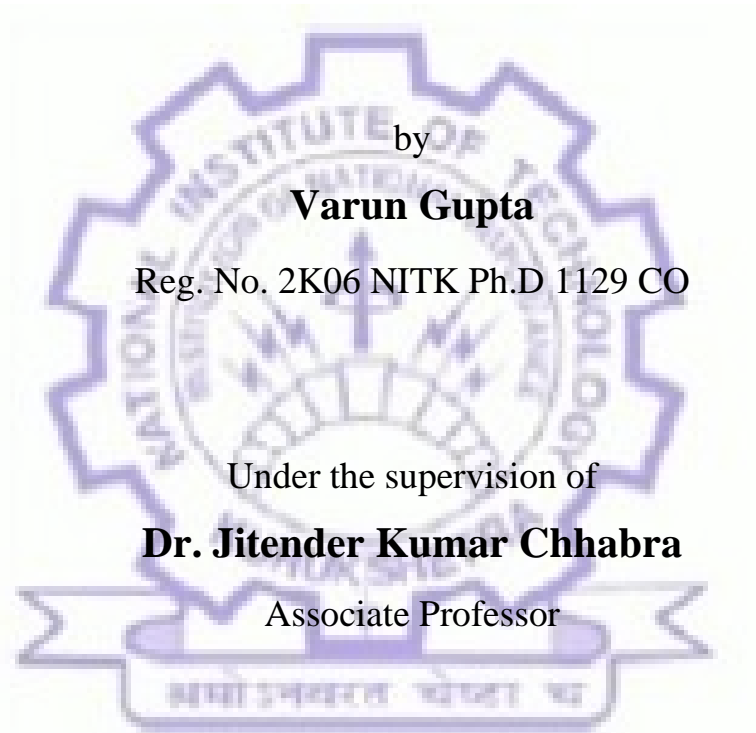


Object-Oriented Static and Dynamic Software Metrics for Design and Complexity

Submitted in fulfillment of the requirement of the degree of

DOCTOR OF PHILOSOPHY



**Department of Computer Engineering
National Institute of Technology
Kurukshetra-136119 India**

June 2010

ACKNOWLEDGEMENTS

I consider myself fortunate and privileged to have Dr. Jitender Kumar Chhabra as my supervisor. I am deeply indebted to him for shaping my path to research by guiding me with his extensive knowledge and his insightful discussions. I thank him for his constant support and encouragement throughout the course of this research work.

I would like to thank Departmental Research Committee members Dr. Mayank Dave (HOD), Dr. A. K. Singh, and Dr. S. K. Jain who gave me suggestions on improving this work.

The research reported in this thesis has benefited from anonymous reviewers. I would like to thank all those reviewers, who through their incisive and useful critic have contributed significantly to improving the submitted papers and consequently the work presented here.

I thank all others who have contributed to my work with material, data, knowledge, viewpoints, advice, comments, suggestions, ideas and other things. I would like to thank my colleagues and friends for inspiring me to focus on my research work.

Lastly and by no means least, I thank my family for their support, encouragement, and understanding during the research work.

VARUN GUPTA

LIST OF PUBLICATIONS OUT OF THE THESIS

Papers Published/Accepted in International Journals

1. "Package Coupling Measurement in Object-Oriented Software", Journal of Computer Science and Technology (Springer), vol. 24(2), pp. 273-283, March 2009.
2. "Object-Oriented Cognitive-Spatial Complexity Measures", International Journal of Computer Science and Engineering, vol. 3(2), pp. 122-129, 2009.
3. "Evaluation of Object-Oriented Spatial Complexity Measures", ACM SIGSOFT Software Engineering Notes, vol. 34(3), pp. 1-5, May 2009.
4. "Evaluation of Code and Data Spatial Complexity Measures", Communications in Computer and Information Science (Springer), vol. 40, pp. 604–614, 2009.
5. "A Survey of Dynamic Software Metrics", Accepted for publication in Journal of Computer Science and Technology (Springer).
6. "Dynamic Cohesion Measures for Object-Oriented Software" Accepted for publication in Journal of Systems Architecture (Elsevier).

Papers Published in International Conferences

7. "Towards Spatial Complexity Measures for Comprehension of Java programs", in Proceedings of the 14th International conference on Advanced Computing & Communications (ADCOM 2006) IEEE Computer Society Press, 2006, pp. 430-433.
8. "Measurement of Dynamic Metrics using Dynamic Analysis of Programs", in Proceedings of the WSEAS Conference on Applied Computing Conference (ACC 2008), Istanbul, Turkey, May 27-30, 2008, pp. 81-86.

Papers Communicated in International Journals

9. "A Novel Approach for Dynamic Coupling Measurement in Object-Oriented Software" communicated to Information Processing Letters (Elsevier).

10. “Object Level Run-Time Cohesion Measurement” communicated to Journal of Digital Information Management.
11. “A Novel Approach for Package Cohesion Measurement” communicated to Journal of Computer Science and Technology (Springer).

Paper to be communicated in International Journal

12. “Dynamic Analysis using Aspect-Oriented Programming” to be communicated.



SYNOPSIS

Software metrics are used to measure software engineering products (design, source code etc.), processes (analysis, design, coding, testing etc.) and professionals (efficiency or productivity of an individual designer). If used properly, software engineering metrics allow us to quantitatively define the degree of success or failure of a product, process, or person. These can also be used to take meaningful and useful managerial and technical decisions related to cost, effort, time, quality etc. Thus, incorporating metrics into software development process is a valuable step towards creating better systems. The most common approach used for software development is the object-oriented approach and mainly two kinds of software metrics exist for object-oriented software - static software metrics and dynamic software metrics. The static software metrics are obtained from static analysis of the software, whereas dynamic software metrics are computed on the basis of data collected during execution of the software. This thesis presents some new static and dynamic software metrics for the measurement of design and complexity of object-oriented software. The research work reported in this thesis is grouped into three categories:- (i) package level metrics, (ii) dynamic metrics, and (iii) cognitive complexity metrics.

Packages are re-usable components for modern object-oriented systems. A package usually consists of classes, interfaces and sub-packages e.g. Java packages. To promote reuse in object-oriented systems and to make deployment and maintenance tasks easy, packages in object-oriented systems should follow the basic principles of design i.e. maximum cohesion and minimum coupling. Very few metrics exist in literature to measure these design characteristics at package level and are primitive in nature. In the thesis, new metrics for the measurement of package cohesion and package coupling are proposed.

The cohesion of a package can be measured in terms of the degree of intra-package dependencies among its elements (classes, interfaces and sub-packages) present at the same hierarchical level. An attempt has been made in the thesis to measure cohesion of packages in object-oriented software on the basis of relations present between the pairs of package elements. The cohesion of a package is measured as the actual number of relations between all ordered and unique pairs of

the package elements, divided by the maximum number of relations possible between ordered pairs of the package elements. The reason of considering ordered pairs is to take care of the direction of relations between package elements. If, there are n elements in a package, then each element of the package may be at the most connected to all other $n-1$ elements of the package. Thus the maximum possible number of relations among n elements is $n*(n-1)$, which represents the count of maximum possible intra-package dependency for a package. The ratio of the actual number of relations and the maximum possible number of relations provides a normalized value for the proposed measure. The hierarchical structure of packages has also been taken into account during the measurement of package cohesion as the dependency between a class element and a subpackage element of a package is defined in terms of relations between the class element and classes present in the subpackage element of the package at the subsequent hierarchical levels. The proposed cohesion metric has been firstly validated theoretically using standard axiomatic framework given by Briand et al. and then empirically using twenty-five packages taken from six open source software projects developed in Java. The computed values of the proposed cohesion metric for these packages have been found to be strongly correlated with external quality factors such as reusability of the packages.

