

**AN IMPROVED HIERARCHICAL CLUSTERING COMBINATION APPROACH
FOR SOFTWARE MODULARIZATION**

RASHID NASEEM

**A thesis submitted in
fulfillment of the requirement for the award of the
Doctor of Philosophy in Information Technology**

**Faculty of Computer Science and Information Technology
Universiti Tun Hussein Onn Malaysia**

JANUARY 2017

To my family.



ACKNOWLEDGEMENT

First, I want to thank ALLAH Almighty for giving me the strength and courage to accomplish my goal. HE has been the biggest source of strength for me.

I would like to express my deepest gratitude to my supervisor Prof. Dr. Mustafa Mat Deris. It has been an honor to be his PhD student. I am grateful for all his contributions of time, ideas, and knowledge to make my research experience productive and exciting.

I benefited from the expertise and support of Dr. Onaiza Mabool of Quaid-I-Azam University, Pakistan, in all aspects of my research work. Her guidance and instructions were particularly valuable during preparing research papers and the thesis.

I would like to thank University Tun Hussein Onn Malaysia (UTHM) for supporting this research under Graduate Researcher Incentive Grant (GIPS) Vote No. U063.

I am grateful to Siraj Muhammad and Abdul Qadoos Abbasi who shared my interests and helped me a lot to clarify my views through conversations and implementation concerning my research and provide me the test systems for my research.

I would like to show my appreciation and pleasure to all my friends from Nigeria, Somalia, Iraq, Yemen, Malaysia, and Pakistan, especially the friends of Parit Raja (our house on rent in Batu Pahat, Malaysia). The loving and caring friends of Parit Raja: Zinda Abad.

Lastly, I would like to thank my family for all their pray, love and encouragement. The moral support of all my family member was the main stream to complete my PhD.

Thanks Malaysia.

Rashid Naseem

ABSTRACT

Software modularization plays an important role in software maintenance phase. Modularization is the breaking down of a software system into sub-systems so that most similar entities (e.g., classes or functions) are collected in clusters to get the modular architecture. To check the accuracy of collected clusters, authoritativeness is calculated which finds the correspondence between collected clusters and a software decomposition prepared by a human expert. To improve the authoritativeness, different techniques have been proposed in the literature. However, *agglomerative hierarchical clusterings* (AHCs) are preferred due to their resemblance with internal tree structure of the software systems because AHC results in a tree like structure, called dendrogram. AHC uses similarity measures to find association values between entities and makes clusters of similar entities. This research addresses the strengths and weakness of existing similarity measures (i.e., Jaccard (JC), JaccardNM (JNM), and Russal&Rao (RR)). For example JC measure produces large number of clusters (NoC) and number of *arbitrary decisions* (AD). Large NoC is considered to be better for improving the authoritativeness but large AD deteriorates it. To overcome this trade-off, new combined binary similarity measures are proposed. To further improve the authoritativeness, this research explores the idea of *hierarchical clustering combination* (HCC) for software modularization which is based on combining results (dendrograms) of individual AHCs (IAHCs). This research proposes an improved HCC approach in which the dendrograms are represented in a 4+N (4 is the number of features and can be extended to N) dimensional Euclidean space (4+NDES). The proposed binary similarity measures and 4+NDES based HCC approach are tested on several test software systems. Experimental results revealed [13.5% - 63.5%] improvement in authoritativeness as compared to existing approaches. Thus the combined measures and 4+NDES-HCC have shown better potential to be used for software modularization.

ABSTRAK

Modularisasi perisian memainkan peranan yang penting dalam fasa penyelenggaraan perisian. Modularisasi adalah pecahan daripada sistem perisian ke dalam sub-sistem supaya kebanyakan entiti yang sama (contohnya, kelas atau fungsi) dikumpulkan dalam kelompok tertentu untuk mendapatkan seni bina modular. Untuk memeriksa ketepatan kelompok yang telah dikumpul, keberkesanan dikira dari segi kesesuaian di antara kelompok yang telah dikumpul dan penguraian perisian yang diperincikan oleh pakar. Untuk meningkatkan keberkesanannya, pelbagai teknik yang berbeza telah dicadangkan dalam literatur. Walau bagaimanapun, *agglomerative hierarchical clusterings* (AHCs) lebih digemari kerana persamaan struktur dalaman sistem perisian kerana AHC membentuk sebuah struktur berbentuk pokok, yang dipanggil dendrogram. AHC menggunakan penilaian persamaan untuk mencari nilai-nilai persatuan antara entiti dan membentuk kelompok entiti yang sama. Kajian ini menangani kekuatan dan kelemahan penilaian persamaan yang sedia ada (iaitu, Jaccard (JC), JaccardNM (JNM), dan Russel&Rao (RR)). Sebagai contoh, teknik penilaian JC menghasilkan sejumlah besar kelompok (NoC) dan beberapa *arbitrary decisions* (AD). NoC dianggap lebih baik untuk meningkatkan keberkesanan, tetapi AD besar kemungkinan merosot dari aspek tertentu. Untuk mengatasi masalah keseimbangan ini, langkah-langkah persamaan binari gabungan baru telah dicadangkan. Untuk meningkatkan lagi kewibawaan, penyelidikan ini meneroka idea *hierarchical clustering combination* (HCC) untuk perisian modularisasi yang berasaskan menggabungkan hasil (dendrograms) individu AHCs (IAHCs). Kajian ini mencadangkan pendekatan HCC bertambah baik di mana dendrograms diwakili dalam $4+N$ (4 adalah bilangan ciri-ciri dan boleh dilanjutkan kepada N) dimensi ruang Euclidean ($4+N$ DES). Kajian ini telah melaksanakan langkah-langkah persamaan binari yang dicadangkan dan $4+N$ DES berdasarkan pendekatan HCC telah diuji pada beberapa sistem perisian ujian. Oleh itu $4+N$ DES-HCC dan langkah-langkah persamaan binari yang dicadangkan adalah lebih berpotensi untuk digunakan untuk perisian modularisasi.

CONTENTS

DECLARATION	ii
DEDICATION	iii
ACKNOWLEDGEMENT	iv
ABSTRACT	v
ABSTRAK	vi
CONTENTS	vii
LIST OF TABLES	xi
LIST OF FIGURES	xiv
LIST OF ALGORITHMS	xvi
LIST OF APPENDICES	xvii
LIST OF SYMBOLS AND ABBREVIATIONS	xviii
LIST OF PUBLICATIONS	xx
 CHAPTER 1 INTRODUCTION	 1
1.1 Overview	1
1.2 Problem Statement	5
1.3 Objectives	6
1.4 Scope	7
1.5 Dissertation Outline	7
 CHAPTER 2 LITERATURE REVIEW	 9
2.1 Introduction	9
2.2 Software Maintenance	9
2.3 Aspects of Software Modularization using Clustering	11
2.4 Application of IAHC for Software Modularization	14
2.4.1 Selection of Entities and Features	14
2.4.2 Selection of Similarity Measure	16

2.4.3	Selection of the Linkage Method	18
2.5	Software Modularization using IAHCs	20
2.6	Software Modularization using Non-IAHCs	34
2.6.1	Search based Software Modularization	34
2.6.2	Graph based Approaches	37
2.6.3	Lexical Information based Approach	39
2.6.4	Other Approaches	40
2.7	Comparative Analysis	41
2.8	Hierarchical Clustering Combinations (HCC)	43
2.9	Works Related to Evaluation Criteria	46
2.10	Overall Scenario	50
2.11	Chapter Summary	51

CHAPTER 3 RESEARCH METHODOLOGY 52

3.1	Introduction	52
3.2	Phase 1	52
3.2.1	Study the Performance of Existing Similarity Measures	54
3.2.2	Combined Binary Similarity Measures	54
3.2.3	Comparison of Combined Binary Similarity Measures	54
3.2.3.1	Test Systems	54
3.2.3.2	Entities and Features	56
3.2.4	Strategies for Comparing Combined Measures	57
3.2.5	Selected Assessment Criteria for Phase 1	59
3.2.5.1	External Assessment	60
3.2.5.2	Expert Decompositions	60
3.2.5.3	Internal Assessment	61
3.2.6	Research Questions	62
3.3	Phase 2	62
3.3.1	Study the Performance of the Existing HCCs	62
3.3.2	4+N Dimensions in Euclidean Space Based HCC	63
3.3.3	Selection of the Best Parameters	63
3.3.4	Comparison of HCC	63
3.3.5	Selected Evaluation Criteria for HCC	65
3.4	Chapter Summary	67

