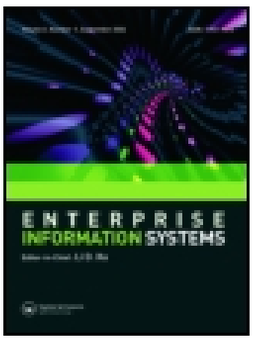

Enterprise Information Systems

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/teis20

# Clustering-based software modularisation models for resource management in enterprise systems

Jiahua Li & Ali Yamini

**To cite this article:** Jiahua Li & Ali Yamini (2020): Clustering-based software modularisation models for resource management in enterprise systems, Enterprise Information Systems, DOI: 10.1080/17517575.2020.1830307

**To link to this article:** https://doi.org/10.1080/17517575.2020.1830307

Published online: 06 Oct 2020.

Submit your article to this journal

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=teis20

ENTERPRISE INFORMATION SYSTEMS
https://doi.org/10.1080/17517575.2020.1830307

Taylor & Francis
Taylor & Francis Group

Check for updates

# Clustering-based software modularisation models for resource management in enterprise systems

Jiahua Li [a] and Ali Yamini [b]

[a]School of Information Engineering, Guangzhou Vocational and Technical University of Science and Technology, Guangzhou, Guangdong 510550, China; [b]Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

**ABSTRACT**

Software module clustering approaches can provide a better understanding of large and complex software systems through decomposing the enterprise resources into classified modules which are smaller, and therefore easier-to-handle. As the dimensions and complexity of enterprise software projects are continuously increasing, handling a large software project is going to be more challenging. The challenge would be more complex if the experienced personnel is considered as well. Therefore, appropriate automatic software modularization clustering methods are required in resource management. This paper provides a Systematic Literature Review (SLR) on the software modularization clustering models. We studied a wide range of papers from 2001 to 2020 to provide our SLR. Also, a technical taxonomy is presented to classify the existing papers on software modularization clustering models and algorithms. The software module clustering methods are categorized into three main classes. Finally, new challenges and forthcoming issues of software modularization clustering models are presented.

**ARTICLE HISTORY**
Received 7 June 2020
Accepted 26 September 2020

**KEYWORDS**
Data mining; software modularisation; clustering method; resource management; enterprise systems

## 1. Introduction

In data mining techniques, clustering as an unsupervised method is considered as one of the important approaches in software development (Xie et al. 2009). The main purpose of the software module clustering process is to divide a large software system into its subsystems and provide an abstract model of software complex architecture. Having an articulated architecture, software developers could have a better analysis of the software modularisation. An enterprise software system should be clustered in such a way that the relationship between modules (Czekster et al. 2019), which are in the same cluster (intra-connections), is maximised and the relationship between the two clusters (inter-connections) is minimal. In other words, in a clustering process, the entities that are more similar to each other are put in the same cluster (Rathore and Kumar 2019).

**CONTACT** Jiahua Li huahua888219@163.com School of Information Engineering, Guangzhou Vocational and Technical University of Science and Technology, Guangzhou, Guangdong 510550, China

© 2020 Informa UK Limited, trading as Taylor & Francis Group

The issue of turning software modules into bigger structures same as container shaped structures is called software module clustering. To split a software system into various modules, the Modularisation method comes to assist. The resulted modules from modularisation are discrete, independent, and also, can handle the assigned tasks (Venkatesan and Sridhar 2019). Generated modules can be utilised as a building base for the whole software and developers intend to create them to be able to be run and/or compiled individually while depending on themselves. Since there are lots of other interests in software modularisation, this design comes with a rule called 'divide and conquer' which is a problem-solving strategy. Scaled software systems can include thousands of modules and there is a need to organise these modules to assist future developments in finding the target module that belongs to a certain task (Spijkman et al. 2019). To achieve this goal, modules will be separated into clusters that if done properly, can assist in recognising responsible modules for each functionality, facilitating pathfinding between software segments, and also improving comprehension. Having these benefits will result in ease of development and maintainability (Sasidharan 2019).

Different software module clustering methods are proposed and employed before, such as hierarchical clustering, probabilistic, density-based, constraint-based, subspace, partitioning relocation, distribution-based, and grid-based clustering (Solorio-Fernández, Carrasco-Ochoa, and Martínez-Trinidad 2019). Amongst them, distribution-based (Preheim et al. 2013), hierarchical (Kamis, Chiclana, and Levesley 2018), and density-based clustering (Campello, Moulavi, and Sander 2013) methods are widely used in the software development process.

Hierarchical clustering forms a tree of clusters or a cluster hierarchy. Each of the cluster nodes comprises sub-clusters; fraternal clusters divide the nodes protected by their parent. This approach allows for discovering data on various levels of granularity (Berkhin 2006). Another category of clustering method is closely related to statistics, in particular, to the distribution model. Distribution-based clustering methods define the clusters as objects belonging to the same distribution. Distribution-based clustering creates complex models that can find and show correlation and dependence among attributes. Although these algorithms put an additional burden on the user, they have attracted software developers' attention (Ramanathan et al. 2018).

To explain density-based clustering, propose an exposed set in the Euclidean environment. This set can be separated into a set of components that are connected. To implement this idea for dividing a finite set of nodes, we require to know what is a boundary, connectivity, and density (Wieland et al. 2007). A cluster is a set of components densely connected. As the density increases in a direction, the cluster is also growing in that direction (Souri et al. 2020). Therefore, density-based clustering can find any shapes.

To the best of our knowledge, there is no comprehensive and detailed review of the software module clustering approaches. In this paper, a Systematic Literature Review (SLR) (Souri et al. 2019) is presented for software module clustering approaches. First, a technical taxonomy is presented to classify the existing software module clustering methods and algorithms in enterprise systems. The software module clustering methods are categorised into three main classes including hierarchical, distribution-based, and density-based clustering methods. The main contributions of the proposed SLR on software module clustering approaches are as follows:

- Presentation of a review and analysis for software module clustering approaches covering 34 research studies.
- Presentation of a technical taxonomy of software module clustering methods and applied algorithms.
- Analyzes and discussion on the technical impacts and features of each research study.
- Presentation of the open issues and challenges on the software module clustering approaches.

The reason that we relate the importance of this paper in enterprise systems is the fact that there are some important issues in software modularisation that have a huge impact on high complex enterprise systems. Solving issues like reducing execution time, and making cost-efficient modularizations can be helpful in lots of enterprise systems problems.

The rest of this paper is organised as follows: Section 2 presents the SLR research finding for present software module clustering studies. Section 3 illustrates a technical taxonomy for software module clustering approaches and a side-by-side analysis and summary for each research study based on the presented taxonomy. Section 4 presents an analytical discussion based on reviewed research studies. Also, some new challenges and open research directions are shown in this section. Finally, the conclusion and limitation metrics are illustrated in Section 5.

## 2. Research planning and methodology

In this section, we offer a research finding method considering Kitchenham et al. (2009) to introduce SLR on the software modularisation methods and our analytical review based on it. According to this approach, Figure 1 illustrates three procedures for collecting, refining, and analytical review. As Figure 1 illustrates, there are three phases in our research plan, including collection, refinement, and analytical review. Inside the collection phase, our study objectives using a tested research finding protocol (Souri, Navimipour,

