MINAR International Journal of Applied Sciences and Technology
ISSN: 2717-8234
Article type: Research Article

Received: 25/12/2023 **Accepted:** 01/02/2024 **Published:** 01/03/2024

SOFTWARE MODULES CLUSTERING: A LITERATURE REVIEW

Ali Hussein ALI[1]
University of Mosul, Iraq
Dujan B. TAHA
University of Mosul, Iraq

Abstract

Maintenance takes the largest cost within software engineering processes. Because understanding the software code is complex and requires a deep understanding of the structure of building the code. To face Such a problem and other problems such as research purposes, or retrieval, restructuring, and understanding the behavior of software entities, today the software component, such as objects, classes, or files that have similarity in feature can be grouped together using unsupervised learning method called software module clustering. This method gives a clear picture of the components of the program and the details of the dependencies between these components, and thus it gives a structure that can be relied upon effectively. This research paper will present a compilation of studies published in this field over the past 23 years. We review 36 well-known research papers in the literature that examines software module clustering techniques and obtain useful results and data. Research containing duplicate ideas was excluded. We will attempt to answer the following questions: What are the modern methods used in the concept of clustering? What are the applications of clustering in software engineering? What are various clustering algorithms? What evaluation methods are available for assessing the quality of clustering techniques?

Keywords: *Software Modules Clustering, Clustering Algorithms, Clustering Applications, Software Clustering.*

http://dx.doi.org/10.47832/2717-8234.18.7

[1] alihoseen90@gmail.com

CC BY

Introduction

The goal of the clustering technique, also known as cluster analysis, is to arrange components from a given dataset into several clusters using an unsupervised data mining approach.[1] Elements within the same cluster have similarities in certain features. Similarities and dissimilarities can be measured. Software clustering in the field of software engineering involves the decomposition of intricate and extensive software systems into smaller subsystems that have high cohesion and low coupling, whose content can be understood, managed, handled and maintained more easily. [2]

It is now challenging for software module clustering to keep up with the rapid pace of software development and the constant changes in application requirements, though. For example, there are few experimental studies on clustering topics for systems that use more than one programming language in their development, as well as systems that call web services available on the Internet. [3][4][5]

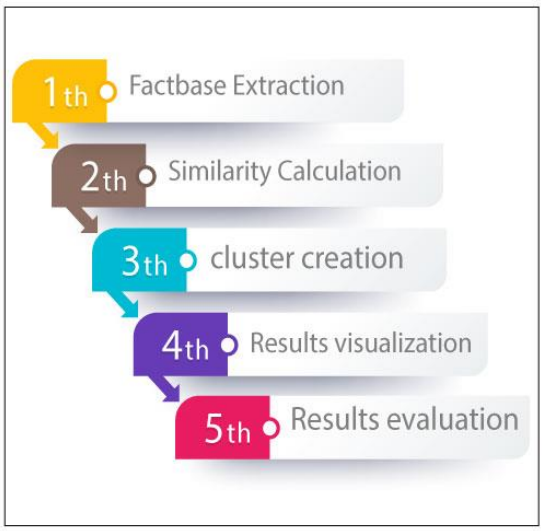

Figure 1. Software module clustering process

The Process of Clustering Software Modules

As shown in Figure 1, the standard clustering process includes five basic stages. The process finish when the completion criterion is met, for example the maximum digit of iterations or the required digit of clusters is reached.

Step1: Factbase Extraction

A factbase is the expression that includes input that the clustering algorithms are expected to receive. The process of factbase extraction comprises the following steps: selecting a software system, choosing a factbase source, preprocessing and filtering, choosing an entity, and choosing features. [6][7][5][8]

1) Select the software system

To initiate the software module clustering process, it is necessary to specify the target systems that will be subjected to clustering. [1]

2) Select the factbase source

The factbase include information about the target software systems, including software components such as classes and their relationships, such as method calls. There are different types of factbase sources. For example source code, Binary code, Bytecode, Human expertise, Configuration files, Dynamic information, Data files, Files organization and Evolutionary (historical) information. Factbases come in a variety of formats from a wide range of sources. Within this context, commonly utilized objects include the "dependency graph," "vector-space model," "software metrics," and the "extended dependency graph." [6][9][5][10]

3) Filtering and Preprocessing

Filtering plays a crucial role as a preprocessing step in all clustering operations within the realm of software engineering. Its goal is to eliminate data that has been retrieved from source codes that is not important or non-textual in order to reduce noise and improve the quality of clustering results. [11][12]

4) Entity Selection.

