MODELLING AND MEASURING STRUCTURAL COMPLEXITY OF PROLOG PROGRAM BASED ON RULE-DEPENDENCY

Alizam Jonie (82595)
Master of Science (Information Technology) 2005

MODELLING AND MEASURING STRUCTURAL COMPLEXITY OF PROLOG PROGRAM BASED ON RULE-DEPENDENCY

This thesis is presented to Faculty of Information Technology in fulfillment of the requirements for the degree of Master of Science (Information Technology), Universiti Utara Malaysia

JABATAN HAL EHWAL AKADEMIK
(DEPARTMENT OF ACADEMIC AFFAIRS)
UNIVERSITI UTARA MALAYSIA

PERAKUAN KERJA/TESIS
(Certification of Thesis Work)

Kami, yang bertandatangan, memperakukan bahawa
(We, the undersigned, certify that)

ALIZAM JONIE

calon untuk Ijazah
(candidate for the degree of) *SARJANA SAINS (TEKNOLOGI MAKLUMAT)*

telah mengemukakan tesis/disertasinya yang bertajuk
(has presented his/her thesis work of the following title)

MODELLING AND MEASURING STRUCTURAL COMPLEXITY OF PROLOG PROGRAM BASED ON RULE-DEPENDENCY

seperti yang tercatat di muka surat tajuk dan kulit tesis/disertasi
(as it appears on the title page and front cover of thesis work)

bahawa tesis/disertasi tersebut boleh diterima dari segi bentuk serta kandungan, dan liputan bidang ilmu yang memuaskan, sebagaimana yang ditunjukkan oleh calon dalam ujian lisan yang diadakan pada : **16 Ogos 2005**

(that the thesis/dissertation is acceptable in form and content, and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate through an oral examination held on

Pengerusi Viva *(Chairman for Viva)*	:	Prof. Madya Dr. Suhaidi Hassan	Tandatangan: *(Signature)*
Pemeriksa Luar *(External Examiner)*	:	Dr. Azuraliza Abu Bakar	Tandatangan: *(Signature)*
Pemeriksa Dalaman *(Internal Examiner)*	:	Prof. Madya Dr. Zulikha Jamaluddin	Tandatangan: *(Signature)*
Penyelia Utama *(Principal Supervisor)*	:	Prof. Madya Dr. Zulkhairi Md. Dahalin	Tandatangan: *(Signature)*
Dekan, Fakulti Teknologi Maklumat *(Dean, Faculty of Information Technology)*	:	Prof. Madya Dr. Zulkhairi Md. Dahalin	Tandatangan: b/p *(Signature)*

Tarikh
(Date) : **16 OGOS 2005**

PERMISSION TO USE

In presenting this thesis as major requirements for a post-graduate degree from Universiti Utara Malaysia, I agree that the University Library may make it freely available for inspection after being submitted for a year. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by my supervisor or, in his absence, by the Director of Center for Graduate Study. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that recognition shall be give to me and to Universiti Utara Malaysia for any scholarly use, which may be made of any material from my thesis.

Request for permission to copy or to make other use of materials in this thesis, in whole or in part, should be addressed to:

Dean of Faculty of Information Technology
Universiti Utara Malaysia
06010 Sintok
Kedah Darul Aman
MALAYSIA

EXECUTIVE SUMMARY

This thesis describes modelling and measuring structural complexity measure of Prolog program based on rule-dependency. Rule-dependency can be defined as relationships or interaction between rules. Usually, Prolog program is constructed by rules. These rules are Horn clause subset of the clausal form of first-order predicate logic. It is believed that rule-dependency is significant element of complexity and this research investigates to corroborate the claim especially on how rule dependency can be used to model and measure Prolog's structural complexity. This research is motivated by the lack of measures developed for Prolog due to the implicit control flow and construct. This lack of explicit control flow and constructs precludes in adapting conventional measures to Prolog program. This thesis shall present models that can be used to partially solve this problem that can enabled direct application of existing measures to Prolog program. To do measurement four criteria are explicitly defined: (1) attribute of entity, (2) abstraction or model, (3) ordering relationships, and (4) order-preserving mapping. These criteria are based on representational approach of measurement theory. The model Prolog's control flow and construct are modelled in the second criteria, while the measure is achieved by completing the process from identification of entity and attribute into numbers.

