A MODEL OF SOFTWARE COMPONENT INTERACTIONS
USING THE CALL GRAPH TECHNIQUE

NOORAINI ISMAIL

MASTER OF SCIENCE (INFORMATION TECHNOLOGY)
UNIVERSITI UTARA MALAYSIA
2013
A MODEL OF SOFTWARE COMPONENT INTERACTIONS USING THE CALL GRAPH TECHNIQUE

A thesis submitted to the UUM College of Arts and Sciences in fulfilment of the requirements for the degree of Master of Science
Universiti Utara Malaysia

by
Nooraini Ismail

© 2013
Permission to Use

In presenting this thesis in fulfilment of the requirements for a postgraduate degree from Universiti Utara Malaysia, I agree that the Universiti Library may make it freely available for inspection. I further agree that permission for the copying of this thesis in any manner, in whole or in part, for scholarly purpose may be granted by my supervisor(s) or, in their absence, by the Dean of Awang Had Salleh Graduate School of Arts and Sciences. It is understood that any copying, 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 due recognition shall be given to me and to Universiti Utara, Malaysia for any scholarly use that may be made of any material from my thesis.

Requests 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 Awang Had Salleh Graduate School of Arts and Sciences
UUM College of Arts and Sciences
Universiti Utara, Malaysia
06010 UUM Sintok
Abstrak

Maklumat interaksi yang berkaitan dengan operasi antara komponen adalah penting, terutama apabila program perlu diubahsuai dan diselenggara. Oleh itu, komponen yang terlibat perlu dikenal pasti dan dipadankan berdasarkan keperluan sistem. Maklumat berkenaan boleh diperolehi menggunakan teknik kod ulasan. Walau bagaimanapun, proses ini mengambil masa yang panjang. Penyelidikan ini mencadangkan suatu model untuk mewakili maklumat tersebut yang mana ia diperolehi secara automatik daripada kod sumber untuk menyediakan paparan yang berkesan untuk perwakilan interaksi antara komponen perisian. Untuk mencapai objektif kajian, metodologi reka bentuk kajian yang mengandungi lima fasa telah diadaptasikan iaitu kesedaran kepada masalah, cadangan, pembangunan, penilaian dan kesimpulan. Fasa pembangunan mempunyai proses yang lebih terperinci yang mana maklumat interaksi antara komponen perlu diperolehi secara automatik menggunakan peralatan kejuruteraan balikan dan program tambahan. Program ini digunakan untuk mendapatkan maklumat perisian, maklumat interaksi komponen dalam program perisian, dan untuk mewakili model dalam bentuk call graph. Graf yang dihasilkan ini dinilai melalui dua cara, iaitu menggunakan alatan penggambaran yang bersesuaian dan juga melalui kajian oleh pakar. Alatan penggambaran digunakan untuk memaparkan graf yang dihasilkan daripada format teks ke paparan grafik. Proses penilaian model pula dilakukan melalui teknik kajian pakar. Hasil kajian ini menunjukkan bahawa model yang terhasil boleh digunakan dan dimanipulasikan untuk tujuan menggambarkan maklumat interaksi antara komponen. Model ini boleh digunakan untuk menyediakan paparan penggambaran bagi penganalisis untuk melihat interaksi maklumat yang relevan dalam komponen perisian. Ia juga dalam meningkatkan pemahaman mengenai integrasi komponen itu sendiri, supaya penganalisis boleh memanipulasi dan mengekalkan perisian untuk tujuan tertentu.

Kata kunci: Model interaksi komponen, Perwakilan call graph, Kefahaman program, Alatan penggambaran
Abstract

Interaction information that is related to operations between components is important, especially when the program needs to be modified and maintained. Therefore, the affected components must be identified and matched based on the requirement of the system. This information can be obtained through performing the code review technique, which requires an analyst to search for specific information from the source code, which is a very time consuming process. This research proposed a model for representing software component interactions where this information was automatically extracted from the source code in order to provide an effective display for the software components interaction representation. The objective was achieved through applying a research design methodology, which consists of five phases: awareness of the problem, suggestion, development, evaluation, and conclusion. The development phase was conducted by automatically extracting the components’ interaction information using appropriate reverse engineering tools and supporting programs that were developed in this research. These tools were used to extract software information, extract the information of component interactions in software programs, and transform this information into the proposed model, which was in the form of a call graph. The produced model was evaluated using a visualization tool and by expert review. The visualization tool was used to display the call graph from a text format into a graphical view. The processed model evaluation was conducted through an expert review technique. The findings from the model evaluation show that the produced model can be used and manipulated to visualize the component interactions. It provides a process that allows a visualization display for analysts to view the interaction of software components in order to comprehend the components integrations that are involved. This information can be manipulated and improved the program comprehension, especially for other software maintenance purposes.

