Execution architecture views for evolving software-intensive systems

Callo Arias, Trosky Boris

Bibliography

About Processes and Threads (Windows): n.d.
URL: http://msdn.microsoft.com/en-us/library/ms681917.aspx

Agarwal, M. K., Appleby, K., Gupta, M., Kar, G., Neogi, A. and Sailer, A.: 2004, Problem determination using dependency graphs and run-time behavior models, *in* A. S. Wu and F. (eds), *15th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Springer, pp. 171–182.

Allen, R. and Garlan, D.: 1997, A formal basis for architectural connection, *ACM Trans. Softw. Eng. Methodol.* **6**(3), 213–249.

Alzamil, Z. A.: 2007, Redundant Coupling Detection Using Dynamic Dependence Analysis, *International Conference on Software Engineering Advances, ICSEA 200*, p. 43.

Automated Drawing of UML Diagrams: n.d.
URL: http://www.umlgraph.org/

Balmas, F., Wertz, H., Chaabane, R. and Artificielle, L.: 2005, Visualizing Dynamic Data Dependences as a Help to Maintain Programs, *International Conference on Software Maintenance*, Citeseer.

Balsamo, S., Di Marco, A., Inverardi, P. and Simeoni, M.: 2004, Model-based performance prediction in software development: A survey, *IEEE Transactions on Software Engineering* **30**(5), 295–310.

Basili, V.: 1996, The role of experimentation in software engineering: past, current, and future, *International Conference on Software Engineering,*, IEEE Computer Society, pp. 442–449.

Bass, L., Clements, P. and Kazman, R.: 2003, *Software architecture in practice*, Vol. 2, Addison-Wesley Professional.

Binkley, D. and Harman, M.: 2005, Locating dependence clusters and dependence pollution, *21 st IEEE International Conference on Software Maintenance* pp. 177–186.

Bohnet, J., Voigt, S. and Döllner, J.: 2009, Projecting code changes onto execution traces to support localization of recently introduced bugs, *ACM Symposium on Applied Computing*, ACM New York, NY, USA, pp. 438–442.

Bootchart: Boot Process Performance Visualization: 2010.
URL: http://www.bootchart.org

Breivold, H. P., Crnkovic, I., Land, R. and Larsson, S.: 2008, Using dependency model to support software architecture evolution, *23rd IEEE/ACM International Conference on Automated Software Engineering-Workshops, ASE Workshops*, pp. 82–91.

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M. and Khalil, M.: 2007, Lessons from applying the systematic literature review process within the software engineering domain, *J. Syst. Softw.* **80**(4), 571–583.

Briand, L., Labiche, Y. and Miao, Y.: 2003, Towards the reverse engineering of UML sequence diagrams, *10th Working Conference on Reverse Engineering*, IEEE Computer Society.

Brown, A., Kar, G. and Keller, A.: 2001, An active approach to characterizing dynamic dependencies forproblem determination in a distributed environment, *IEEE/IFIP International Symposium on Integrated Network Management Proceedings*, pp. 377–390.

Callo Arias, T. B., America, P. and Avgeriou, P.: 2009a, Constructing a Resource Usage View of a Large and Complex Software-Intensive System, *16th Working Conference on Reverse Engineering*, IEEE Computer Society, pp. 247–255.

Callo Arias, T. B., America, P. and Avgeriou, P.: 2009b, Defining execution viewpoints for a large and complex software- intensive system., *in* R. Kazman (ed.), *Joint 8th Working IEEE/IFIP Conference on Software Architecture & 3rd European Conference on Software Architecture*, pp. 1–10.

Callo Arias, T. B., America, P. and Avgeriou, P.: 2010, Defining and Documenting Execution Viewpoints for a Large and Complex Software-Intensive System, *Journal of Systems and Software* **In Press,**(Available online 25 November).

Callo Arias, T. B., Avgeriou, P. and America, P.: 2008, Analyzing the Actual Execution of a Large Software-Intensive System for Determining Dependencies, *15th Working Conference on Reverse Engineering*, IEEE Computer Society, pp. 49–58.