When the Clustering aim is Software comprehension then the Input entities are Functions and their call statements. On the other hand, Software classes, packages, modules, and files are the input entities when the clustering goal is architectural recovery.

5) Feature Selection

For example, Formal and nonformal features are the two groups into which a class entity's features can be divided. A class entity's nonformal qualities include things like the quantity of variables and lines of code (LOC). While on another level, formal features consist of attributes like method invocations and entity relationships. [13] [6] [14]

Step2: Similarity Calculation

Software module clustering involves the utilization of similarity metrics to evaluate the attributes of each component to determine which is the most similar or different. The most commonly used measures are "Euclidean distance", "Jaccard distance" and "Cosine distance". [6] [15]

Step3: Cluster configuration

At this stage, clustering algorithms are applied to groups of similar entities in the target system. Choosing an algorithm is itself a difficult process. In [7] authors propose a method to help in this area. The study identifies the most common types of clustering algorithms frequently utilized in the literature:

1) Hierarchical Clustering

Hierarchical clustering algorithms can be broadly categorized into two types: agglomerative clustering and divisive clustering.

***Agglomerative algorithms** also known as bottom-up algorithms. Starts with singletons (each element) and combine them until a single cluster is gained.*

***Divisive algorithms**, also known as top-down algorithms, initiate the clustering process with a single cluster that contains all n entities. The algorithm then proceeds to divide this initial cluster iteratively until n clusters are obtained.[16]*

2) Partitional clustering

This approach involves partitioning the set of entities into distinct and mutually ex-clusive groups, ensuring that each entity is assigned to only one group. Each data point is assigned to exactly one cluster, and the objective is to maximize or minimize a particular criterion, like the inter-cluster or intra-cluster distance. Popular partitional clustering algorithms include k-means, k-medoids, and fuzzy c-means.

Step 4: Visualization of the Clustering Results

After the software module clustering process, the Visualization is taking place to show the results of clustering as dendrograms, graphs, or distribution maps [17]. The purpose of result visualization is to make it easier for software engineers to explore the results efficiently and easily.

Step 5: Evaluation Metrics of the Clustering

A variety of methods can be utilized to assess the efficacy of software clustering algorithms. Figure 2 shows the evaluation methods most frequently used in the papers we studied. The assessment techniques are summed up as follows:

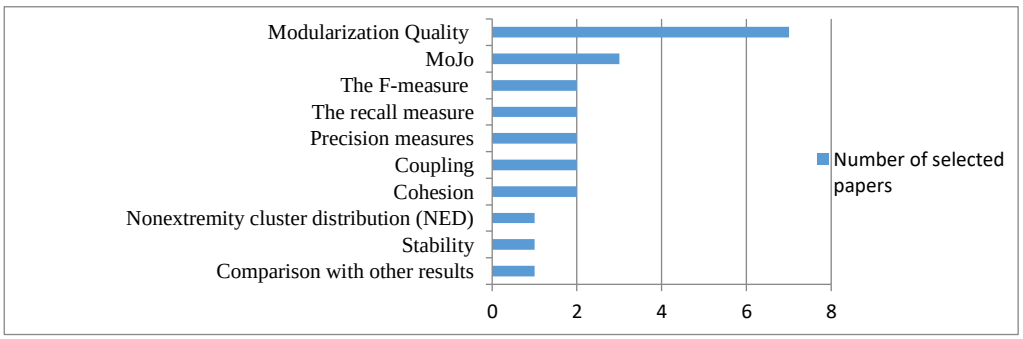

Figure 2. Number of selected papers vs. metrics of module clustering evaluation

Comparison with other results.

