STATIC ANALYSER FOR JAVA-BASED OBJECT-ORIENTED SOFTWARE METRICS

HASAN MUGBIL KHALAF ABU AL-ESE

FSKTM 1999 4

STATIC ANALYSER FOR JAVA-BASED OBJECT-ORIENTED SOFTWARE METRICS

By

HASAN MUGBIL KHALAF ABU AL-ESE

Thesis Submitted in Fulfilment of the Requirements for the Degree of Master of Science in the Faculty of Computer Science and Information Technology
Universiti Putra Malaysia

February 1999

بِسْمِ ٱللَّهِ ٱلرَّحْمَٰنِ ٱلرَّحِيمِ

وَيُزَكِّيكُمْ وَيُعَلِّمُكُمُ ٱلْكِتَٰبَ وَٱلْحِكْمَةَ وَيُعَلِّمُكُم

مَّا لَمْ تَكُونُوا۟ تَعْلَمُونَ ﴿١٥١﴾

Surat Al-Baqara.

To My Late Father's Pure Spirit.

ACKNOWLEDGEMENTS

In the name of Allah, Most Gracious, Most Merciful

I would like to take this opportunity to convey my sincere thanks and deepest gratitude to my chairman supervisor Dr. Abdul Azim Abd. Ghani for his advice, comments, suggestions, help, and invaluable guidance throughout my research. I am also indebted to my co-supervisors Dr. Ramlan Mahmod and Dr. Md. Yazid Saman for their constant support all the time throughout my research. I am very grateful to all of them for all the help and invaluable guidance, fruitful discussions, patience and continued encouragement provided to me at every stage of this thesis. A lot of thanks and grateful to all of them for their kindness and having taken so much of their valuable time for studying, correcting and restructuring the preliminary drafts.

I would like to convey my appreciation to the Faculty of Computer Science and Information Technology, the Library of the University, the Graduate School Office, and the Laboratories Technicians. I am also indebted to UM University and UKM University for allowing to use their Libraries.

My special thanks go to the Embassy of Jordan in Kuala Lumpur and to all of its employees. I would like in particular to thank the ambassador Mr. Abdul Ilah Al-Kurdi, Mr. Sudqi Al-Omoush and his family for their helping and moral support.

I would like to express my deepest gratitude to Prof. Mohammad Adnan Al-bakheet, the president of Al al-Bayt University for his encouragement. I am also pleased to thank my friends in Computer Center and Faculty of Science for their encouragement and helping.

I am also very grateful and wish to thank my Arab friends at UPM; Ahmad Sulaiman Al-Khawaldeh, Atef Ali Al-Khawaldeh, Belal Barham, Rafa Al-Qutasih, Abdulla Al-Shbil, Malik Al-Qdah, Hamed Fawareh, Jehad Al-Khaldi, Faraj Abu Alaiwi, Abdul Nasir Abu Alaiwi, Khalil Al-Hasan and Ahmad Zahran for their encouragement and support.

I am also pleased to convey my deepest thanks to those who have been giving me their invaluable time and who have been sharing me my interesting time, my special thanks go to Isam Qudsieh and his sister Hanan (UPM), Moneef Al-Jazzar (UPM), Ahmad Al-Esa (USM), Sharaf Al-Horani (UNIMAS), Hayel Al-Fuqara (UM), Mohammad Abu Ghazleh (UM), Imad Hamadneh (UKM). Not forgetting my Malaysian friends Syed Hussien Mahmud, Zulkifli Zainul and Mahdi Boon. Sincere thanks to all of them for their kindness and moral encouragement.

I would like to convey my sincere thanks and deepest gratitude to my closed friend who stood beside me all the time, indeed I am indebted to my brother Saleh Ayed Ahmad Al-Khawaldeh who gave me his invaluable time, sincere thanks to him and for his great-hearted. Not forgetting my closed friends Omar Abu Al-Ese, Mohammad Al-Smadi and Abdul Salam Al-Hussari who stood beside me all the time. Sincere thanks to all of them and for their noble deeds.

Finally, I would like to express my most sincere and warmest gratitude to my mother, sisters, brothers; Yaser, Mohammad, Ahmad, Ihsan, and Raja, sister in law, brothers in law; Saleh, Fawaz, and Farhan, uncles, aunts, cousins, nephews, and nieces for their prayers, love, generous moral and financial support during my study.

TABLE OF CONTENTS

APPENDIX

LIST OF TABLE

LIST OF FIGURES

Abstract of thesis presented to the Senate of Universiti Putra Malaysia in fulfilment of the requirements for the degree of Master of Science.