CHAPTER 4	COMBINED BINARY SIMILARITY MEASURES	68
4.1	Introduction	68
4.2	Analysis of the JC, JNM, and RR Measures	68
4.2.1	JC with CL Clustering Process	69
4.2.2	JNM with CL Clustering Process	70
4.2.3	RR with CL Clustering Process	70
4.2.4	Discussion on the Results of JC, JNM and RR Measures	72
4.3	Design of the Combined Binary Similarity Measures	76
4.3.1	Combination of the JC and JNM Measures: “CJCJNM” Similarity Measure	76
4.3.2	Analysis of CJCJNM Measure	77
4.3.3	Combination of the JC and RR Measures: “CJCRR” Similarity Measure	83
4.3.4	Combination of the JNM and RR Measures: “CJNMRR” Similarity Measure	83
4.3.5	Addition of the JC and JNM and RR Measures: “CJCJNMRR” Similarity Measure	84
4.4	Chapter Summary	85
CHAPTER 5	EUCLIDEAN SPACE BASED HIERARCHICAL CLUSTERING COMBINATION	86
5.1	Introduction	86
5.2	4+NDES-HCC Algorithm	87
5.2.1	Step 1, Initialization of the Variables	87
5.2.2	Step 2, Repeat	88
5.2.3	Step 3, Applying Base Clusterer (IAHC)	88
5.2.4	Step 4, Dendrogram Descriptor	88
5.2.4.1	(4+N) dimensions in a Euclidean Space Based Descriptor:	89
5.2.4.2	BE Features	90
5.2.4.3	ES Features	91
5.2.5	Step 5, Loop Termination	92
5.2.6	Step 6, Vectors Addition	92
5.2.7	Step 7, Euclidean Distance	93
5.2.8	Step 8, Recovery Technique	94
5.3	Illustration of (4+NDES-HCC) Algorithm	94
5.4	Complexity Analysis	94
5.5	Chapter Summary	96

CHAPTER 6	RESULTS AND ANALYSIS	97
6.1	Introduction	97
6.2	Assessment of the Combined Similarity Measures	97
6.2.1	Arbitrary Decisions	97
6.2.2	Number of Clusters	106
6.2.3	Authoritativeness	115
6.2.4	Overview of Results	119
6.2.5	Comparison with COUSM	121
6.2.6	Significance of the Authoritativeness	122
6.3	Assessment of the 4+NDES-HCC	123
6.3.1	Number of Clusters	123
6.3.2	Cluster-to-Cluster (c2c) Comparison	126
6.3.3	Authoritativeness	128
6.3.4	Significance of the Results	131
6.4	Overview of the Results for NDES	132
6.5	Chapter Summary	134
CHAPTER 7	CONCLUSIONS AND FUTURE WORKS	135
7.1	Introduction	135
7.2	Objectives Accomplished	136
7.3	Contributions	139
7.4	Directions for Future Work	140
7.4.1	Integration of More Similarity Measures	140
7.4.2	Integration of Partitioning Algorithms with Hierarchical Algorithms	141
7.4.3	Application of Consensus Based Techniques	141
7.4.4	Feature Extraction	141
7.4.5	Description of Dendrogram	141
7.4.6	Experiments on Other Test Systems	142
7.4.7	Use of Other Assessment Criteria	142
7.4.8	Big Data	142
	REFERENCES	143
	APPENDIX A	160
	VITAE	169

LIST OF TABLES

2.1	An Example Feature Matrix	15
2.2	Similarity Matrix of Table 2.1 using the JC Measure	18
2.3	Iteration 1: Updated Similarity Matrix from Table 2.2 using the CL Method	18
2.4	Details of the Literatures	29
3.1	Details of the Test Software Systems	55
3.2	Relationships Between Classes that were used for Experiments	57
3.3	Statistics for the Indirect Features of Classes in Industrial Test Software Systems that were Used for Experiments	57
3.4	Clustering Stratagems for IAHCs	59
3.5	Personnel Statistics	61
3.6	Clusterers Stratagems for HCC	66
4.1	Iteration 2: Updated Similarity Matrix from Table 2.3	69
4.2	Iteration 3: Updated Similarity Matrix from Table 4.1	69
4.3	Iteration 4: Updated Similarity Matrix from Table 4.2	69
4.4	Iteration 5: Updated Similarity Matrix from Table 4.3	70
4.5	Iteration 6: Updated Similarity Matrix from Table 4.4	70
4.6	Iteration 7: Updated Similarity Matrix from Table 4.5	70
4.7	Similarity Matrix of Feature Matrix in Table 2.1 Using the JNM Measure	71
4.8	Iteration 1: Updated Similarity Matrix from Table 4.7 Using the CL Method	71
4.9	Iteration 2: Updated Similarity Matrix from Table 4.8	71
4.10	Iteration 3: Updated Similarity Matrix from Table 4.9	71
4.11	Iteration 4: Updated Similarity Matrix from Table 4.10	72
4.12	Iteration 5: Updated Similarity Matrix from Table 4.11	72
4.13	Iteration 6: Updated Similarity Matrix from Table 4.12	72
4.14	Iteration 7: Updated Similarity Matrix from Table 4.13	72
4.15	Similarity Matrix of Feature Matrix in Table 2.1 Using the RR Measure	73
4.16	Iteration 1: Updated Similarity Matrix from Table 4.15 Using the CL Method	73

4.17	Iteration 2: Updated Similarity Matrix from Table 4.16	73
4.18	Iteration 3: Updated Similarity Matrix from Table 4.17	73
4.19	Iteration 4: Updated Similarity Matrix from Table 4.18	74
4.20	Iteration 5: Updated Similarity Matrix from Table 4.19	74
4.21	Iteration 6: Updated Similarity Matrix from Table 4.20	74
4.22	Iteration 7: Updated Similarity Matrix from Table 4.21	74
4.23	Similarity Matrix of Feature Matrix in Table 2.1 using the CJCJNM Measure	79
4.24	Iteration 1: Updated Similarity Matrix from Table 4.23 using the CL Method	79
4.25	Iteration 2: Updated Similarity Matrix from Table 4.24	79
4.26	Iteration 3: Updated Similarity Matrix from Table 4.25	79
4.27	Iteration 4: Updated Similarity Matrix from Table 4.26	80
4.28	Iteration 5: Updated Similarity Matrix from Table 4.27	80
4.29	Iteration 6: Updated Similarity Matrix from Table 4.28	80
4.30	Iteration 7: Updated Similarity Matrix from Table 4.29	80
6.1	Experimental Results using Arbitrary Decisions for all Similarity Measures	104
6.2	Average Number of Arbitrary Decisions for all Similarity Measures	106
6.3	Experimental Results using Number of Clusters for all Similarity Measures	111
6.4	A f_{vc} for PLC Test System	114
6.5	Average Number of Clusters for All Similarity Measures	115
6.6	MoJoFM Results for all Similarity Measures	116
6.7	Correlations of MoJoFM with Arbitrary Decisions (AD) and Number of Clusters (NoC)	117
6.8	Average MoJoFM Results for all Similarity Measures	119
6.9	Experimental Results using MoJoFM for COUSM (J-JNM) and Combined Measures	121
6.10	Statistics of f_{cv_1} , f_{vc_2} and f_{vc_3}	122
6.11	The T-test Values between the MoJoFM Results of the Combined and Existing Measures using CL and SL Method	122
6.12	Descriptive Statistics of Results Achieved for Number of Clusters	125
6.13	Percentage Improvement of Number of Clusters for NDES over CD and IAHCs	126
6.14	Descriptive Statistics of Results Achieved for c2c	127
6.15	Percentage Improvement of c2c Values for NDES over CD and IAHCs	128

6.16	Descriptive Statistics of Results Achieved for MoJoFM	128
6.17	Percentage Improvement of MoJoFM Values for NDES over CD and IAHCs	130
6.18	The T-test Values between the MoJoFM Results of the NDES and, CD and IAHC	132



