).

[109] M.T. Maybury. Generating summaries from event data. *Inf. Process. Manage.*, 31(5):735–751, September 1995. (cited on page 195).

[110] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998. (cited on page 26).

[111] S. Medini, G. Antoniol, Y.-G. Guéhéneuc, M. Di Penta, and P. Tonella. SCAN: An approach to label and relate execution trace segments. In *Proceedings of the 19th Working Conference on Reverse Engineering (WCRE 2012)*, pages 135–144, Oct 2012. (cited on page 137).

[112] G.A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, (2):81–97, March 1956. (cited on page 4).

[113] T.M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997. (cited on pages 22 and 23).

[114] L. Moreno, J. Aponte, G. Sridhara, A Marcus, L. Pollock, and K. Vijay-Shanker. Automatic generation of natural language summaries for java classes. In *Proceedings of the IEEE 21st International Conference on Program Comprehension (ICPC)*, pages 23–32, May 2013. (cited on page 137).

[115] NetBeans. Netbeans Integrated Development Environment (IDE). `https://netbeans.org/features/index.html`, 2014. (cited on page 165).

[116] D. Ng, D.R. Kaeli, S. Kojarski, and D.H. Lorenz. Program comprehension using aspects. In *ICSE 2004 Workshop WoDiSEE'2004*, 2004. (cited on page 12).

[117] A. Nugroho. *The effects of UML modeling on the quality of software*. PhD thesis, Leiden University, the Netherlands, 2010. (cited on page 73).

[118] A. Nugroho and M.R.V. Chaudron. A survey of the practice of design – Code correspondence amongst professional software engineers. In *Proceedings of the First International Symposium on Empirical Software Engineering and Measurement*, ESEM '07, pages 467–469, Washington, DC, USA, 2007. IEEE Computer Society. (cited on pages 48, 70 and 118).

[119] Object Management Group (OMG). `http://www.omg.org`, 2014. (cited on page 32).

[120] M. Orr. Introduction to radial basis function networks. Technical report, Institute for Adaptive and Neural Computation, Edinburgh University, 1996. (cited on page 26).

[121] M.H. Osman. Answered questionnaire. `http://www.liacs.nl/~hosman/ValidationAll.rar`, 2014. (cited on pages 177 and 186).

[122] M.H. Osman. An example of the text dictionary. `www.liacs.nl/~hosman/TextDictionary.csv`, 2014. (cited on pages 6 and 141).

[123] M.H. Osman. The list of common and uncommon words. `http://www.liacs.nl/~hosman/CommonAndUncommon.xlsx`, 2014. (cited on page 142).

[124] M.H. Osman. SAAbs tool demonstration. http://www.youtube.com/watch?v=dHBB5wA2wDI, 2014. (cited on pages 152 and 165).

[125] M.H. Osman and M.R.V. Chaudron. Correctness and completeness of case tools in reverse engineering source code into uml model. *GSTF Journal on Computing*, 2(1):193–201, 2012. (cited on pages 36 and 118).

[126] M.H. Osman and M.R.V. Chaudron. An assessment of reverse engineering capabilities of UML CASE tools. In *Proceedings of 2nd Annual International Conference on Software Engineering and Application (SEA 2011)*, pages 7–12, December 12-13, 2011. (cited on page 92).

[127] M.H. Osman, M.R.V. Chaudron, and P. van der Putten. An analysis of machine learning algorithms for condensing reverse engineered class diagrams. In *Proceedings of the 29th IEEE International Conference on Software Maintenance (ICSM 2013)*, pages 140–149, Sept 2013. (cited on pages 157, 163 and 187).

[128] M.H. Osman, M.R.V. Chaudron, and P. van der Putten. Condensing reverse engineered class diagram using text mining. Technical report, Leiden University, the Netherlands (`http://www.liacs.nl/~hosman/TechnicalReport-2014-02.pdf`), 2014. (cited on pages 162 and 163).

[129] M.H. Osman and A. van Zadelhoff. Original questionnaire. `http://www.liacs.nl/~hosman/Questionnaire.rar`, 2012. (cited on page 72).

[130] M.H. Osman and A. van Zadelhoff. Structured questionnaire. `http://www.liacs.nl/~hosman/The_Presence_of_Classes_in_Class_Diagrams.pdf`, 2012. (cited on page 95).