Modularization Quality (MQ). It evaluates the level of cohesion and coupling among modules.

MoJo similarity metrics. Are employed to measure the degree of similarity between the section produced by an expert and the piece produced by the software clustering procedure.

Execution Time

Nonextremity cluster distribution (NED). A good clustering process is one that does not contain a part with a large number of entities, and does not contain a part with a single entity [18]

Cohesion, referred to as intraconnectivity, **Evaluates the extent of connections among modules within a cluster**.[19]

Coupling, also known as interconnectivity Quantifies the level of communication or interaction between software modules located in different clusters [19].

Precision measures (in the same cluster) .The percentage of module pairs that are assigned to different clusters by the clustering algorithm compared to the expert decomposition.[20]

The recall measure**, in the context of different clusters. Is the percentage of module pairs in the expert decomposition that the clustering algorithm properly detected.** It indicates how well the clustering algorithm captures the modules identified by the expert. [20] [6]

Stability. If there are minor alterations between consecutive versions of a developing software system, does the clustering process produces similar partitions and that what is called Stability measures. [21] [6]

Human Experts. The weighted average of recall and precision is calculated to determine the F-measure, a metric used to assess the accuracy of clustering techniques**. It provides a balanced assessment of their performance by considering both the ability to correctly identify relevant items (recall) and the precision in avoiding false positives.** [22] [6]

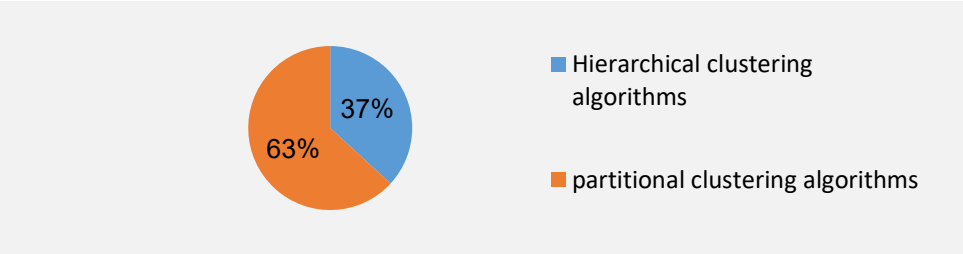

Figure 3 Paper count vs. method of clustering

Related work

The authors highlighted a variety of software clustering methods and their applications in the field of software engineering. They also examined several approaches for evaluating software clustering results and discussed the challenges associated with enhancing its performance.

Name of researcher	algorithm	Application Areas	Quality measure	Result
"Jian Feng Cui, Heung Seok Chae", 2011,[16]	Agglomerative hierarchical clustering (AHC) algorithms	Evaluate the algorithms for component identification in software reengineering	Size, coupling, and cohesion criteria	Different clustering algorithms produced different clustering results.
"Fabian Beck and Stephan Diehl" , 2010 , [21]	Bunch, a graph-based clustering algorithm	Support information recovery	MoJoFM metric	The traditional approach provides better results
"linhui Zhong, jing He, nengwei Zhang , peng Zhang , jing Xia" , 2016, [23]	(AHC) algorithm	Maintainability of software	Module Quality- MQ	The method improves the accuracy and aids in the refactoring
"Anna Corazza, Sergio Di Martino, Valerio Maggio, Giuseppe Scanniello" , 2019, [24]	Hierarchical clustering algorithm	Support information recovery	Authoritativeness(Auth) and (NED)	Software system clustering. Compared to structural-based solutions, it produces better results
"Amit Rathee and Jitender Kumar Chhabra", 2017, [25]	Hierarchical Agglomerate Clustering (HAC)	Maintainability	Precision, recall, and F-measure	The proposed technique of software remodularization showed higher accuracy against the corresponding software gold standard .