Keywords: Component interaction model, Call graph representation, Program comprehension, Visualization tool
Acknowledgement

Alhamdulillah. A thousand gratitude to Allah S.W.T for giving me strengths to finish my studies.

First of all, I would like to express my deepest gratitude to my supervisor, Dr Nor Laily Hashim for her excellent guidance, moral support and advice throughout my studies.

For my thesis examiners, who give me their valuable comments and feedbacks.

I would like to acknowledge my sponsor, Ministry of High Education for granting sponsorship of my research study.

My dearest thanks I own to my husband Mohd Hafiz Abdullah, for his encouragement, patient, prayers and support throughout my studies.

I also thanks to my family, my masters colleagues and friends for their help and encouraged me through this thesis. May Allah bless us, InsyaAllah.
## Table of Contents

Permission to Use .......................... i
Abstrak .................................. ii
Abstract .................................. iii
Acknowledgement ......................... iv
Table of Contents .......................... v
List of Tables ............................... ix
List of Figures ............................. x

### CHAPTER ONE: INTRODUCTION 1

1.1 Overview ................................ 1
1.2 Introduction ............................ 1
1.3 Problem Statement ..................... 3
1.4 Research Questions ..................... 5
1.5 Research Objectives .................... 5
1.6 Research Scope ......................... 5
1.7 Significant of Study ................... 7
1.8 Overview of Thesis ..................... 8
1.9 Summary ............................... 9

### CHAPTER TWO: LITERATURE REVIEW 10

2.1 Introduction ............................ 10
2.2 Software Maintenance .................. 10
2.3 Software Comprehension ............... 14
   2.3.1 Cognitive models of program comprehension strategies 15
2.4 Software Component .................... 18
   2.4.1 Existing study concerning software component extraction techniques 21
2.5 Existing studies on related component extraction tools ............... 25
   2.5.1 Appropriate tool to be used ........... 26
2.6 Software representation
   2.6.1 Techniques Representation 29
   2.6.2 Program Visualization 30
   2.6.3 Technique for program visualization 34
      2.6.3.1 Visual Model 34
   2.6.4 Technique for model visualization 37
2.7 Dependence Graph 38
   2.7.1 Call graph 41
   2.7.2 Related studies using call graph 44
   2.7.3 Call graph techniques 45
   2.7.4 Call Graph Interpretation 49
   2.7.5 Application model 52
2.8 Graphviz as a visualization tool 52
   2.8.1.1 Model elaboration 54
      2.8.1.2 Representation for vertices and edges 54
2.9 Summary 55

CHAPTER THREE: RESEARCH METHODOLOGY 58
3.1 Introduction 58
3.2 Research Design 58
   3.2.1 Phase 1: Awareness of Problem 59
   3.2.2 Phase 2: Suggestion 60
      3.2.2.1 Research Framework 61
   3.2.3 Phase 3: Development 62
      3.2.3.1 Process 1: Sampling 63
      3.2.3.2 Process 2: Reverse engineering to extract software information 66
      3.2.3.3 Process 3: Component Extraction 67
   3.2.4 Phase 4: Evaluation 68
      3.2.4.1 Evaluation 1: Goal-based evaluation using existing tool 68
CHAPTER SIX: CONCLUSION

6.1 Achievement of Objectives
  6.1.1 Research Objective 1
  6.1.2 Research Objective 2
  6.1.3 Research Objective 3
6.2 Research Contributions
6.3 Limitation of study
6.4 Future Work

REFERENCES

APPENDIX A: RESPONDENT’S PROFILE
APPENDIX B: QUESTIONNAIRE FOR HEURISTIC REVIEW
APPENDIX C: DECLARATION
List of Tables

Table 2.1: Advantages of static analysis and dynamic extraction analysis 23
Table 2.2: Types of dependence graph representation 40
Table 2.3: Existing tools for software representation using graph 44
Table 3.1: Criteria used in the comparative study (adopted from Beacker, 1998) 73
Table 4.1: Symbol in model representation 76
Table 4.2: Parsing for parameters in ExtractJavaGenerator program 82
Table 4.3: Parsing for type of relationships in ExtractJavaGenerator program 82
Table 4.4: ExtractJavaGenerator tables descriptions 83
Table 4.5: Parsing for parameters in ExtractTestDataNew program 84
Table 4.6: Type of relationships in ExtractTestDataNew program 85
Table 4.7: ExtractTestDataNew tables descriptions 85
Table 4.8: The number of classes and interfaces that are extracted from the program 87
Table 4.9: The number of classes and interfaces that are extracted from the program
No. of classes 88
Table 5.1: Result of comparative study 103
List of Figures