[131] M.H. Osman and A. van Zadelhoff. Survey data. `http://www.liacs.nl/~hosman/SurveyData.rar`, 2012. (cited on page 74).

[132] M.H. Osman, A. van Zadelhoff, and M.R.V. Chaudron. UML class diagram simplification - A survey for improving reverse engineered class diagram comprehension. In *MODELSWARD*, pages 291–296, 2013. (cited on page 160).

[133] M.H. Osman, A. van Zadelhoff, D.R. Stikkolorum, and M.R.V. Chaudron. UML class diagram simplification: What is in the developer's mind? In *Proceedings of the Second Edition of the International Workshop on Experiences and Empirical Studies in Software Modelling*, EESSMod '12, pages 5:1–5:6, New York, NY, USA, 2012. ACM. (cited on pages 43, 160 and 165).

[134] M.H. Osman and L. Wei. The SAAbs tool. https://github.com/aislimau/SAAbs, 2014. (cited on pages 6 and 165).

[135] N. Pennington. Stimulus structures and mental representations in expert comprehension of computer programs. *Cognitive psychology*, 19(3):295–341, 1987. (cited on page 12).

[136] F. Perin, L. Renggli, and J. Ressia. Ranking software artifacts. In *4th Workshop on FAMIX and Moose in Reengineering (FAMOOSr 2010)*, 2010. (cited on pages 120 and 133).

[137] S.L. Pfleeger and B.A. Kitchenham. Principles of survey research: Part 1: Turning lemons into lemonade. *SIGSOFT Softw. Eng. Notes*, 26(6):16–18, November 2001. (cited on page 5).

[138] H. Pirzadeh, A. Hamou-Lhadj, and M. Shah. Exploiting text mining techniques in the analysis of execution traces. In *Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM 2011)*, pages 223–232, Sept 2011. (cited on page 137).

[139] D. Pollet, S. Ducasse, L. Poyet, I Alloui, S. Cimpan, and H. Verjus. Towards a process-oriented software architecture reconstruction taxonomy. In *Proceedings of the 11th European Conference on Software Maintenance and Reengineering (CSMR '07)*, pages 137–148, March 2007. (cited on page 195).

[140] M.F. Porter. Readings in information retrieval. chapter An Algorithm for Suffix Stripping, pages 313–316. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997. (cited on page 141).

[141] F.J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 445–453, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. (cited on page 154).

[142] F.D. Rácz and K. Koskimies. Tool-supported compression of UML class diagrams. In *«UML»'99—The Unified Modeling Language*, pages 172–187. Springer, 1999. (cited on page 4).

[143] A. Rajaraman and J.D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, New York, NY, USA, 2011. (cited on page 140).

[144] C. Riva. Reverse architecting: an industrial experience report. In *Proceedings of the 7th Working Conference on Reverse Engineering (WCRE2000)*, pages 42–50, 2000. (cited on page 195).

[145] S. Robitaille, R. Schauer, and R.K. Keller. Bridging program comprehension tools by design navigation. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM 2000)*, pages 22–32, 2000. (cited on page 13).

[146] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-oriented Modeling and Design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1991. (cited on page 18).

[147] J. Rumbaugh, I. Jacobson, and G. Booch, editors. *The Unified Modeling Language Reference Manual*. Addison-Wesley Longman Ltd., Essex, UK, UK, 1999. (cited on page 19).

[148] P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Softw. Engg.*, 14(2):131–164, April 2009. (cited on page 5).

[149] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:211–229, 1959. (cited on page 23).

[150] T. Schäfer, M. Eichberg, M. Haupt, and M. Mezini. The SEXTANT software exploration tool. *IEEE Trans. Softw. Eng.*, 32(9):753–768, 2006. (cited on page 13).

[151] B. Sharif and J.I. Maletic. The effect of layout on the comprehension of UML class diagrams: A controlled experiment. In *5th IEEE International Workshop on Visualizing Software for Understanding and Analysis (VISSOFT 2009)*, pages 11–18. IEEE, 2009. (cited on page 21).

[152] B. Shneiderman and R.E. Mayer. Interactions in programmer behavior: A model and experimental results. *International Journal of Parallel Programming*, 8(3):219–238, 1979. (cited on page 12).