Callo Arias, T. B., Avgeriou, P. and America, P.: 2009, Tech. Report: Documenting a Catalog of Viewpoints to Describe the Execution Architecture of a Large Software-Intensive System for the ISO/IEC 42010 Standard.
URL: `http://www.esi.nl/projects/darwin/publications.asp`

Callo Arias, T. B., Avgeriou, P., America, P., Blom, K. and Bachynskyy, S.: 2010, A top-down strategy to reverse architecting execution views for a large and complex software-intensive system: An experience report, *Science of Computer Programming* **In Press**(Available online 8 December).

Cataldo, M., Herbsleb, J. D. and Carley, K. M.: 2008, Socio-Technical Congruence: A Framework for Assessing the Impact of Technical and Work Dependencies on Software Development Productivity, *2nd International Symposium on Empirical Software Engineering and Measurement*, ACM, Kaiserslautern, Germany, pp. 2–11.

Chen, K. and Rajlich, V.: 2000, Case study of feature location using dependence graph, *Program Comprehension, 2000. Proceedings. IWPC 2000. 8th International Workshop on*, pp. 241–247.

Chen, Z., Xu, B., Yang, H., Liu, K. and Zhang, J.: 2000, An approach to analyzing dependency of concurrent programs, *Proceedings of the The First Asia-Pacific Conference on Quality Software (APAQS'00)*, IEEE Computer Society Washington, DC, USA, p. 39.

Chen, Z., Xu, B. and Zhao, J.: 2002, An overview of methods for dependence analysis of concurrent programs, *ACM Sigplan Notices* **37**(8), 45–52.

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Reed Little, Nord, R. and Stafford, J. A.: 2002, *Documenting software architectures: views and beyond*, Addison-Wesley Professional.

Code Profiler - AQtime: 2010.
URL: `http://www.automatedqa.com/products/aqtime/`

Cornelissen, B., Holten, D., Zaidman, A., Moonen, L., van Wijk, J. and van Deursen, A.: 2007, Understanding execution traces using massive sequence and circular bundle views, *15th IEEE International Conference on Program Comprehension.*, IEEE Computer Society,, pp. 49–58.

Cornelissen, B., Zaidman, A., van Deursen, A., Moonen, L. and Koschke, R.: 2009, A Systematic Survey of Program Comprehension through Dynamic Analysis, *IEEE Transactions on Software Engineering* **35**(5), 684–702.

Cossette, B. and Walker, R.: 2007, Polylingual Dependency Analysis Using Island Grammars: A Cost Versus Accuracy Evaluation, *IEEE International Conference on Software Maintenance, 2007. ICSM 2007* pp. 214–223.

Cox, L., Delugach, H. and Skipper, D.: 2001, Dependency analysis using conceptual graphs, *Proceedings of the 9th International Conference on Conceptual Structures, ICCS*, Citeseer, pp. 117–130.

De Souza, C. R. B.: 2005, *On the relationship between software dependencies and coordination: field studies and tool support*, PhD thesis.

Devarakonda, M. and Iyer, R.: 1989, Predictability of process resource usage: A measurement-based study on UNIX, *IEEE Transactions on Software Engineering* **15**, 1579–1586.

Dong, X. and Godfrey, M.: 2007, System-level usage dependency analysis of object-oriented systems, *Proc. of the Intl. Conference on Software Maintenance*, pp. 375–384.

Ducasse, S. and Pollet, D.: 2009, Software Architecture Reconstruction: A Process-Oriented Taxonomy, *IEEE Transactions on Software Engineering* **35**(July/August), 573–591.

Eclipse Test & Performance Tools Platform Project: 2010.
URL: `http://www.eclipse.org/tptp/`

Egyed, A.: 2003, A scenario-driven approach to trace dependency analysis, *IEEE Transactions on Software Engineering* **29**(2), 116–132.

Eisenbarth, T., Koschke, R. and Simon, D.: 2001, Aiding program comprehension by static and dynamic feature analysis, *Proceedings of the International Conference on Software Maintenance*, pp. 602–611.

Eisenbarth, T., Koschke, R. and Simon, D.: 2003, Locating features in source code, *IEEE Transactions on Software Engineering* **29**(3), 210–224.

