IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 11, 2019, accepted May 20, 2019, date of publication June 3, 2019, date of current version June 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2920427

Software Architecture Module-View Recovery Using Cluster Ensembles

CHOONGKI CHO[1], KI-SEONG LEE[2], MINSOO LEE[1], AND CHAN-GUN LEE[1]
[1]Department of Computer Science and Engineering, Chung-Ang University, Seoul 156-756, South Korea
[2]Da Vinci College of General Education, Chung-Ang University, Seoul 156-756, South Korea

Corresponding author: Chan-Gun Lee (cglee@cau.ac.kr)

This work was supported in part by the Chung-Ang University Excellent Student Scholarship, in part by the National Research Foundation of Korea (NRF) under Grant NRF-2017R1E1A1A01075803, and in part by the Korea Atomic Energy Research Institute (Development and Operation of ICT-based Nuclear Energy Safety Validation System) funded by the Ministry of Science and ICT under Grant 524320-18.

ABSTRACT Software architecture documents are valuable assets supporting the maintenance process for software systems. Unfortunately, in many projects, software architecture documentation is not conducted properly or the documents become obsolete due to a discrepancy with the current architecture. To address this, various automated methods to recover software architecture have been proposed in the literature. We argue that most previous studies have not considered cluster ensembles but relied on a single clustering algorithm. In this paper, we propose to take advantage of cluster ensembles for software architecture recovery. Our experiments on five open-source projects are reported and the results are analyzed.

INDEX TERMS Software architecture recovery, module-view, cluster ensembles.

I. INTRODUCTION

The scale and complexity of modern software systems have grown dramatically. The cost of maintaining software systems has thus been increasing. Especially when it comes to large software systems, the complexity of maintenance becomes enormous and requires architectural knowledge.

Software architecture documents are valuable assets that support effective maintenance work. The documents provide software engineers with several architectural views of a software system from which important design decisions and their rationales can be understood.

Among the information provided by architecture documents, one of the most critical pieces of information for software engineers is the architecture module-view [4], which explains the set of modules and their high-level structure.

Unfortunately, software architecture documentation is rarely conducted properly and the documents often become obsolete due to a discrepancy with the current architecture. It is known that manual architecture recovery is labor-intensive and requires enormous effort even for experienced recovery engineers. To address this problem, there have been active studies on automated software architecture recovery [1], [16], [17], [19]–[21], [31], [44]–[47]. Many of these have primarily focused on recovery of the architecture module-view.

Most automated architecture recovery techniques rely on clustering algorithms to decompose a software system into meaningful clusters [27]. So far, a number of clustering algorithms including search-based clustering algorithms, hierarchical clustering algorithms, and graph-based clustering algorithms have been investigated in various studies on architecture recovery [1], [6], [16], [20], [21], [31]. Each clustering algorithm has unique characteristics and exhibits different performance depending on the given data set; hence, it is not a good strategy to rely on a single clustering algorithm and expect quality results for a wide spectrum of software projects.

In fields of study such as pattern recognition and data mining, many researchers began to study cluster ensembles combining several base clusterings into a consensus clustering [7], [8], [25], [28], [29] in the early 2000s. These studies were triggered by the success of an ensemble classifier that combines the results of several classifiers. They showed that cluster ensembles can contribute to consistently better performance compared to methods relying on a single clustering algorithm. Unfortunately, most previous studies on software architecture recovery have not explored using cluster ensembles yet [43].

In this paper, we propose to take advantage of cluster ensembles for architecture recovery to complement the

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Zhang.

weaknesses of single clustering techniques and achieve better recovery accuracy. The purpose of this work is not only to study the effects of cluster ensembles for software architecture recovery but also to provide guidance for the application of the proposed recovery process. We clarify the goals of our work through the following research questions:

RQ1 - Do cluster ensemble-based recovery methods perform better than methods using a single clustering algorithm?

RQ2 – Which consensus method performs well for software architecture recovery?

RQ3 – How does the generation method for base clusterings affect the recovery performance of cluster ensemble-based recovery methods?

RQ4 - How does the number of base clusterings affect the recovery performance of cluster ensemble-based recovery methods?