STATIC ANALYSER FOR JAVA-BASED OBJECT-ORIENTED SOFTWARE METRICS

By

HASAN MUGBIL KHALAF ABU AL-ESE

February 1999

Chairman : Abdul Azim Abd. Ghani, Ph.D.

Faculty : Computer Science and Information Technology.

Software metrics play a major role in the software development. Not only software metrics help in understanding the size and complexity of software systems, but they are also helpful in improving the quality of software systems. For object-oriented systems, a large number of metrics have been established. These metrics should be supported by automated collection tools. Automated tools are useful for measuring and improving the quality of software systems. One such tool is a static analyser.

A static analyser has been developed for a subset of Java language. A number of object-oriented software metrics has been evaluated using attribute grammar approach. Attribute grammar approach is considered as a well-defined approach to the software metrics evaluation since it is based on the measurement of the source code itself. New definitions for a number of object-oriented metrics have been specified using attribute grammars.

This tool has been built using C language. Lexical analyser and syntax analyser have been generated using *lex* and *yacc* tools under *linux* operating system. Four object-oriented metrics have been evaluated. These metrics are Depth of Inheritance Tree metric, Number of Children metric, Response For a Class metric, and Coupling Between Object Classes metric. The software metrics will be produced in the common metrics format that is used in SCOPE project.

Abstrak tesis yang dikemukakan kepada Senat Universiti Putra Malaysia sebagai memenuhi keperluan untuk ijazah Master Sains.

PENGANALISIS STATIK BAGI METRIK PERISIAN BERORIENTASI OBJEK BERDASARKAN BAHASA JAVA

Oleh

HASAN MUGBIL KHALAF ABU AL-ESE

Februari 1999

Pengerusi : Abdul Azim Abd. Ghani, Ph.D.

Fakulti : Sains Komputer dan Teknologi Maklumat

Metrik perisian memainkan peranan yang penting di dalam pembangunan perisian. Metrik perisian bukan hanya membantu di dalam memahami sesuatu saiz dan kesukaran sesuatu sistem perisian, tetapi ia juga membantu untuk memperbaiki kualiti sesuatu sistem perisian. Untuk sistem berorientasikan objek, sebilangan besar metrik telah dibina. Kesemua metrik perlu disokong oleh peralatan pengumpulan automasi. Peralatan automasi sangat berguna untuk mengukur dan memperbaiki kualiti sisitem perisian. Salah satu alat tersebut adalah penganalisis statik.

Satu penganalisis statik telah dibangunkan untuk subset kepada bahasa Java. Sebilangan metrik perisian berorientasi objek telah dinilai dengan menggunakan pendekatan nahu atribut. Pendekatan nahu atribut ini dianggap sebagai satu pendekatan yang sesuai untuk proses penilaian metrik perisian kerana ia adalah berdasarkan pengukuran kod sumbernya tersendiri. Definisi baru untuk sebilangan metrik berorientasi objek telah dikenal pasti dengan menggunakan nahu atribut ini.

Perisian ini dibina dengan menggunakan bahasa C. Penganalisis leksikal dan sintak telah dihasilkan menggunakan peralatan *lex* dan *yacc* di bawah sistem pengoperasian *linux.* Empat metrik berorientasi objek telah dinilai. Metrik ini terdiri dari metrik Kedalaman Pepohon Pewarisan, metrik Bilangan Anak, metrik Tindakbalas Untuk Kelas dan metrik Pasangan Untuk Kelas Objek. Metrik perisian akan dihasilkan dalam format metrik yang biasa seperti yang digunakan dalam projek SCOPE.

CHAPTER I

INTRODUCTION

Measurement in Software Engineering

Software engineering describes the collection of techniques that apply an engineering approach to the construction and support of software product. Software engineering activities include managing, costing, planning, modelling, analysing, specifying, designing, implementing, testing, and maintaining (Fenton and Pfleeger, 1997). Engineering approach means that each activity is understood and controlled. Software engineering focuses on implementing software in a controlled and scientific ways. Software engineering needs measurement in order to quantify the aspects of software development and maintenance.

It is clear that measurement can be effective, if not essential, in making characteristics and relationships more visible, in assessing the magnitude of problems, and in fashioning a solution to problems.

Measurement and Software Metrics

Today, computers play a primary role in almost every area of our life. The increased importance of software also places more requirements on it. Thus, it is necessary to have precise, predictable, and repeatable control over the software development process and product. Software measures are tools to measure the quality of software. The area of software measurement is also known as software metrics. The terms, metric and measure are used as synonyms.

The background for software measures and software measurement was established in the sixties (Rubey and Hartwick, 1968), and mainly in the seventies (McCabe, 1976; Halstead, 1977; Albrecht, 1979). And from these earlier works, further results have emerged in the eighties and nineties.