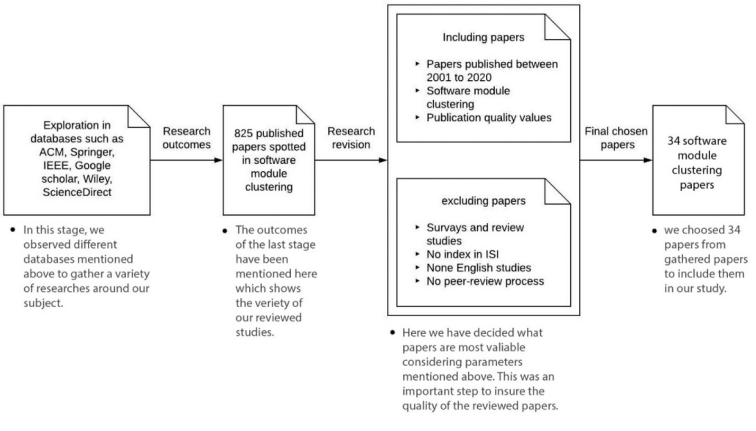


**Figure 1.** Research selection strategy based on SLR for software module clustering.

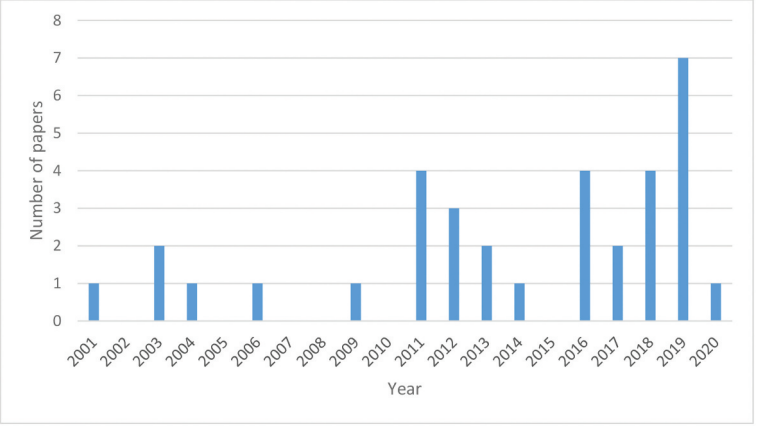


**Figure 2.** Publication per year analysis for software module clustering approaches.

and Rahmani 2018). In this convention, the leading keywords will be applied to perform a search inside scientific databases. These keywords are [12]:

'Software' | 'System' | 'Application' & 'Module' | 'Modularisation' & 'Clustering'

The confirmation of the papers that are chosen to our keywords in scientific databases will be done directly which these databases are publishers such as, IEEE, Springer, Wiley, Elsevier, ACM. In the next phase which is refinement, the evaluation and comparing of the selected research studies to the response of analytical questions will be done. In the end, the analytical review stage, the finishing outcomes will be stated and the final research studies will be gathered.

As shown in Figure 2, a statistical time scale of software modularisation clustering papers per publication year is presented. 34 papers were gathered in the final paper collection.

In the following some analytical questions are provided for answering technical features of the SLR approach on the software module clustering methods, to evaluate the review analysis for each paper study:

RQ1: Which techniques and applied algorithms are considered for existing studies?

RQ2: What are the evaluation factors for examining software modularisation models?

RQ3: Which machine learning methods were applied to evaluate software modularisation models?

RQ4: Which tools and environments are considered in software modularisation models?

## 3. Software modularisation models

One of the problematic issues of automatic decomposition of software units into their modules is software module clustering. It helps to enhance the structure of a software and improves the readability of software. Software module clustering results in easier navigation and easier tracking among software components and enhances system understandings. Therefore, a good distribution of the modules simplifies the process of software development and maintenance. Several studies on software module clustering have been

conducted with different goals such as: to analyse large systems, to increase economic profits, to make development cycles shorter, to study interactions among entities in an object-oriented system, to enhance the comprehensibility of software module clustering results; as reviewed in the rest of the current subsection.

Figure 3 presents a taxonomy for categorising existing machine learning-based software modularisation models concerning three main classes including hierarchical, distribution-based, and density-based clustering methods. According to the proposed taxonomy, the whole software module clustering is categorised into three sections including hierarchical-based software module, density-based software module, and distributed-based software module. Also, hierarchical-based software module has over three clustering method which includes meta-heuristic clustering, classical clustering, and cooperative clustering. The density-based software module consists of two methods including fuzzy clustering and meta-heuristic clustering.

### 3.1. Hierarchical clustering software modularisation

Hierarchical clustering forms a tree of clusters or a cluster hierarchy. Each of the cluster nodes comprises sub-clusters; fraternal clusters divide the nodes protected by their parent. This approach allows for discovering data on various levels of granularity (Souri et al. 2019a; Zhao et al. 2019).

Sun and Ling (2018) proposed a software algorithm that uses contingency selection for software module clustering. They changed the problem of software module clustering to graph clustering problem. They solved the software module clustering problem through SPS which is one of the clustering algorithms of the software module.

On the other hand, Alkhalid, Alshayeb, and Mahmoud (2011) presented a software-based refactoring through clustering methods. The paper also illustrated two approaches in the software level. The first one utilises the same number of software in clustering, and the other one utilises the changeable number of software in clustering. The proposed algorithm in the first approach was a k-nearest neighbour (KNN). Based on results it enhances software unity and decreases software pairing. Besides, applied clustering methods in the second approach are as follows: WPGMA, CLINK, and namely Slink. The

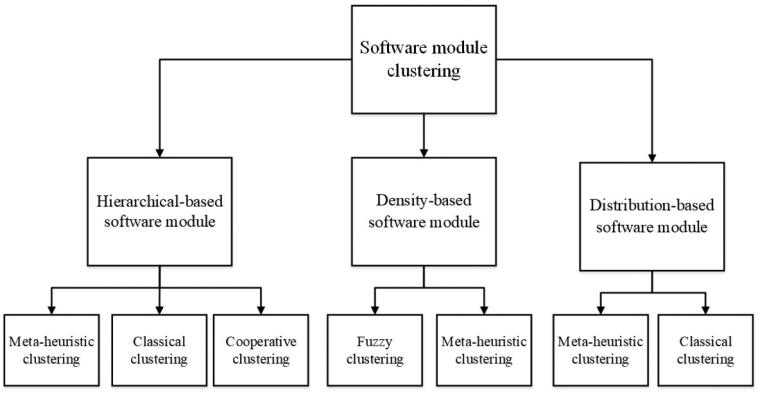


**Figure 3.** Proposed taxonomy for classifying machine learning-based software modularisation models.

applied tool in this research was Trama which demonstrates various graphical UI (user interface) to graphically work with the matrices. The main positive point of the study was presenting new methods that make less the complexity of the computation through competitive performance. However, the paper did not compare the refactoring by using the proposed approaches.

However, Anquetil and Lethbridge (2003) proposed three decisions that need to be made when clustering: the choice of abstract details of the entities to be clustered, metrics to gauge coupling between the entities, and clustering algorithms. Muhammad, Maqbool, and Abbasi (2012) presented different methods for the automatic modulation and architecture recovery of software systems because selecting an appropriate algorithm is very important and has a significant impact on the quality of the result.

Siddique and Maqbool (2012) presented a technique that increases the comprehensibility of software module clustering results. They used TWS to determine which schemes in the software domain work well, and the identification of software characteristics

Huang and Liu (2016) proposed an algorithm that modularises the quality of measure for software module clustering issues, and for evaluating the solutions they designed two evaluation based on software design requirements. Naseem, Maqbool, and Muhammad (2013) proposed a method for software module based on the collaboration of more than one parallel measures. They presented this method for both binary and non-binary features' software module clustering strategy.

Also, Zhao and Zou (2011) proposed an algorithm that analyzes dependencies in data and tasks to extract software modular structures from business processes and group them into a software component automatically. Mitchell, Traverso, and Mancoridis (2001) discussed the way of individuating nodes in a computer network to be used as a collection of connected processing elements for improving the performance of a software engineering tool which is developed. Naseem, Deris, and Maqbool (2014) explored the idea of Cooperative Clustering for software module which joins the strengths of likeness and coldness measures together at the same process. Pavithr and Garg (2011) suggested a novel objective algorithm that achieves the best answer for the software automatic clustering issue. The author also proposed Filtered Turbo to find the library modules which filtered from the useful clusters.

Bishnoi and Singh (2016) presented a multi-objective method for software modularisation which uses Particle Swarm Optimisation. For analysing the performance of the algorithm, they have used open-source java software systems

Seo and Huh (2019) have introduced a module clustering approach to create the structure of the software system in module shape. Their GUI-build module clustering method can create GUI-made structures and by utilising dynamic software analysis they were able to recognise software modules that are associated with functionalities of GUI modules. The GUI-build method is compared to SHA, SSA, and THA approaches in Weka.