The main contributions of this paper are the following:

- We propose to utilize cluster ensembles for software architecture recovery and show how to apply this technique to existing approaches.
- We perform systematic experiments to answer the above research questions and present the experimental results and analysis.

The remainder of this paper is composed as follows. We discuss the background and previous studies related to our approach in Section 2. In Section 3, we outline the process of architecture recovery using cluster ensembles. Sections 4 describes our case study and Section 5 presents the experimental results and analysis. Section 6 describes threats to validity, which is followed by a summary of the paper in Section 7.

II. BACKGROUND AND RELATED WORK

A. SOFTWARE ARCHITECTURE RECOVERY USING SOFTWARE CLUSTERING

Clustering is a technique for identifying clusters composed of entities with similar characteristics in a given data set. In order to recover the module-view of a software architecture, software clustering decomposes a large software system into a set of smaller subsystems to help the software engineers manage the system effectively. Architecture recovery is generally performed through the following steps [27]:

- Fact extraction
- Similarity computation
- Application of software clustering
- Evaluation

The details of each step are discussed below. In the remainder of the paper, we shall use the following terminologies.

Cluster: A bundle containing several entities.

Clustering: Grouping data entities into clusters or decomposing a given data set into clusters. Note that the term clustering may also be used to indicate the results produced by a clustering algorithm. When used in this sense, it is understood as a clustering result, a partition or a decomposition. They can easily be distinguished by the context.

Cluster ensemble: An approach combining different clustering results into a single consolidated result. Generally, cluster ensembles consist of a generation step producing different clustering results and a consensus step combining the generated clustering results into a single output.

Base clustering: The generation phase of a cluster ensemble produces a number of clustering results, which are referred to as base clusterings.

1) FACT EXTRACTION

Prior to applying software clustering, we should define software entities to be clustered. In the context of software architecture recovery, typical software entities include packages, files, classes, and functions [27]. The appropriate entities depend on the purpose of the recovery. In case a detailed level is needed, a function can be selected as a software entity. If a higher level view is needed, a class or a package can be chosen.

After determining the entities, we should identify their features. A feature is an attribute of a software entity and contains information about relationships with other entities [9], [19], [27]. The selected features are utilized for setting a criterion or similarity for clustering.

The relationships among software entities can be classified into three categories. The first is structural relationships such as association, dependency, composition, and generalization between classes. The second is semantic relationships retrieved from text information such as identifier names and comments in the source code [17]. The last is change couplings, which signify that two or more software entities have been changed together frequently during the evolution of the software [5]. Structural relationships can be extracted through static analysis of the source or binary code and dynamic analysis of the program. Semantic relationships can be extracted through natural language processing such as topic analysis. Change couplings can be collected from a software configuration management system. These extracted software entities and features are referred to as facts and are given to clustering algorithms.

2) SIMILARITY COMPUTATION

Software clustering algorithms calculate the similarities between entities using its features and put the most similar entities into the same cluster. The features can be categorized into binary features, categorical features, and numerical features according to the range or type of values that the features can take. Binary features can take only 0 or 1 values, while categorical features allow a small number of discrete values. Numerical features can have real numbers. In calculating the similarities between entities based on binary features, binary similarity coefficients such as the Jacquard coefficient are used. For numerical features, numerical resemblance coefficients such as Euclidian and Manhattan distance are used. The categorical feature is often represented by a set of binary features. Therefore, those similarities can be easily computed by the binary similarity coefficients.

3) APPLICATION OF SOFTWARE CLUSTERING

After completing the above processes, we can apply software clustering to achieve a module view of the software architecture. A variety of software clustering algorithms have been proposed in the literature. Tzerpos et al. proposed an Algorithm for Comprehension-Driven Clustering (ACDC) that recovers architectures based on subsystem patterns [31]. Mitchell et al. proposed Bunch, which performs architecture recovery using a search-based clustering algorithm [21]. Maqbool et al. proposed the Weighted Combine Algorithm (WCA), which uses an Agglomerative Hierarchical Clustering Algorithm (AHCA). Based on these pioneering studies, several architecture recovery techniques using clustering algorithms have been proposed [16], [17], [19], [44]–[47].