LIST OF FIGURES

1.1	Hierarchical clustering approaches and dendrogram	3
1.2	General framework of the HCC	4
2.1	Direct and indirect relationships in a C++ input/output libraries ¹	13
2.2	IAHC Approach	16
2.3	Dendrogram	20
2.4	General HCC approach	44
3.1	Proposed research framework	53
3.2	Improved IAHC	58
3.3	IAHCs with combined similarity measures and linkage methods	58
3.4	Proposed 4+NDES-HCC approach	64
4.1	Number of clusters and arbitrary decisions created by JC and JNM	75
5.1	The 4+N dimensions in euclidean space based HCC model	88
5.2	Different steps of the 4+NDES-HCC algorithms. Two dendrograms (D^1 and D^2) are described using 4+NDES and then V-Matrices are aggregated into AV-Matrix. Lastly a recovery tool is used to get the final hierarchy FH	95
6.1	Experimental results of arbitrary decisions for all binary measures using IAHCs	99
6.1	Experimental results of arbitrary decisions for all binary measures using IAHCs (Continued)	100
6.1	Experimental results of arbitrary decisions for all binary measures using IAHCs (Continued)	101
6.2	Arbitrary decisions taken during clustering process by all binary similarity measures using CL and SL methods for different test software systems.	105
6.3	Experimental results for the number of clusters	107
6.3	Experimental results for the number of clusters (continued)	108
6.3	Experimental results for the number of clusters (continued)	109

6.4	Number of clusters created by all binary similarity measures during clustering process using CL and SL methods for different test software systems.	113
6.5	Size of the clusters created using DDA test system and CL method	118
6.6	Results for number of clusters created by different stratagems	124
6.7	Number of clusters created by different stratagems in each iteration	125
6.8	Results for c2c created by different stratagems	127
6.9	c2c results for each iteration created by different stratagems	129
6.10	Results for authoritativeness created by different stratagems	129
6.11	Authoritativeness results for each iteration created by different stratagems	131
6.12	Percentage improvement of NDES with respect to existing measures (JC, JNM, and RR) using CL, SL, and WL methods	133



LIST OF ALGORITHMS

1	Individual Agglomerative Hierarchical Clustering (IAHC) Algorithm	15
2	Complete Linkage (CL) Method	19
3	Single Linkage (SL) Method	19
4	Weighted Average Linkage (WL) Method	19
5	4+N Dimensions in Euclidean Space based HCC	89



LIST OF APPENDICES

A	Propositions	160
----------	---------------------	------------



LIST OF SYMBOLS AND ABBREVIATIONS

AHC	–	Agglomerative Hierarchical Clustering
ACDC	–	Algorithm for Comprehension Driven Clustering
c2c	–	Cluster to Cluster
CLH	–	Cannot Link Hard
CH	–	Cluster Cohesion
CC/G	–	Graph-based Clustering Approach
CPCC	–	Co-Phenetic Correlation Co-efficiency
CL	–	Complete Linkage
COUSM	–	Cooperative Only Update Similarity Matrix
DDA	–	Document Designer Application
DSM	–	Dependency Structure Matrix
dsm	–	design structure matrix clustering
EdgeSim	–	Edge Similarity
eb	–	edge betweenness clusterin
ECA	–	equal size cluster approach
FCA	–	Formal Concept Analysis
FES	–	Fact Extractor System
FACA	–	Fast Community Algorithm
GA	–	Genetic Algorithm
HCC	–	Hierarchical Clusterers Combinations
IAHC	–	Individual Agglomerative Hierarchical Clustering
IEEE	–	The Institute of Electrical and Electronics Engineers
IL	–	Information Loss
J-JNM	–	Jaccard and JaccardNM
JC	–	Jaccard
JNM	–	JaccardNM
LIMBO	–	Scalable Information Bottleneck
LSI	–	Latent Semantic Indexing
Max	–	Maximum
MQ/mq	–	Modularization Quality
MLH	–	Must Link Hard
MCA	–	Maximizing Cluster Approach

MST	–	Minimum Spanning Tree
MMST	–	Modified Minimum Spanning Tree
MoJoFM	–	Move and Join Effectiveness Measure
MoJo	–	Move and Join
MED	–	Maximum Edge Distance
MeCl	–	Merge Clusters
MATCH	–	Min-transitive Combination of Hierarchical Clusterings
NAHC	–	Nearest hill-climbing
NoC	–	Number of Clusters
NDES	–	N Dimensional Euclidean Space
OAM	–	Optimal Algorithm for MoJo
PEDS	–	Power Economic Dispatch System
PLC	–	Printer Language Converter
PLP	–	Print Language Parser
RR	–	Russal&Rao
SAVT	–	Statistical Analysis Visualization Tool
SBSE	–	Search Based Software Engineering
SBSC	–	Search Based Software Clustering
SL	–	Single Linkage
SAHC	–	Steepest-Ascend Hill-Climbing
SD	–	Sorenson-Dice
std.dev.	–	Standard Deviation
sig.	–	Significance
TF.IDF	–	Term Frequency Inverse Document Frequency
WCA	–	Weighted Combined Algorithm
WL	–	Weighted Average Linkage

LIST OF PUBLICATIONS

- Rashid Naseem, Mustafa Mat Deris, Onaiza Maqbool, Jing-peng Li, Sara Shahzad, and Habib Shah (2016), Improved binary similarity measures for software modularization, *Frontiers of Information Technology & Electronic Engineering (FITEE)*, In press (ISI and Scopus Indexed, Springer Journal)
- Rashid Naseem, Mustafa Mat Deris and Onaiza Maqbool (2014), Software Modularization Using Combination of Multiple Clustering, *IEEE 17th International Multi-Topic Conference (INMIC)*, pp. 277 - 281, IEEE Conference
- Rashid Naseem and Mustafa Mat Deris (2016), A New Binary Similarity Measure Based on Integration of the Strengths of Existing Measures: Application to Software Clustering, *International Conference on Soft Computing and Data Mining (SCDM)*, pp. 304 - 315, Springer Conference



CHAPTER 1

INTRODUCTION

1.1 Overview

The software development life cycle comprises of many phases, perhaps the most important phase being maintenance, because it increases the operational life of a software after it has been deployed. Software maintenance is considered to be a difficult phase since it dominates other phases in terms of cost and efforts required (Garcia *et al.*, 2011). Bavota *et al.* (2013) stated that a typical system's maintenance costs up to 90% of the total project expenses. In addition to that, Kumari *et al.* (2013) reported that maintenance phase requires 50% of the human and computer resources. According to Antonellis *et al.* (2009), United States spends \$60 billion per year of its economy on the maintenance of software systems. Keeping in view the above, software maintenance may be considered a competitive research area to reduce the efforts required for maintenance.

In the software maintenance phase, "*understanding the architecture*" of software systems is an important and challenging activity to start adding new requirements or to start reverse engineering (Muhammad *et al.*, 2012). However, when new requirements are incorporated in software systems, the systems' size and complexity increases (Shtern & Tzerpos, 2014), while architecture may deteriorate and diverge from the original documentation (Shtern & Tzerpos, 2014; Chardigny *et al.*, 2008) due to the many reasons: 1) software development without an architecture design phase; 2) original developer may not be available; 3) documentation may not be uptodated and; 4) copied source code without understanding the code segments.

Therefore, it becomes difficult to understand the systems and introduce changes to them. Corazza *et al.* (2011), Cornelissen *et al.* (2009) and Kuhn *et al.* (2007) reported that “understanding” the systems costs up to 60% of the whole maintenance cost. Hence, researchers have explored the problem and are working to solve it using automated techniques for the last two decades. These techniques support the software maintenance team in terms of gathering and presenting the basic architectural information for better understanding and improvement of the software systems. Architectural information comprises of a software architecture (structures of the systems, e.g., module architecture), which plays a vital role to understand the software systems (Ducasse & Pollet, 2009). Therefore, a number of approaches have been developed and proposed in the literature to recover the architecture of software systems mainly from their source code in order to improve the authoritativeness. Authoritativeness determines how much the automatically obtained decomposition is similar to manual decomposition prepared by human expert (Wu *et al.*, 2005). Besides hierarchical clustering (Maqbool & Babri, 2007b; Cui & Chae, 2011; Andritsos & Tzerpos, 2005), these approaches include, supervised clustering (Hall *et al.*, 2012), optimization techniques (Praditwong *et al.*, 2011), role based recovery (Dugerdil & Jossi, 2009), graph based techniques (Bittencourt & Guerrero, 2009), association based approaches (Vasconcelos & Werner, 2007), spectral method (Xanthos & Goodwin, 2006), rough set theory (Jahnke, 2004), concept analysis (Tonella, 2001), and visualization tools (Synytskyy *et al.*, 2005).