Ferrante, J., Ottenstein, K. J. and Warren, J. D.: 1987, The program dependence graph and its use in optimization, *ACM Trans. Program. Lang. Syst.* **9**(3), 319–349.

Fowler, M.: 1999, *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, Boston, MA, USA.

Fowler, M.: 2003, Who needs an architect?, *IEEE Software* **20**(5), 11–13.

Fundel, K., Kuffner, R. and Zimmer, R.: 2007, RelEx–relation extraction using dependency parse trees, *Bioinformatics* **23**(3), 365.

Gao, J., Kar, G. and Kermani, P.: 2004, Approaches to building self healing systems using dependency analysis, *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, Vol. 1, pp. 119–132 Vol.1.

Garlan, D. and Perry, D.: 1995, Introduction to the special issue on software architecture, *IEEE Transactions on Software Engineering* **21**(4), 269–274.

Garousi, V., Briand, L. C. and Labiche, Y.: 2006, Analysis and Visualization of Behavioral Dependencies Among Distributed Objects Based on UML Models, *MoDELS'06*, Springer Berlin / Heidelberg, Genova, Italy, pp. 365–379.

Gavin, D.: 1998, Performance Monitoring Tools for Linux, *Linux Journal* (56).

Glass, R. L.: 2004, Matching methodology to problem domain, *Commun. ACM* **47**(5), 19–21.

Glorie, M., van Deursen, A., Zaidman, A. and Hofland, L.: 2009, Splitting a large software repository for easing future software evolution- an industrial experience report, *Journal of Software Maintenance and Evolution: Research and Practice* **21**(2), 113–141.

Gotel, O. C. Z. and Finkelstein, A. C. W.: 1994, An Analysis of the Requirements Traceability Problem, *Proc. First Intl Conf. Requirements Eng.*, pp. 94–101.

Graphviz - Graph Visualization Software: n.d.
URL: `http://www.graphviz.org/`

Gupta, M., Neogi, A., Agarwal, M. K. and Kar, G.: 2003, Discovering dynamic dependencies in enterprise environments for problem determination, *in* M. B. Keller and A. (eds), *14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*.

Hamou-Lhadj, A., Braun, E. and Amyot, D.: 2005, Recovering behavioral design models from execution traces, *Ninth European Conference on Software Maintenance and Reengineering, CSMR*, pp. 112 – 121.

Hamou-Lhadj, A. and Lethbridge, T. C.: 2004, A survey of trace exploration tools and techniques, *CASCON'04*, IBM Press, pp. 42–55.

Hassan, A. and Holt, R.: 2004, Predicting change propagation in software systems, *Proceedings of 26th International Conference on Software Maintenance (ICSM04)*, Citeseer.

Hofmeister, C., Kruchten, P., Nord, R., Obbink, H., Ran, A. and America, P.: 2007, A general model of software architecture design derived from five industrial approaches, *Journal of Systems and Software* **80**(1), 106–126.

Hofmeister, C., Nord, R. and Soni, D.: 2000, *Applied software architecture*, Addison-Wesley Professional.

Holmes, R. and Walker, R.: 2007, Task-specific source code dependency investigation, *4th IEEE International Workshop on Visualizing Software for Understanding and Analysis, 2007. VISSOFT 2007*, pp. 100–107.

Holz, H. J., Applin, A., Haberman, B., Joyce, D., Purchase, H. and Reed, C.: 2006, Research Methods in Computing: What are they, and how should we teach them?, *SIGCSE Bull.* **38**(41), 96–114.

Hopkins, R. and Jenkins, K.: 2008, *Eating the IT elephant: moving from greenfield development to brownfield*, IBM Press.

Huang, L. and Song, Y.: 2007, Precise dynamic impact analysis with dependency analysis for object-oriented programs, *Fifth International Conference on Software Engineering Research, Management and Applications*, pp. 374–381.

Hunt, G., Aiken, M., F\\"ahndrich, M., Hawblitzel, C., Hodson, O., Larus, J., Levi, S., Steensgaard, B., Tarditi, D. and Wobber, T.: 2007, Sealing OS processes to improve dependability and safety, *2nd ACM SIGOPS/EuroSys European Conference on Computer Systems*, ACM, pp. 341–354.

