

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

ON THE CLASSIFICATION OF CYCLIC DEPENDENCIES IN JAVA PROGRAMS

A thesis presented in partial fulfilment of the
requirements for the degree of

Master of Science
in
Computer Science

at Massey University, Manawatū, New Zealand



HUSSAIN ABDULLAH A. AL-MUTAWA

2013

ABSTRACT

Software engineering guidelines and rules discourage cyclic dependencies between modules, yet empirical studies have shown that many software systems are burdened with them. This might indicate that not all cycles are as detrimental to software quality as previously thought. Clearly, a better understanding of the types of cyclic dependencies and their effect on software quality is required. As a first step in this direction, we look closely at the shapes formed by software dependency graphs containing cyclic dependencies. Such cyclic dependencies correspond to the concept of strongly connected components in graph theory. We propose an approach to classify strongly connected components according to their topologies. This allows us to distinguish between dense and sparse, symmetric and asymmetric structures. We extend on previous studies and investigate the relationship between cyclic dependencies and the package containment tree. We validate our approach with experiments based on a corpus of 103 open-source Java systems. We find that cyclic dependencies tend to form in branches of the package containment tree around parent packages that are not critical according to some researchers.

ACKNOWLEDGEMENTS

I am in debt to my advisors, Jens Dietrich, Catherine McCartin, and Stephen Marsland for supervising me throughout the course of this thesis. This thesis would not have been possible without their guidance and continual help. When I had been doing my undergraduate degree, I once asked Jens: *“why cyclic dependency matters?”* Jens’s response was: *“why don’t you do some postgraduate research with me and find out for yourself?”* My deepest thanks goes to Jens for motivating me towards doing this research and for his continual and invaluable help in putting this thesis together. My thanks also goes to Catherine and Stephen for being tolerant and supportive and for the warm encouragement and guidance they provided.

My sincere thanks also goes to Janet George, Russell Johnson and Briony Coote for their help in proofreading this thesis.

Last but not least, my heartfelt thanks goes to my wife, for her kindness and support she has shown during the past eighteen months it has taken me to finalise this thesis. I would also like to thank my daughters Ayah and Alaa, and my son Ali, for their patience during my study. Furthermore, I would like to offer my special thanks to my parents for teaching me how to be persistent.

This project has been supported by funding from the Ministry of Higher Education, Saudi Arabia as part of the King Abdullah Foreign Scholarship Program (KASP).

CONTENTS

1	Introduction	1
1.1	The Growing Size and Complexity of Software	2
1.2	Coping with Complexity	5
1.2.1	Layered Design	5
1.2.2	The Acyclic Dependency Principle	5
1.2.3	The Acyclic Dependency Principle and the Package Containment Tree	8
1.3	Research Questions	9
1.4	Thesis Outline	9
2	Background and Related Work	11
2.1	Cycles in Directed Graphs	12
2.2	Standard Metrics from Graph Theory	14
2.2.1	Density (DENSE)	14
2.2.2	Ratio of number of vertices to number of edges	15
2.2.3	Minimum Feedback Edge Set	15
2.2.4	Mean Degree Centrality (DEG)	16
2.2.5	Diameter (DIAM)	16
2.2.6	Longest Path Length (LONG)	17
2.2.7	Betweenness Centrality	17
2.2.8	Tangledness (TANGL)	19
2.2.9	Transitivity (TRANS)	19
2.2.10	Size of the Automorphism Group (AUTO)	20
2.3	The Package Containment Tree	20
2.3.1	Building the Package Containment Tree	22
2.3.2	Metrics on Package Containment Tree	22
2.3.3	Reduced Package Containment Trees	23
2.4	Graph Topology	23
2.5	Removal of Cyclic Dependencies – Refactoring	25
2.6	Conclusion	26
3	Dependency Graphs and Tangles	29