Sadat Jalali, Izadkhah, and Lotfi (2019) have proposed a fitness function to do software modularisation from source code. The multi-objective function's goal is to modularise software systems considering features that can be either structural or non-structural. The evolutionary algorithms are enabled by using the presented objective function. The outcomes of this study have been examined using various criteria such as Mojo and MojoFM.

Kargar, Isazadeh, and Izadkhah (2019) has introduced a unified structural and semantic concept to enhance the quality of software modularisation. In the paper, dependency graphs such as Call Dependency Graph (CDG), Semantic Dependency Graph (SDG), and Nominal Similarity Graph (NDG) are made through source code. Also, a genetic algorithm is introduced to help in modularising programs that are multilingual. Mozilla Firefox is used to show experimental outcomes from utilising SDG, NDG, and structural build graphs.

Imran (2019) discusses the design smells by utilising an Abstract Syntax Tree (AST) build tool. The tool helps to recognise exact design qualities and makes relationships among breaches of those along with the existence of design smells. The outcomes of applying the tool on class files and a big segment of the java project demonstrated that normally there is a relation between an exact software, quality properties, and the existence of design smells. Also, some design smells are more common in software development.

Tabrizi and Izadkhah (2019) have introduced a combination of search-based and hierarchical algorithm to use the benefits of both algorithms in software modularisation. The outcomes of testing the algorithm on software systems show that the quality is enhanced using the following method. Also, using this method's hierarchical characteristics understanding software structure is simpler.

Kargar, Isazadeh, and Izadkhah (2020) proposed semantic dependency and nominal similarity graphs that are made through the syntax of programming languages. Also, a hybrid graph called SNGA is introduced that combines the semantic graph and nominal similarity graph. The experimental outcomes that come from tests on Mozilla Firefox, implied that modularity quality is enhanced compared to the semantic graph.

As shown in Table 1, hierarchical-based software module clustering models can be compared and analysed based on different main ideas, clustering methods, applied algorithms, simulation environments, and evaluation factors respectively. These factors are used in data mining and computation techniques. Some of the papers bellow have

**Table 1.** Comparison of the hierarchical software module clustering models.

| Research | Main idea | Applied algorithm | Simulation environment | Evaluation factors |
|---|---|---|---|---|
| Sun and Ling (2018) | Software module based on the complexity of the structure | SPS | GGA, GNE, MCA, ECA, MAEA-SMCPs | Time |
| Alkhalid, Alshayeb, and Mahmoud (2011) | Software-based refactoring through clustering method | A-KNN | Trama | Dencity |
| Anquetil and Lethbridge (2003) | Representation of heuristic algorithm for software remodularization | Hill-climbing algorithm | GCC, Linux and Mosaic | Precision |
| Muhammad, Maqbool, and Abbasi (2012) | Study relationships between entities in an object-oriented system | WA, UWA, CL, CB, WC, BUNCH, ACDC | DDA (c ++ program), LIMBO, Jedit v4.1, Jhotdraw v5.3 and Jfreechart v1.0.13 | Precision, recall, time |

(*Continued*)

**Table 1.** (Continued).

| Research | Main idea | Applied algorithm | Simulation environment | Evaluation factors |
|---|---|---|---|---|
| Siddique and Maqbool (2012) | Enhancing the comprehensibility of software module clustering results | TWS, LIMBO | C/C++ Systems: Xfig, Chocolate Doom, Mozilla, Weka, compost | Accuracy, time |
| Huang and Liu (2016) | Similarity-based modularisation quality measure | HC, GA, MAEA | HC GA MAEA | Error rate, time |
| Naseem, Maqbool, and Muhammad (2013) | Cooperative clustering for software modularisation | CCT | Jaccard-NM NFV | Time, number of clusters |
| Xulin Zhao and Zou (2011) | Generating software Modules automatically | ACDC | BPE | Accuracy, precision, time |
| B. Mitchell, Traverso, and Mancoridis (2001) | Distributing computation of clustering | SAHC NAHC GA | C++ Java RMI MDG Bunch | Accuracy, precision, error rate, time |
| Naseem, Deris, and Maqbool (2014) | Cooperative-based clustering method | CC | CA | Time |
| Pavithr and Garg (2011) | Automatic Clustering of Software-Intensive Systems | FTMQ | MDG ETMQ Graphviz | Time |
| Bishnoi and Singh (2016) | Modularising software systems using PSO | - | Java Bunch | Error rate, time |
| Tabrizi and Izadkhah (2019) | Software modularisation using genetic and hierarchical | GA Single-linkage | Bison Boxer Compiler ISpell Mini-Tunis | Time |
| Kargar, Isazadeh, and Izadkhah (2019) | Multi-programming language | SGNA Bunch DAGC HC ECA SGA | Mozilla Firefox 3.7 | Precision, recall, time |
| Kargar, Isazadeh, and Izadkhah (2020) | Improving modularisation quality | SDGA SNDGA | Mozilla Firefox | Precision, recall, time |
| Imran (2019) | Smell Analysis for Java Software | - | - | Accuracy, recall, time |
| Sadat Jalali, Izadkhah, and Lotfi (2019) | Inheritance dependency-based software modularisation | GA, Hill climbing | Mozilla Firefox | Time |
| Seo and Huh (2019) | GUI-based software modularisation in edge computing | HCA | Java SWT/Swing Weka | Time |

used fuzzy clustering methods to use this method's feature that each data point can belong to more than one cluster. Also, some classical clustering methods have been considered to recognise groups of observations that have similarities respecting some number of variables.

### 3.2. Density-based clustering software modularisation

To explain density-based clustering, propose an exposed set in the Euclidean environment. This set can be separated into a set of components that are connected (Xu and Chen 2014).

Mitchell and Mancoridis (2003) proposed an evaluation approach based on the search metaheuristic software module clustering algorithms. They examined the Bunch cause, specifies subsystem hierarchy by search techniques and that makes useful results for different systems.

Also, Köhler, Fampa, and Araújo (2012) presented a solution for the software module clustering problem with programming formulations of mixed-integer linear. They formulated the SCP as a sum of linear fractional problem, then reformulated Mixed-Integer Linear Programming (MILP) problems by applied two different linearisation procedures. Wan and Wu (2009) proposed the Fuzzy decision and support vector clustering applied to optimal reliability allocation for a modular software system. This model boosts economic benefits and applied to the NC system for reliability distribution. Mitchell and Mancoridis (2006) designed a framework for maintainers which helps to understand the big and complex software system that uses search techniques to perform clustering.

Praditwong, Harman, and Yao (2011) proposed a solution including cohesion and coupling for the software module clustering problem as a multi-search. Certainly, both of these approaches are better than the single-objective solution and may improve performance.

While Kumari and Srinivas (2016) answered the question of solving the multi-objective software module clustering issue using the presented approach. This issue was researched by two multi-objective approaches to clustering and five objectives for every one of them. They planned to work on another method on software module clustering problems. A Kumari, Srinivas, and Gupta (2013) presented an algorithm to find a solution for software module clustering issues which is based on a multi-objective genetic algorithm. This approach maintains the correct stability between exploration and exploitation of the search space due to the author's claim.

Amarjeet and Chhabra (2017) presented an algorithm for software module clustering problems that cover a great number of objective functions and conclusions by using the many-objective algorithm. Mu, Sugumaran, and Wang (2019) have discussed the maintainability of software architectures and emphasises on a model to perform an automatic software remodularization on systems. The hybrid genetic algorithm (HGA) can automatically recognise solutions for high-quality software modularisation. Along with that, the proposed solution is enhanced using a customised genetic algorithm (GA). The paper also compares HGA with hill-climbing algorithm (HCA) and the genetic algorithms having group number encoding (GNE).

As shown in Table 2, density-based software module clustering models can be compared and analysed based on different main ideas, clustering methods, applied algorithms, simulation environments, and evaluation factors respectively. These factors are used in data mining and computation techniques.