RINGKASAN EKSEKUTIF

Tesis ini menghuraikan permodelan dan pengukuran kekompleksan struktur aturcara Prolog berdasarkan pergantungan aturan. Pergantungan aturan boleh didefinisikan sebagai hubungan atau interaksi di antara aturan-aturan. Pada kebiasaannya, aturcara Prolog dibangunkan mengunakan aturan-aturan. Aturan-aturan ini merupakan klausa Horn subset kepada *clausal form of first-order predicate logic.* Adalah dipercayai bahawa pergantungan aturan adalah elemen penting kompleksiti dan tesis ini menyediakan bukti sokongan kepada dakwaan tersebut terutama sekali bagaimana pergantungan aturan boleh digunakan untuk memodel dan mengukur kekompleksan struktur aturcara Prolog. Penyelidikan ini didorong oleh kekurangan ukuran-ukuran yang dibangunkan khas untuk Prolog, berikutan kawalan aliran dan struktur Prolog yang tidak jelas. Masalah ini mengakibatkan, adaptasi ukuran sedia ada terus ke aturcara Prolog tidak dapat dilaksanakan. Tesis ini akan menghuraikan model-model yang digunakan untuk menyelesaikan sebahagian daripada masalah ini yang membolehkan aplikasi terus ukuran-ukuran sedia ada kepada Prolog. Untuk membuat pengukuran empat kriteria diambil kira: (1) atribut sesuatu entiti, (2) abstrak atau model, (3) hubungan tatasusunan, dan (4) *order-preserving mapping.* Kriteria-kriteria ini berdasarkan interpretasi teori pengukuran berdasarkan pendekatan perwakilan. Model aliran kawalan dan struktur Prolog diketengahkan dalam kriteria kedua, sementara ukuran pula diperoleh dengan identifikasi atau takrifan kepada setiap kriteria.

ACKNOWLEDGEMENT

Praise be to Allah S.W.T for blessing me with healthiness and awareness in completing this thesis.

Firstly I would like to thank my supervisor, Prof Madya Dr. Zulkhairi Md Dahlin, for steering me in the right direction and for always making the time to help me throughout the development of this work.

Thanks to all my research colleagues at the School of Information Technology of University Utara Malaysia for their friendship and for providing an environment where I could learn so much, *bird of the feathers*: Mazlyda, Rohaya, Siti Salmah, Adillah, Nurasiah, Noorul, Aida Liza, Izhar, Zuraidy, Mahyuddin, Farkhana, Mohd Fadhli, Zulhazlin, Mohd Zaidil, Noraina, Rokiyah and Nurmayani. Also thanks to all my friends who keep me sane throughout this period particularly: Syamsul Bahrin, Harryizman and Zainizam.

Special thanks are due to Wan Hussain for giving me a knock in the head to stop me from being mad scientist and for the time to discuss the difficult-to-grasp matters. Also special thanks to Shair Abdullah, Austin Melton, Norman Fenton, Trevor Moores, Valerie Barr, Steve Worthington, and John McDermid for helping me with the book, article and idea that I certainly needed for the research.

Finally, I would like to thank my parents, Jonie Awang and Siti Latiah Serin, and my Brothers, Firdaus and Mohd Syufaat. Their continued encouragement and support has made the task of completing this thesis possible.

In for a penny, in for a pound.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

$\sum$	Summation
$\in$	Member of
AOG	AND/OR Graph
BNF	Backus-Naur Form
CC	Cyclomatic Complexity
DE	Decision Count
FIFO	Fan-in and Fan-out
ISO	International Standard Organization
MT	Measurement Theory
PRAM	Prolog Automatic Marker
P-structure	Prolog Local Structure
RD	Rule-dependency
RFI	Rule Fan-in
RFO	Rule Fan-out
SE	Software Engineering
SM	Software Measurement
SPM	Software Project Model
S-structure	Local Standard Structure
UPN	Unique Predicate Name

CHAPTER ONE

INTRODUCTION

Measurement is an integral part of everyday life. Measurement provides objective information (McGarry *et al.*, 2002) which cannot be provided by other means (Shepperd, 1995). A chef used measurement to make a good proportion of cake. For parents, they measure their children height to buy the right size of clothes. In scientific and engineering discipline measurement allows the acquisition of information that can be used for developing theories and models, devising, assessing, and using methods and techniques (Morasca, 2000). It would be difficult to imagine how the disciplines of electrical, mechanical and civil engineering could have evolved without a central role of measurement (Fenton, 1991).