The coupling between packages is the degree of interdependence between them. Theoretically every package is a stand-alone unit, but in reality packages may depend on each other as either they require services from other packages or provide services to other packages. Thus, coupling between packages cannot be completely avoided but can only be controlled. Coupling between packages is an important factor that affects external quality attributes of software e.g. understandability, maintainability, reusability etc. But still, only a few quantitative studies of the actual use of coupling have been conducted at the package level. In this thesis, new metrics are proposed for the measurement of package level coupling based on formal definitions and properties of the packages in order to achieve good quality software systems. The coupling between two packages is defined as the total number of directed connections between all pairs of their elements, which can be classes or sub-packages. The total coupling of a package at a hierarchical level can be defined as the summation of couplings of the package with all other packages present at the same hierarchical level. The proposed package coupling metrics have been validated

theoretically using standard theoretical framework given by Briand et al. for coupling measures. Then, these metrics have been validated empirically using eighteen packages taken from two open source software systems. The results obtained from this study show strong correlation between package coupling and the effort required to understand a package, which suggest that the proposed package coupling metrics could be used to predict external software quality factors such as understandability.

The conventional static metrics have been found to be inadequate for modern object-oriented software due to improper/insufficient contribution of object-oriented features such as polymorphism, dynamic binding, inheritance and unused code towards their measurement. This fact motivates to focus on dynamic metrics in addition to the traditional static metrics. In this thesis, a few new dynamic metrics are proposed for the measurement of cohesion and coupling at run-time. Moreover, different approaches for the dynamic analysis of programs required for collection of run-time data for the measurement of dynamic metrics are compared. Dynamic analysis of software can be performed in many ways: - using profilers, from dynamic models and using aspect-oriented programming (AOP). Some other less popular techniques like AST rewriting based approach, pre-processor based approach, method-wrappers based approach and hybrid approach can also be used for this purpose. From this study, it is found that AOP approach provides a balanced method for the dynamic analysis of programs. In addition, this approach is easier to implement and at the same time an efficient technique for dynamic analysis without any side effects. In the thesis, AOP approach has been used for the computation of dynamic cohesion and coupling metrics.

Most of the class cohesion measures proposed in literature are static in nature and very few attempts have been made to measure cohesion of a class at run-time. Moreover, the already existing run-time cohesion metrics are nothing but direct extensions of the existing static cohesion metrics. In the thesis, new measures have been defined for dynamic cohesion measurement for objects as well as classes and the proposed dynamic cohesion measures take into consideration the key features of object-orientation like inheritance, polymorphism and dynamic binding. The necessity of dynamic metrics for these features can be understood from the fact that the actual polymorphic invocations can only be determined at run-time. The inclusion of effect of these features in computation of dynamic cohesion makes them more accurate than

the existing static cohesion measures. The method to measure dynamic cohesion is to instrument the source code to log all occurrences of interactions among object-members at run-time, while the application is being executed. The dynamic cohesion of an object is measured on the basis of four types of relationships among elements of an object: - (a) the write dependence of attributes on methods, (b) the read dependence of methods on attributes, (c) the call dependence between methods, and (d) the reference dependence between attributes. In the definitions of the measures, the elements (i.e. attributes and methods) as well as the relationships among the elements of a class are precisely defined. The methods such as constructors, access methods, delegation methods and destructors, which do not contribute to the cohesiveness of objects and classes, are excluded from the cohesion measurement. The dynamic cohesion has been measured first at the object level and then at the class level. Class level cohesion is obtained by aggregating cohesion values of all objects belonging to that class. The proposed measures have been demonstrated to satisfy the four cohesion properties defined by Briand et al. A dynamic analyzer tool has been developed using aspect-oriented programming (Aspectj) to perform dynamic analysis of Java applications for the purpose of collecting run-time data needed for the proposed dynamic cohesion metrics. Using this tool, the empirical values of the proposed metrics have been gathered for source code APIs of Java Development Kit (JDK) and their correlation with the change-proneness of the classes has been investigated in the thesis over different versions of JDK. The proposed metrics are found to be better indicators of change-proneness of classes than the existing cohesion metrics. Moreover, the proposed dynamic cohesion measures are more accurate than their static counterparts as they are defined at run-time and take into consideration the actual interactions taking place rather than the potential interactions. The working of the metrics is such that the scope of the measurement can be limited to a single object or even a scenario. This precision of measurement might be useful in testing and impact analysis whereas other existing cohesion metrics cannot provide such usage, as they measure cohesion up to the class level only.

Just like cohesion, dynamic coupling measurement has also caught little attention of researchers. Existing dynamic coupling metrics take into account only method-method invocations between objects of classes at run-time. In this thesis, new dynamic coupling metrics have been proposed which take into account all major types of relations: - (a) run-time aggregation relations, (b) run-time inheritance relations, (c)

run-time method-attribute reference relations, and (d) run-time method-method invocation relations, between the objects of different classes during measurement. A dynamic coupling tracer application has been developed in Aspectj for the purpose of computation of the proposed metrics. With the help of this tracer application, the proposed metrics have been validated empirically using six open source projects developed in Java. In the empirical study, it has been found that average dynamic coupling per class (CDC/NOC) is a useful indicator of the degree of dynamic coupling for a project. The study has further shown that there is no correlation between the number of classes (NOC) and amount of dynamic coupling (CDC) in a project. Thus, number of classes in a project cannot predict the amount of dynamic coupling in a project.

In addition to coupling and cohesion, another important dimension of software quality is its complexity. The software can have better testability, readability and maintainability if it possesses low coupling, high cohesion and less complexity. There are many facets of complexity. Some target the control flow complexity, few concentrate on operators and operands and others focus on measuring comprehension efforts on the basis of spatial and cognitive complexity. Spatial complexity metrics indicate the difficulty of understanding the logic of the program in terms of lines of code that the reader is required to traverse to follow control or data dependencies as they build a mental model. Spatial complexity of object-oriented software is the combination of class spatial complexity and object spatial complexity. The spatial complexity metrics are more difficult to trust and validate due to less-understood effect of human factors and the human mind's working towards comprehension of the programs. In order to improve confidence in some of such metrics, there are certain necessary properties and evaluation criteria widely used by many researchers that software complexity metrics should satisfy. In the thesis, Weyuker's properties and Briand et al. criteria are used for the evaluation of the object-oriented spatial complexity metrics. These metrics are found to satisfy all nine properties given by Weyuker and all five complexity properties required by Briand et al. evaluation criteria. The object-oriented spatial complexity measures proposed in literature were formulated by keeping C++ language in mind, and there were no spatial complexity measures available for the Java language. Keeping in view the increasing popularity of Java, this thesis attempts to define the spatial complexity measures for Java

programs. Although both C++ and Java are object-oriented languages, still in many aspects Java programs completely differ from C++ programs e.g. way of defining a method. The spatial metrics proposed for Java in the thesis have been formulated in a way so as to handle all such differences. As programming in Java revolves around definitions of classes and objects, comprehension of a Java program requires understanding of Java classes as well as objects. Thus, spatial complexity of Java code is unification of spatial complexities of classes and objects. In the thesis, two categories of spatial complexity measures for Java programs are proposed. These metrics are termed as Spatial Complexity of Java Class (SCJC) and Spatial Complexity of Java Object (SCJO). These metrics have been computed for ten different projects developed using Java. The results indicate that the proposed spatial complexity metrics influence the comprehension process of Java programs to a great extent and understandability of Java programs does not depend on their size only but also on the values of SCJC and SCJO. Lower values of Spatial Complexity of Java Class (SCJC) and Spatial Complexity of Java Object (SCJO) are desirable for better understandability of Java programs.