**Table 2.** Comparison of the density-based software module clustering models.

| Research | Main idea | Applied algorithm | Simulation environment | Evaluation factors |
|---|---|---|---|---|
| Mitchell and Mancoridis (2003) | Search landscape using a bunch | Search algorithms | Bunch, Kerberos5, | Time |
| Köhler, Fampa, and Araújo (2012) | Linear programming-based modularisation | Linear Programming | C++ CPLEX | Time |
| Wan and Wu (2009) | Increasing economic profit development cycles | SVM | – | Time |
| Mitchell and Mancoridis (2006) | Automatic modularisation of software systems | Hill-climbing | MDG (C++ program) | Recall, time |
| Praditwong, Harman, and Yao (2011) | Heuristic-based software module clustering | MQ (hill-climbing) | ECA, MDG | Time |
| Kumari and Srinivas (2016) | Hyper-heuristic for multi-objective module | MHypEA MCA ECA | MDG | Accuracy, time |
| Kumari, Srinivas, and Gupta (2013) | Hyper-heuristic for multi-objective module | MCA Genetic algorithm | Bunch Graph drawing tool | Time |
| Amarjeet and Chhabra (2017) | Artificial bee colony algorithm for large-scale modules | MaABC | MCA ECA | Error rate, time |
| Mu, Sugumaran, and Wang (2019) | A hybrid genetic algorithm for re-modularisation | GA | - | Time |

### 3.3. Distribution-based clustering software modularisation

Distribution-based clustering methods define the clusters as objects belonging to the same distribution. Distribution-based clustering creates complex models that can find and show correlation and dependence among attributes.

Parsa and Bushehrian (2004) presented the DAGC software environment that helps to automatic modularisation of software systems in research works by genetic clustering algorithms designing and development. Due to the distinctive features of DAGC, a clustering algorithm was evolved for the Bunch genetic clustering algorithm components by trying different schemes. Prajapati and Chhabra (2017) presented the applicability and usefulness of particle swarm optimisation (PSO) based module clustering (PSOMC) to solve the software module clustering problems (SMCPs). They redesigned the particle situation and rapidity of the PSO algorithm according to the SMCPs and redefined particle update procedure.

Monçores, Alvim, and Barros (2018) evaluated a method to solve the module clustering issues. Their study evaluated some alternatives, strategies, and distinct values for a different part of the method.

Amarjeet and Chhabra (2018) presented a fuzzy-Pareto (FP) method for resolving many objective software improvement issues which can make a qualified software module clustering result related to other algorithms. In this research, the evaluation factors have been introduced into the ABC to help the strategy for leading to a premiere search area. The outcomes represented the ability of the FP strategy for capturing an improved module clustering solution.

Huang, Liu, and Yao (2016) proposed an algorithm for solving the software module clustering problems. The trials showed a good presentation and the contrast presented that MAEA-SMCPs outperform two existing single-objective algorithms and two existing multi-objective algorithms in terms of MQ. Varghese, Raimond, and Lovesum (2019)

**Table 3.** Comparison of the distribution-based software module clustering models.

| Research | Main idea | Applied algorithm | Simulation environment | Evaluation factors |
|---|---|---|---|---|
| Parsa and Bushehrian (2004) | Facilitation research work of genetic clustering algorithms | Genetic algorithm | DAGC | Time, number of clusters |
| Prajapati and Chhabra (2017) | Intercluster dependency-based modularisation | PSO | Java | Accuracy, recall |
| Monçores, Alvim, and Barros (2018) | Large neighbourhood search applied to the | LNS SMC | JodaMoney | Error rate, time |
| Amarjeet and Chhabra (2018) | Using FP-ABC Algorithm for Objective Software Module Clustering | F-ABC | - | Error rate, time |
| Huang, Liu, and Yao (2016) | Multi-agent evolutionary algorithm | MAEA | Bunch MQ | Error rate,, time |
| Varghese, Raimond, and Lovesum (2019) | Automatic re-modularisation of software using colony optimisation | ACO | Bunch | Time |
| Zamli et al. (2019) | Software module clustering base on fuzzy adaptive learning | Fuzzy teaching-learning algorithm | - | Time |

presented an approach to re-modularise software systems automatically by utilising an expanded Ant Colony Optimisation (ACO) algorithm. This is because maintaining the high cohesion and low coupling is important to have adhered to the basic theories of software modularisation. Authors evaluate their method by different software systems and the outcomes demonstrate the advantage of it over methods such as I-GAs, Bunch-GA, and Bunch-HC.

Zamli et al. (2019) demonstrated the productivity of the Adaptive Fuzzy Teaching Learning Based Optimisation (ATLBO) algorithm compared to Fuzzy Adaptive Teaching Learning Based Optimisation (FATLBO) algorithm in software module clustering related applications. Since Teaching Learning Based Optimisation (TLBO) algorithm doesn't have the best performance on controlling exploration, exploration, The ATLBO was able to be even more efficient from the TLBO which is the original idea in making the best MQ unit.

According to Table 3, distribution-based software module clustering models can be compared and analysed based on different main ideas, clustering methods, applied algorithms, simulation environments, and evaluation factors respectively. These factors are used in data mining and computation techniques.

## 4. Discussion

The section will discuss existing software modularisation methods comparatively and technically. As it is stated by analytical questions in segment 3, technical and statistical answers are responded as follows:

RQ1: Which techniques and applied algorithms are considered for existing studies?

As illustrated in Figure 4, the hierarchical clustering strategies are utilised more than density-based and distribution-based strategies. Since those are some of the powers of hierarchical clustering, ease of using, understanding, and clear mathematical calculations are possible. The dendrogram tree for software module clustering is the leading output of the hierarchical clustering approach. In the writing of this paper, it is illustrated that software architecture recovery, software module, and software metric approaches, mostly use a hierarchical model, since the structure of the data is hierarchical it should be utilised

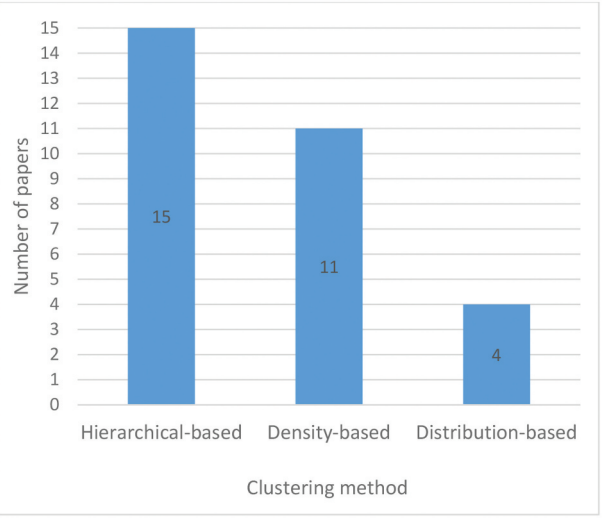


**Figure 4.** Percentage of a clustering method for software module clustering approaches in the literature.

for testing approaches. Generally, density-based and distribution-based are more suitable alternatives because the hierarchical approach should be utilised for non-numerical and data that is not independent.

Figure 5 shows that various algorithms have been presented in the research studies which we have seen algorithms by the percentage that they have been utilised for the case study. Most of these algorithms that have been utilised were clustering algorithms such as

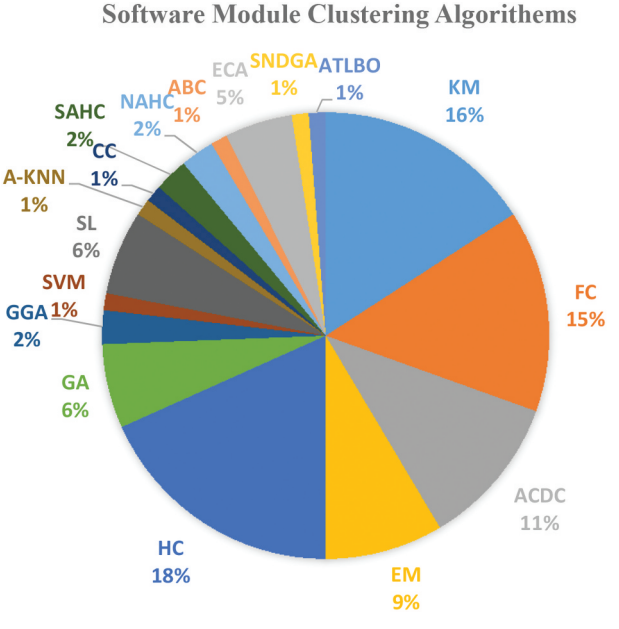


**Figure 5.** Comparison of applied algorithms in the software modularisation models.

K-mean and Fuzzy. When we have great variables K-means show that they can produce tight clusters along with their faster computation compared to others. The fuzzy algorithm has its best used as a software module clustering algorithm since it is simple and flexible and we consider a degree of truth instead of usual true or false Boolean logic. Fuzzy can solve problems that have minimum data and it can handle them at a low cost. On the other hand, these algorithms might be utilised less than algorithms such as TB, GN, SAHC, NA, LA, ABC, HCA which are illustrated in Table 2, but they can be optimised to be used more frequently in the future. You can see a brief description of these algorithms in Table 4.