Measurement in software engineering (SE) is called software engineering measurement or in short, *software measurement* (SM). SE can be regarded as the use of sound engineering principles to develop economical software that is reliable and works efficiently on real machines (Pressman, 1992). Most of the SE methods that have been proposed and developed provide rules, tools, and heuristics for producing software products (Fenton, 1991). One of the reasons to measure software is to indicate the quality of the software product according to particular structural principles or program structure[1] (Pressman, 1992).

Program structure in this research concerns on the construction of a program based on control constructs. It is believed that a program with a large number of control constructs is relatively more complex and difficult to understand (Conte *et al.*, 2002). This is known as structural complexity. In academia, a program with higher structural complexity may indicate bad programming habits (Mansouri, 1998) e.g.

[1] Examples of program structure are modularity, re-use, coupling, cohesiveness, redundancy, *D*-structuredness, hierarchic and structured programming.

The contents of the thesis is for internal user only

REFERENCES

Apt, K. R. & Smaus, J. G. (2001). Rule-based versus Procedure-based View of Programming. *Joint Bulletin of the Novosibirsk Computing Center and Institute of Informatics Systems Series: Computer Science*, 16, p. 75 – 97.

Barr, V. (1996). Rule-Based System Testing with Control and Data Flow Techniques. *International Software Quality-Week 1996.*

Baker, L. A., Bieman J. M, Fenton, N, Gustafson, D. A., Melton, A., & Whitty, R. (1990). A Philosophy for Software Measurement (electronic version). *Journal of Systems and Software*, 12, p. 277 – 281.

Bieman, J. M., Fenton, N, Gustafson, D. A, Melton, A. & Ott, L. M. (1995). Fundamental Issues in Software Measurement, In Austin Melton (Ed.). *Software Measurement*, p. 39 – 52. UK: International Thomson Computer.

Bieman, J. M. (1995). Metric Development for Object-Oriented Software. In Austin Melton (Ed.). *Software Measurement*, p. 75 – 92. UK: International Thomson Computer.

Bratko, I. (2001). *PROLOG: Programming for Artificial Intelligence 3rd ed.* Singapore: Addison-Wesley.

Briand, L., Morasca, S., & Basili, V. (1994). Property Based Software Engineering Measurement (electronic version). *Technical Report CS-TR-119.* University of Maryland.

Briand, L., Emam, K. E., & Morasca, S. (1995a). On the Application of Measurement Theory in Software Engineering (electronic version). *Technical Report ISERN-95-04*. International Software Engineering Research Network.

Briand, L., Emam, K. E. & Morasca, S. (1995b). Theoretical and Empirical Validation of Software Product Measures. *Technical Report ISERN-95-03*. International Software Engineering Research Network.

Cardoso, A.I., Kokol, P. & Crespo R. G. (2000). Two different views about software complexity. *11th European Software Control and Metrics Conference*, p. 433 – 438.

Cardoso, A. I., Kokol, P. & Crespo, R. G (2002). How to measure the complexity of a program text. *Proceedings of the third International Symposium on Engineering of intelligent Systems*, p. 119 – 122.

Carson, P. A. & Hunter, R. B. (1992). Source-Code Models and Their Use in Assessing Software Quality. *EUROMETRICS' 92*, p. 83 – 84.

Chen, Jeng-Rung & Cheng, A. M. K. (1994). A Fast, Partially Parallelizable Algorithm for Predicting Execution Time of EQL Rule-Based Programs, In Jagdish Chandra (Ed.). *Proceedings of International Conference on Parallel Processing*, p. 17 – 20.

Ciancarini, P. & Levi, G. (1995). Applications of Logic Programming in Software Engineering (electronic version). *PAP Workshop on Logic Programming and Software Engineering*. Retrieved: (n.d.) from site: http://citeseer.nj.nec.com/326955.html

Clocksin, W. F. & Mellish, C. S. (1984). *Programming in Prolog 2nd ed.* New York: Springer-Verlag.

Conte, S. D., Dunsmore, H. E. & Shen, V. Y. (1986). *Software Engineering Metrics And Models*. UK: Benjamin Cummings Publishing.

Covington, M. A., Nute, D. & Vellino, A. (1997). *Prolog Programming in Depth.* London: Prentice-Hall International.

