UNIVERSITI PUTRA MALAYSIA

INCORPORATING OBJECT-ORIENTED METRICS INTO A REVERSE ENGINEERING TOOL

NIDAL BASHIR ESHAH

FSKTM 2003 3

INCORPORATING OBJECT-ORIENTED METRICS INTO A REVERSE ENGINEERING TOOL

By

NIDAL BASHIR ESHAH

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Master of Science

April 2003

INCORPORATING OBJECT-ORIENTED METRICS INTO A REVERSE ENGINEERING TOOL

By

NIDAL BASHIR ESHAH

Thesis Submitted to the School of Graduate Studies, Universiti Putra Malaysia, in Fulfilment of the Requirements for the Degree of Master of Science

April 2003

To my parents

Abstract of thesis presented to the Senate of University Putra Malaysia in fulfillment of the requirements for the degree of Master of Science

INCORPORATING OBJECT-ORIENTED METRICS INTO A REVERSE ENGINEERING TOOL

By
NIDAL BASHIR ESHAH

April 2003

Chairman: Associate Professor Abdul Azim Abdul Ghani, Ph.D.
Faculty: Computer Science and Information Technology

This work explains the use of object-oriented software product metrics with their thresholds and how they could be incorporated into a reverse engineering tool that visualizes the architectural components of a software system. Visualizing Object-Oriented C++ (VOO++), reverse engineering tool that visualizes C++ object-oriented source code, is enhanced and reproduced to become a Visualizing and Measuring C++ (VMCPP) tool that visualizes and measures object-oriented C++ files. VMCPP assists the software developer in extracting and interpreting the components of a software system. Unified Modeling Language (UML) class diagrams are produced to graphically represent the classes involved in implementing a software system. Thresholds are used within VMCPP to separate the extracted metrics values into normal values and critical values.

Abstrak tesis dipersembahkan kepada Senat Universiti Putra Malaysia sebagai memenuhi syarat keperluan untuk ijazah Master Sains

PENGGABUNGAN METRIK OBJECT-ORIENTED KEPADA SATU ALAT KEJURUTERAAN SONGSANG

Daripada
NIDAL BASHIR ESHAH

April 2003

Pengerusi: Profesor Madya Abdul Azim Abdul Ghani, Ph.D.
Fakulti: Sains Komputer dan Teknologi Maklumat

Tesis ini menerangkan penggunaan metrik perisian berorientasi objek melalui penggunaan takat dan bagaimana ia dapat digabungkan dengan kejuruteraan songsang untuk membolehkan pengamatan komponen dalam sistem perisian. Memperlihatkan berorientasi objek C++ (VOO++), satu alat kejuruteraan songsang yang memperlihatkan kod sumber berorientasi objek C++, dipertingkat dan dibangunkan semula untuk menghasilkan, penganatan dan mengukur fail C++ berorientasi objek (VMCPP) yang membolehkan penganatan dan mengukur fail C++ berorientasi objek. VMCPP membantu pembangun perisian dalam mengambil dan mentafsir komponen sistem perisian. Kelas diagram, UML dihasilkan untuk mewakilkan secara grafik kelas yang terlibat dalam melaksanakan sesuatu sistem perisian. Takat digunakan dalam VMCPP untuk mengasingkan nilai-nilai metrik kepada nilai-nilai normal dan kritikal.

ACKNOWLEGEMENTS

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 supervisor Assoc. Prof. Dr. Abdul Azim bin Abd. Ghani for his advises, comments, suggestions, help, and invaluable guidance throughout my research.

I am also indebted to Assoc. Prof. Hj. Mohd Hasan bin Selamat and Ms. Hazura Zulzalil, members of the supervising committee, for their technical support and suggestions. Their insights were priceless.

I am greatly indebted to the Libyan Ministry of Education for the financial support during my study. Also, the contribution of the Libyan Embassy staff in Kuala Lumpur is highly appreciated.

Special thanks to the School of graduate studies staff at UPM, especially Mr. Rustam who without his help this thesis wouldn't be finished in time.

I am very Thankful to Mr. Ali Mresa who kept helping me understanding the source code of VOO++.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CASE	Computer Aided Software Engineering
HN	Hungarian Notation
LOC	Lines of Code
LCOM	Lack of Cohesion between methods
DIT	Depth of inheritance in the inheritance tree
RFC	Response for Class
WMC	Weighted Methods per Class
CC	Cyclomatic Complexity
CP	Comment Percentage
NOM	Number of Methods in Class
NOA	Number of Attributes in Class
NOPA	Number of Public Attributes in Class
NOPM	Number of Public Methods in Class
Fan-In	Flow of Information In, Module, Class, or Method
Fan-out	Flow of Information Out of, Module, Class, or Method
TFC	Total of Function Calls
NOC	Number of Children for Class
NRP	Number of Return Points in a Method
NOI	Number of Instances for Class
VOO++	A reverse engineering tool that visualizes object-oriented C++ source code
VMCPP	A reverse engineering tool that visualizes and measures object-oriented C++ source code

CHAPTER 1

INTRODUCTION

"The degree to which you can express something in numbers is the degree to which you really understand it." Lord Kelvin

1.1 Background

The concepts of object-oriented paradigm like encapsulation, inheritance and polymorphism made the object-oriented paradigm is more desirable by software developers than the traditional programming. Many of the traditional programming problems have been solved using the object-oriented paradigm (Pressman, 1997).