RQ2: What are the evaluation factors for examining software modularisation models?

Figure 6 shows that in clustering methods factors such as accuracy, precision, recall, error rate, and response time are being utilised to have an accurate evaluation and comparison. These evaluations implied that response time and accuracy were the most important quality factors and F-score had the lowest contribution in these evaluations compared to the other two factors. At large, six main factors are presented here, these six factors are used greatly in the studied papers since they can be great at reviewing different aspects of quality in software modularisation models, but to have more precise evaluation other factors can be applied as well such as privacy and performance.

RQ3: Which machine learning methods were applied to evaluate software modularisation models?

Based on the usage of the SLR method on the software module clustering papers in Figure 7, IEEE has the biggest number of articles investigation, since 12 papers from 34 studies have been published by IEEE. Figures 8 and 9 demonstrates the distribution of papers among authors and their countries.

**Table 4.** List of applied algorithms with full name.

| Abbreviation | Full name |
|---|---|
| KM | K-mean |
| FC | Fuzzy Clustering |
| ACDC | Algorithm for Comprehension Driven Clustering |
| EM | Expectation Maximization |
| GA | Genetic Algorithm |
| GGA | Grouping Genetic Algorithm |
| SVM | Support Vector Machine |
| HC | Hill Climbing |
| BA | Bunch Algorithm |
| CL | Complete Linkage |
| SL | Single Linkage |
| A-KNN | Adaptive K-nearest Neighbour |
| CC | Cooperative Clustering |
| NAHC | Next Ascent Hill Climbing Algorithm |
| SNDGA | Sequential Nominal Dynamic Genetic Algorithm |
| ABC | Artificial Bee Colony |
| ATLBO | Adaptive Fuzzy Teaching Learning Based Optimisation |
| SPS | Simple Filtering Algorithm |
| WC | Weighted Clustering Algorithm |
| CB | Cluster-Based Algorithm |
| WA | Weighted Average Algorithm |
| UWA | Unweighted Average Algorithm |
| TWS | Term Weighting Schemes |
| MAEA | Multi-Agent Evolutionary Algorithm |
| CCT | Cooperative Clustering Technique |
| SAHC | Steepest Ascent Hill Climbing |
| FTMQ | Filtered Turbo MQ |

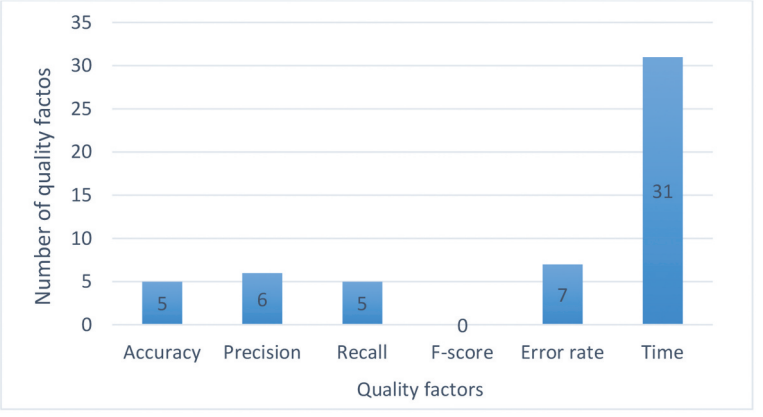


**Figure 6.** Comparison of quality factors in the software modularisation models.

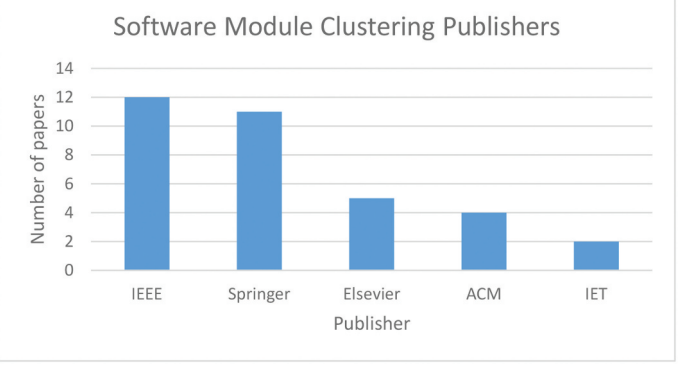


**Figure 7.** A number of publishers in existing papers of the software modularisation models.

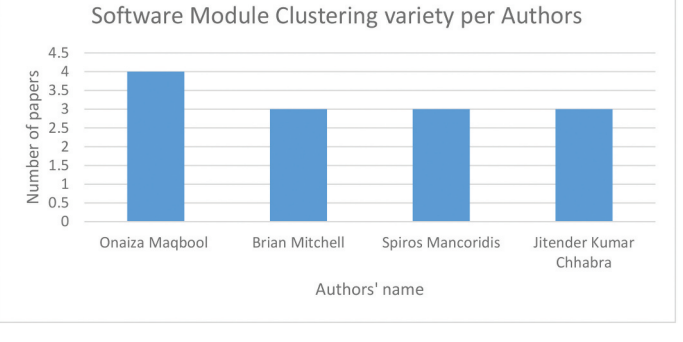


**Figure 8.** A number of authors in existing papers of the software modularisation models.

Figure 10 points out that, Pakistan has the first rank among countries of authors contributing in these 34 papers, and authors Onaiza Maqbool published 4 papers among these researchers. According to the current research papers, Onaiza Maqbool emphasises on subjects including software modularisation clustering, software

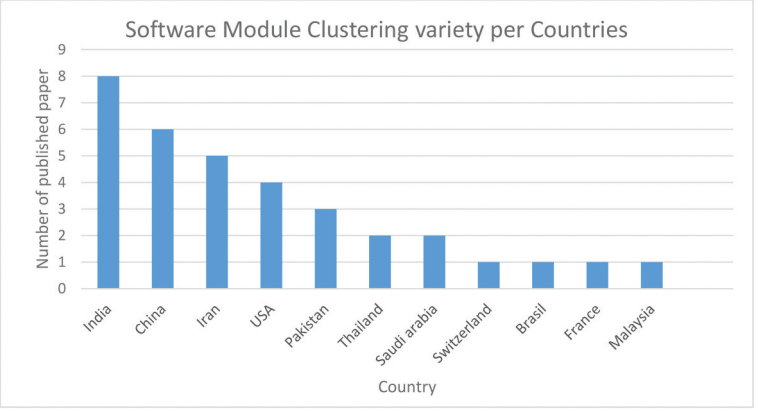


**Figure 9.** A number of countries in existing papers of the software modularisation models.

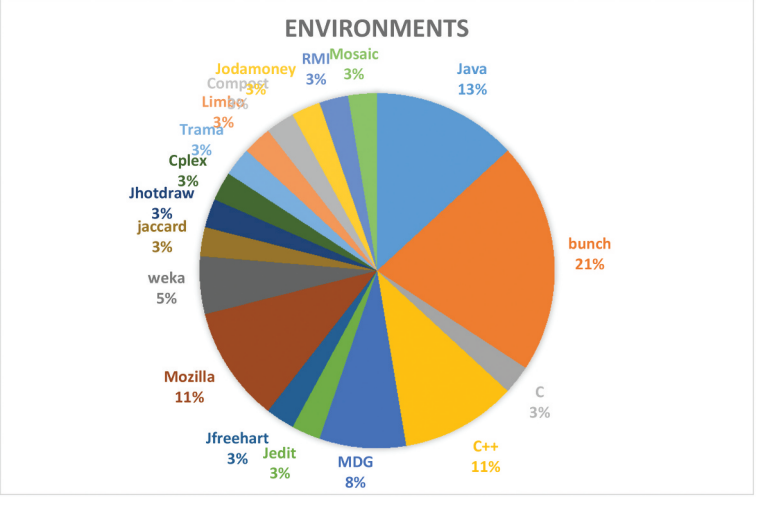


**Figure 10.** Comparison of environments, programming languages, and tools in the software module clustering approaches.

architecture recovery, and software metric measurement using cluster labelling, meta-heuristic analysis on graph dependencies and multiple combined clustering algorithms. Software module clustering, software architecture recovery, and software metric measurement approaches.