Ishio, T., Kusumoto, S. and Inoue, K.: 2004, Debugging support for aspect-oriented program based on program slicing and call graph, *Proc. 20th IEEE International Conference on Software Maintenance*, Citeseer, pp. 178–187.

ISO/IEC: 2007, ISO/IEC Std. 42010: Recommended Practice for Architectural Description of Software-intensive Systems.
URL: `http://www.iso-architecture.org/ieee-1471/`

Ivkovic, I. and Kontogiannis, K.: 2006, Towards automatic establishment of model dependencies using formal concept analysis, *International Journal of Software Engineering and Knowledge Engineering* **16**(4), 499–522.

Jacsó, P.: 2008, Google Scholar revisited, *Online Information Review* **32**(1), 102–114.

Jasz, J., Beszedes, A., Gyimothy, T. and Rajlich, V.: 2008, Static Execute After/Before as a Replacement of Traditional Software Dependencies, *IEEE International Conference on Software Maintenance, 2008. ICSM 2008*, pp. 137–146.

Jiang, T., Gold, N., Harman, M. and Li, Z.: 2008, Locating dependence structures using search-based slicing, *Information and Software Technology* **50**(12), 1189–1209.

Jiang, Z., Hassan, A. and Hamann, G.: 2008, An automated approach for abstracting execution logs to execution events, *Journal of Software Maintenance and Evolution: Research and Practice* **20**(4), 249–267.

Jourdan, G.-v., Ritthiruangdech, P. and Ural, H.: 2006, Test Suite Reduction Based on Dependence Analysis, *in* A. L. E. Al. (ed.), *21th International Symposium Computer and Information Sciences*, number Computer and Information Sciences ISCIS, Springer, pp. 1021–1030.

Kagdi, H. and Maletic, J.: 2007, Combining single-version and evolutionary dependencies for software-change prediction, *Fourth International Workshop on Mining Software Repositories*, IEEE Computer Society, p. 17.

Kazman, R., Abowd, G., Bass, L. and Clements, P.: 1996, Scenario-based analysis of software architecture, *Software, IEEE* **13**(6), 47–55.

Keller, A. and Kar, G.: 2000, Dynamic dependencies in application service management, *International Conference on Parallel and Distributed Processing Techniques and Applications*.

Keller, A., Kar, G. and Blumenthal, U.: 2000, Classification and computation of dependencies for distributed management, *Fifth International Conference on Computers and Communications (ISCC 2000)*.

Khan, S., Greenwood, P., Garcia, A. and Rashid, A.: 2008, On the Impact of Evolving Requirements-Architecture Dependencies: An Exploratory Study, *in* Z. B. Léonard and M. (eds), *20th International Conference Advanced Information Systems Engineering*, pp. 243–257.

Kitchenham, B.: 2004a, Software Productivity Measurement Using Multiple Size Measures, *IEEE Trans. Softw. Eng.* **30**(12), 1023–1035.

Kitchenham, B. A.: 2004b, Procedures for Performing Systematic Reviews, *Technical report*, Keele University and NICTA.

Kitchenham, B., Brereton, O. P., Budgen, D., Turner, M., Bailey, J. and Linkman, S.: 2009, Systematic literature reviews in software engineering - A systematic literature review, *Information and Software Technology* **51**(1), 7–15.

Koning, H. and van Vliet, H.: 2006, A method for defining {IEEE Std 1471} viewpoints, *Journal of Systems and Software* **79**(1), 120–131.

Kontogiannis, K., Linos, P. and Wong, K.: 2006, Comprehension and Maintenance of Large-Scale Multi-Language Software Applications, *ICSM '06*, IEEE Computer Society, Washington, DC, USA, pp. 497–500.

Korel, B., Tahat, L. and Vaysburg, B.: 2002, Model based regression test reduction using dependence analysis, *Proceedings. International Conference on Software Maintenance, 2002.*, pp. 214–223.

Koschke, R.: 2009, Architecture reconstruction: Tutorial on reverse engineering to the architectural level, *Lecture Notes in Computer Science* **5413**, 140–173.

Kossiakoff, A. and Sweet, W. N.: 2002, *Systems engineering: principles and practice*, Wiley-Interscience.