Dick, R. (1993). Subjective Software Measurement - Tools for the Human Assessor. *Esprit Project SCOPE Report*. Department of Computer Science, University of Strathclyde, Glasgow. Retrieved: Jun 29, 2002 from site: http://citeseer.nj.nec.com/433040.html

Emam, K. E. (2000). A Methodology for Validating Software Product Metrics (electronic version). *Technical Report NRC 44142.* National Research Council of Canada. Retrieved September 25, 2001 from http://iit-iti.nrc-cnrc.gc.ca/publications/nrc-44142_e.html

Fenton, N. E. & Melton, A. (1995). Measurement Theory and Software Measurement, In Austin Melton (Ed.). *Software Measurement*, p. 27 – 38. UK: International Thomson Computer.

Fenton, N. E. (1991). *Software Metrics a Rigorous Approach.* London: Chapman & Hall.

Fenton, N. E. & Pfleeger, S. L. (1997). *Software Metrics a Rigorous Approach 2nd Edition.* London: International Thomson.

Gordon, R. D. & Halstead, M. H. (1976). An Experiment Comparing Fortran Programming Time with the Software Physics Hypothesis. *Proceeding AFIPS*, p. 935 – 937.

Gross, J. & Yellen, J. (1998). *Graph Theory and Its Applications*. USA: CRC Press.

Gustafson, D. A., Tan, J. T. & Weaver, P. (1995). Software Metric Specifications. In Austin Melton (Ed.). *Software Measurement*, p. 179 – 195. UK: International Thomson Computer.

Hao, W. (1999). *Software Metrics Collection Techniques for Product Assessment*. Retrieved August 27, 2001 from Wang Hao's Homepage site [offline]: http://www.comp.nus.edu.sg/~wanghao/ic52a5_content.html

Hass, M. & Hassel, J. (1983). A Proposal for a Measure of Program Understanding. *Proceedings of the Fourteenth SIGCSE Technical Symposium on Computer Science Education*, p. 7 – 13.

Henry, S. & Kafura, D. (1981). Software Structure Metrics based on Information Flow. *IEEE Transactions on Software Engineering*, 7(5), p. 510 – 518.

Kaposi, A. A. (1993). Measurement Theory, In John A. McDermid (Ed.). *Software Engineer's Reference Book*. London: Butterworth-Heinemann.

Kearney, J. P., Sedlmeyer, R. L., Thompson, W. B., Gray, M. A. & Adler, M. A. (1986). Software Complexity Measurement. *Communications of the ACM*, 29(11), p. 1044 – 1050.

Luger, G. F. (2002). *Artificial Intelligence: Structures and Strategies for Complex Problem Solving 4^{th} ed.* Essex: Pearson Education.

Mansouri, F. Z., Gibbon, C. A. & Higgins, C. A. (1998). PRAM: Prolog Automatic Marker. *Proceedings of the Sixth Annual Conference on the Teaching of Computing*, p. 166 – 170.

Marco, L. (1997). Measuring Software Complexity (electronic version). *Enterprise System Journal*. Retrieved August 27, 2001 from http://cispom.boisestate.edu/cis320emaxson/metrics.htm

Markusz, Z. & Kaposi, A. A. (1985). Complexity Control in Logic-based Programming (electronic version). *The Computer Journal*, 28(5), p. 487 – 495.

McCabe, T. J. (1976). A Complexity Measure (electronic version). *IEEE Transactions on Software Engineering*, 2(4), p. 308 – 320.

McCauley, R. A & Edwards, W. R. (1995). Analysis and Measurement Techniques For Logic-Based Languages. In Austin Melton (Ed.), *Software Measurement*, p. 93 – 133. UK: International Thomson Computer.

McDermid, J. A. (2000). Complexity: Concept, Causes and Control. *Proceedings Sixth IEEE International Conference on Engineering of Complex Computer Systems*, p. 2 – 9.

McGarry, J., Card, D., Jones, C., Layman, B., Clark, E., Dean, J. & Hall, F. (2002). *Practical Software Measurement*. USA: Addison-Wesley.

Melton, A. C., Gustafson, D. A, Bieman, J. M., & Baker, A. L. (1990). A Mathematical Perspective of Software Measures Research. *IEE/BCS Software Engineering Journal*, 5(5), p. 246 – 254.