The understandability of object-oriented software has been reported in literature to be dependent on architectural aspects as well along with spatial distances. The architectural aspect of software has been reflected in cognitive complexity with the help of using weights of various types of Basic Control Structures (BCS) present in the source code. The measures of cognitive complexity proposed by various authors considered only these weights, which were reflection of architectural viewpoint only and did not look into the spatial aspect at all. On the other hand, the importance of spatial distance towards complexity has already been established in literature. Thus there is need to combine the impact of architectural as well as spatial aspects of the software to compute the cognitive complexity. This type of software complexity has been termed as cognitive-spatial complexity of the software in the thesis and new cognitive-spatial complexity measures are proposed for object-oriented software. The proposed measures take spatial as well as architectural complexity of the software into account for estimation of the cognitive effort required for the software comprehension process. The spatial complexity has been taken into consideration using the lexical distances (in LOC) between different program elements and architectural complexity of the software has been taken into account using the cognitive weights of various control structures present in the control flow of

the program. The proposed measures have been evaluated against the standard axiomatic frameworks given by Weyuker and Briand et al. to prove their usefulness. Further, the proposed measures have been compared with the existing cognitive and spatial complexity measures for object-oriented software. This comparative study has shown that the proposed measures are better indicators of the cognitive effort required for program comprehension than the corresponding existing cognitive and spatial complexity measures.



CONCLUSIONS AND FUTURE SCOPE

Conclusions

In this thesis, new static and dynamic metrics have been proposed, evaluated and validated for measurement of cohesion, coupling and complexity for object-oriented software. Major contributions of the work reported in this thesis are:

- New metrics for the measurement of package cohesion and package coupling have been proposed and validated.
- Aspect-oriented approach of collection of run-time data has been found to be better than other approaches for this purpose.
- New dynamic cohesion and coupling metrics have been proposed and validated.
- A dynamic analyzer tool has been developed using aspect-oriented programming (Aspectj) to perform dynamic analysis of Java applications.
- New spatial complexity measures for the estimation of comprehensibility of Java programs have been defined.
- New object-oriented cognitive-spatial complexity metrics have been proposed and evaluated.

Future Scope

During working in this area of research, a lot of scope for future work has been observed. There is need of further empirical investigations of the proposed package level metrics in order to establish their relations with other external software quality factors such as maintainability. As dynamic metrics have the advantage of being more precise, but they are difficult to compute in comparison to static ones. Thus, there is clear opportunity for researchers in hybrid approach where the dynamic analysis results can be augmented by static information for collection of metrics data. This hybrid approach can combine static as well as dynamic approaches to make the measurement process more cost-effective. In future work in the area of dynamic cohesion metrics, relationships between the proposed dynamic cohesion metrics and other quality attributes such as fault-proneness could be explored. Moreover, in

future, an alternative approach for selection of the values of the weights assigned to different relations could be explored to use the dynamic cohesion metrics in predictive models and estimate coefficients of the predictive models, which could be used as weights. Further, the impact of the dynamic coupling metrics on the external attributes of software applications needs to be explored.

The cognitive complexity measures proposed in the thesis can be extended to take into account the semantic aspect of the code. There is also an opportunity to study and integrate the impact of special features such as exception handling and dynamic polymorphism etc. into the cognitive metrics. Even the possibility of proposing dynamic cognitive metrics can also be explored.



REFERENCES

1. F.B. Abreu, "The MOOD Metrics Set", *Proc. Ninth European Conf. Object-Oriented Programming (ECOOP'95) Workshop on Metrics*, Aarhus, Denmark, 1995.
2. F.B. Abreu, M. Goulão, and R. Esteves, "Toward the Design Quality Evaluation of Object-Oriented Software Systems", *Proc. 5th International Conference on Software Quality*, Austin, Texas, USA, October 1995.
3. F.B. Abreu, G. Pereira, and P. Sousa, "A Coupling-Guided Cluster Analysis Approach to Reengineer the Modularity of Object-Oriented Systems", *Proc. 4th European Conference on Software Maintenance and Reengineering (CSMR'2000)*, Zurich, Switzerland, 2000, pp. 13.
4. F.B. Abreu, and M. Goulão, "A Merit Factor Driven Approach to the Modularization of Object-Oriented Systems", *L'Objet*, vol. 7, no. 4, pp. 1-23, 2001.
5. K.K. Aggarwal, Y. Singh, and J.K. Chhabra, "A Dynamic Software Metric and Debugging Tool", *ACM SIGSOFT Software Engineering Notes*, vol. 28, no. 2, pp. 1-4, 2003.
6. K.K. Aggarwal, Y. Singh, and J.K. Chhabra, "Complete Dependency Matrix for Object-Oriented Software", *International Journal of Management and System (IJOMAS)*, vol. 19, no.1, pp. 43-54, 2003.
7. R.L. Akers, "Using Build Process Intervention to Accommodate Dynamic Instrumentation of Complex Systems", *Proc. 1st International Workshop on Program Comprehension through Dynamic Analysis*, pp. 1-5, 2005.
8. E. Allen and T. Khoshgoftaar, "Measuring Coupling and Cohesion, An Information-Theory Approach", *Proc. IEEE International Symposium on Software Metrics*, Boca Raton, Fla, pp. 119-127, 1999.
9. H. Aman, T. Yanaru, M. Nagamatsu, and K. Miyamoto, "A Metric for Class Structure Complexity Focusing on Relationships among Class Members", *IEICE Transactions on Information and System*, vol. E81-D, no.12, pp. 1364-1373, 1998.