This technology requires not only new programming languages but also new approaches and techniques to refine it. For example software metrics applied to the traditional programming languages are no longer useful for the object-oriented approach because of some fundamentally different issues (encapsulation, inheritance, and polymorphism). As a result, new metrics have been introduced and applied to measure the products of the object-oriented approach.

1.2 Reverse Engineering

Reverse engineering for software is the process of analyzing a program in an effort to create a representation of the program at a higher level of abstraction than the source code (Pressman, 1997). The key to reverse engineering is its ability to abstract specifications from the detailed source code implementation (Pfleeger, 1998).

Most of the software reverse engineering tools extracts data and architectural design from software products to increase the understanding of the subject system (Hausi et al, 2000). The purpose of software reverse engineering product tools is to extract the architectural components, to explore and visualize, to measure, and to re-document existing software systems.

1.3 Software Measurement

A key element of any engineering process is measurement. We use measurements to better understand the attributes of the models that we create, but most important, we use measurement to assess the quality of the engineered products or systems that we build (Pressman, 1997).

In software engineering, software measurement tools gather software metrics to understand the subject software, to provide guidelines that recommend an action to improve the quality of software components, and to estimate a software product quality.

1.4 Problem Statement

The aim of developing reverse engineering software product tools that visualize software products is to graphically represent software attributes and components to the software developer by extracting them from the subject software. In object-oriented world the software components are classes, and visualizing the classes involved in implementing a specific software system will allow the software developer to explore the subject software classes and their relationships to each other. Such tools are built to assist software developers to make good decisions when evaluating the subject software by producing a graphical view of the architectural components of a software system. As a result a high-level of abstraction of the subject software will be produced showing the subject software system as a whole ignoring the low-level entities that in fact, makes the subject software system works.

On the other hand, software measurement tools treat the software products as a source of data that needs to be collected and then presented in terms of numbers. These tools lack the ability of representing software system components at a higher level of abstraction. Up to now there are no available tools that would deal with the subject software on both levels.

If software metrics are useful in a forward software engineering environment, then they are quite vital in a reverse engineering environment (Zhou et. al, 1999). Incorporating software metrics with their thresholds into reverse engineering software tools that visualize software products will be the ultimate solution to this problem (evaluating subject software system on both levels), which in turn will

enrich the knowledge of the software developer graphically and numerically about the subject software for better decisions.

1.5 Research Objectives

The objectives of this research are:

- To extend the functionality of a reverse engineering tool by incorporating software metrics with their thresholds. The chosen tool for this purpose is the available VOO++, Visualizing object-oriented C++ files. This tool has been developed by Mresa (2000) to explore and visualize the architectural components of object-oriented C++ program files. VOO++ developer has recommended that measurement techniques can be incorporated into VOO++ to estimate the quality of software systems and monitor its progress (Mresa, 2000).
- To introduce a new object-oriented metric that measures the understandability from the perspective of identifiers names. For this, a new version of the reverse engineering VOO++ tool will be developed and named VMCPP, Visualizing and Measuring C++ files. This tool will help the software developer understand the subject software system by:
    - Separating the metrics values of the subject software system components into critical values and normal values.
    - Visualizing the architectural components of a subject software system using the Unified Modeling Language (UML).

1.6 Thesis Organization

The remainder of this thesis is structured into five chapters. Chapter 2 explores the object-oriented metrics in general with emphasis on product metrics. Various object-oriented product metrics are discussed and explained in detail with examples to show their use. After that some thresholds have been collected with focusing on object-oriented product metrics. Chapter 3 explains the methodology chosen for building VMCPP. Chapter 4 presents the design, development, and usage of the VMCPP tool. Chapter 5 shows the validation of VMCPP and the results of applying the thresholds on real software projects. In chapter 6, the conclusions are discussed and the areas of future research as well as extensions to VMCPP are identified.

CHAPTER 2

SOFTWARE MEASUREMENT:

METRICS, THRESHOLDS AND REVERSE ENGINEERING TOOLS

"You can't control what you can't measure" Tom DeMacro.

2.1 Software Measurement

Measurement can be defined as 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 (Fenton & Pfleeger, 1996).

We use measurement equipments in our daily life to measure time, weight temperature…etc. These equipments provide us with very valuable information. For example, one could look at the home thermometer to decide whether the weather is suitable for out-door picnic or not. So, he may describe that day's temperature as "nice day," the word "nice" used to represent that day's temperature. People naturally have some kind of scaling in their minds, so they can imagine the weather without even going out and examine the weather by themselves.

In software engineering, software measurement tools are the equipments that measure software systems. These tools gather software metrics to understand the software system under developing, to provide guidelines that suggest a specific action to improve the quality of different system components, and to estimate quality of a software product.

2.2 Classification of Software Metrics

Software metrics are classified into two categories. The first one is the process metrics, those metrics deal with the developing process phases; they are used to estimate cost, time, and effort needed to complete a specific software project.

The second category is the product metrics, which is divided into two sections, internal product metrics, and external product metrics. While the internal product metrics deals mostly with the size, complexity, and style of the software under developing, the external product metrics deals with functionality, usability, and performance of the software. Figure 2.1 shows software metrics classifications.

2.3 Object-Oriented Product Metrics

Because of its flexibility, object technology has been widely adopted in the first half of the nineties. Since then, object technology becomes the ultimate choice for many software product builders. New engineering approaches have been introduced in every phase of the software life cycle for this new technology.

For software measurement professionals, the need for new metrics to measure object-oriented systems was raised dramatically, many new metrics have been proposed to specifically evaluate and quantify the object-oriented process and its products.