"Chun Yong Chong and Sai Peck Lee", 2015, [26]	Unweighted Pair-Group Method using Arithmetic Average (UPGMA)	Support information recovery	MoJoFM, Sorensen-Dice coefficient	The results of the evaluation showed that the algorithm successfully handled constraints and provided a better understanding of the analyzed software system
"Monika Bishnoi and Paramvir Singh", 2016, [18]	Weighted Combined Algorithm (WCA) and PSO clustering techniques	Maintainability	TurboMQ	The results demonstrated that augmenting the number of iterations in the Particle Swarm Optimization (PSO) did not have a substantial influence on the overall outcomes

As previously stated. Hierarchical clustering and partitional clustering are two common forms of clustering algorithms that have been utilized in the literature, and they will both be covered in the sections that follow. "Partitional Clustering" is the most widely utilized kind of clustering, as Figure 3 illustrates.

A. Clustering based on hierarchy

A study was carried out, according to [16], utilizing a range of hierarchical clustering

Table 1. Hierarchical clustering algorithms

principles, to examine how old object-oriented systems evolved into a different type of module-based systems. The goal of this study is to discover whether there is a better clustering method. Next, carry out a series of experiments for numerous outdated object-oriented systems using various clustering algorithms. Using a

variety of parameters, including coupling and cohesion, an evaluation of the relative strengths and weaknesses of the different hierarchical agglomerative clustering methods was given based on the clustering findings.

The biggest focus of this study was on analyzing algorithms in light of software re-engineering. The experimental findings demonstrated that different weighting schemes, connection techniques, and similarity measurements had different effects on the component identification outcomes. Results from several clustering algorithms varied in terms of clustering. Talking about [21], Authors compares the results of evolutionary approaches with traditional

structural code dependency approaches. The study showed that the traditional approach provides better results. In [23], a method is proposed for software clustering that incorporates software evolution information. An expanded software dependency model is then constructed, and software is clustered using the Agglomerative Hierarchical Clustering (AHC) algorithm. Tests conducted on two publicly-available projects demonstrate that the technique enhances the precision of software clustering and facilitates business software reworking. In the [24], the usefulness of lexical information in software system clustering is examined. The study demonstrated that appropriately weighted lexical information can be successfully used for software clustering, yielding superior results than structural-based solutions. The study used a dataset of thirteen open-source Java software systems.

The paper [25] proposes a software remodularization technique by eliciting conceptual similarity between software component using structural and semantic coupling measures. Hierarchical Agglomerate Clustering (HAC). The proposed technique of software re_modularization, which combines structural and semantic coupling measurements, showed higher accuracy against the corresponding software gold standard. The paper [26] proposes a constrained agglomerative hierarchical clustering algorithm that merge pair-wise constraints to improve the quality of software clustering. The algorithm determines if software components belong to the same functional group by maximizing the satisfaction of must-link and cannot-link requirements. The suggested algorithm's efficacy was assessed through two experiments utilizing real-world software systems; the outcomes demonstrated its capacity to manage restrictions and enhance the quality of clustering.. The paper [18] introduces a methodology for enhancing software modularization by applying Particle Swarm Optimization (PSO) to optimize the Weighted Combined Algorithm (WCA). [Table 1] Summarizes the methods of hierarchical clustering algorithms.