10. H.H. Ammar, T. Nikzadeh, and J. Dugan. "A Methodology for Risk Assessment of Functional Specification of Software Systems using Coherent Petri Nets", *Proc. Fourth International Software Metrics Symposium, Metrics'97*, Albuquerque, New Mexico, pp.108-117, 1997.
11. The Apache Jakarta Project, available at <http://jakarta.apache.org>
12. E. Arisholm, D.I.K. Sjøberg, M. Jørgensen, "Assessing the Changeability of Two Object-Oriented Design Alternatives– A Controlled Experiment", *Empirical Software Engineering*, vol. 6, pp. 231–277, 2001.
13. E. Arisholm, *Empirical Assessment of Changeability in Object-Oriented Software*, PhD Thesis, University of Oslo. 2001.
14. E. Arisholm, "Dynamic Coupling Measures for Object-Oriented Software", *Proc. Eighth IEEE Symposium Software Metrics (METRICS '02)*, pp. 33-42, 2002.
15. E. Arisholm, L.C. Briand, A. Foyen, "Dynamic Coupling Measures for Object-Oriented Software", *IEEE Transactions on Software Engineering*, vol. 30, no. 8, pp. 491–506, 2004.
16. AspectJ < <http://www.eclipse.org/aspectj> >
17. C.S. Atole and K.V. Kale, "Assessment of Package Cohesion and Coupling Principles for Predicting the Quality of Object Oriented Design", *Proc. 1st International Conference on Digital Information Management*, Bangalore, India, pp. 1-5, Dec. 2006.
18. A.L. Baker and S.H. Zweben, "A Comparison of Measures of Control Flow Complexity", *IEEE Transactions on Software Engineering*, vol. SE–6, no. 6, pp. 506-511, 1980.
19. T. Ball and J.R. Larus, "Using Paths to Measure, Explain and Enhance Program Behavior", *IEEE Computer*, vol. 33, no. 7, pp. 57-65, July 2000.
20. J. Bansiya, L.H. Etzkorn, C.G. Davis, and W. Li, "A Class Cohesion Metric for Object-Oriented Designs", *Journal of Object-Oriented Programming*, vol. 11, no. 8, pp. 47-52, 1999.
21. V.R. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data", *IEEE Transactions of Software Engineering*, vol. 10, no. 11, pp. 728-738, 1984.

22. V.R. Basili and D.H. Rombach, "The Tame Project: Towards Improvement-Oriented Software Environments", *IEEE Transactions of Software Engineering*, vol. 14, no. 6, pp. 758-773, 1988.
23. V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", *IEEE Transactions on Software Engineering*, vol. 22, no. 10, pp. 751-761, 1996.
24. Byte Code Engineering Library (BCEL), Retrieved from <http://jakarta.apache.org/bcel/index.html>
25. R.K. Bhatia, M. Dave, and R.C. Joshi, "Ant Colony Based Rule Generation for Reusable Software Component Retrieval", *ACM SIGSOFT Software Engineering Notes*, vol. 35, no. 2, pp. 1-5, March 2010.
26. J.M. Bieman and L.M. Ott, "Measuring Functional Cohesion", *IEEE Transactions on Software Engineering*, vol. 20, no. 8, pp. 644-657, 1994.
27. J.M. Bieman and B.K. Kang, "Cohesion and Reuse in an Object-Oriented System", *Proc. ACM Symposium on Software Reusability (SSR'95)*, Seattle, WA, 1995, pp. 259-262. Reprinted in *ACM SIGSOFT Software Engineering Notes*, 1995.
28. G. Booch, *Object Oriented Analysis and Design with Applications*, 2nd ed., Benjamin Cummings Publishing Company Inc., Redwood City, LA, USA, 1994.
29. G. Booch, J. Rumbaugh, and I. Jacobson, *The Unified Modeling Language Users Guide*, Addison-Wesley, 1998.
30. J. Brant, B. Foote, R. E. Johnson, and D. Roberts, "Wrappers to the Rescue", *Proc. ECOOP '98*, vol. 1445, LNCS, pp. 396-417, Springer-Verlag, 1998.
31. L. Briand, S. Morasca, and V. Basili, "Measuring and Assessing Maintainability at the End of High-Level Design", *IEEE Conference on Software Maintenance*, Montreal, Canada, pp. 88 - 97, September 1993.
32. L. Briand, S. Morasca, and V. Basili, "Defining and Validating High-Level Design Metrics", *Technical Report*, University of Maryland, CS-TR 3301, 1994.
33. L.C. Briand, S. Morasca, and V. Basili, "Property based Software Engineering Measurement", *IEEE Transactions of Software Engineering*, vol. 22, no. 1, pp. 68-86, 1996.

34. L.C. Briand, P. Devanbu, and W. Melo, "An Investigation into Coupling Measures for C++", *Proc. 19th International Conference on Software Engineering (ICSE 97)*, Boston, USA, pp. 412-421, 1997.
35. L.C. Briand, J.W. Daly, and J. Wust, "A Unified Framework for Cohesion Measurement in Object-Oriented Systems", *Empirical Software Engineering: An International Journal*, vol. 3, no. 1, pp. 65-117, 1998.
36. L.C. Briand, J.W. Daly, and J. Wust, "A Unified Framework for Coupling Measurement in Object-Oriented Systems", *IEEE Transactions on Software Engineering*, vol. 25, no. 1, pp. 91-121, Jan/Feb 1999.
37. L.C. Briand, S. Morasca, and V.R. Basili. "Defining and Validating Measures for Object-Based High-Level Design", *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 722-743, 1999.
38. L. Briand, S. Morasca, and V. Basili, "An Operational Process for Goal-Driven Definition of Measures", *IEEE Transactions on Software Engineering*, vol. 28, no. 12, pp. 1106-1125, 2002.
39. Bean Scripting Framework (BSF) <<http://jakarta.apache.org/bsf/index.html>>
40. R. Buschermöhle and W. Hasselbring, "Taking Advantage of the Symbiotic Relationship between Tools and Processes to Support Executable Processes Models", *Proc. Software Engineering Research and Practice*, pp. 279-285, 2003.
41. CaffeineMark <<http://www.benchmarkhq.ru/cm30/info.html>>
42. H.S. Chae and Y.R. Kwon. "A Cohesion Measure for Classes in Object-Oriented Systems", *Proc. Fifth International Software Metrics Symposium*, IEEE Computer Society Press, Bethesda, MD, USA, pp. 158-166, 1998.
43. H.S. Chae, Y.R. Kwon, and D.H. Bae. "A Cohesion Measure for Object-Oriented Classes", *Software Practice & Experience*, vol. 30, no. 12, pp. 1405-1431, 2000.
44. H.S. Chae, Y.R. Kwon, and D.H. Bae, "Improving Cohesion Metrics for Classes by Considering Dependent Instance Variables", *IEEE Transactions on Software Engineering*, vol. 30, no. 11, pp. 826-832, Nov 2004.

45. Z. Chen, Y. Zhou, B. Xu, J. Zhao, and H. Yang, "A Novel Approach to Measuring Class Cohesion Based on Dependence Analysis", *Proc. International Conference on Software Maintenance*, IEEE Computer Society Press, Montreal, Canada, pp. 377-384, 2002.
46. J.K. Chhabra, K.K. Aggarwal, and Y. Singh: "Code and Data Spatial Complexity: Two Important Software Understandability Measures", *Information and Software Technology*, vol. 45, no.8, pp. 539-546, 2003.
47. J.K. Chhabra, K.K. Aggarwal, and Y. Singh, "Measuring Maintainability of Object-Oriented Software", *International Journal of Management and Systems (IJOMAS)*, vol. 20, no. 2, pp. 139-156, May-Aug 2004.
48. J.K. Chhabra, K.K. Aggarwal, and Y. Singh: "Measurement of Object Oriented Software Spatial Complexity", *Information and Software Technology*, vol. 46, no. 10, pp. 689-699, 2004.
49. J.K. Chhabra, K.K. Aggarwal, and Y. Singh, "A Unified Measure of Complexity of Object-Oriented Software", *Journal of the Computer Society of India*, vol. 34, no.3, pp. 2-13, 2004.
50. S.R. Chidamber and C.F. Kemerer, "Towards a Metrics Suite for Object Oriented Design", *Proc. 6th ACM Conf. Object-Oriented Programming: Systems, Languages and Applications (OOPSLA)*, Phoenix, AZ, pp. 197-211, Oct. 1991.
51. S.R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object-Oriented Design", *IEEE Transactions on Software Engineering*, vol. 20, no. 6, pp. 476-493, 1994.
52. E.S. Cho, C.J. Kim, S.D. Kim, and S.Y. Rhew, "Static and Dynamic Metrics for Effective Object Clustering", *Proc. Asia-Pacific Software Engineering Conference*, Taiwan, IEEE Computer Society, pp. 78-85, 1998.
53. N.I. Churcher and M.J. Shepperd, "Comments on A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering*, vol. 21, no. 3, pp. 263-264, 1995.
54. J. Cleland-Hunang, C.K. Chang, H. Kim, and A. Balakrishnan, "Requirements-Based Dynamic Metric in Object-Oriented Systems", *Proc. IEEE International Symposium on Requirements Engineering (RE'01)*, pp. 212-219, 2001.
55. P. Coad and E. Yourdon, *Object-Oriented Analysis*, second edition. Prentice Hall, 1991.
56. P. Coad and E. Yourdon, *Object-Oriented Design*, first edition. Prentice Hall, 1991.