Figure 2.1. Ten knowledge areas adapted from Abran et al. (2004) .................................................. 10
Figure 2.2. Area of software maintenance adapted from Abran et al. (2004) ....................................... 13
Figure 2.3. Graphical description of the interaction interface between two components .......................... 20
(adopted from Broy & Kruger, 1998)
Figure 2.5. Different techniques in representing software ................................................................. 29
Figure 2.6. Example of a matrix view showing component patterns for the top developers (adopted from Eick et al., 2002) .......................................................... 31
Figure 2.7. Example of a cityscape view (adopted from Eick et al., 2002) ............................................. 32
Figure 2.8. Example of a bar chart: numbers of software changes ...................................................... 33
Figure 2.9. Basic concept of dependence graph .................................................................................... 39
Figure 2.10(a). Contact-Insensitive Figure 2.10(b). Contact-Sensitive .................................................... 42
Figure 2.11. Design Behaviour Tree (DBT) ......................................................................................... 46
Figure 2.12. Edit behaviour tree (EBT) ................................................................................................. 46
Figure 2.13. Representing change nodes adopted from Lin et al. (2009) ............................................... 48
Figure 2.14. Transformation dependence call graph to expression language (adapted from Kuck et al., 1981) ........................................................................................................... 50
Figure 2.15. Transformation of dependence call graph (adapted from Ferrante et al., 1987) .................. 51
Figure 2.4. Graphical representation for call graph ................................................................................. 54
Figure 3.1. Research Design Methodologies (Vaishnavi & Kuechler, 2008) ........................................... 59
Figure 3.2. Research Framework ......................................................................................................... 61
Figure 3.3. Flow chart of the development process ................................................................................ 63
Figure 3.4. Screenshot of a JUnit HTML report (Source: http://docs.codehaus.org) ............................. 64
Figure 3.5. Screenshot example of UML2 0.9 (Source: http://www.sourceforge.net) ............................ 65
Figure 3.6. Interface Ptidej screenshot (Source: http://www.ptidej.net) ............................................... 67
Figure 3.7. Screenshot of the extract component program ...................................................................... 68
Figure 3.8. Graphviz screenshot ........................................................................................................... 69
Figure 4.1. Model representing a call graph .......................................................................................... 75
Figure 4.2. Screenshot of example output file in text format ................................................................. 78
Figure 4.3. Screenshot of example output file in UML document 79
Figure 4.4. Screenshot of ExtractJavaGenerator program 81
Figure 4.5. Screenshot of ExtractTestDataNew program 84
Figure 4.6. Screenshot of createGraph program 86
Figure 4.7. Screenshot dot file for JUnit 4.7 88
Figure 4.8. Screenshot dot file for UML2 0.9 89
Figure 4.9. Call graph produced for getName service from TestClass interface for JUnit 4.7 92
Figure 4.10. Call graph produced for new Instance service from Log Monitor Adapter interface for UML2 9.0 96
Figure 5.1. Evaluation process 99
CHAPTER ONE
INTRODUCTION

1.1 Overview
This chapter includes an overall research plan of this study by introducing the research foundation and motivation to be undertaken in this research. It also includes a detailed description of the issues to be studied, the research objectives, scope of the research, the research framework, and the contribution expected to be gained in this research.

1.2 Introduction
A software component can be a single element of software that can be integrated with other components (Szyperski, 1998). Two components are integrated if they can potentially react to the same events (Fiege, 2005), which is bypassing messages through their interfaces when the components are provided or required for specific events (Inverardi et al., 2003). The communication between components typically is realized by procedure calls or any kind of messaging (Bure et al., 2009).

When new components are integrated, a newly added component has an effect on another component and it can be used by other components. Because of this situation, the program may crash or immediately stop the execution of the system. For this reason, a programmer must scan through the program and investigate which components are causing the errors.
The contents of the thesis is for internal user only
REFERENCES


123