Clustering is an emerging research area to acquire different types of knowledge from the data by forming meaningful clusters (groups). Thus, entities within a cluster have similar characteristics or features, and are dissimilar from entities in other clusters. To determine similarity based on features of an entity, a similarity measure is employed. Finding a good clustering is an NP-complete problem (Tumer & Agogino, 2008). To address the problem, a number of clustering algorithms exist. Moreover algorithms have been proposed for various domains, e.g., DNA analysis (Avogadri & Valentini, 2009), software modularization (Wang *et al.*, 2010), bioinformatics (Janssens *et al.*, 2007), image processing (Gong *et al.*, 2013) and information retrieval

(Campos *et al.*, 2014) and for general data mining (Liao *et al.*, 2012). Each clustering algorithm produces results according to some criteria and bias of the technique (Faceli *et al.*, 2007). For example, Complete linkage clustering algorithm creates small size of clusters because this algorithm considers maximum distance between clusters while Single linkage algorithm considers minimum distance between the clusters and therefore makes large size clusters.

Clustering techniques can be mainly divided into two categories: 1) partitional and 2) hierarchical. Partitional clustering makes flat partitions (or clusters) in the input data, while hierarchical clustering results in a tree like nested structure of clusters called “*dendrogram*”. The two main types of hierarchical clustering are divisive and agglomerative. The divisive approach starts by considering the whole data as one big cluster and then iteratively splits the clusters into two nested clusters in a top-down manner. Agglomerative hierarchical clustering (AHC) starts with singleton clusters and merges two most similar clusters at every step in a bottom-up manner as shown in Figure 1.1.

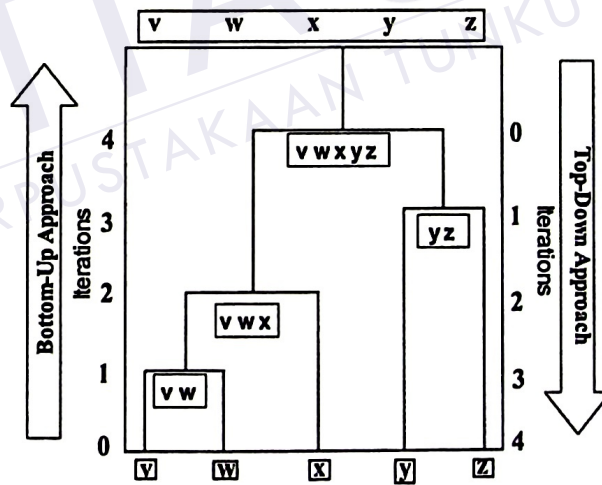


Figure 1.1: Hierarchical clustering approaches and dendrogram

Due to the ill-posed problem of clustering (Kashef & Kamel, 2010; Ghaemi *et al.*, 2009), different clustering algorithms usually produce different results from the given data. For example, if a clustering algorithm produces results which are stable (results are stable if affect of slight modification in input is also slight on output

of algorithm), it may produce less authoritative results and vice versa. To improve clustering results many of the existing methods have been modified and improved (Chen *et al.*, 2014; Wang & Su, 2011; Forestier *et al.*, 2010; Kashef & Kamel, 2010). Some efforts have been made to improve the results by using prior information, e.g., user input and number of clusters, but this information is very difficult to obtain in advance (Ghaemi *et al.*, 2009).

Recently, combining the individual AHC (IAHC) algorithms in an ensemble fashion has gained the attention of the researchers to boost the clustering accuracy, known as Hierarchical Clustering Combination (HCC). This ensemble clustering is achieved by combining the dendrograms created by different IAHCs, to obtain a consensus result (Rashedi *et al.*, 2015; Zheng *et al.*, 2014). Generally this combination has four phases as shown in Figure 1.2. In the first phase the IAHCs are applied to the input data to create dendrograms. Then the dendrograms are translated into an intermediate format (mostly in a matrix format called *Description Matrix*), so that they can be aggregated into a single intermediate format. Lastly a recovery tool is applied to recover a single consensus integrated dendrogram for the input data.

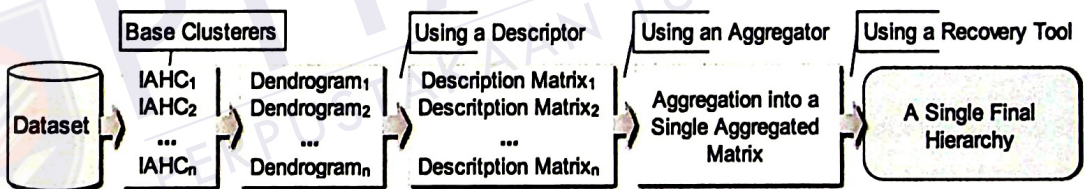


Figure 1.2: General framework of the HCC

For software modularization AHC algorithms have been widely used by researchers to cluster the software systems in order to improve the authoritativeness (Muhammad *et al.*, 2012; Shtern & Tzerpos, 2010; Patel *et al.*, 2009; Maqbool & Babri, 2007b; Mitchell, 2006). However, the similarity measures used by AHCs perform better for one assessment criteria and poor for another. To overcome this trade off, integration of the existing similarity measures are performed. Moreover, HCC which has never been explored for software modularization is introduced. In addition to that, the existing HCC approach has the limitation of describing dendrogram using only

a single feature and considering that feature as a distance value to make the clusters for final dendrogram. Therefore, in this research, a new improved HCC approach is proposed which is based on Euclidean space theory and is named as 4+NDES-HCC. This approach translate the dendrogram using four new features as vector dimensions in Euclidean space. The entity points in dendrograms are represented as vectors. Then, the corresponding vectors of two dendrograms are added using vector addition property to get the aggregated vector matrix. Then, Euclidean distance measure is applied to get the distance matrix. At the end a recovery tool is used such as IAHC to get the final consensus dendrogram which provides high authoritativeness.

1.2 Problem Statement

HCC has gain the attention of the researcher due to its promising results (Rashedi *et al.*, 2015; Zheng *et al.*, 2014; Rashedi & Mirzaei, 2013; Ghosh & Acharya, 2011). HCC bases on the results (dendrogram) of IAHCs while IAHC bases on a linkage method and a similarity measure. However, similarity measures have a major influence on the clustering results as compared to a linkage method (Shtern & Tzerpos, 2012). For software modularization comparative studies reported that Jaccard (JC), Jaccard-New-Measure(JNM), and Russal&Rao (RR) binary similarity measures produced better clustering results as compared to Euclidean, Simple and Rogers&Tanimoto measures (Naseem *et al.*, 2013; Shtern & Tzerpos, 2012; Cui & Chae, 2011). These similarity measures have different characteristics, for example, JC creates large number of clusters with large number arbitrary decisions while JNM reduces the arbitrary decisions but at the cost of reducing number of clusters. Similarly RR has the strengths to reduce the arbitrary decisions where JC creates. Producing large number of clusters and reducing the arbitrary decisions ensue into better and qualitative software modularization results (Naseem *et al.*, 2013; Shtern & Tzerpos, 2012; Cui & Chae, 2011). This research is motivated by the idea of “integrating the strengths of these measures (i.e., large number of clusters while taking less arbitrary decisions) in a single binary similarity measures to improve the clustering quality”.