57. J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. Lawrence Erlbaum Publishing Co, Mahwah, N.J., 1988.
58. S.D. Conte, H.E. Dunsmore, and V.Y. Shen, "Software Engineering Metrics and Models", Benjamin-Cummings, Publishing Co., Inc., Redwood City, CA, USA, 1986.
59. S. Counsell and S. Swift, "The Interpretation and Utility of Three Cohesion Metrics for Object-Oriented Design", *ACM Transactions on Software Engineering and Methodology*, vol. 15, no. 2, pp. 123–149, 2006.
60. T. DeMarco, *Controlling Software Projects*, Yourdon Press, New York, USA, 1982.
61. C.R. Douce, P.J. Layzell, and J. Buckley, "Spatial Measures of Software Complexity," *Proc. 11th Meeting of Psychology of Programming Interest Group*, 1999. <http://www.ppig.org/workshops/11th-programme.html>
62. B. Dufour, K. Driesen, L.J. Hendren, and C. Verbrugge, "Dynamic Metrics for Java", *Proc. Conference on Object-Oriented Programming Systems, Languages and Applications*, Anaheim, CA, USA, pp. 149–168, 2003.
63. Element Construction Set (ECS) <<http://jakarta.apache.org/ecs/index.html>>
64. J. Eder, G. Kappel, and M. Schrefl, "Coupling and Cohesion in Object-Oriented Systems", *Technical Report*, University of Klagenfurt, 1994.
65. J. Engel, *Programming for the Java Virtual Machine*, Addison-Wesley, 1999.
66. L.H. Etzkorn, C.G. Davis, W. Li, "A Practical Look at the Lack of Cohesion in Methods Metric", *Journal of Object-Oriented Programming*, vol. 11, no. 5, pp. 27–34, 1998.
67. N. Fenton and M. Neil, "Software Metrics: Successes Failures and New Directions", *Journal of Systems and Software*, vol. 47, no. 2-3, pp. 149-157, 1999.
68. X. Franch and J.P. Carvalho, "A Quality-Model-Based Approach for Describing and Evaluating Software Packages", *Proc. IEEE Joint International Conference on Requirements Engineering (RE'02)*, pp. 1-8, 2002.
69. X. Franch and N.A.M. Maiden. "Modeling Component Dependencies to Inform Their Selection", *Lecture Notes in Computer Science*, vol. 2580, H. Erdogmus and T. Weng (Eds.): ICCBSS 2003, Springer-Verlag Berlin Heidelberg, pp. 81–91, 2003.
70. N.E. Gold and P.J. Layzell, "Spatial Complexity Metrics: an Investigation of Utility", *IEEE Transactions on Software Engineering*, vol. 3, no. 1, pp. 203-212, 2005.

71. J. Gosling and F. Yellin, *The Java Application Programming Interface, Vol. #1 (Core Packages) / #2 (Window Toolkit and Applets)*, Reading, Addison-Wesley, Massachusetts, USA, 1996.
72. R.G. Gowda and L.E. Winslow. "An Approach for Deriving Object-Oriented Metrics", *Proc. IEEE National Aerospace and Electronics Conference (NAECON)*, pp. 897-904, 1994.
73. I.M. Graham. "Making Progress in Metrics", *Object Magazine*, vol. 6, no. 8, pp. 68-73, October 1996.
74. O. Greevy and S. Ducasse, "Correlating Features and Code using a Compact Two-Sided Trace Analysis Approach", *Proc. 9th European Conference on Software Maintenance and Reengineering (CSMR 2005)*, IEEE Computer Society, pp.314–323, 2005.
75. L. Grunske and B. Kaiser, "An Automated Dependability Analysis Method for COTS-Based Systems", *Lecture Notes in Computer Science (LNCS)*, vol. 3412, X. Franch and D. Port (Eds.): ICCBSS 2005, Springer-Verlag Berlin Heidelberg, pp. 178-190, 2005.
76. G. Gui and P.D. Scott, "Coupling and Cohesion Measures for Evaluation of Component Reusability", *Proc. International Workshop on Mining Software Repositories*, Shanghai, China, pp. 18-21, 2006.
77. G. Gui, "Component Reusability and Cohesion Measures in Object-Oriented Systems", *Proc. Information and Communication Technologies*, pp. 2878-2882, 2006.
78. R. Gunnalan, M. Shereshevsky, and H.H. Ammar, "Pseudo Dynamic Metrics", *Proc. third ACS/IEEE International Conference on Computer Systems and Applications*, pp. 117-vii, 2005.
79. N. Gupta and P. Rao, "Program Execution based Module Cohesion Measurement," *Proc. 16th International Conference on Automated on Software Engineering (ASE '01)*, San Diego, USA, pp. 144-153, November 2001.
80. M.H. Halstead, *Elements of Software Science (Operating and Programming Systems Series)*, Elsevier Science Inc. New York, NY, USA, 1977.
81. W. Harrison, "An Entropy-based Measure of Software Complexity", *IEEE Transactions on Software Engineering*, vol. 18, no. 11, pp. 1025-1029, 1992.