3.1	The Level of Dependencies in Software	30
3.1.1	Statement Level Dependency Graph	30
3.1.2	Method Level Dependency Graph	31
3.1.3	Class Level Dependency Graph	31
3.1.4	Package Level Dependency Graph	33
3.1.5	Component Level Dependency Graph	33
3.2	Types and Levels of Tangles	33
3.2.1	Extracting Tangles from Dependency Graph	34
3.2.2	Class Tangles	36
3.2.3	Top-Level-Class Tangles	37
3.2.4	Weak Package Tangles	38
3.2.5	Strong Package Tangles	38
3.3	Building Dependency Graphs	40
3.4	Conclusion	41
4	Tangles Classification	43
4.1	Introduction	44
4.2	Reference Topologies	45
4.2.1	Symmetric Topologies	45
4.2.2	Asymmetric Topologies	48
4.3	A Set of Custom Metrics	49
4.3.1	The Depth of the SCC Decomposition Graph	49
4.3.2	Immediate Back-reference (BCKREF)	51
4.3.3	Starness (STAR)	51
4.3.4	Chainness (CHAIN)	52
4.3.5	Hubs (HUB)	52
4.4	Robustness Analysis	54
4.5	Classification of Tangles	57
4.6	Validating the Classifier	58
4.7	Classification of Qualitas Corpus Tangles	58
4.7.1	Raw Result Data	59
4.7.2	The Number of Tangles	60
4.7.3	The Size of Tangles	62
4.7.4	The Occurrence of Topologies	63
4.7.5	Tangles Morphology	65
4.8	Conclusion	67

5	Tangles and the Package Containment Tree	69
5.1	Introduction	70
5.2	Parent Centrality	70
5.3	The Shape of Tangles and the Package Containment Tree	71
5.4	Conclusion	73
6	Conclusions and Future Work	75
6.1	Conclusions	76
6.2	Future Work	77
A	Glossaries	93
B	ANTLR and Hibernate Version Data	97
B.1	ANTLR	97
B.2	Hibernate	97
C	Qualitas Corpus Dataset	99

LIST OF FIGURES

Microsoft Windows operating system size over a decade	2
A small example graph with eight vertices and nine edges	3
PDGs of different versions of ANTLR.	4
PDGs of different versions of Hibernate.	4
Layered design	5
A comparison between sparse and dense Strongly Connected Components (SCCs) formed by cyclic dependencies	6
PDGs of different versions of ANTLR with emphasis on cyclic dependencies (bold edges).	7
PDGs of different versions of Hibernate with emphasis on cyclic dependencies (bold edges).	7
A partial Package Containment Tree (PCT) of <code>java.awt</code> , <code>javax.swing</code> and some of their sub-packages.	8
A partial PCT of <code>java.awt</code> , <code>javax.swing</code> and some of their sub-packages. The labels on edges represent the number of class-to-class dependencies made from one package to another.	8
Three families of directed graphs, namely: disconnected, connected and strongly connected.	12
A dependency graph that contains three tangles	13
The relationship between <code>RATIO</code> , <code>DENSE</code> and the shape of tangle.	15
The relationship between Minimum Feedback Edge Set (MFES) and the shape of tangle.	16
The relationship between <code>DEG</code> and the shape of tangles.	16
The relationship between <code>DIAM</code> , <code>LONG</code> and the shape of tangles.	17
The relationship between vertex betweenness and the shape of tangles	18
The package containment tree for selected core Java packages.	21
The normal and reduced PCTs of NekuHTML-1.9.14 system.	23
Cycle, tall, flat and balanced binary Package Dependency Graphs (PDGs).	24
Controlled vs. Pancaked structure.	24
An example of call graph	31
Example classes and packages (UML class diagram)	34
The package graph G_p	34