REFERENCES

- Abebe, S. L., Haiduc, S., Marcus, A., Tonella, P., & Antoniol, G. (2009). Analyzing the Evolution of the Source Code Vocabulary. In *2009 13th European Conference on Software Maintenance and Reengineering*. IEEE. pp. 189–198.
- Abi-Antoun, M., Ammar, N., & Hailat, Z. (2012). Extraction of ownership object graphs from object-oriented code. In *Proceedings of the 8th international ACM SIGSOFT conference on Quality of Software Architectures - QoSA '12*. New York, New York, USA: ACM Press. p. 133.
- Altman, D. G. (1997). *Practical Statistics for Medical Research*. (Chapman & Hall/CRC Texts in Statistical Science).
- Andreopoulos, B., & Tzerpos, V. (2005). Multiple Layer Clustering of Large Software Systems. In *12th Working Conference on Reverse Engineering*. IEEE. pp. 79–88.
- Andritsos, P., & Tzerpos, V. (2003). Software clustering based on information loss minimization. In *10th Working Conference on Reverse Engineering*. IEEE. pp. 334–344.
- Andritsos, P., & Tzerpos, V. (2005). Information-theoretic software clustering. *IEEE Transactions on Software Engineering*, 31(2), 150–165.
- Anquetil, N., & Lethbridge, T. C. (2003). Comparative study of clustering algorithms and abstract representations for software remodularisation. *IEE Proceedings-Software*, 150(3), 185—201.
- Antonellis, P., Antoniou, D., Kanellopoulos, Y., Makris, C., Theodoridis, E., Tjortjis, C., & Tsirakis, N. (2009). Clustering for Monitoring Software Systems Maintainability Evolution. *Electronic Notes in Theoretical Computer Science*, 233, 43–57.

- Avogadri, R., & Valentini, G. (2009). Fuzzy ensemble clustering based on random projections for DNA microarray data analysis. *Artificial Intelligence in Medicine*, 45(2-3), 173–183.
- Barros, M. d. O. (2014). An experimental evaluation of the importance of randomness in hill climbing searches applied to software engineering problems. *Empirical Software Engineering*, 19(5), 1423–1465.
- Bauer, M., & Trifu, M. (2004). Architecture-aware adaptive clustering of OO systems. *Eighth European Conference on Software Maintenance and Reengineering, 2004. CSMR 2004. Proceedings.*, pp. 3–14.
- Bavota, G., De Lucia, A., Marcus, A., & Oliveto, R. (2013). Using structural and semantic measures to improve software modularization. *Empirical Software Engineering*, 18(5), 901–932.
- Bavota, G., Di Penta, M., & Oliveto, R. (2014). Search Based Software Maintenance: Methods and Tools. In *Evolving Software Systems*, vol. 1, chap. 4, pp. 103–137. Italy: Springer, 1st ed.
- Beck, F., & Diehl, S. (2010). Evaluating the Impact of Software Evolution on Software Clustering. In *2010 17th Working Conference on Reverse Engineering*. IEEE. pp. 99–108.
- Beck, F., & Diehl, S. (2013). On the impact of software evolution on software clustering. *Empirical Software Engineering*, 18(5), 970–1004.
- Belle, T. B. V. (2004). *Modularity and the Evolution of Software Evolvability..* Ph.D. thesis, University of New Mexico.
- Bittencourt, R. A., & Guerrero, D. D. S. (2009). Comparison of Graph Clustering Algorithms for Recovering Software Architecture Module Views. In *2009 13th European Conference on Software Maintenance and Reengineering*. IEEE. pp. 251–254.
- Burd, E., & Munro, M. (2000). Supporting program comprehension using dominance trees. *Annals of Software Engineering*, 9(1-2), 193–213.
- Cai, Z., Yang, X., Wang, X., & Wang, Y. (2009). A systematic approach for layered component identification. *2009 2nd IEEE International Conference*

on Computer Science and Information Technology, pp. 98–103.

- Campos, R., Dias, G., Jorge, A. M., & Jatowt, A. (2014). Survey of Temporal Information Retrieval and Related Applications. *ACM Computing Surveys*, 47(2), 1–41.
- Ceccato, M., Marin, M., Mens, K., Moonen, L., Tonella, P., & Tourwé, T. (2006). Applying and combining three different aspect Mining Techniques. *Software Quality Journal*, 14(3), 209–231.
- Chardigny, S., Seriai, A., Oussalah, M., & Tamzalit, D. (2008). Search-Based Extraction of Component-Based Architecture from Object-Oriented Systems. In R. Morrison, D. Balasubramaniam, & K. Falkner (Eds.) *Software Architecture SE - 28*, vol. 5292 of *Lecture Notes in Computer Science*, pp. 322–325. Springer Berlin Heidelberg.
- Chen, Y., Sanghavi, S., & Xu, H. (2014). Improved Graph Clustering. *IEEE Transactions on Information Theory*, 60(10), 6440–6455.
- Chen, Y.-F. R., Gansner, E. R., & Koutsofios, E. (1997). A C++ data model supporting reachability analysis and dead code detection. *ACM SIGSOFT Software Engineering Notes*, 22(6), 414–431.
- Chikofsky, E., & Cross, J. (1990). Reverse engineering and design recovery: a taxonomy. *IEEE Software*, 7(1), 13–17.
- Chong, C. Y., & Lee, S. P. (2015). Constrained Agglomerative Hierarchical Software Clustering with Hard and Soft Constraints. In *International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*. Malaysia: IEEE. pp. 177 – 188.
- Chong, C. Y., Lee, S. P., & Ling, T. C. (2013). Efficient software clustering technique using an adaptive and preventive dendrogram cutting approach. *Information and Software Technology (IST)*, 55(11), 1994–2012.
- Corazza, A., Di Martino, S., & Scanniello, G. (2010). A Probabilistic Based Approach towards Software System Clustering. In *2010 14th European Conference on Software Maintenance and Reengineering*. IEEE. pp. 88–96.
- Corazza, A., Martino, S. D., Maggio, V., & Scanniello, G. (2011). Investigating the

- Use of Lexical Information for Software System Clustering. In *European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE. pp. 35–44.
- Cornelissen, B., Zaidman, A., van Deursen, A., Moonen, L., & Koschke, R. (2009). A Systematic Survey of Program Comprehension through Dynamic Analysis. *IEEE Transactions on Software Engineering*, 35(5), 684–702.
- Cui, J. F., & Chae, H. S. (2011). Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems. *Information and Software Technology (IST)*, 53(6), 601–614.
- Davey, J., & Burd, E. (2000). Evaluating the suitability of data clustering for software remodularisation. In *Working Conference on Reverse Engineering*. IEEE. pp. 268–276.
- Dayani-fard, H., Yu, Y., & Mylopoulos, J. (2005). Improving the Build Architecture of Legacy C / C ++ Software Systems. In *Fundamental Approaches to Software Engineering*, pp. 96–110.
- De Lucia, A., Deufemia, V., Gravino, C., & Risi, M. (2007). A Two Phase Approach to Design Pattern Recovery. In *11th European Conference on Software Maintenance and Reengineering*. IEEE. pp. 297–306.
- Ducasse, S., & Pollet, D. (2009). Software Architecture Reconstruction: A Process-Oriented Taxonomy. *IEEE Transactions on Software Engineering*, 35(4), 573–591.
- Dugerdil, P., & Jossi, S. (2009). Reverse-Architecting Legacy Software Based on Roles : An Industrial Experiment. In *Software and Data Technologies*, pp. 114–127. Springer.
- El-Ramly, M., Iglinski, P., Stroulia, E., Sorenson, P., & Matichuk, B. (2001). Modeling the system-user dialog using interaction traces. In *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*. pp. 208–217.
- Erdemir, U., & Buzluca, F. (2014). A learning-based module extraction method for object-oriented systems. *Journal of Systems and Software (JSS)*, 97(2014), 156–177.