82. R. Harrison, S. Counsell, and R. Nithi, "Coupling Metrics for Object-Oriented Design", *Proc. 5th International Software Metrics Symposium Metrics*, pp. 150-156, 1998.
83. R. Harrison, S.J. Counsell, and R.V. Nithi, "An Evaluation of the MOOD Set of Object-Oriented Metrics", *IEEE Transactions of Software Engineering*, vol. 24, no. 6, pp. 491-496, June 1998.
84. Y. Hassoun, R. Johnson, and S. Counsell, "Empirical Validation of a Dynamic Coupling Metric", Technical Report BBKCS-04-03, School of Computer Science and Information Systems, Birkbeck College, University of London, UK, March, 2004.
85. Y. Hassoun, R. Johnson, and S. Counsell, "A Dynamic Runtime Coupling Metric for Meta-level Architectures", *Proc. European Conference on Software Maintenance and Reengineering (CSMR'04)*, IEEE Computer Society Press, pp.339-346, 2004.
86. Y. Hassoun, R. Johnson, and S. Counsell, "Dynamic Coupling Metric-Proof of Concept", *IEE Proc.-Software*, vol. 152, no. 6, 2005.
87. E. Hautus, "Improving Java Software through Package Structure Analysis" *Proc. International Conference on Software Engineering and Applications*, 2002.
88. B. Henderson-Sellers, *Software Metrics*, Prentice Hall, Hemel Hempstead, U.K., 1996.
89. B. Henderson-Sellers, *Object-Oriented Metrics—Measures of Complexity*, Prentice Hall PTR, Upper Saddle River, New Jersey, 1996.
90. M. Hitz and B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Systems", *Proc. Third International Symposium on Applied Corporate Computing (ISACC'95)*, Monterrey, Mexico, pp. 25-27, 1995.
91. M. Hitz and B. Montazeri, "Measuring Coupling in Object-Oriented Systems", *Object Currents*, vol. 1, no. 4, pp. 124-136, 1996.
92. M. Hitz and B. Montazeri, "Chidamber & Kemerer's Metrics Suite: A Measurement Theory Perspective", *IEEE Transactions on Software Engineering*, vol. 22, no. 4, pp. 276-270, April 1996.
93. ISO/IEC 9126-1, Software Product Quality, Part 1: Quality Model, 1991.
94. Jakarta Servlet <<http://cvs.apache.org/viewcvs.cgi/jakarta-servletapi-5/>>
95. Java Grande Forum Benchmark Suite (JGFBS)
<<http://www.epcc.ed.ac.uk/research/javagrande/benchmarking.html>>

96. JPDA <<http://java.sun.com/products/jpda>>
97. JUnit <<http://sourceforge.net/projects/junit/>>
98. H. Kabaili, R. Keller, and F. Lustman, "Cohesion as Changeability Indicator in Object-Oriented Systems", *Proc. IEEE Conference on Software Maintenance and Reengineering (CSRM)*, pp. 39-46, 2001.
99. B.K. Kang and J.M. Bieman, "Design-Level Cohesion Measures: Derivation, Comparison, and Applications", *Proc. 20th Computer Software and Applications Conference*, IEEE Computer Society, Seoul, Korea, pp. 92-97, August 1996.
100. T.M. Khoshgoftaar, J.C. Munson, and D.L. Lanning, "Dynamic System Complexity", *Proc. Software Metrics Symposium*, Baltimore, MD, USA, pp. 129-140, 1993.
101. J.M. Kleinberg, "Authoritative Sources in a Hyperlinked Environment", *Journal of the ACM*, vol. 46, no. 5, pp. 604-632, 1999.
102. C.R. Kothari, *Research Methodology: Methods & Techniques*, revised second ed., New Age International Publishers, New Delhi, 2007.
103. D.S. Kushwaha and A.K. Misra, "Robustness Analysis of Cognitive Information Complexity Measure using Weyuker's Properties", *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 1, pp. 1-6, 2006.
104. S.T. Lai, "A Software Metric Combination Model for Software Reuse", *Proc. Asia Pacific Software Engineering Conference*, Taipei, pp. 70-77, 1998.
105. K.B. Lakshmanian, S. Jayaprakash, and P.K. Sinha, "Properties of Control-Flow Complexity Measures", *IEEE Transactions on Software Engineering*, vol. 17, no. 2, pp. 1289-1295, 1991.
106. 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", *Proc. International Conference on Software Quality*, Maribor, Slovenia, pp. 81-90, 1995.
107. J.K. Lee, S.J. Seung, S.D. Kim, W. Hyun, and D.H. Han, "Component Identification Method with Coupling and Cohesion", *Proc. Eighth Asia-Pacific on Software Engineering Conference (APSEC'01)*, pp. 79-79, 2001.
108. S. Lewis, *The Art and Science of Smalltalk*, Prentice-Hall, 1995.

109. W. Li and S. Henry, "Object Oriented Metrics that Predict Maintainability", *Journal of Systems and Software*, vol. 23, no. 2, pp. 111-122, 1993.
110. W. Li and S. Henry, "Maintenance Metrics for the Object-Oriented Paradigm", *Proc. first International Software Metrics Symposium*, pp. 52-60, 1993.
111. W. Li, S. Henry, D. Kafura, and R. Schulman, "Measuring Object-Oriented Design", *Journal of Object-Oriented Programming*, vol. 8, no. 4, pp. 48-55, 1995.
112. W. Li, "Another Metric Suite for Object-Oriented Programming", *Journal of Systems and Software*, vol. 44, pp. 155-162, 1998.
113. K.J. Lieberherr, D.H. Lorenz, and M. Mezini, "Building Modular Object-Oriented Systems with Reusable Collaborations", *Proc. International Conference on Software Engineering*, Limerick Ireland, pp. 821-821, 2000.
114. M. Lorenz and J. Kidd, *Object-Oriented Software Metrics: A Practical Guide*, Prentice Hall, Englewood Cliffs, New Jersey, 1994.
115. S. Mancoridis, B.S. Mitchell, Y. Chen, and E.R. Gansner, "Bunch: A Clustering Tool for the Recovery and Maintenance of Software System Structures", *Proc. International Conference on Software Maintenance*, IEEE Computer Society Press, Oxford, England, 1999.
116. R. Martin, "OO Design Quality Metrics-An Analysis of Dependencies", Position Paper, *Proc. Workshop on Pragmatic and Theoretical Directions in Object-Oriented Software Metrics (OOPSLA'94)*, Oct. 1994.
117. R. Martin, "Object Oriented Design Quality Metrics: An Analysis of Dependencies", *ROAD*, vol. 2, no. 3, 1995.
118. R. Martin, *Agile Software Development, Principles, Patterns, and Practices*, Prentice Hall, 2002.
119. T.J. McCabe: "A Complexity Measure", *IEEE Transactions on Software Engineering*, vol. SE-2, no. 4, pp. 308-320, 1976.
120. V.B. Mišić, "Cohesion is Structural, Coherence is Functional: Different Views, Different Measures", *Proc. Seventh International Software Metrics Symposium (METRICS-01)*, pp. 135-144, 2001.

121. S. Misra and A.K. Misra, "Evaluating Cognitive Complexity Measure with Weyuker Properties", *Proc. third IEEE International Conference on Cognitive Informatics (ICCI2004)*, pp. 103-108, 2004.
122. S. Misra, "Modified Cognitive Complexity Measure", *21st ISCIS'06, LNCS*, vol. 4263, pp. 1050-59, 2006.
123. S. Misra, "A Complexity Measure based on Cognitive Weights", *Proc. International Journal of Theoretical and Applied Computer Science*, vol.1, no.1, pp. 1-10, 2006.
124. S. Misra and A.K. Misra, "Evaluation and Comparison of Cognitive Complexity Measure", *ACM SIGSOFT Software Engineering Notes*, vol. 32, no. 2, pp. 1-5, 2007.
125. S. Misra, "Validating Modified Cognitive Complexity Measure", *ACM SIGSOFT Software Engineering Notes*, vol. 32, no. 3, pp. 1-5, 2007.
126. D. Mishra and A. Mishra, "Object-Oriented Inheritance Metrics: Cognitive Complexity Perspective", *Lecture Notes in Computer Science (LNCS)*, Springer Berlin/Heidelberg, vol. 5589, 2009.
127. A. Mitchell and J.F. Power, "Run-Time Cohesion Metrics for the Analysis of Java Programs", *Technical Report Series no. NUIM-CS-TR-2003-08*, National University of Ireland, Maynooth, Co. Kildare, Ireland, 2003.
128. A. Mitchell and J.F. Power, "Toward a Definition of Run-Time Object-Oriented Metrics", *Proc. 7th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, Darmstadt, Germany, pp. 1-7, 2003.
129. A. Mitchell and J.F. Power, "Run-Time Cohesion Metrics: An Empirical Investigation", *Proc. International Conference on Software Engineering Research and Practice (SERP'04)*, Las Vegas, Nevada, June 21-24, pp. 532-537, 2004.
130. A. Mitchell and J.F. Power, "An Empirical Investigation into the Dimensions of Run-Time Coupling in Java Programs", *Proc. 3rd Conference on the Principles and Practice of Programming in Java*, Las Vegas, Nevada, pp. 9-14, 2004.
131. A. Mitchell and J.F. Power, "An Approach to Quantifying the Run-Time Behaviour of Java GUI Applications", *Proc. Winter International Symposium on Information and Communication Technologies*, Cancun, Mexico, pp. 1-6, 2004.