Kruchten, P.: 1995, The 4+ 1 view model of architecture, *IEEE software* **12**(6), 42–50.

Kruchten, P.: 2000, *The rational unified process: an introduction*, second edn, Addison-Wesley, Boston, MA, USA.

Kruchten, P., Obbink, H. and Stafford, J. A.: 2006, The past, present, and future for software architecture, *IEEE software* **23**(2), 22–30.

Kuhn, A. and Greevy, O.: 2006, Exploiting the analogy between traces and signal processing, *Proceedings of the 22nd IEEE International Conference on Software Maintenance*, IEEE Computer Society, pp. 320 – 329.

Law, J. and Rothermel, G.: 2003a, Incremental dynamic impact analysis for evolving software systems, *Proceedings of the International Symposium on Software Reliability Engineering*, Citeseer, pp. 430–441.

Law, J. and Rothermel, G.: 2003b, Whole program path-based dynamic impact analysis, *25th International Conference on Software Engineering*, IEEE Computer Society Washington, DC, USA, pp. 308–318.

Lehman, M.: 1996, *Laws of software evolution revisited*, Vol. 1149, Springer Berlin / Heidelberg, pp. 108–124.

Leitch, R. and Stroulia, E.: 2003, Assessing the maintainability benefits of design restructuring using dependency analysis, *Proc. of the 9th International Symposium on Software Metrics*, Citeseer, pp. 309–322.

Lewandowski, D.: 2010, Google Scholar as a tool for discovering journal articles in library and information science, *accepted for publication in Online Information Review* **34**.

Li, B., Zhou, Y., Wang, Y. and Mo, J.: 2005, Matrix-based component dependence representation and its applications in software quality assurance, *ACM SIGPLAN Notices* **40**(11), 29–36.

Li, Y., Zhang, M. and Hou, C.: 2005, An Active Method to Building Dynamic Dependency Model for Distributed Components, *ICAC '05: Proceedings of the Second International Conference on Automatic Computing*, IEEE Computer Society, Washington, DC, USA, pp. 337–338.

Liangli, M., Houxiang, W. and Yansheng, L.: 2006, The Design of Dependency Relationships Matrix to improve the testability of Component-based Software, *Sixth International Conference on Quality Software, QSIC 2006.*, pp. 93–98.

Lienhard, A., Greevy, O. and Nierstrasz, O.: 2007, Tracking objects to detect feature dependencies, *15th IEEE International Conference on Program Comprehension*, pp. 59–68.

Lind-Nielsen, J., Andersen, H. R., Hulgaard, H., Behrmann, G., Kristoffersen, K. and Larsen, K. G.: 2001, Verification of large state/event systems using compositionality and dependency analysis, *Formal Methods in System Design* **18**(1), 5–23.

Linux Trace Toolkit (LTTng) Project: 2010.
URL: `http://ltt.polymtl.ca/`

Loyall, J. P. and Mathisen, S. A.: 1993, Using Dependence Analysis to Support the Software Maintenance Process, *ICSM '93*, IEEE Computer Society, Washington, DC, USA, pp. 282–291.

Mao, C., Zhang, J. and Lu, Y.: 2007, Using Dependence Matrix to Support Change Impact Analysis for CBS, *Fifth International Conference on Computational Science and Applications (ICCSA 2007)*, pp. 192–200.

Maule, A., Emmerich, W. and Rosenblum, D.: 2008, Impact analysis of database schema changes, *30th international conference on Software engineering*, pp. 451–460.

McComb, D., Robe, S., Hoare, S. and Crawford-Hines, S.: 2002, Dependency analysis and visualization as tools to prolong system life, *26 th Annual International Computer Software and Applications Conference (COMPSAC02)*, pp. 463–465.

Mehta, N. R., Medvidovic, N. and Phadke, S.: 2000, Towards a Taxonomy of Software Connectors, *ICSE '00*, IEEE Computer Society, Los Alamitos, CA, USA, p. 178.