- Erdemir, U., Tekin, U., & Buzluca, F. (2011). Object Oriented Software Clustering Based on Community Structure. In *Asia-Pacific Software Engineering Conference (APSEC)*. IEEE. pp. 315–321.
- Faceli, K., de Carvalho, A. C. P. L. F., & de Souto, M. C. P. (2007). Multi-Objective Clustering Ensemble with Prior Knowledge. In *Advances in Bioinformatics and Computational Biology*, pp. 34–45.
- Forestier, G., Gançarski, P., & Wemmert, C. (2010). Collaborative clustering with background knowledge. *Data & Knowledge Engineering*, 69(2), 211–228.
- François-Joseph Lapointe, P. L. (1995). Comparison tests for dendrograms: A comparative evaluation. *Journal of Classification*, 12(2), 265–282.
- Gansner, E. R., Koutsofios, E., North, S., & Vo, K.-P. (1993). A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3), 214–230.
- Garcia, J., Ivkovic, I., & Medvidovic, N. (2013). A comparative analysis of software architecture recovery techniques. In *IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE. pp. 486–496.
- Garcia, J., Popescu, D., Mattmann, C., & Medvidovic, N. (2011). Enhancing architectural recovery using concerns. In *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*. IEEE. pp. 552–555.
- Garlan, D. (2000). Software architecture. In *Proceedings of the conference on The future of Software engineering*. New York, New York, USA: ACM Press. pp. 91–101.
- Ghaemi, R., Sulaiman, M. N., Ibrahim, H., & Mustapha, N. (2009). A survey: clustering ensembles techniques. *World Academy of Science, Engineering and Technology*, 2009(1), 636—645.
- Ghosh, J., & Acharya, A. (2011). Cluster ensembles. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(4), 305–315.
- Girvan, M., & Newman, M. E. J. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12), 7821–7826.

- Glorie, M., Zaidman, A., van Deursen, A., & Hofland, L. (2009). Splitting a large software repository for easing future software evolution-an industrial experience report. *Journal of Software Maintenance and Evolution: Research and Practice*, 21(2), 113–141.
- Gong, M., Liang, Y., Shi, J., Ma, W., & Ma, J. (2013). Fuzzy C-Means Clustering With Local Information and Kernel Metric for Image Segmentation. *IEEE Transactions on Image Processing*, 22(2), 573–584.
- Gueheneuc, Y.-G., & Antoniol, G. (2008). DeMIMA: A Multilayered Approach for Design Pattern Identification. *IEEE Transactions on Software Engineering*, 34(5), 667–684.
- Gunqun, Q., Lin, Z., & Li, Z. (2008). Applying Complex Network Method to Software Clustering. In *2008 International Conference on Computer Science and Software Engineering*. IEEE. pp. 310–316.
- Gutierrez, C. I. (1998). *Integration Analysis of Product Architecture to Support Effective Team Co-location*. Ph.D. thesis, Massachusetts Institute of Technology.
- Hall, M., Walkinshaw, N., & McMinn, P. (2012). Supervised software modularisation. In *IEEE International Conference on Software Maintenance (ICSM)*. IEEE. pp. 472–481.
- Hamdouni, A.-e. E., Seriai, A. D., & Huchard, M. (2010). Component-based Architecture Recovery from Object Oriented Systems via Relational Concept Analysis. In *7th International Conference on Concept Lattices and Their Applications*. pp. 259–270.
- Han, Z., Wang, L., Yu, L., Chen, X., Zhao, J., & Li, X. (2009). Design pattern directed clustering for understanding open source code. *2009 IEEE 17th International Conference on Program Comprehension*, pp. 295–296.
- Handrakanth, C. P. (2012). Software Module Clustering using Single and Multi-Objective Approaches. *International Journal of Advanced Research in Computer Engineering & Technology*, 1(10), 86–90.
- Harman, M., Swift, S., & Mahdavi, K. (2005). An empirical study of the robustness of

- two module clustering fitness functions. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, New York, USA: ACM Press. p. 1029.
- Hartigan, J. A., & Wong, M. A. (1979). A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1), 100–108.
- Hoffman, K. (2013). *Analysis in Euclidean space*. Courier Corporation.
- Huselius, J. G. (2007). *Reverse Engineering of Real-Time Legacy Systems-An Automated Approach Based on Execution-Time Recording*. Phd, Mälardalen University Press.
- Hussain, I., Khanum, A., Abbasi, A. Q., & Javed, M. Y. (2015). A Novel Approach for Software Architecture Recovery Using Particle Swarm Optimization. *The International Arab Journal of Information Technology*, 12(1), 1–10.
- Ibrahim, A., Rayside, D., & Kashef, R. (2014). Cooperative based software clustering on dependency graphs. In *Canadian Conference on Electrical and Computer Engineering (CCECE)*. Canada: IEEE. pp. 1–6.
- Jahnke, J. (2004). Reverse engineering software architecture using rough clusters. In *IEEE Annual Meeting of the Fuzzy Information*. Ieee. pp. 4–9 Vol.1.
- Janssens, F., Glänzel, W., & De Moor, B. (2007). Dynamic hybrid clustering of bioinformatics by incorporating text mining and citation analysis. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '07*. New York, New York, USA: ACM Press. p. 360.
- Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3), 241–254.
- Kanellopoulos, Y., Antonellis, P., Tjortjis, C., & Makris, C. (2007). k-Attractors: A Clustering Algorithm for Software Measurement Data Analysis. In *19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*. IEEE. pp. 358–365.
- Karypis, G., & Kumar, V. (1999). Chameleon: hierarchical clustering using dynamic modeling. *Computer*, 32(8), 68–75.

- Kashef, R. F., & Kamel, M. S. (2010). Cooperative clustering. *Pattern Recognition*, 43(6), 2315–2329.
- Kienle, H. M., & Müller, H. A. (2010). RigiAn environment for software reverse engineering, exploration, visualization, and redocumentation. *Science of Computer Programming*, 75(4), 247–263.
- Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science (New York, N.Y.)*, 220(4598), 671–80.
- Kobayashi, K., Kamimura, M., Kato, K., Yano, K., & Matsuo, A. (2012). Feature-gathering dependency-based software clustering using Dedication and Modularity. In *IEEE International Conference on Software Maintenance (ICSM)*. IEEE. pp. 462–471.
- Koschke, R., & Eisenbarth, T. (2000). A framework for experimental evaluation of clustering techniques. In *International Workshop on Program Comprehension*. IEEE Comput. Soc. pp. 201–210.
- Krishnamurthy, B. (1995). *Practical reusable UNIX software*. Wiley.
- Kuhn, A., Ducasse, S., & Gîrba, T. (2007). Semantic clustering: Identifying topics in source code. *Information and Software Technology*, 49(3), 230–243.
- Kumari, A. C., Srinivas, K., & Gupta, M. P. (2013). Software module clustering using a hyper-heuristic based multi-objective genetic algorithm. In *IEEE International Advance Computing Conference (IACC)*. India: IEEE. pp. 813–818.
- Lakhotia, A. (1997). A unified framework for expressing software subsystem classification techniques. *Journal of Systems and Software*, 36(3), 211–231.
- Langfelder, P., Zhang, B., & Horvath, S. (2008). Defining clusters from a hierarchical cluster tree: the Dynamic Tree Cut package for R. *Bioinformatics (Oxford, England)*, 24(5), 719–20.
- Lesot, M.-J., & Rifqi, M. (2009). Similarity measures for binary and numerical data : a survey. *Int. J. Knowledge Engineering and Soft Data Paradigms*, 1(1).
- Liao, S.-H., Chu, P.-H., & Hsiao, P.-Y. (2012). Data mining techniques and applications A decade review from 2000 to 2011. *Expert Systems with*

Applications, 39(12), 11303–11311.

Lundberg, J., & Löwe, W. (2003). Architecture Recovery by Semi-Automatic Component Identification. *Electronic Notes in Theoretical Computer Science*, 82(5), 98–114.

Lung, C.-H., Zaman, M., & Nandi, A. (2004). Applications of clustering techniques to software partitioning, recovery and restructuring. *Journal of Systems and Software*, 73(2), 227–244.

Lutellier, T., Chollak, D., Garcia, J., Tan, L., Rayside, D., Medvidovic, N., & Kroeger, R. (2015). Comparing Software Architecture Recovery Techniques Using Accurate Dependencies. In *IEEE International Conference on Software Engineering (ICSE)*, vol. vol.2. USA, Canada: IEEE, ACM. pp. 69–78.

Mahmoud, A., & Niu, N. (2013). Evaluating software clustering algorithms in the context of program comprehension. In *International Conference on Program Comprehension (ICPC)*. USA: IEEE. pp. 162–171.

Mancoridis, S., Mitchell, B. S., Chen, Y., & Gansner, E. R. (1999). Bunch: a clustering tool for the recovery and maintenance of software system structures. In *International Conference on Software Maintenance*. IEEE. pp. 50–59.