132. A. Mitchell and J.F. Power, "Using Object-Level Run-Time Metrics to Study Coupling between Objects", *Proc. ACM Symposium on Applied Computing*, Santa Fe, NM, pp. 1456–1462, 2005.
133. A. Mitchell and J.F. Power, "A Study of the Influence of Coverage on the Relationship between Static and Dynamic Coupling Metrics". *Science of Computer Programming*, vol. 59, pp. 4–25, 2006.
134. K.L. Morris, *Metrics for Object-Oriented Software development Environments*, unpublished Master Thesis, M.I.T. Cambridge, MA, 1988.
135. S. Moser, V. B. Mistic, "Measuring Class Coupling and Cohesion: A Formal Meta model Approach", *Proc. Asia Pacific Software Engineering Conference and International Computer Science Conference*, Hong Kong, IEEE Computer Society Press, pp. 31-40, 1997.
136. J.C. Munson and T.M. Khoshgoftaar, "Measuring Dynamic Program Complexity", *IEEE Software*, vol. 9, no. 6, pp. 48-55, 1992.
137. J.C. Munson and T.M. Khoshgoftaar, *Software Metrics for Reliability Assessment, Handbook of Software Reliability Engineering*, Michael Lyu (edt.), McGraw-Hill, pp. 493-529, 1996.
138. J.D. Musa, A. Iannino, K. Okumoto, *Software Reliability: Measurement, Prediction, Application*, New York: McGraw-Hill, Inc., 1987.
139. V.L. Narasimhan and B. Hendradjaya, "Some Theoretical Considerations for a Suite of Metrics for the Integration of Software Components", *Information Sciences*, vol. 177, no. 3, pp. 844–864, 2007.
140. P. Naughton and H. Schildt, *The Complete Reference Java 2*, 3rd edition, McGraw-Hill Companies Inc., New York, 2000.
141. M. Northcott and M. Vigder, "Managing Dependencies between Software Products", *Lecture Notes in Computer Science (LNCS)*, vol. 3412, X. Franch and D. Port (Eds.): ICCBSS 2005, Springer Berlin Heidelberg, pp. 201-211, 2005.
142. M.P. O'Brien and J. Buckley: "Inference-Based and Expectation based Processing in Program Comprehension", *Proc. Ninth IEEE Int'l Workshop Program Comprehension*, pp. 71-78, 2001.

143. F. Ohata, K. Hirose, M. Fujii, and K. Inoue: "A Slicing Method for Object-Oriented Programs Using Lightweight Dynamic Information", *Proc. 8th Asia Pacific Software Engineering Conference*, pp. 273-280, 2001.
144. Object Management Group (OMG), "UML™ Resource Page" available at: <http://www.omg.org/uml/>
145. Jakarta-ORO <<http://jakarta.apache.org/oro/index.html>>
146. L.M. Ott, J.M. Bieman, B.K. Kang, and B. Mehra, "Developing Measures of Class Cohesion for Object-oriented Software" *Proc. 7th Annual Oregon Workshop on Software Metrics*, Oregon, Portland, 1995.
147. L.M. Ott and J.M. Bieman, "Program Slices as an Abstraction for Cohesion Measurement", *Journal of Information and Software Technology*, vol. 40, no. 11-12, pp. 691-699, 1998.
148. S. Patel, W.C. Chu, and R. Baxter, "A Measure for Composite Module Cohesion", *Proc. 14th International Conference on Software Engineering*, pp. 38-48, 1992.
149. H. Pham and X. Zhang, "A Software Cost Model with Warranty and Risk Costs", *IEEE Transactions on Computers*, vol. 48, no. 1, pp. 71-75, Jan 1999.
150. L. Ponisio and O. Nierstrasz, "Using Contextual Information to Assess Package Cohesion", *Technical Report No. IAM-06-002, 2006*, Institute of Applied Mathematics and Computer Sciences, University of Berne, 2006.
151. C. Rajaraman and M.R. Liu, "Some Coupling Measures for C++ programs", *Proc. Eighth international conference on Technology of object oriented languages and systems*, pp. 225-234, Santa Barbara, California, United States, 1992.
152. C. Rajaraman and M.R. Liu, "Reliability and Maintainability Related Software Coupling Metrics in C++ Programs", *Proc. 3rd IEEE Int. Symposium on Software Reliability Engineering (ISSRE'92)*, Research Triangle Park, NC, pp. 303-311, October 8-9, 1992,
153. RIAC Document: Procedures for Performing Failure Mode Effects and Criticality Analysis. US MIL-STD-1629 Nov 1974, US MIL-STDJ629A Nov 1980, US MIL_STD_1629A/Notice 2 Nov 1984.
154. S. Sabharwal, *Software Engineering Principles, Tools and Techniques*, Umesh Publications, Delhi, 2005.

155. B. Selic, G. Gullekson, and P. Ward, *Real-Time Object Oriented Modeling*, John Wiley & Sons, Inc., 1994
156. O. Seng, M. Bauer M, Biehl, and G. Pache, "Search-based Improvement of Subsystem Decompositions", *Proc. Genetic and Evolutionary Computation Conference (GECCO'05)*, Washington, DC, USA, pp. 1045-1051, 25–29 June 2005.
157. S.D. Sheetz, D.P. Tegarden, D.E. Monarchi, "Measuring Object-Oriented System Complexity", *Proc. First Workshop on Information Technologies and Systems*, pp. 285-307, Dec 1991.
158. Y. Singh: *Metrics and Design Techniques for Reliable Software*, PhD Thesis, Kurukshetra University, Kurukshetra, 1995.
159. R.K. Singh, P. Chandra, and Y. Singh, "An Evaluation of Boolean Expression Testing Techniques", *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 5, pp. 1-6, 2006.
160. R. V Solingen, "The Goal/Question/Metric Approach", *Encyclopedia of Software Engineering—2 Volume Set*, pp. 578-583, 2002.
161. SpecJVM98 <<http://www.spec.org/org/jvm98>>
162. W. Stevens, G. Myers, and L. Constantine, "Structured Design", *IBM Systems Journal*, vol. 13, no. 2, pp. 115-139, 1974.
163. B. Stroustrup, *The C++ Programming Language*, Third Edition, Addison-Wesley Publishing Company, Massachusetts, USA, 1997.
164. C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, ACM Press / Addison-Wesley, New York, 1998.
165. N. Tagoug, "Object-Oriented System Decomposition Quality" *Proc. 7th IEEE International Symposium on High Assurance Systems Engineering*, pp. 230-235, 2002.
166. Tapestry < <http://linux.cs.lewisu.edu/apache/jakarta/tapestry/source/3.0-beta-1/>>
167. D.P. Tegarden, S.D. Sheetz, and D.E. Monarchi, "The Effectiveness of Traditional Metrics for Object-Oriented Systems", *Proc. Twenty-Fifth Hawaii International Conference on System Sciences*, vol. IV, IEEE Computer Society Press, 1992.
168. D.P. Tegarden, S.D. Sheetz, D.E. Monarchi, "A Software Complexity Model of Object-Oriented Systems", *Decision Support Systems: the International Journal*, vol. 13, pp. 241-262, 1995.