4) EVALUATION

Various evaluation methods have been proposed in the literature. Generally, there are two methods for the evaluation of architecture recovery methods: external and internal validation [27]. External validation requires a ground truth because the architectural decomposition obtained from an automated recovery method is compared to it. The ground truth is manually constructed by software experts in the subject or extracted from architecture documents. In internal validation, we evaluate an architecture recovery method merely on the basis of the properties of its output, such as the stability of the recovered architectures in a series of evolutions of the software system and the extremity of cluster distribution [41].

B. CLUSTER ENSEMBLES

The basic idea of cluster ensembles is to combine different clusterings to create a final result with better quality. In general, a cluster ensemble technique is performed in two phases. The first phase is to generate different clustering results for a given data set and the second is to consolidate the generated clustering results into a single result (see Fig. 1).

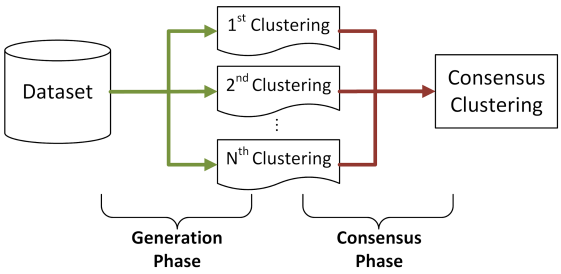


FIGURE 1. The general process of the cluster ensemble technique consists of two phases: Generation and consensus.

1) GENERATION PHASE

In this stage, different clustering results are generated from a given data set. These multiple clustering results are referred to as base clusterings.

It is widely accepted that the supply of diverse base clusterings is critical to the quality of an ensemble [32]. If there is a limited number of unique base clusterings, even if the best consensus method is applied, the ensemble method is doomed to produce an output that is similar to those of homogeneous base clusterings.

There are two major approaches to generate diverse base clusterings. The first is to use different clustering algorithms. The other approach is to run a clustering algorithm with different parameter settings several times.

2) CONSENSUS PHASE

Several consensus methods have been proposed in the literature. They can be categorized into two major approaches: objects co-occurrence and median partition [32].

The idea of the objects co-occurrence approach is that the entities frequently being assigned to the same clusters in the base clusterings should belong to the same cluster in the single consolidated result. We can define new similarities between entities through the frequency of entity co-occurrence. A clustering algorithm is then performed based on the similarities between entities to obtain a single consolidated result.

The median partition approach treats the consensus process as an optimization problem of finding a median partition. The median partition is a clustering result maximizing the sum of the similarities between the clustering result and all the base clusterings.

C. ARCHITECTURE RECOVERY APPROACHES BASED ON CLUSTER ENSEMBLES

To the best of our knowledge, previous studies have not exploited cluster ensembles for architecture recovery except in the following two approaches.

Naseem et al. proposed the Cooperative Clustering Technique (CCT) for architecture recovery [23]. They utilized two similarity measures during the process of the Agglomerative Hierarchical Clustering Algorithm (AHCA). For a typical AHCA, only one similarity measure, such as the Jaccard measure, is employed. However, Naseem et al. identified feature vector cases in which the Jaccard measure alone did not suffice, and proposed the Jaccard-NM to solve the problem. Depending on the feature vector cases, they selectively applied either of the measures on each iteration of the AHCA. The same authors later proposed a more advanced version of the CCT in their subsequent work [43]. Note that their approaches do not use cluster ensembles to consolidate the results of multiple clustering results into a single result, unlike our approach, but use multiple similarities or distance measures within a step of the clustering process.

Ibrahim et al. proposed Cooperative Clustering based on Graphs (CC/G) [12]. CC/G is based on Cooperative Clustering (CC) proposed by Kashef and Kamel [15], which is one of the cluster ensemble algorithms based on the object co-occurrence approach. We argue that their study proved the feasibility of a cluster ensemble approach, however, their experimental results failed to show the benefit of the approach.