Microsoft Corporation: 2010a, Fast Boot / Fast Resume Design.
URL: `http://www.microsoft.com/whdc/system/sysperf/fastboot/default.mspx`

Microsoft Corporation: 2010b, Sysinternals Suite.
URL: `http://technet.microsoft.com/en-us/sysinternals/`

Microsoft Corporation: 2010c, Windows Performance Analyzer (WPA).
URL: `http://msdn.microsoft.com/en-us/library/ff191077.aspx`

Moe, J. and Carr, D. A.: 2001, Understanding distributed systems via execution trace data, *International Workshop on Program Comprehension, IWPC 2001.*, IEEE Computer Society Press, pp. 60–69.

Moise, D. L. and Wong, K.: 2005, Extracting and Representing Cross-Language Dependencies in Diverse Software Systems, *WCRE '05*, IEEE Computer Society, Washington, DC, USA, pp. 209–218.

Moraes, R. L. O., Martins, E. and Mendes, N. V.: 2005, Fault Injection Approach Based on Dependence Analysis, *COMPSAC'05*, IEEE Computer Society, Washington, DC, USA, pp. 181–188.

Moriconi, M. and Winkler, T. C.: 1990, Approximate Reasoning about the semantic effects of program changes, *IEEE Trans. Softw. Eng.* **16**(9), 980–992.

Muller, G.: 2004a, *CAFCR: A Multi-view Method for Embedded Systems Architecting; Balancing Genericity and Specificity*, Phd, Technical University Delft.

Muller, G.: 2004b, *Conceptual View*, chapter 8, p. 81.
URL: `http://www.gaudisite.nl/ArchitecturalReasoning.html`

Muller, G.: 2009, Gaudí System Architecting - A Reference Architecture Primer.
URL: `http://www.gaudisite.nl/info/ReferenceArchitecturePrimer.info.html`

Murphy, G., Notkin, D. and Sullivan, K.: 2001, Software reflexion models: Bridging the gap between design and implementation, *IEEE Transactions on Software Engineering* **27**(4), 364–380.

Muskens, J. and Chaudron, M.: 2004, Prediction of run-time resource consumption in multi-task component-based software systems, *Lecture Notes in Computer Science* **3054**(Component-Based Software Engineering), 162–177.

Myers, G. J.: 1975, *Reliable Software Through Composite Design*, Petrocelli/Charter, New York.

Nagappan, N. and Ball, T.: 2007, Using software dependencies and churn metrics to predict field failures: An empirical case study, *First International Symposium on Empirical Software Engineering and Measurement, ESEM 2007.*, pp. 364–373.

Narayanasamy, S., Pereira, C. and Calder, B.: 2006, Recording shared memory dependencies using strata, *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, ACM, p. 240.

.netCHARTING: n.d.
URL: http://www.dotnetcharting.com/

NetworkX: n.d.
URL: http://networkx.lanl.gov/

Neuvo, Y.: 2004, Cellular Phones as Embedded Systems, *IEEE International Solid-State Circuits Conference*, pp. 32–37.

Obbink, H., Kruchten, P., Kozaczynski, W., Hilliard, R., Ran, A., Postema, H., Lutz, D., Kazman, R., Tracz, W. and Kahane, E.: 2002, Software Architecture Review and Assessment (SARA) Report Version 1.0.
URL: http://philippe.kruchten.com/architecture/SARAv1.pdf

Pfaltz, J. L.: 2006, Using concept lattices to uncover causal dependencies in software, *in* R. M. Schmid and J. (eds), *4th International Conference Formal Concept Analysis*, Vol. 3874, Springer, p. 233.

Philips Healthcare: 2010, Magnetic Resonance Imaging.
URL: http://www.healthcare.philips.com/main/products/mri/systems/

Podgurski, A. and Clarke, L. A.: 1990, A Formal Model of Program Dependences and Its Implications for Software Testing, Debugging, and Maintenance, *IEEE Trans. Softw. Eng.* **16**(6), 965–979.

Potts, C.: 1993, Software-Engineering Research Revisited, *IEEE Software* **10**(5), 19–28.

Registry (Windows): n.d.
URL: http://msdn.microsoft.com/en-us/library/ms724871.aspx

Riva, C., Selonen, P., Systä, T. and Xu, J.: 2009, A profile-based approach for maintaining software architecture: an industrial experience report, *Journal of Software Maintenance and Evolution: Research and Practice* **21**(2), n/a–n/a.