Measurement is important for three basic activities (Fenton and Pfleeger, 1997). First, there are measures that help us to understand what is happening during development and maintenance. Projects without clear goals will not achieve their goals clearly (Gilb, 1988). Second, the measurement allows us to control what is happening in our projects. You can neither predict nor control what you cannot measure (DeMarco, 1982). Third, measurement encourages us to improve our processes and products.

According to Finkelstein (1984) measurement is defined as:

> *"Measurement is the process by which numbers or symbols are assigned to attributes of entities in the real world in such a way as to describe them according to clearly defined rules."*

Ince *et al.* (1993) defined the software metrics as numerical values of quality which can be used to characterize how good or bad that the product is in terms of properties such as its proneness to error.

Moreover, Fenton and Pfleeger (1997) defined it formally as:

> *"Measurement is as a mapping from the empirical world to the formal, relational world. Consequently, a measure is the number or symbol assigned to an entity by this mapping in order to characterize an attribute."*

Fenton and Pfleeger (1997) classified three classes of entities:

1. *Processes*: are collections of software-related activities.
2. *Products*: are any artifacts, deliverables, or documents that result from a process activity.
3. *Resources*: are entities required by a process activity.

Within each class of entities, there is a distinguish between two types of attributes (Fenton and Pfleeger, 1997):

1. *Internal attributes* of a product, process, or resource are those that can be measured purely in terms of the product, process, or resource itself. In other words, an internal attribute can be measured by examining the product, process, or resource on its own, separate from its behaviour.
2. *External attributes* of a product, process, or resource are those that can be measured only with respect to how the product, process, or resource relates to its environment. Here, the behaviour of the process, product, or resource is important rather than the entity itself.

Grady and Caswell (1989) summarized the advantages of software metrics. They determined that software metrics help the developer to:

1. Understand software development process better.
2. Measure progress.
3. Provide common terminology for key controlling elements of the process.
4. Identify complex software elements.
5. Make software management more objective and less subjective.
6. Enable the engineers and manager to estimate and schedule better.

7. Better evaluate the competitive position.
8. Understand where automation is needed.
9. Identify engineering practices, which lead to highest quality and productivity.
10. Make critical decisions earlier in the development process.
11. Eliminate fundamental causes of defects.
12. Encourage the use of software engineering techniques by the engineers and managers.
13. Encourage the definition of long-term software development strategy based upon a measured understanding of current practices and needs.
14. Be more competitive.

Software Metrics in an Object-Oriented Environment

It is quite clear that measurement is necessary for the software development process to be successful. The recent movement toward object-oriented technology must also include the processes that control object-oriented development, namely software measures.

Object-oriented systems contain many significant architectural features that are not adequately captured by existing metrics. Firstly, code and design metrics developed for structured software assume a separation between data and procedure which does not occur in object-oriented software. Secondly, the process of object-oriented design tends to differ, for example the boundaries between analysis and design tend to be less rigid, thus metrics developed for traditional systems are unlikely to be applicable, at least not without modification (Henderson-Sellers, 1991; Shepperd and Cartwright, 1997).

Fetcke (1995) investigated the properties of object-oriented software metrics (Zuse and Fetcke, 1995) and summarized the following:

> *"The result of this investigation is that a large set of object-oriented metrics have properties that are completely different from properties of metrics for procedural languages."*

Moreover, Berard (1996) mentioned that object-oriented software engineering metrics are different because of localization, encapsulation, information hiding, inheritance, and object abstraction techniques.

At the end of the eighties, software measures for the object-oriented environment were proposed. A very early investigation of object-oriented measures can be found in Rocacher (1988), Morris (1988) and Pfleeger (1989). The first book about object-oriented software metrics was in 1994 (Lorenz and Kidd, 1994).

Jones (1997), in his paper about strengths and weaknesses of software metrics, mentioned the following strengths for object-oriented metrics:

1. They are psychologically attractive within the object-oriented community.
2. They appear to be able to distinguish simple from complex object-oriented projects.

However, he mentioned the following weaknesses:

1. They do not support studies outside of the object-oriented paradigm.
2. They do not deal with full life-cycle issues.
3. They have not yet been applied to testing.
4. They have not yet been applied to maintenance.
5. They have no conversion rules to lines of code metrics.
6. They have no conversion rules to function point metrics.
7. They lack automation.
8. They are difficult to enumerate.
9. They are not supported by software estimating tools.

However, in the area of object-oriented systems it is not clear what an object-oriented program makes difficult to understand, to test, or to maintain.