Name of researcher	algorithm	Application Areas	Quality measure	Result
"Masoud Kargar,Ayaz Isazadeh, Habib Izadkhah ",2019,[3]	Genetic algorithm	Maintaining and evolving software systems	Precision, Recall, FM, and MoJoFM	Modularization close to human experts
"Shohag Barman, HiraLal Gope, M M Manjurul Islam, MdMehedi Hasan, Umme Salma", 2016, [27]	Genetic Algorithm-based Software Modularization Clustering (GASMC)	Maintenance and improve the program structure	(MQ)	Higher MQ and lower standard deviations
"Amarjeet , Jitender Kumar Chhabra ", 2015, [28]	Non-dominated Sorting Genetic Algorithms (NSGA-II)	Improve the modularization quality and organizing large and complex systems	NA	Better modularization quality with minimum modification
"Simone Romano, Giuseppe Scanniello, Michele Risi,Carmine Gravino" , 2011 [11]	Fuzzy c-means clustering algorithm called Fanny for design pattern recover	To improve the recovery of design patterns in source code.	NA	Improves the correctness of the results, while preserving the number of design pattern instances correctly identified.
"Pradeep Tomar and Jagdeep Kaur" , 2016 [12]	Fuzzy clustering	Organizing and managing software components	NA	The results of the algorithm are represented as fuzzy clusters, where are components that tend to lie in more than two clusters
"Shumail Arshad, Christos Tjortjis", 2014, [29]	*K-Means* clustering algorithm	Maintainability	NA	Supports the identification of potentially problematic code parts
"Shumail Arshad and Christos Tjortjis", 2008, [8]	K-Means clustering	Maintainability	NA	The approach supports the process of identifying potentially

				problematic parts of the code that require additional inspection and proactive maintenance
"Amarjeet Prajapati · Jitender Kumar Chhabra", [2017],[30]	Particle Swarm Optimization (PSO) algorithm	To support information recovery	(MQ), (NED), coupling, and cohesion measures	PSOMC approach is effective for solving SMCPs, with higher MQ values
"Jimin Hwa and Shin Yoo", 2017,[2]	Hill-climbing algorithm	Comprehension and maintenance of complex systems	(MQ)	Multi-factor approach allows for module clusters of different natures, capturing both semantic and structural bondings.
"Masoud Kargar,Ayaz Isazadeh, Habib Izadkhah" , 2017, [31]	Hill-Climbing Algorithms	Support information recovery	Turbo MQ	(SDG) can be used as a replacement for (CDG) in software clustering.
"Abdulaziz Alkhalid , Mohammad Alshayeb , Sabri Mahmoud "2010 [13]	Adaptive K-Nearest Neighbor (A-KNN)	Enhance the understandability, reduce the effort of the maintenance, and minimize the costs associated with software evolution and complexity.	Comparison with other results	The results showed that the proposed A-KNN algorithm demonstrated competitive performance with the other three algorithms while requiring less computational complexity.
"Jinhuang Huang, Jing Liu", 2017, [32]	Multi-agent evolutionary algorithm called MAEA-SMCPs	Software maintenance, reusability, understandability, software testing, and debugging	MQ	MAEA-SMCPs is a highly effective algorithm for software module clustering problems

B. Partitional clustering

Multi-programming language modularization was introduced, according to [3]. Next, a method for modularizing applications written in several programming languages is presented. The outcomes show that the suggested method can extract a modularization that is comparable to that of human specialists. Provide only software clustering strategies in [27] that break down huge software systems into smaller subsystems as research guidelines. The GASMC method outperformed both Hill Climbing and GGA in terms of mean quality values (MQ) and standard deviations. It displayed fewer standard deviations, indicating steady performance, and higher MQ values, suggesting a better division of modules into clusters. Weighted class connections and a multi-objective optimization approach are used by Chhabra and Jitender Kumar [28] to improve the package structure of object-oriented applications. The paper suggests a multi-objective optimization strategy to reduce the amount of class migration across current packages and enhance the modularization quality of object-oriented systems.

The results demonstrate that the proposed approach improves modularization quality while requiring the least amount of alteration to the original package structure. By utilizing the concept of fuzzy relations, the authors of [12][11] present a method for selecting software modules based on fuzzy clustering. Next, they use fuzzy clustering with lexical information to increase the precision of source code design pattern recovery. Authors in [29] offered data mining as a solution to the issue of gathering and evaluating metric values for big software systems since it may extract information and uncover hidden patterns. Discrete particle swarm optimization-based module clustering (PSOMC), a software module clustering technique based on particle swarm optimization (PSO), is introduced by [30] in an effort to more effectively and efficiently handle the large-scale SMCPs. In comparison to other competing approaches, the suggested PSOMC approach produces clustering solutions with higher MQ values, indicating its effectiveness and promise in solving SMCPs.