The class graph G_c	34
The top-level-class graph G_{tlc}	34
Class tangles, $T_c = \{\{A, B, C, C\$1\}\}$	36
Top Level Class tangles, $T_{tlc} = \{\{A, B, C\}\}$	37
Weak package tangles, $T_p^w = \{\{P1, P2, P3\}\}$	38
Strong package tangles, $T_p^s = \{\{P1, P2\}\}$	39
Symmetric tangle topologies.	46
Asymmetric tangle topologies.	48
A SCC and its elementary cycles.	50
Decomposition graph.	50
SCC decomposition graphs of some tangles.	51
Graphical representation of the Gini coefficient.	53
Some example of tangles and their HUB values.	53
HUB metric can be misleading on very dense tangles.	54
TANGL score for variations of the circle and star topologies	55
Classification algorithm	57
An example dependency graph of the system shown in Listing 4.8.	60
Distribution of tangle topologies on the corpus.	61
A package can be a member of more than one strong package tangle.	62
Package boundaries and strong package tangles.	62
The distribution of tangles topologies on the corpus.	64
Star-like package dependencies in Swing.	64
Top-level-class tangle derived from non tangled class graph.	66
Summary of tangles morphology.	67
Parent centrality results on the corpus.	71
The package containment tree for some selected core Java packages.	71
ACLOSE distribution in package tangles.	72
DCLOSE distribution in package tangles.	72
Strongly connected components extracted from dependency graph.	94

LIST OF TABLES

Some differences between source code and byte code.	40
Some tools and research done in analysing cyclic dependencies.	41
List of projects in Qualitas Corpus which contain multiple classes that have the same qualified name.	45
Boxplots of metrics scores on different tangles topologies	56
Metrics scores on different tangles topologies	56
Un-mutated tangles classification confusion matrix	58
Mutated tangles classification confusion matrix	58
System specifications of the workstation used in performing the experiment.	59
Execution time of the classification algorithm on the corpus in milliseconds.	59
Breakdown of tangles and their topologies.	61
The size of class tangles.	63
The size of package tangles.	63
Class to top-level-class tangles topologies change in morphology	65
Top-level-class to strong package tangles topologies change in morphology	65
Weak package to strong package tangles topologies change in morphology	66
Package containment tree metrics.	72
Some measures among different versions of ANTLR system.	97
Some measures among different versions of Hibernate system.	97

LISTINGS

2.1	An example of extends reference	13
2.2	An example of uses reference	13
3.1	An example of data dependence	31
3.2	An example of control dependence	31
3.3	An example Java Program	31
3.4	An example of a static nested class	32
3.5	An example of an inner class	32
3.6	An example of an anonymous class	33
3.7	Tarjan's algorithm implementation in Java	35
3.8	The method used to build tangles from dependency graphs.	36
3.9	The method used to build class tangles.	36
3.10	The method used to build the top-level-class tangles.	37
3.11	The method used to build the weak package tangles.	38
3.12	The method used to build the strong package tangles.	39
3.13	The source code and byte-code of class A	40
3.14	The source code and byte-code of class B	40
4.1	Tiny tangle generation function.	46
4.2	Circle tangle generation function.	46
4.3	Clique tangle generation function.	47
4.4	Chain tangle generation function.	47
4.5	Star tangle generation function.	48
4.6	Noisfy tangle function.	48
4.7	Semi-clique tangle generation function	49
4.8	Dependency graph, tangles and their topologies of a system stored in JSON data format.	60

ACRONYMS

ADP Acyclic Dependency Principle

ANTLR ANother Tool for Language Recognition

ASPL Average Shortest Path Length

AWT Abstract Window Toolkit

DAG Directed Acyclic Graph

DFS Depth First Search

JAR Java Archive

JRE Java Runtime Environment

JSON JavaScript Object Notation

LOC Lines of Code

MFES Minimum Feedback Edge Set

ORM Object-Relational Mapping

OSI Open System Interconnection

PCT Package Containment Tree

PDG Package Dependency Graph

SCC Strongly Connected Component

SVM Support Vector Machine

UML Unified Modelling Language