Moores, T. T. (1998). Applying Complexity Measures to Rule-based Prolog Programs. *Journal of Systems and Software*, 44(1), p. 45 – 52.

Morasca, S. (2001). Software Measurement. In S.K. Chang (Ed.), *Handbook Of Software Engineering And Knowledge Engineering Volume 1: Fundamentals,* p. 239 – 276. USA: World Scientific.

Neal, R. D., Coppins, R. J., & Weistroffer, H. R. (1997). The Assignment of Scale to Object-Oriented Software Measures. *Working Report NCCW-0040*. NASA.

Nilsson, U. & Maluszyński, J. (2000). *Logic, Programming And Prolog 2nd ed* (electronic version). Retrieved December 28, 2003 from http://www.ida.liu.se/~ulfni/lpp

O'Neal, M. B & Edward, W. R. (1994). Complexity Measures For Rule-Based Programs. *IEEE Transactions on Knowledge Data Engineering*, 6(5), p. 669 – 680.

Otto, K. N. (1994). Measurement Methods for Product Evaluation. *Research in Engineering Design*, p. 157 – 166.

Ott, L. M. (1995). The Early Days of Software Metrics: Looking Back After 20 Years, In Austin Melton (Ed.). *Software Measurement*, p. 7 – 25. UK: International Thomson Computer.

Ottenstein, K. J. & Ottenstein, L. M. (1984). The Program Dependence Graph in a Software Development Environment. *ACM Software Engineering Notes*, 9(3), p. 177 – 184.

Oviedo, E. (1980). Control Flow, Data Flow and Program Complexity. *Proceedings COMPSAC 80*, p. 146 – 152.

Prather R. E. (1984). An Axiomatic Theory of Software Complexity Measure (electronic version). *The Computer Journal*, 27(4), p. 340 – 347.

Prather, R. E. (1995). The Role of Recursion and Symmetry in the Design and Analysis of Software Metrics, In Austin Melton (Ed.). *Software Measurement*, p. 145 – 156. UK: International Thomson Computer.

Pressman, R. S. (1992). *Software Engineering: A Practitioner's Approach 3rd ed.* New York: McGraw-Hill.

Riguzzi, F. (1996). A Survey of Software Metrics. *Technical Report DEIS-LIA-96-010*. DEIS. Università degli Studi di Bologna.

Robinson, J. A. (1965). A Machine-Oriented Logic Based on the Resolution Principle. *Journal of the ACM*, 12(1), p. 23 – 41.

Rubey, R. J. & Hartwick R. D. (1968). Quantitative Measurement of Program Quality. *Proceedings of the 23rd ACM National Conference*, p. 671 – 677.

Sebesta, W. Robert (1996). *Concepts of Programming Languages 3rd ed.* USA: Addison-Wesley.

Sellappan, P. (2000). *Software Engineering: Management & Methods*. Malaysia: Sejana Publishing.

Shepperd, M. (1995). *Foundations Of Software Measurement.* London: Prentice Hall.

Shepperd, M. (1993). Software Measurement: Past, Present and Future. In Martin Shepherd (Ed.), *Software Engineering Metrics Volume I: Measures & Validations*. UK: McGraw-Hill.

Tsai, Jefferey J. P. & Jen, Yuan-Heng (1997). *Evaluation of Rule-Based Systems* (electronic version). Retrieved May 23, 2001 from [offline]: http://kel3.eecs.uic.edu/USAF/cplxdf3/doccplxdf.html

Van den Berg, K.G & Van den Broek, P.M. (1995). Axiomatic Validation in the Software Metric Development Process. In Austin Melton (Ed.), *Software Measurement*, p. 157 – 177. UK: International Thomson Computer.

Weyuker, E. J. (1988). Evaluating Software Complexity Measures (electronic version). *IEEE Transaction of Software Engineering*, 14(9), p. 1357 - 1365.

Zhao, J., Cheng, J. & Ushijima, K. (1998). A Metrics Suite for Concurrent Logic Programs (electronic version). *Proceedings of the 2nd Euromicro Conference on Software Maintenance & Reengineering,* p. 172 – 178. Retrieved February 28, 2001, from http://citeseer.nj.nec.com/35967.html

Zuse, Horst. (1995). *History of Software Measurement.* Retrieved June 3, 2001 from Dr. –Ing. Horst Zuse, Software Measurement - Software Metrics site: http://irb.cs.tu-berlin.de/~zuse/sme.html