[153] SparxSystems. Enterprise Architect. `http://www.sparxsystems.com.au/`, 2013. (cited on pages 34, 72 and 94).

[154] D. Steidl, B. Hummel, and E. Juergens. Using network analysis for recommendation of central software classes. In *Proceedings of the 19th Working Conference on Reverse Engineering*, WCRE '12, pages 93–102, Washington, DC, USA, 2012. IEEE Computer Society. (cited on pages 120, 133, 157 and 196).

[155] P. Stevens. Small-scale XMI programming: A revolution in UML tool use? *Automated Software Eng.*, 10(1):7–21, January 2003. (cited on pages 22 and 159).

[156] M.-A.D. Storey, F.D. Fracchia, and H.A. Müller. Cognitive design elements to support the construction of a mental model during software exploration. *Journal of Systems and Software*, 44(3):171 – 185, 1999. (cited on pages 15, 156, 195 and 205).

[157] M.-A.D. Storey, F.D. Fracchia, and H.A. Mueller. Cognitive design elements to support the construction of a mental model during software visualization. In *Proceedings of the 5th International Workshop on Program Comprehension (WPC '97)*, WPC '97, pages 17–, Washington, DC, USA, 1997. IEEE Computer Society. (cited on pages 14, 132, 168 and 195).

[158] H. Störrle. On the impact of layout quality to understanding UML diagrams. In *Proceedings of the 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 135–142. IEEE, 2011. (cited on pages 21 and 193).

[159] H. Störrle. On the impact of layout quality to understanding UML diagrams: Size matters. In Juergen Dingel, Wolfram Schulte, Isidro Ramos, Silvia Abrahão, and Emilio Insfran, editors, *Model-Driven Engineering Languages and Systems*, volume 8767 of *Lecture Notes in Computer Science*, pages 518–534. Springer International Publishing, 2014. (cited on pages 21 and 193).

[160] D. Sun and K. Wong. On evaluating the layout of UML class diagrams for program comprehension. In *Proceedings of 13th International Workshop on Program Comprehension (IWPC 2005)*, pages 317–326. IEEE, 2005. (cited on page 21).

[161] R.S. Sutton. Introduction: The challenge of reinforcement learning. *Machine Learning*, 8(3-4):225–227, 1992. (cited on page 24).

[162] T. Systä. *Static and Dynamic Reverse Engineering Techniques for Java Software Systems*. PhD thesis, University of Tampere, Finland, 2000. (cited on page 4).

[163] M.-H. Tang, M.-H. Kao, and M.-H. Chen. An empirical study on object-oriented metrics. In *Software Metrics Symposium, 1999. Proceedings. Sixth International*, pages 242–249. IEEE, 1999. (cited on page 93).

[164] F. Thung, D. Lo, M.H. Osman, and M.R.V. Chaudron. Condensing class diagrams by analyzing design and network metrics using optimistic classification. In *Proceedings of the 22nd International Conference on Program Comprehension*, ICPC 2014, pages 110–121, New York, NY, USA, 2014. ACM. (cited on pages 157, 187, 196 and 198).

[165] Tigris.org. ArgoUML. `http://argouml.tigris.org/`, 2014. (cited on page 39).

[166] S. Tilley and S. Huang. A qualitative assessment of the efficacy of UML diagrams as a form of graphical documentation in aiding program understanding. In *Proceedings of the 21st annual international conference on Documentation*, pages 184–191. ACM, 2003. (cited on page 21).

[167] S.R. Tilley, P. Santanu, and D.B. Smith. Towards a framework for program understanding. In *Proceedings of 4th Workshop on Program Comprehension (WPC'96)*, pages 19–28. IEEE, 1996. (cited on page 168).

[168] K.M. Ting. Matching model versus single model: A study of the requirement to match class distribution using decision trees. In J.-F. Boulicaut, F. Esposito, F. Giannotti, and D. Pedreschi, editors, *Machine Learning: ECML 2004*, volume 3201 of *Lecture Notes in Computer Science*, pages 429–440. Springer Berlin Heidelberg, 2004. (cited on page 28).

[169] F. Tip. A survey of program slicing techniques. *Journal of programming languages*, 3(3):121–189, 1995. (cited on page 199).