Figure 9 shows the number of articles of software module clustering in each country. Based on the nationality of leading authors in this study, India, China, and Iran have more research activity in these specific topics.

Considering Table 5, information from research papers and publishers that focus on software module clustering approaches have been extracted. Table 5 also, is useful to understand the impact factor of related journals. Table 5 presents some of the contributed journals in the published papers that have been studied in this paper. The table also compares the impact factor of each journal along with their quartile.

**Table 5.** Top Journals and publishers in existing papers of the software module clustering approach in 2019.

| Journal | Publisher | Impact factor | Quartile |
|---|---|---|---|
| Proceeding of IEEE | IEEE | 10.25 | Q1 |
| IEEE Transactions On Cybernetics | IEEE | 10.38 | Q1 |
| IEEE Transactions On Dependable And Secure Computing | IEEE | 6.8 | Q1 |
| Information Sciences | Elsevier | 5.52 | Q1 |
| Applied Soft Computing | Elsevier | 4.87 | Q1 |
| Soft Computing | Springer | 4.87 | Q2 |
| Empirical software engineering | Springer | 4.45 | Q2 |
| IEEE access | IEEE | 4.09 | Q1 |
| IEEE Transactions On Software Engineering | IEEE | 3.33 | Q2 |
| Computers & Operations Research | Elsevier | 3.00 | Q1 |
| Information And Software Technology | Elsevier | 2.92 | Q2 |
| The Journal Of Systems And Software | Elsevier | 2.55 | Q1 |
| Automated software engineering | Springer | 2.20 | Q3 |
| Cluster computing | Springer | 1.85 | Q2 |
| Computer Languages, Systems & Structure | Elsevier | 1.84 | Q2 |
| ACM Transactions on Computer Systems | ACM | 1.76 | Q1 |
| ACM Transactions on Information Systems | ACM | 1.76 | Q1 |
| Arabian Journal for Science and Engineering | Springer | 1.51 | Q3 |
| Journal of Software evaluation and process | Wiley | 1.30 | Q2 |
| Electronic Notes In Theoretical Computer Science | Elsevier | 1.2 | Q2 |
| IET software | IET | 1.07 | Q3 |
| Computer | IEEE | 0.98 | Q1 |
| The computer journal | Oxford | 0.98 | Q3 |
| ACM Transactions on Computer-Human Interaction | ACM | 0.97 | Q3 |

RQ4: Which tools and environments are considered in software modularisation models?

Figure 10 shows that in the papers listed in this paper, various tools, programming languages, and environments were utilised for software module clustering approaches. In Figure 10, a comparison between the simulation environments of these papers is illustrated. Programming languages including C++, Java, and some others were in the related researches.

According to Figure 10, some various languages and environments have been used by each paper. As illustrated, Bunch with 21% has a big share of usage in studied papers along with Java with 13% of usage.

### *4.1. Open issues and new challenges*

According to the above analytical results, there are some open issues an new challenges in the software modularisation clustering which have not been analysed comprehensively to cover the way for upcoming studies. We explain some open issues for software modularisation clustering models briefly as follows:

In the software module clustering, finding an optimal dependency graph between modules is an open issue to decrease inter-connections between independent modules and maximise intra-connections between dependent models.

- Finding the optimal similarity matrix for software modularisation is another open issue to minimise the complexity and size of the software system for software module clustering problems. To solve these problems, several improved binary similarity measures have been introduced (Shen et al. 2016; Zhang, Ding, and Zhang 2020).

- Detecting the cohesion metric using meta-heuristic algorithms is one of the main challenges to evaluate the grade of intra-dependability between software modules (Wang and Chen 2020; Zhao et al. 2014).
- Formal analysis of software module clustering can be useful to evaluate the correctness of the inter-dependability and intra-dependability between modules (Souri et al. 2019b). Also, nature-inspired computing can be applied to find the optimal number of dependencies in enterprise-based software modules (Wang et al. 2017).
- Verification of cohesion and Coupling metrics is very essential to support reachability conditions for monitoring level of inter-dependability and intra-dependability between modules (Rodriguez, Piattini, and Ebert 2019).
- Reducing execution time for software modularisation is an important challenge for highly complex enterprise systems with million lines of codes using meta-heuristic algorithms (Y. Xu et al. 2019).
- Cost-efficient modularisation can be enhanced on dynamic decision making for enterprise systems. Evolutionary algorithms and fuzzy logic can be applied to increase the efficiency of dynamic decision making for software modularisation (Chen et al. 2020; Kataev et al. 2020).

## 5. Conclusion and future work

Clustering technics in software engineering can be very effective in decreasing development stage errors and increases the performance, effectively managing and can be utilised for other development schemes as well. Software modularisation clustering is the act of categorising software modules and application phases into a set of batches. In the development stages of a software modularisation, clustering can provide a notable effect on managing, evaluating, and monitoring mechanisms.

Selecting the correct method for clustering is an important and challenging phase in software modularisation. The parameters that are used in software module clustering can be often in contrast with each other and this has a negative effect on other parameters. In this paper, we reviewed systematically over 34 software module clustering approaches from 2001 to 2020 that had an analytical comparison.

From the review papers, we had a very high rate of relevant publishes from 2001 to 2020. The most published papers belong to IEEE with 12 of published papers. After IEEE, Springer and Elsevier are next publishers with 11 and 5 papers.

The approaches that are chosen have been compared by some factors like accuracy, precision, error rate, recall, F-Score. Also, considering these cases, a comparative analysis was performed on clustering algorithms. The outcomes implied that papers tried to enhance time, accuracy. Also, results illustrated that topics such as user preferences, privacy, and security issues were not considered in many of the reviewed papers. Research on different clustering techniques in software engineering will help to create new clustering approaches to decrease errors and enhance usability along with management. Also, by comparing each software module clustering paper considering their country of origin, papers from India along with China had the most of the contribution in our reviewed papers.

There are also some limitations in the context of this review. First of all, the review has been performed based on some keywords in our research. These keywords include "software module clustering, application clustering, and software cluster algorithm.

Secondly, this study only covers English papers, and Non-English papers are not covered. We also didn't cover chapter books, thesis, and non-index journals in our paper's evaluation. We assume that software computing approaches in computer engineering have been discussed in different languages as well.

For future work of this study, many directions can be considered for future studies. One direction is modularisation-based design for innovative product-related industrial service. Also, a review of the benefits of software modularisation for small and medium-sized enterprises would be a good direction.

## Disclosure statement

No potential conflict of interest was reported by the authors.

## Funding

'Innovation Research on Cross-border E-commerce Shopping Guide Platform Based on Big Data and AI Technology', Funded by Ministry of Education Humanities and Social Sciences Research and Planning Fund (18YJAZH042); Key Research Platform Project of Guangdong Education Department (2017GWTSCX064); The 13th Five-Year Plan Project of Philosophy and Social Science Development in Guangzhou (2018GZGJ208).

## ORCID

Jiahua Li http://orcid.org/0000-0002-0222-0619
Ali Yamini http://orcid.org/0000-0003-0643-4048

## References

Alkhalid, A., M. Alshayeb, and S. A. Mahmoud. 2011. "Software Refactoring at the Package Level Using Clustering Techniques." *IET Software* 5 (3): 274. doi:10.1049/iet-sen.2010.0070.

Amarjeet, and J. K. Chhabra. 2017. "Many-objective Artificial Bee Colony Algorithm for Large-scale Software Module Clustering Problem." *Soft Computing* 22 (19): 6341–6361. doi:10.1007/s00500-017-2687-3.