Robillard, M.: 2008, Topology analysis of software dependencies, *ACM Transactions on Software Engineering and Methodology (TOSEM)* **17**(4), 18.

Robinson, M. L. and Wusteman, J.: 2007, Putting Google Scholar to the test: a preliminary study, *Program: electronic library and information systems* **41**(1), 71–80.

Ronen, I., Dor, N., Porat, S. and Dubinsky, Y.: 2006, Combined static and dynamic analysis for inferring program dependencies using a pattern language, *Conference of the Center for Advanced Studies on Collaborative research*, ACM, p. 3.

Rozanski, N. and Woods, E.: 2005, *Software systems architecture: working with stakeholders using viewpoints and perspectives*, Addison Wesley.

Ryser, J. and Glinz, M.: 2000, Using dependency charts to improve scenario-based testing, *Proceedings of the 17th International Conference on Testing Computer Software (TCS2000)*, Citeseer.

Safyallah, H. and Sartipi, K.: 2006, Dynamic Analysis of Software Systems using Execution Pattern Mining, *ICPC '06*, IEEE Computer Society, Washington, DC, USA, pp. 84–88.

Sangal, N., Jordan, E., Sinha, V. and Jackson, D.: 2005, Using dependency models to manage complex software architecture, *20th annual ACM SIGPLAN Conference on object-oriented programming, systems, languages, and applications*, ACM, pp. 167–176.

Shaw, M.: 2002, What makes good research in software engineering?, *International Journal on Software Tools for Technology* **4**(1), 1–7.

Shaw, M. and Garlan, D.: 1996, *Software architecture: perspectives on an emerging discipline*, Prentice Hall Englewood Cliffs, NJ.

Sozer, H. and Tekinerdogan, B.: 2008, Introducing recovery style for modeling and analyzing system recovery, *Seventh Working IEEE/IFIP Conference on Software Architecture (WICSA)*, IEEE, pp. 167–176.

Stafford, J. A., Wolf, A. and Caporuscio, M.: 2003, The application of dependence analysis to software architecture descriptions, *in* M. Bernardo and P. Inverardi (eds), *Third International School on Formal Methods for the Design of Computer, Communication and Software Systems: Software Architectures*, pp. 52–62.

Stafford, J. A. and Wolf, A. L.: 1998, Architecture-level dependence analysis in support of software maintenance, *ISAW '98*, ACM Press, New York, NY, USA, pp. 129–132.

Stafford, J. A. and Wolf, A. L.: 2001, Architecture-level dependence analysis for software systems, *International Journal of Software Engineering and Knowledge Engineering* **11**(4), 431–451.

Steinle, M., Aberer, K., Girdzijauskas, S. and Lovis, C.: 2006, Mapping moving landscapes by mining mountains of logs: novel techniques for dependency model generation, *Proceedings of the 32nd international conference on Very large data bases*, VLDB Endowment, p. 1102.

Stevens, R.: 1998, *Systems engineering: coping with complexity*, Prentice Hall.

Stevens, W. P., Myers, G. J. and Constantine, L. L.: 1974, Structured Design, *IBM Systems Journal* **13**(2), 115–139.

Stoermer, C., L. O'Brien and Verhoef, C.: 2002, Practice patterns for architecture reconstruction, *Ninth Working Conference on Reverse Engineering*, pp. 151–160.

Systä, T.: 2000, *Static and dynamic reverse engineering techniques for java software systems*, Phd, University of Tampere.

Tallam, S. and Gupta, R.: 2007, Unified control flow and data dependence traces, *ACM Transactions on Architecture and Code Optimization (TACO)* **4**(3), 19–es.
URL: `http://portal.acm.org/citation.cfm?id=1275943`

Technet-Microsoft: 2003a, Memory Object: Core Services.
URL: `http://technet.microsoft.com/en-us/library/cc778082(WS.10).aspx`

Technet-Microsoft: 2003b, Process Object: Core Services.
URL: `http://technet.microsoft.com/en-us/library/cc780836(WS.10).aspx`