Next paper [31], Authors are proposing a new dependency graph, called semantic dependency graph (SDG) to extraction tool sets for large-scale software systems. The results of the paper show that the semantic dependency graph (SDG) can be used as a replacement for the Call Dependency Graph (CDG) in software clustering. On the other hand, [2] proposes a multi-factor module clustering method that addresses the following issues by enabling the creation of module groups depending on numerous parameters: When using the technique, the user must choose a factor without knowing which will work best. While some modules establish more structural bondings, others may generate semantic ones. The paper [13] proposes a new algorithm called Adaptive K-Nearest Neighbor (A-KNN) for function-level software refactoring. We present MAEA-SMCPs in [32], a multi-agent evolutionary algorithm that uses module relationships to determine acceptable grade software module clustering. Using software metrics and data mining, Arshad and Tjortjis[8] suggest an automated approach to software maintenance. In order to find difficult and complicated classes that could be prone to errors and need proactive maintenance, the study use clustering algorithms. [Table 2] summarizes the methods of partitional clustering algorithms.

C. Methods for Analyzing Purposes

Shtern and Tzerpos [7][33] reviewed the latest findings of software clustering algorithms, discuss the advantages and weaknesses, and clarify directions for further research in the field of software clustering, including algorithm development and improvement, in addition to evaluating current algorithms. They also stated that the process of selecting appropriate clustering algorithms is difficult, so they focused their study on software clustering algorithms. Accordingly, they presented a method of selection based on specific needs.

The study provides formal descriptive templates for software clustering algorithms. The same templates can be used to improve existing algorithms. The output of these algorithms is called software system decomposition. In [9] the paper proposes a novel static concept location technique that combines textual information and structural dependencies in source code, and it outperforms a baseline approach in terms of effectiveness. It is utilized to enhance the identification of locations where modifications are to be made in response to requests for modifications. Researchers [20] present an automated technique that applies self-adaptive evolution strategies to increase the quality of software clustering, which helps in decomposing complex software systems into smaller subsystems that can be controlled and managed. Compared with the genetic algorithm-based approach, it shows better quality in test results. The paper [22] proposes the use of object-oriented analysis and clustering operations in addition to the chi-square test to predict software quality. This method aims to increase the prediction accuracy of software quality. by highlighting the relationships between software components and their attributes. Usage Pattern Based cohesiveness (UPBC), a new cohesiveness metric for object-oriented software that is calculated at the module level, is proposed in [34]. It utilized to break down the complexity of the software's modules to increase maintainability.

According to [4], It highlights the application of clustering tools and methods to rank web services based on the similarity of clusters. Thus, we can obtain the required service faster from large data sets, The study also emphasizes the significance of similarity computation and actual user surveys in assessing the utility and efficacy of clustering techniques in web service discovery. Next paper [14] proposes an approach to evaluate software clustering algorithms in the context of program comprehension by utilizing interaction logs from previous maintenance sessions. The study evaluates the performance of different clustering techniques.

Conclusion

This paper demonstrates cutting-edge experimental contributions in software module clustering. As a result, the methods and tools utilized for these purposes were identified in order to ensure the applicability of modern clustering techniques in the field of software engineering and to enable the clustering process. A total of 36 publications from literature databases that were released between 2008 and 2020 were taken into consideration for the software module clustering analysis in this study. The studies were carefully and professionally reviewed and analyzed from different perspectives to achieve sufficient understanding. Next, the software cluster applications were classified.

We discuss all the algorithms, target software systems, and evaluations that enabled the compilation process. In conclusion, new researchers may find it difficult to handle many facets of the topic of software module clustering due to the volume of research works in the area. As a result, we suggest using this analytical survey as a useful resource to help with the process of obtaining the most pertinent data.

REFERENCES

[1] A. Adolfsson, M. Ackerman, and N. C. Brownstein, "To Cluster, or Not to Cluster: An Analysis of Clusterability Methods," Aug. 2018, doi: 10.1016/j.patcog.2018.10.026.