Mancoridis, S., Mitchell, B. S., Rorres, C., Chen, Y., & Gansner, E. R. (1998). Using automatic clustering to produce high-level system organizations of source code. In *International Workshop on Program Comprehension*. IEEE. pp. 45–52.

Maqbool, O., & Babri, H. (2004). The weighted combined algorithm: a linkage algorithm for software clustering. In *Eighth European Conference on Software Maintenance and Reengineering*. IEEE. pp. 15–24.

Maqbool, O., & Babri, H. (2007a). Bayesian Learning for Software Architecture Recovery. In *2007 International Conference on Electrical Engineering*. IEEE. pp. 1–6.

Maqbool, O., & Babri, H. (2007b). Hierarchical Clustering for Software Architecture Recovery. *IEEE Transactions on Software Engineering*, 33(11), 759–780.

- Maqbool, O., Babri, H., Karim, A., & Sarwar, S. (2005). Metarule-guided association rule mining for program understanding. *IEEE Proceedings*, 152(6), 281–296.
- Medvidovic, N., & Taylor, R. N. (2010). Software architecture: foundations, theory, and practice. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, vol. 2. New York, New York, USA: ACM Press. p. 471.
- Mirzaei, A., & Rahmati, M. (2010). A Novel Hierarchical-Clustering-Combination Scheme Based on Fuzzy-Similarity Relations. *IEEE Transaction on Fuzzy Systems*, 18(1), 27–39.
- Mirzaei, A., Rahmati, M., & Ahmadi, M. (2008). A new method for hierarchical clustering combination. *Intelligent Data Analysis*, 12, 549–571.
- Mitchell, B. S. (2003). A heuristic approach to solving the software clustering problem. *International Conference on Software Maintenance, 2003. ICSM 2003. Proceedings.*, pp. 285–288.
- Mitchell, B. S. (2006). Clustering Software Systems to Identify Subsystem Structures. Tech. rep.
- Mitchell, B. S., & Mancoridis, S. (1998). Clustering Module Dependency Graphs of Software Systems Using the Bunch Tool. Tech. rep.
- Mitchell, B. S., & Mancoridis, S. (2001). Comparing the decompositions produced by software clustering algorithms using similarity measurements. In *International Conference on Software Maintenance*. IEEE. pp. 744–753.
- Mitchell, B. S., & Mancoridis, S. (2002). Using Heuristic Search Techniques to Extract Design Abstractions from Source Code. In *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2. pp. 1375—1382.
- Mitchell, B. S., & Mancoridis, S. (2003). Modeling the Search Landscape of Metaheuristic Software Clustering Algorithms. In E. Cantú-Paz, J. Foster, K. Deb, L. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. Potter, A. Schultz, K. Dowsland, N. Jonoska, & J. Miller (Eds.) *Genetic and Evolutionary Computation GECCO 2003 SE - 153*, vol. 2724 of *Lecture Notes*

- in Computer Science*, pp. 2499–2510. Springer Berlin Heidelberg.
- Mitchell, B. S., & Mancoridis, S. (2006). On the automatic modularization of software systems using the Bunch tool. *IEEE Transactions on Software Engineering*, 32(3), 193–208.
- Mitchell, B. S., Mancoridis, S., & Traverso, M. (2002). Search based reverse engineering. In *Proceedings of the 14th international conference on Software engineering and knowledge engineering*. New York, New York, USA: ACM Press. p. 431.
- Morokoff, W. J., & Caflisch, R. E. (1994). Quasi-Random Sequences and Their Discrepancies. *SIAM Journal on Scientific Computing*, 15(6), 1251–1279.
- Muhammad, S., Maqbool, O., & Abbasi, A. Q. (2010). Role of relationships during clustering of object-oriented software systems. In *2010 6th International Conference on Emerging Technologies (ICET)*. IEEE. pp. 270–275.
- Muhammad, S., Maqbool, O., & Abbasi, A. Q. (2012). Evaluating relationship categories for clustering object-oriented software systems. *IET Software*, 6(3), 260.
- Murphy, G., & Notkin, D. (1997). Reengineering with reflexion models: a case study. *Computer*, 30(8), 29–36.
- Murphy, G., Notkin, D., & Sullivan, K. (2001). Software reflexion models: bridging the gap between design and implementation. *IEEE Transactions on Software Engineering*, 27(4), 364–380.
- Murtagh, F. (1983). A Survey of Recent Advances in Hierarchical Clustering Algorithms. *The Computer Journal*, 26(4), 354–359.
- Naseem, R., Maqbool, O., & Muhammad, S. (2010). An Improved Similarity Measure for Binary Features in Software Clustering. In *2010 Second International Conference on Computational Intelligence, Modelling and Simulation*. IEEE. pp. 111–116.
- Naseem, R., Maqbool, O., & Muhammad, S. (2011). Improved Similarity Measures for Software Clustering. In *European Conference on Software Maintenance and Reengineering (CSMR)*. Pakistan: IEEE. pp. 45–54.

- Naseem, R., Maqbool, O., & Muhammad, S. (2013). Cooperative clustering for software modularization. *Journal of Systems and Software (JSS)*, 86(8), 2045–2062.
- Ngonga Ngomo, A.-C., & Schumacher, F. (2009). BorderFlow: A Local Graph Clustering Algorithm for Natural Language Processing. In *Proceedings of the 10th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing '09*. Berlin, Heidelberg: Springer-Verlag. pp. 547–558.
- Patel, C., Hamou-Lhadj, A., & Rilling, J. (2009). Software Clustering Using Dynamic Analysis and Static Dependencies. In *2009 13th European Conference on Software Maintenance and Reengineering*. IEEE. pp. 27–36.
- Podani, J. (2000). Simulation of Random Dendrograms and Comparison Tests: Some Comments. *Journal of Classification*, 17(1), 123–142.
- Praditwong, K., Harman, M., & Yao, X. (2011). Software Module Clustering as a Multi-Objective Search Problem. *IEEE Transactions on Software Engineering (TSE)*, 37(2), 264–282.
- Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical Recipes 2nd Edition: The Art of Scientific Computing*. Cambridge University Press.
- Pukelsheim, F. (1994). The Three Sigma Rule. *The American Statistician*, 48(2), 88–91.
- Rashedi, E., & Mirzaei, A. (2013). A hierarchical clusterer ensemble method based on boosting theory. *Knowledge-Based Systems*, 45, 83–93.
- Rashedi, E., Mirzaei, A., & Rahmati, M. (2015). An information theoretic approach to hierarchical clustering combination. *Neurocomputing*, 148, 487–497.
- Reynolds, C. W. (1987). Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87*. New York, New York, USA: ACM Press. pp. 25–34.
- Saeed, M., Maqbool, O., Babri, H., Hassan, S., & Sarwar, S. (2003). Software

- clustering techniques and the use of combined algorithm. In *Seventh European Conference on Software Maintenance and Reengineering*. IEEE Comput. Soc. pp. 301–306.
- Sajnani, H. S., & Lopes, C. V. (2014). Probabilistic component identification. In *India Software Engineering Conference (ISEC)*. USA: ACM. pp. 1–10.
- Sartipi, K., & Kontogiannis, K. (2003a). On modeling software architecture recovery as graph matching. In *International Conference on Software Maintenance*. IEEE Comput. Soc. pp. 224–234.
- Sartipi, K., & Kontogiannis, K. (2003b). Pattern-based Software Architecture Recovery. In *In Proc. of the Second ASERC Workshop on Software Architecture*. pp. 1–7.
- Sartipi, K., Kontogiannis, K., & Mavaddat, F. (2000). A pattern matching framework for software architecture recovery and restructuring. In *International Workshop on Program Comprehension*. IEEE. pp. 37–47.
- Scanniello, G., D'Amico, A., D'Amico, C., & D'Amico, T. (2010a). Architectural Layer Recovery for Software System Understanding and Evolution. *Softw. Pract. Exper.*, 40(10), 897–916.
- Scanniello, G., & Erra, U. (2013). Software entities as bird flocks and fish schools. In *IEEE Working Conference on Software Visualization (VISSOFT)*, I. IEEE. pp. 1–4.
- Scanniello, G., & Marcus, A. (2011). Clustering Support for Static Concept Location in Source Code. In *International Conference on Program Comprehension (ICPC)*. Ieee. pp. 1–10.
- Scanniello, G., Risi, M., & Tortora, G. (2010b). Architecture Recovery Using Latent Semantic Indexing and K-Means: An Empirical Evaluation. In *2010 8th IEEE International Conference on Software Engineering and Formal Methods*. IEEE. pp. 103–112.
- Seriai, A., Sadou, S., & Sahraoui, H. A. (2014). Enactment of Components Extracted from an Object-Oriented Application. In *The European Conference on Software Architecture (ECSA)*. pp. 234–249.