TNO/IDATE: 2005, Software Intensive Systems in the Future.
URL: `http://www.itea2.org/attachments/150/ITEA_SIS_in__the%_future__Final_Report.pdf`

van de Laar, P., America, P., Rutgers, J., van Loo, S., Muller, G., Punter, T. and Watts, D.: 2007, The Darwin Project: Evolvability of Software-Intensive Systems, *Workshop on Evolvability at International Conference on Software Maintenance*, pp. 48–53.

van de Laar, P., Douglas, A. U. and America, P.: 2010, *Researching Evolvability*, Springer, chapter Researchin.

van Deursen, A.: 2002, Software architecture recovery and modelling, *ACM SIGAPP Applied Computing Review* **10**(1), 4–7.

van Deursen, A., Hofmeister, C., Koschke, R., Moonen, L. and Riva, C.: 2004, Symphony: View-driven software architecture reconstruction, *Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture*, IEEE Computer Society, Washington, DC, USA, pp. 122–134.

van Ommering, R.: 2002, Building product populations with software components, *ICSE '02: Proceedings of the 24th International Conference on Software Engineering*, ACM, New York, NY, USA, pp. 255–265.

Vasilache, S. and Tanaka, J.: 2005, Bridging the Gap between Analysis and Design Using Dependency Diagrams, *Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA05)*.

Vieira, M., Dias, M. and Richardson, D.: 2001, Describing Dependencies in Component Access Points, *23rd Intern. Conf. on Software Engineering*, pp. 115–118.

Vieira, M. and Richardson, D.: 2002, Analyzing dependencies in large component-based systems, *17th IEEE International Conference on Automated Software Engineering, ASE 2002*, pp. 241–244.

Watkins, R. and Neal, M.: 1994, Why and How of Requirements Tracing, *IEEE Softw.* **11**(4), 104–106.

Wood, T., Cherkasova, L., Ozonat, K. and Shenoy, P.: 2008, Profiling and modeling resource usage of virtualized applications, *9th ACM/IFIP/USENIX International Conference on Middleware*, Springer-Verlag New York, Inc., pp. 366–387.

Woodside, C.: 2002, Software resource architecture and performance evaluation of software architectures, *34th Annual Hawaii International Conference on System Sciences*, IEEE, p. 10.

Xiao, C. and Tzerpos, V.: 2005, Software clustering based on dynamic dependencies, *Proceedings of the Ninth European Conference on Software Maintenance and Reengineering*, Vol. pp, IEEE Computer Society Washington, DC, USA, pp. 124–133.

Xiao, Y. and Urban, S. D.: 2008, Recovery of Concurrent Processes in a Service Composition Environment Using Data Dependencies, *Proceedings of the OTM 2008 Confederated*, Springer, pp. 139–156.

Xin, B. and Zhang, X.: 2007, Efficient online detection of dynamic control dependence, *Proceedings of the 2007 international symposium on Software testing and analysis*, ACM, p. 195.

Xing, Z. and Stroulia, E.: 2006, Understanding the evolution and co-evolution of classes in object-oriented systems, *International Journal of Software Engineering and Knowledge Engineering* **16**(1), 23–51.

Yantzi, D. and Andrews, J.: 2007, Industrial evaluation of a log file analysis methodology, *5th International Workshop on Dynamic Analysis*, pp. 4–4.

YourKit .NET & Java Profiling: 2010.
URL: http://www.yourkit.com/

Zhang, W. and Ryder, B.: 2007, Discovering accurate interclass test dependences, *7th ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*, pp. 55–61.

Zhao, J.: 2001, Using dependence analysis to support software architecture understanding, *New Technologies on Computer Software* .
URL: http://arxiv.org/abs/cs/0105009

Zhao, J.: 2002, Change impact analysis for aspect-oriented software evolution, *International Workshop on Principles of Software Evolution*, ACM New York, NY, USA, pp. 108–112.

Zimmermann, T. and Nagappan, N.: 2007, Predicting subsystem failures using dependency graph complexities, *Proceedings of the The 18th IEEE International Symposium on Software Reliability*, IEEE Computer Society, pp. 227–236.

Zimmermann, T. and Nagappan, N.: 2008, Predicting defects using network analysis on dependency graphs, *Proceedings of the 30th international conference on Software engineering* pp. 531–540.