[2] J. Hwa, S. Yoo, Y. S. Seo, and D. H. Bae, "Search-Based Approaches for Software Module Clustering Based on Multiple Relationship Factors," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 7, pp. 1033–1062, Sep. 2017, doi: 10.1142/S0218194017500395.

[3] M. Kargar, A. Isazadeh, and H. Izadkhah, "Multi-programming language software systems modularization," *Computers and Electrical Engineering*, vol. 80, Dec. 2019, doi: 10.1016/j.compeleceng.2019.106500.

[4] Adhiparasakthi Engineering College. Department of Electrical and Electronics Engineering, Adhiparasakthi Engineering College, and Institute of Electrical and Electronics Engineers, *2017 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*.

[5] V. Singh, "Software module clustering using metaheuristic search techniques: A survey," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, 2016, pp. 2764–2767.

[6] Q. I. Sarhan, B. S. Ahmed, M. Bures, and K. Z. Zamli, "Software Module Clustering: An In-Depth Literature Analysis," Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.01057

[7] M. Shtern and V. Tzerpos, "Methods for Selecting and Improving Software Clustering Algorithms."

[8] S. Arshad and C. Tjortjis, "Clustering software metric values extracted from C# code for maintainability assessment," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, May 2016. doi: 10.1145/2903220.2903252.

[9] G. Scanniello and A. Marcus, "Clustering support for static concept location in source code," in *IEEE International Conference on Program Comprehension*, 2011, pp. 1–10. doi: 10.1109/ICPC.2011.13.

[10] "EEE 4 th International Conference on Knowledge-Based Engineering and Innovation (KBEI) I," 2017.

[11] S. Romano, G. Scanniello, M. Risi, and C. Gravino, "Clustering and lexical information support for the recovery of design pattern in source code," in *IEEE International Conference on Software Maintenance, ICSM*, 2011, pp. 500–503. doi: 10.1109/ICSM.2011.6080818.

[12] IEEE Staff, *2016 1st India International Conference on Information Processing (IICIP)*. IEEE, 2016.

[13] A. Alkhalid, M. Alshayeb, and S. Mahmoud, "Software refactoring at the function level using new Adaptive K-Nearest Neighbor algorithm," *Advances in Engineering Software*, vol. 41, no. 10–11, pp. 1160–1178, 2010, doi: 10.1016/j.advengsoft.2010.08.002.

[14] IEEE Staff, *2013 IEEE 21st International Conference on Program Comprehension*. IEEE, 2013.

[15] G. Scanniello, A. D'Amico, C. D'Amico, and T. D'Amico, "Using the Kleinberg algorithm and vector space model for software system clustering," in *IEEE International Conference on Program Comprehension*, 2010, pp. 180–189. doi: 10.1109/ICPC.2010.17.

[16] J. F. Cui and H. S. Chae, "Applying agglomerative hierarchical clustering algorithms to component identification for legacy systems," *Inf Softw Technol*, vol. 53, no. 6, pp. 601–614, Jun. 2011, doi: 10.1016/j.infsof.2011.01.006.

[17] B. Nasim Adnan, "CLUSTERING SOFTWARE SYSTEMS TO IDENTIFY SUBSYSTEM STRUCTURES USING KNOWLEDGEBASE," 2010.

[18] International Conference on Computational Techniques in Information and Communication Technologies 2016 Delhi, International Conference on Computational Techniques in Information and Communication Technologies 2016.03.11-13 New Delhi, and ICCTICT 2016.03.11-13 New Delhi, *2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT) proceedings : March 11-March 13, 2016, New Delhi, India*.

[19] A. C. Kumari and K. Srinivas, "Hyper-heuristic approach for multi-objective software module clustering," *Journal of Systems and Software*, vol. 117, pp. 384–401, Jul. 2016, doi: 10.1016/j.jss.2016.04.007.