Amarjeet, and J. K. Chhabra. 2018. "FP-ABC: Fuzzy-Pareto Dominance Driven Artificial Bee Colony Algorithm for Many-objective Software Module Clustering." *Computer Languages, Systems & Structures* 51: 1–21. doi:10.1016/j.cl.2017.08.001.

Anquetil, N., and T. Lethbridge. 2003. "Comparative Study of Clustering Algorithms and Abstract Representations for Software Remodularisation." *IEE Proceedings-Software* 150 (3): 185–201. doi:10.1049/ip-sen:20030581.

Berkhin, P. 2006. "A Survey of Clustering Data Mining Techniques." In *Grouping Multidimensional Data: Recent Advances in Clustering*, edited by J. Kogan, C. Nicholas, and M. Teboulle, 25–71. Berlin, Heidelberg: Springer Berlin Heidelberg.

Bishnoi, M., and P. Singh 2016. "Modularizing Software Systems Using PSO Optimized Hierarchical Clustering." Paper presented at the 2016 International Conference on Computational Techniques in Information and Communication Technologies (ICCTICT), United states.

Campello, R. J., D. Moulavi, and J. Sander 2013. "Density-based Clustering Based on Hierarchical Density Estimates." Paper presented at the Pacific-Asia conference on knowledge discovery and data mining, Australia.

Chen, H., Q. Zhang, J. Luo, Y. Xu, and X. Zhang. 2020. "An Enhanced Bacterial Foraging Optimization and Its Application for Training Kernel Extreme Learning Machine." *Applied Soft Computing* 86: 105884. doi:10.1016/j.asoc.2019.105884.

Czekster, R. M., T. Webber, A. H. Jandrey, and C. A. M. Marcon. 2019. "Selection of Enterprise Resource Planning Software Using Analytic Hierarchy Process." *Enterprise Information Systems* 13 (6): 895–915. doi:10.1080/17517575.2019.1606285.

Huang, J., and J. Liu. 2016. "A Similarity-based Modularization Quality Measure for Software Module Clustering Problems." *Information Sciences* 342: 96–110. doi:10.1016/j.ins.2016.01.030.

Huang, J., J. Liu, and X. Yao. 2016. "A Multi-agent Evolutionary Algorithm for Software Module Clustering Problems." *Soft Computing* 21 (12): 3415–3428. doi:10.1007/s00500-015-2018-5.

Imran, A. 2019. "Design Smell Detection and Analysis for Open Source Java Software." Paper presented at the 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), 29 Sept.-4 Oct, USA.

Kamis, N. H., F. Chiclana, and J. Levesley. 2018. "Geo-uninorm Consistency Control Module for Preference Similarity Network Hierarchical Clustering Based Consensus Model." *Knowledge-Based Systems* 162: 103–114. doi:10.1016/j.knosys.2018.05.039.

Kargar, M., A. Isazadeh, and H. Izadkhah. 2019. "Multi-programming Language Software Systems Modularization." *Computers & Electrical Engineering* 80: 106500. doi:10.1016/j.compeleceng.2019.106500.

Kargar, M., A. Isazadeh, and H. Izadkhah. 2020. "Improving the Modularization Quality of Heterogeneous Multi-programming Software Systems by Unifying Structural and Semantic Concepts." *The Journal of Supercomputing* 76 (1): 87–121. doi:10.1007/s11227-019-02995-3.

Kataev, M., L. Bulysheva, L. Xu, Y. Ekhlakov, N. Permyakova, and V. Jovanovic. 2020. "Fuzzy Model Estimation of the Risk Factors Impact on the Target of Promotion of the Software Product." In *Enterprise Information Systems*, 1–15. Taylor and Francis.

Kitchenham, B., O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman. 2009. "Systematic Literature Reviews in Software Engineering–a Systematic Literature Review." *Information and Software Technology* 51 (1): 7–15. doi:10.1016/j.infsof.2008.09.009.

Köhler, V., M. Fampa, and O. Araújo. 2012. "Mixed-Integer Linear Programming Formulations for the Software Clustering Problem." *Computational Optimization and Applications* 55 (1): 113–135. doi:10.1007/s10589-012-9512-9.

Kumari, A. C., and K. Srinivas. 2016. "Hyper-heuristic Approach for Multi-objective Software Module Clustering." *Journal of Systems and Software* 117: 384–401. doi:10.1016/j.jss.2016.04.007.

Kumari, A. C., K. Srinivas, and M. Gupta 2013. "Software Module Clustering Using a Hyper-heuristic Based Multi-objective Genetic Algorithm." Paper presented at the 2013 3rd IEEE International Advance Computing Conference (IACC), India.

Mitchell, B., M. Traverso, and S. Mancoridis 2001. "An Architecture for Distributing the Computation of Software Clustering Algorithms." Paper presented at the Proceedings Working IEEE/IFIP Conference on Software Architecture, Netherlands.

Mitchell, B. S., and S. Mancoridis 2003. "Modeling the Search Landscape of Metaheuristic Software Clustering Algorithms." Paper presented at the Genetic and Evolutionary Computation Conference, USA.

Mitchell, B. S., and S. Mancoridis. 2006. "On the Automatic Modularization of Software Systems Using the Bunch Tool." *IEEE Transactions on Software Engineering* 32 (3): 193–208. doi:10.1109/TSE.2006.31.

Monçores, M. C., A. C. F. Alvim, and M. O. Barros. 2018. "Large Neighborhood Search Applied to the Software Module Clustering Problem." *Computers & Operations Research* 91: 92–111. doi:10.1016/j.cor.2017.10.004.

Mu, L., V. Sugumaran, and F. Wang. 2019. "A Hybrid Genetic Algorithm for Software Architecture Re-Modularization." *Information Systems Frontiers*. doi:10.1007/s10796-019-09906-0.

Muhammad, S., O. Maqbool, and A. Q. Abbasi. 2012. "Evaluating Relationship Categories for Clustering Object-oriented Software Systems." *IET Software* 6 (3). doi:10.1049/iet-sen.2011.0061.

Naseem, R., M. B. M. Deris, and O. Maqbool 2014. "Software Modularization Using Combination of Multiple Clustering." Paper presented at the 17th IEEE International Multi Topic Conference 2014, Pakistan.

Naseem, R., O. Maqbool, and S. Muhammad. 2013. "Cooperative Clustering for Software Modularization." *Journal of Systems and Software* 86 (8): 2045–2062. doi:10.1016/j.jss.2013.03.080.

Parsa, S., and O. Bushehrian 2004. "A Framework to Investigate and Evaluate Genetic Clustering Algorithms for Automatic Modularization of Software Systems." Paper presented at the International conference on computational science, Poland.

Pavithr, R. S., and A. Garg 2011. "ETMQ: A Novel Objective function—Automatic Clustering of Software Intensive Systems." Paper presented at the 2011 Annual IEEE India Conference, India.

Praditwong, K., M. Harman, and X. Yao. 2011. "Software Module Clustering as a Multi-Objective Search Problem." *IEEE Transactions on Software Engineering* 37 (2): 264–282. doi:10.1109/tse.2010.26.

Prajapati, A., and J. K. Chhabra. 2017. "A Particle Swarm Optimization-Based Heuristic for Software Module Clustering Problem." *Arabian Journal for Science and Engineering* 43 (12): 7083–7094. doi:10.1007/s13369-017-2989-x.

Preheim, S. P., A. R. Perrotta, A. M. Martin-Platero, A. Gupta, and E. J. Alm. 2013. "Distribution-based Clustering: Using Ecology to Refine the Operational Taxonomic Unit." *Applied and Environmental Microbiology* 79 (21): 6593–6603. doi:10.1128/AEM.00342-13.

Ramanathan, L., G. Parthasarathy, K. Vijayakumar, L. Lakshmanan, and S. Ramani. 2018. "Cluster-based Distributed Architecture for Prediction of Student's Performance in Higher Education." *Cluster Computing*. doi:10.1007/s10586-017-1624-7.