[170] P. Tonella, M. Torchiano, B. Du Bois, and T. Systä. Empirical studies in reverse engineering: State of the art and future trends. *Empirical Softw. Engg.*, 12(5):551–571, October 2007. (cited on page 118).

[171] P. van der Putten and M. van Someren. A Bias-variance analysis of a real world learning problem: The CoIL challenge 2000. *Mach. Learn.*, 57(1-2):177–195, October 2004. (cited on pages 25 and 26).

[172] A. Van Deursen, C. Hofmeister, R. Koschke, L. Moonen, and C. Riva. Symphony: View-driven software architecture reconstruction. In *Proceedings of the 4th IEEE/IFIP Working Conference on Software Architecture (WICSA 2004)*, pages 122–132. IEEE, 2004. (cited on pages 3 and 70).

[173] A. von Mayrhauser and A.M. Vans. From code understanding needs to reverse engineering tool capabilities. In *Proceedings of the Sixth International Workshop on Computer-Aided Software Engineering (CASE '93)*, pages 230–239, Jul 1993. (cited on page 13).

[174] A. Von Mayrhauser and A.M. Vans. Program comprehension during software maintenance and evolution. *Computer*, 28(8):44–55, 1995. (cited on pages 13, 14 and 205).

[175] E. Voorhees. Natural language processing and information retrieval. In M. Pazienza, editor, *Information Extraction*, volume 1714 of *Lecture Notes in Computer Science*, pages 32–48. Springer Berlin Heidelberg, 1999. (cited on page 139).

[176] W3C. Extensible markup language (XML). `http://www.w3.org/XML/`, 2014. (cited on page 21).

[177] B.E. Wampler. *The Essence of Object-Oriented Programming with Java and UML*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001. (cited on pages 52 and 57).

[178] I.H. Witten, E. Frank, and M.A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, Amsterdam, 3 edition, 2011. (cited on pages 23, 25, 26 and 121).

[179] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, Z.-H. Zhou, M. Steinbach, D.J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowl. Inf. Syst.*, 14(1):1–37, December 2007. (cited on page 26).

[180] J. Wüst. SDMetrics. `http://www.sdmetrics.com/`, 2013. (cited on pages 35, 50, 94, 95, 121, 159 and 209).

[181] B. Xu, J. Qian, X. Zhang, Z. Wu, and L. Chen. A brief survey of program slicing. *SIGSOFT Softw. Eng. Notes*, 30(2):1–36, March 2005. (cited on page 199).

[182] K. Yatani, E. Chung, C. Jensen, and K.N. Truong. Understanding how and why open source contributors use diagrams in the development of Ubuntu. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 995–1004, New York, NY, USA, 2009. ACM. (cited on page 33).

[183] S. Yusuf, H. Kagdi, and J.I. Maletic. Assessing the comprehension of UML class diagrams via eye tracking. In *Proceedings of the 15th IEEE International Conference on Program Comprehension*, ICPC '07, pages 113–122, Washington, DC, USA, 2007. IEEE Computer Society. (cited on pages 21 and 70).

[184] A. Zaidman and S. Demeyer. Automatic identification of key classes in a software system using webmining techniques. *J. Softw. Maint. Evol.*, 20(6):387–417, November 2008. (cited on pages 119 and 133).

[185] M.V. Zelkowitz and D.R. Wallace. Experimental models for validating technology. *Computer*, 31(5):23–31, May 1998. (cited on page 6).

[186] W. Zhang, Y. Yang, and Q. Wang. Network analysis of OSS evolution: An empirical study on ArgoUML project. In *Proceedings of the 12th International Workshop on Principles of Software Evolution and the 7th Annual ERCIM Workshop on Software Evolution*, IWPSE-EVOL '11, pages 71–80, New York, NY, USA, 2011. ACM. (cited on page 42).

[187] J. Zhi, V. Garousi-Yusifoğlu, B. Sun, G. Garousi, S. Shahnewaz, and G. Ruhe. Cost, benefits and quality of software development documentation: A systematic mapping. *Journal of Systems and Software*, 99(0):175 – 198, 2015. (cited on page 32).

[188] Y. Zhou and H. Leung. Empirical analysis of object-oriented design metrics for predicting high and low severity faults. *IEEE Transactions on Software Engineering*, 32(10):771–789, 2006. (cited on pages 121 and 196).