- Shah, Z., Naseem, R., Orgun, M., Mahmood, A. N., & Shahzad, S. (2013). Software Clustering Using Automated Feature Subset Selection. In H. Motoda, Z. Wu, L. Cao, O. Zaiane, M. Yao, & W. Wang (Eds.) *International Conference on Advanced Data Mining and Applications (ADMA)*, vol. 8347 of *Lecture Notes in Computer Science*. Italy; Pakistan; Australia: Springer. pp. 47–58.
- Shokoufandeh, A., Mancoridis, S., Denton, T., & Maycock, M. (2005). Spectral and meta-heuristic algorithms for software clustering. *Journal of Systems and Software*, 77(3), 213–223.
- Shokoufandeh, A., Mancoridis, S., & Maycock, M. (2002). Applying spectral methods to software clustering. In *Ninth Working Conference on Reverse Engineering*. IEEE Comput. Soc. pp. 3–10.
- Shtern, M., & Tzerpos, V. (2010). On the Comparability of Software Clustering Algorithms. In *2010 IEEE 18th International Conference on Program Comprehension*. IEEE. pp. 64–67.
- Shtern, M., & Tzerpos, V. (2011). Evaluating software clustering using multiple simulated authoritative decompositions. In *IEEE International Conference on Software Maintenance (ICSM)*. IEEE. pp. 353–361.
- Shtern, M., & Tzerpos, V. (2012). Clustering Methodologies for Software Engineering. *Advances in Software Engineering (ASE)*, 2012, 1–18.
- Shtern, M., & Tzerpos, V. (2014). Methods for selecting and improving software clustering algorithms. *Software: Practice and Experience*, 44(1), 33–46.
- Siddique, F., & Maqbool, O. (2012). Enhancing comprehensibility of software clustering results. *IET Software*, 6(4), 283.
- Siff, M., & Reps, T. (1999). Identifying modules via concept analysis. *IEEE Transactions on Software Engineering*, 25(6), 749–768.
- Silva, L. L., Valente, M. T., & Maia, M. D. a. (2014). Assessing modularity using co-change clusters. In *International conference on Modularity (MODULARITY)*. Brazil: ACM. pp. 49–60.
- Sindhgatta, R., & Pooloth, K. (2007). Identifying Software Decompositions by Applying Transaction Clustering on Source Code. In *31st Annual*

- International Computer Software and Applications Conference, Compsac.* IEEE. pp. 317–326.
- Slonim, N., & Tishby, N. (2000). Agglomerative information bottleneck. *Advances in neural information processing systems*, 12(1), 617–23.
- Sora, I., Glodean, G., & Gligor, M. (2010). Software architecture reconstruction: An approach based on combining graph clustering and partitioning. In *2010 International Joint Conference on Computational Cybernetics and Technical Informatics*. IEEE. pp. 259–264.
- Spek, P., & Klusener, S. (2011). Applying a dynamic threshold to improve cluster detection of LSI. *Science of Computer Programming (SCP)*, 76(12), 1261–1274.
- Synytskyy, N., Holt, R. C., & Davis, I. (2005). Browsing Software Architectures With LSEdit. In *13th International Workshop on Program Comprehension*. IEEE. pp. 176–178.
- Thangaraj, R., Pant, M., Abraham, A., & Badr, Y. (2009). Hybrid Evolutionary Algorithm for Solving Global Optimization Problems. In E. Corchado, X. Wu, E. Oja, Á. Herrero, & B. Baroque (Eds.) *Hybrid Artificial Intelligence Systems SE - 37*, vol. 5572 of *Lecture Notes in Computer Science*, pp. 310–318. Springer Berlin Heidelberg.
- Tonella, P. (2001). Concept analysis for module restructuring. *IEEE Transactions on Software Engineering*, 27(4), 351–363.
- Tucker, A., Swift, S., & Liu, X. (2001). Variable grouping in multivariate time series via correlation. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 31(2), 235–45.
- Turner, K., & Agogino, A. K. (2008). Ensemble clustering with voting active clusters. *Pattern Recognition Letters*, 29(14), 1947–1953.
- Tzerpos, V. (2003). An optimal algorithm for MoJo distance. In *Proceedings of the 11th IEEE International Workshop on Program Comprehension*. IEEE Comput. Soc. pp. 227–235.

- Tzerpos, V., & Holt, R. C. (1999). MoJo: a distance metric for software clusterings. In *Working Conference on Reverse Engineering*. IEEE. pp. 187–193.
- Tzerpos, V., & Holt, R. C. (2000a). ACCD: an algorithm for comprehension-driven clustering. In *Working Conference on Reverse Engineering*. IEEE. pp. 258–267.
- Tzerpos, V., & Holt, R. C. (2000b). On the stability of software clustering algorithms. In *International Workshop on Program Comprehension*. IEEE. pp. 211–218.
- Vasconcelos, A., & Werner, C. (2007). Architecture Recovery and Evaluation Aiming at. In *Software Architectures, Components, and Applications*, pp. 72–89. Springer.
- Veal, B. W. G. (2011). *Binary Similarity Measures and their Applications in Machine Learning*. Ph.D. thesis, London School of Economics.
- von Detten, M., & Becker, S. (2011). Combining clustering and pattern detection for the reengineering of component-based software systems. In *Proceedings of the joint ACM SIGSOFT conference – QoSA and ACM SIGSOFT symposium – ISARCS on Quality of software architectures – QoSA and architecting critical systems – ISARCS - QoSA-ISARCS '11*. New York, New York, USA: ACM Press. p. 23.
- Wang, J., & Su, X. (2011). An improved K-Means clustering algorithm. In *2011 IEEE 3rd International Conference on Communication Software and Networks*. IEEE. pp. 44–46.
- Wang, L., Han, Z., He, J., Wang, H., & Li, X. (2012). Recovering design patterns to support program comprehension. In *Proceedings of the 2nd international workshop on Evidential assessment of software technologies - EAST '12*. New York, New York, USA: ACM Press. p. 49.
- Wang, Y., Liu, P., Guo, H., Li, H., & Chen, X. (2010). Improved Hierarchical Clustering Algorithm for Software Architecture Recovery. *2010 International Conference on Intelligent Computing and Cognitive Informatics*, pp. 247–250.
- Waters, R. L. (2004). *Obtaining Architectural Descriptions from Legacy Systems-The Architectural Synthesis Process*. Ph.D. thesis, Georgia Institute of Technology.

- Wen, Z., & Tzerpos, V. (2004a). An effectiveness measure for software clustering algorithms. In *Proceedings. 12th IEEE International Workshop on Program Comprehension, 2004.*. IEEE. pp. 194–203.
- Wen, Z., & Tzerpos, V. (2004b). Evaluating similarity measures for software decompositions. In *20th IEEE International Conference on Software Maintenance, 2004. Proceedings.*. IEEE. pp. 368–377.
- Wiggerts, T. (1997). Using clustering algorithms in legacy systems remodularization. In *Working Conference on Reverse Engineering.* IEEE. pp. 33–43.
- Wu, J., Hassan, A., & Holt, R. C. (2005). Comparison of clustering algorithms in the context of software evolution. In *21st IEEE International Conference on Software Maintenance.* IEEE. pp. 525–535.
- Xanthos, S., & Goodwin, N. (2006). Clustering Object-Oriented Software Systems using Spectral Graph Partitioning. *Urbana*, 51(1), 1–5.
- Xia, C., & Tzerpos, V. (2005). Software Clustering Based on Dynamic Dependencies. In *Ninth European Conference on Software Maintenance and Reengineering.* IEEE. pp. 124–133.
- Zheng, L. I., Li, T. A. O., & Ding, C. (2014). A Framework for Hierarchical Ensemble Clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(2), 1–23.