Rathore, S. S., and S. Kumar. 2019. "A Study on Software Fault Prediction Techniques." *Artificial Intelligence Review* 51 (2): 255–327. doi:10.1007/s10462-017-9563-5.

Rodriguez, M., M. Piattini, and C. Ebert. 2019. "Software Verification and Validation Technologies and Tools." *IEEE Software* 36 (2): 13–24. doi:10.1109/MS.2018.2883354.

Sadat Jalali, N., H. Izadkhah, and S. Lotfi. 2019. "Multi-objective Search-based Software Modularization: Structural and Non-structural Features." *Soft Computing* 23 (21): 11141–11165. doi:10.1007/s00500-018-3666-z.

Sasidharan, S. 2019. "Reconceptualizing Knowledge Networks for Enterprise Systems Implementation: Incorporating Domain Expertise of Knowledge Sources and Knowledge Flow Intensity." *Information & Management* 56 (3): 364–376. doi:10.1016/j.im.2018.07.010.

Seo, Y.-S., and J.-H. Huh. 2019. "GUI-based Software Modularization through Module Clustering in Edge Computing Based IoT Environments." *Journal of Ambient Intelligence and Humanized Computing*. doi:10.1007/s12652-019-01455-3.

Shen, L., H. Chen, Z. Yu, W. Kang, B. Zhang, H. Li, D. Liu, and D. Liu. 2016. "Evolving Support Vector Machines Using Fruit Fly Optimization for Medical Data Classification." *Knowledge-Based Systems* 96: 61–75. doi:10.1016/j.knosys.2016.01.002.

Siddique, F., and O. Maqbool. 2012. "Enhancing Comprehensibility of Software Clustering Results." *IET Software* 6 (4): 283. doi:10.1049/iet-sen.2012.0027.

Solorio-Fernández, S., J. A. Carrasco-Ochoa, and J. F. Martínez-Trinidad. 2019. "A Review of Unsupervised Feature Selection Methods." *Artificial Intelligence Review*. doi:10.1007/s10462-019-09682-y.

Souri, A., A. Hussien, M. Hoseyninezhad, and M. Norouzi. 2019. "A Systematic Review of IoT Communication Strategies for an Efficient Smart Environment." In *Transactions on Emerging Telecommunications Technologies*, e3736. Wiley.

Souri, A., A. S. Mohammed, M. Y. Potrus, M. H. Malik, F. Safara, and M. Hosseinzadeh. 2020. "Formal Verification of a Hybrid Machine Learning-based Fault Prediction Model in Internet of Things Applications." *IEEE Access* 8: 23863–23874. doi:10.1109/ACCESS.2020.2967629.

Souri, A., N. J. Navimipour, and A. M. Rahmani. 2018. "Formal Verification Approaches and Standards in the Cloud Computing: A Comprehensive and Systematic Review." *Computer Standards & Interfaces* 58: 1–22. doi:10.1016/j.csi.2017.11.007.

Souri, A., A. M. Rahmani, N. J. Navimipour, and R. Rezaei. 2019a. "Formal Modeling and Verification of a Service Composition Approach in the Social Customer Relationship Management System." *Information Technology & People* 32: 1591–1607. doi:10.1108/ITP-02-2018-0109.

Souri, A., A. M. Rahmani, N. J. Navimipour, and R. Rezaei. 2019b. "A Symbolic Model Checking Approach in Formal Verification of Distributed Systems." *Human-centric Computing and Information Sciences* 9 (1): 4. doi:10.1186/s13673-019-0165-x.

Spijkman, T., S. Brinkkemper, F. Dalpiaz, A.-F. Hemmer, and R. van de Bospoort 2019. "Specification of Requirements and Software Architecture for the Customisation of Enterprise Software: A Multi-case Study Based on the RE4SA Model." Paper presented at the 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), South Korea.

Sun, J., and B. Ling. 2018. "Software Module Clustering Algorithm Using Probability Selection." *Wuhan University Journal of Natural Sciences* 23 (2): 93–102. doi:10.1007/s11859-018-1299-9.

Tabrizi, A. H. F., and H. Izadkhah 2019. "Software Modularization by Combining Genetic and Hierarchical Algorithms." Paper presented at the 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI), 28 Feb.-1 March, Iran.

Varghese, R., B. G. K. Raimond, and J. Lovesum. 2019. "A Novel Approach for Automatic Remodularization of Software Systems Using Extended Ant Colony Optimization Algorithm." *Information and Software Technology* 114: 107–120. doi:10.1016/j.infsof.2019.06.002.

Venkatesan, D., and S. Sridhar. 2019. "A Rationale for the Choice of Enterprise Architecture Method and Software Technology in A Software Driven Enterprise." *International Journal of Business Information Systems* 32 (3): 272–311. doi:10.1504/IJBIS.2019.103080.

Wan, Y., and C. Wu 2009. "Optimal Reliability Allocation for Modular Software Systems Basis on Support Vector Clustering and Fuzzy Decision." Paper presented at the 2009 International Conference on Artificial Intelligence and Computational Intelligence, China.

Wang, M., and H. Chen. 2020. "Chaotic Multi-swarm Whale Optimizer Boosted Support Vector Machine for Medical Diagnosis." *Applied Soft Computing* 88: 105946. doi:10.1016/j.asoc.2019.105946.

Wang, M., H. Chen, B. Yang, X. Zhao, L. Hu, Z. Cai, C. Tong, and C. Tong. 2017. "Toward an Optimal Kernel Extreme Learning Machine Using a Chaotic Moth-flame Optimization Strategy with Applications in Medical Diagnoses." *Neurocomputing* 267: 69–84. doi:10.1016/j.neucom.2017.04.060.

Wieland, S. C., J. S. Brownstein, B. Berger, and K. D. Mandl. 2007. "Density-equalizing Euclidean Minimum Spanning Trees for the Detection of All Disease Cluster Shapes." *Proceedings of the National Academy of Sciences* 104 (22): 9404–9409. doi:10.1073/pnas.0609457104.

Xie, T., S. Thummalapenta, D. Lo, and C. Liu. 2009. "Data Mining for Software Engineering." *Computer* 42 (8): 55–62. doi:10.1109/MC.2009.256.

Xu, X., and H.-L. Chen. 2014. "Adaptive Computational Chemotaxis Based on Field in Bacterial Foraging Optimization." *Soft Computing* 18 (4): 797–807. doi:10.1007/s00500-013-1089-4.

Xu, Y., H. Chen, J. Luo, Q. Zhang, S. Jiao, and X. Zhang. 2019. "Enhanced Moth-flame Optimizer with Mutation Strategy for Global Optimization." *Information Sciences* 492: 181–203. doi:10.1016/j.ins.2019.04.022.

Zamli, K. Z., F. Din, N. Ramli, and B. S. Ahmed. 2019. Software Module Clustering Based on the Fuzzy Adaptive Teaching Learning Based Optimization Algorithm Intelligent and Interactive Computing. In *2nd International Conference on Intelligent and Interactive Computing 2018 (IIC 2018)*, Aug 8, 2018 - Aug 9, 2018, Melaka, Malaysia, 167–177.

Zhang, F., H. Ding, and N. Zhang. 2020. "Productive Service Demands Modularization for CNC Machine Tools Based on the Improved AP Clustering Algorithm." *Neural Computing & Applications* 32 (6): 1567–1579. doi:10.1007/s00521-019-04173-1.

Zhao, X., D. Li, B. Yang, C. Ma, Y. Zhu, and H. Chen. 2014. "Feature Selection Based on Improved Ant Colony Optimization for Online Detection of Foreign Fiber in Cotton." *Applied Soft Computing* 24: 585–596. doi:10.1016/j.asoc.2014.07.024.

Zhao, X., X. Zhang, Z. Cai, X. Tian, X. Wang, Y. Huang, L. Hu, and L. Hu. 2019. "Chaos Enhanced Grey Wolf Optimization Wrapped ELM for Diagnosis of Paraquat-poisoned Patients." *Computational Biology and Chemistry* 78: 481–490. doi:10.1016/j.compbiolchem.2018.11.017.

Zhao, X., and Y. Zou. 2011. "A Business Process-driven Approach for Generating Software Modules." *Software: Practice & Experience*. doi:10.1002/spe.1068.