[20] B. Khan, S. Sohail, and M. Y. Javed, "Evolution strategy based automated software clustering approach," in *Proceedings of the 2008 Advanced Software Engineering and its Applications, ASEA 2008*, 2008, pp. 27–34. doi: 10.1109/ASEA.2008.17.

[21] F. Beck and S. Diehl, "Evaluating the Impact of Software Evolution on Software Clustering."

[22] International Conference on Applied and Theoretical Computing and Communication Technology 1. 2015 Davangere, S. K. Niranjan, M. Aradhya, Institute of Electrical and Electronics Engineers Bangalore Section, International Conference on Applied and Theoretical Computing and Communication Technology 1 2015.10.29-31 Davangere, and ICATccT 1 2015.10.29-31 Davangere, *Proceedings of the 2015 International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT) 29-31 October 2015, Davangere, Karnataka, India*.

[23] L. Zhong, J. He, N. Zhang, P. Zhang, and J. Xia, "Software evolution information driven service-oriented software clustering," in *Proceedings - 2016 IEEE International Congress on Big Data, BigData Congress 2016*, Institute of Electrical and Electronics Engineers Inc., Oct. 2016, pp. 493–500. doi: 10.1109/BigDataCongress.2016.75.

[24] A. Corazza, S. Di Martino, V. Maggio, and G. Scanniello, "Investigating the use of lexical information for software system clustering," in *Proceedings of the European Conference on Software Maintenance and Reengineering, CSMR*, 2011, pp. 35–44. doi: 10.1109/CSMR.2011.8.

[25] A. Rathee and J. K. Chhabra, "Software remodularization by estimating structural and conceptual relations among classes and using hierarchical clustering," in *Communications in*

Computer and Information Science, Springer Verlag, 2017, pp. 94–106. doi: 10.1007/978-981-10-5780-9_9.

[26] C. Y. Chong and S. P. Lee, "Constrained agglomerative hierarchical software clustering with hard and soft constraints," in *ENASE 2015 - Proceedings of the 10th International Conference on Evaluation of Novel Approaches to Software Engineering*, SciTePress, 2015, pp. 177–188. doi: 10.5220/0005344001770188.

[27] S. Barman, H. Gope, M. M. Manjurul Islam, M. Hasan, and U. Salma, "Clustering techniques for software engineering," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 4, no. 2, pp. 465–472, Nov. 2016, doi: 10.11591/ijeecs.v4.i2.pp465-472.

[28] Amarjeet and J. K. Chhabra, "Improving package structure of object-oriented software using multi-objective optimization and weighted class connections," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 3, pp. 349–364, Jul. 2017, doi: 10.1016/j.jksuci.2015.09.004.

[29] S. Arshad and C. Tjortjis, "Clustering software metric values extracted from C# code for maintainability assessment," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, May 2016. doi: 10.1145/2903220.2903252.

[30] A. Prajapati and J. K. Chhabra, "A Particle Swarm Optimization-Based Heuristic for Software Module Clustering Problem," *Arab J Sci Eng*, vol. 43, no. 12, pp. 7083–7094, Dec. 2018, doi: 10.1007/s13369-017-2989-x.

[31] IEEE Staff, *2017 International Symposium on Computer Science and Software Engineering Conference (CSSE)*. IEEE, 2017.

[32] J. Huang, J. Liu, and X. Yao, "A Multi-agent Evolutionary Algorithm for Software Module Clustering Problems." [Online]. Available: http://see.xidian.edu.cn/faculty/liujing/

[33] M. Shtern and V. Tzerpos, "Clustering Methodologies for Software Engineering," *Advances in Software Engineering*, vol. 2012, pp. 1–18, May 2012, doi: 10.1155/2012/792024.

[34] A. Rathee and J. K. Chhabra, "Improving Cohesion of a Software System by Performing Usage Pattern Based Clustering," in *Procedia Computer Science*, Elsevier B.V., 2018, pp. 740–746. doi: 10.1016/j.procs.2017.12.095.