169. M. Telles, *C# Black Book*, The Coriolis Group, Nov., 2001
170. J. Tian, and M.V. Zelkowitz, "A Formal Program Complexity Model and its Application", *Journal of Systems and Software*, vol. 17, no. 3, pp. 253-266, 1992.
171. Tomcat Jasper <<http://jakarta.apache.org/tomcat>>
172. TulipChain <<http://ostermiller.org/tulipchain>>
173. Velocity <<http://jakarta.apache.org/velocity>>
174. B. Wadhwa, S. Andrei, and S.Y. Jien, *Software Engineering: An Object-Oriented Approach*, McGraw Hill, Singapore, 2007.
175. Y. Wang, and J. Shao, "Measurement of the Cognitive Functional Complexity of Software", *Proc. IEEE International Conference on Cognitive Informatics (ICCI'03)*, pp. 67-71, 2003.
176. Y. Wang, and J. Shao, "A New Measure of Software Complexity based on Cognitive Weights", *Canadian Journal of Electrical & Computer Engineering*, vol. 28, no. 2, pp. 69-74, 2003.
177. J. Wang, Y. Zhou, L. Wen, Y. Chen, H. Lu, and B. Xu, "DMC: A More Precise Cohesion Measure for Classes", *Information and Software Technology*, vol. 47, no. 3, pp. 167-180, 2005.
178. H. Washizaki, H. Yamamoto, and Y. Fukazawa, "A Metrics Suite for Measuring Reusability of Software Components", *Proc. Ninth International Software Metrics Symposium (METRICS'03)*, pp. 211, 2003.
179. W. Boggs and M. Boggs, *Mastering UML with Rational Rose*, Sybex Computer Books Inc., USA, 1999.
180. E.J. Weyuker, "Evaluating Software Complexity Measure", *IEEE Transactions on Software Engineering*, vol. 14, no. 9, pp. 1357-1365, 1988.
181. G. Woo, H.S. Chae, J.F. Cui, and J.H. Ji, "Revising Cohesion Measures by Considering the Impact of Write Interactions between Class Members", *Information and Software Technology*, vol. 51, no. 2, pp. 405-417, 2009.
182. XGen Source Code Generator <<http://sourceforge.net/projects/xgen/>>
183. B. Xu and Y. Zhou, "Comments on A Cohesion Measure for Object-Oriented Classes", *Software Practice & Experience*, vol. 31, no. 14, pp. 1381-1388, 2001.

184. B. Xu, Z. Chen, and J. Zhao, "Measuring Cohesion of Packages in Ada95", *Proc. Annual ACM SIGAda International Conference on Ada: The Engineering of Correct and Reliable Software for Real-Time & Distributed Systems using Ada and Related Technologies*, San Diego, California, USA, pp. 62-67, 2003.
185. T. Xu, K. Qian K, and X. He, "Service Oriented Dynamic Decoupling Metrics", *Proc. International Conference on Semantic Web and Web Services (SWWS'06)*, Las Vegas, USA, June 26-29, 2006.
186. S. Yacoub, H. Ammar, and T. Robinson, "Dynamic Metrics for Object-Oriented Designs", *Proc. 5th International Software Metrics Symposium*, Boca Raton, Florida, USA, pp.50-61, 1999.
187. S. Yacoub, H. Ammar, and T. Robinson, "A Methodology for Architectural-Level Risk Assessment using Dynamic Metrics", *Proc. 11th Int'l Symp. Software Reliability Eng.*, pp. 210-221, 2000.
188. X. Yang, *Research on Class Cohesion Measures*, M.S. Thesis, Department of Computer Science & Engineering, Southeast University, April 2002.
189. E. Yourdon and L. Constantine, *Structured Design*, Prentice Hall, 1979.
190. L.G. Yu and S. Ramaswamy, "Component Dependency in Object-Oriented Software", *Journal of Computer Science and Technology*, vol. 22, no. 3, pp. 379-386, May 2007.
191. E. Yu and J. Mylopoulos, "An Actor Dependency Model of Organizational Work- With Application to Business Process Reengineering", *Proc. Conference on Organizational Computing Systems*, Milpitas, Calif., USA, pp. 258-268, Nov. 1-4, 1993.
192. E. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering", *Proc. 3rd IEEE Int. Symposium on Requirements Engineering (RE'97)*, Washington D.C., USA, pp. 226-235, Jan. 6-8, 1997.
193. A. Zaidman and S. Demeyer, "Analyzing Large Event Traces with the Help of Coupling Metrics", *Proc. 5th International Workshop on OO Reengineering*, 2004.
194. A. Zaidman, B.D. Bois, and S. Demeyer, "How Webmining and Coupling Metrics can Improve Early Program Comprehension", *Proc. 14th International Conference on Program Comprehension (ICPC'06)*, IEEE Computer Society, pp. 74-78, 2006.

195. A. Zaidman, B. Adams, K. De Schutter, S. Demeyer, G. Hogman, and B. De Ruyck, "Regaining Lost Knowledge through Dynamic Analysis and Aspect Orientation- An Industrial Experience Report", *Proc. 10th Conference on Software Maintenance and Reengineering (CSMR'06)*, IEEE Computer Society, pp.89–98, 2006.
196. A. Zaidman and S. Demeyer, "Automatic Identification of Key Classes in a Software System using Webmining Techniques", *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 20, no. 6, pp. 387-417, Nov. 2008.
197. A. Zaidman, *Scalability Solutions for Program Comprehension through Dynamic Analysis*, PhD Thesis, Univ. of Antwerp, 2006.
198. Y. Zhou, B. Xu, J. Zhao, and H. Yang, "ICBMC: An Improved Cohesion Measure for Classes", *Proc. International Conference on Software Maintenance*, IEEE Computer Society Press, Montreal, Canada, pp. 44-53, 2002.
199. Y. Zhou, L. Wen, J. Wang, and Y. Chen, "DRC: A Dependence Relationships based Cohesion Measure for Classes", *Proc. Tenth Asia-Pacific Software Engineering Conference (APSEC'03)*, pp. 215-223, 2003.
200. Y. Zhou, J. Lu, H. Lu, and B. Xu, "A Comparative Study of Graph Theory-based Class Cohesion Measures", *ACM SIGSOFT Software Engineering Notes*, vol. 29, no. 2, pp. 13-13, 2004.
201. T. Zhou, B. Xu, L. Shi, Y. Zhou, and L. Chen, "Measuring Package Cohesion Based on Context", *Proc. IEEE International Workshop on Semantic Computing and Systems*, pp. 127-132, 2008.