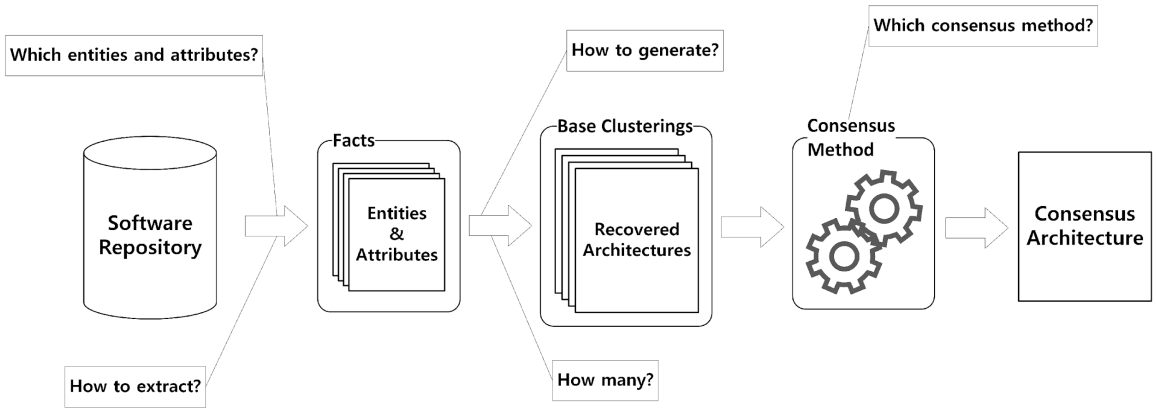


FIGURE 2. An overview of the cluster ensemble-based recovery process and questions that should be answered in the process.

Our approach is motivated by Ibrahim *et al.* [12], and we extend their study by considering combinations of representative architecture recovery techniques and several cluster ensemble algorithms. Most importantly, our study explores a large spectrum of options in cluster ensembles and presents extensive experimental results providing useful guidance on the application of cluster ensembles to software architecture recovery. In addition, the previous approach [12] evaluated CC/G through a non-normalized measure, MoJo distance [30], and treated the package structures of subject projects as ground-truth decompositions. In contrast, we employ MoJoFM [33], an improved version of MoJo distance, and treat the module decompositions for open source projects [9], [19] reconstructed by software experts as ground-truth module-views.

III. ARCHITECTURE RECOVERY METHOD

Fig. 2 shows the proposed architecture recovery process. First, we extract software entities and features, also known as facts, from the software repository. The extracted facts are then utilized to create several base clusterings, each of which can also be considered as a recovered module-view. Applying an appropriate consensus technique to the base clusterings results in a single consensus module-view.

At first glance, it might seem trivial to apply cluster ensembles to architecture recovery. However, it should be noted that there are several points that must be considered in applying cluster ensembles.

(1) Which software entities and features are used?
(2) What is the best way to extract software entities and features?
(3) What is the best way to create multiple recovery architectures for base clusterings?
(4) How many base clusterings are generated?
(5) Which consensus method should be used?

Among the above five questions, the points (3), (4), and (5) should be considered only when the cluster ensemble is applied, while the points (1) and (2) are common to any architecture recovery approach. To answer the questions listed above and the research questions stated in Section 1, we conduct a case study on five open source software projects.

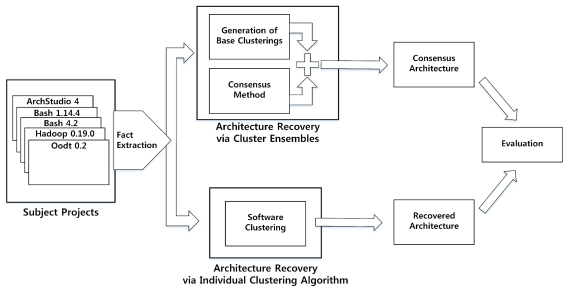


FIGURE 3. An overview of our case study.

IV. CASE STUDY DESIGN

Fig. 3 shows an overview of the case study performed in this paper. First, we extract facts from the five target projects. The extracted facts are given to two different recovery approaches. The first is the cluster ensemble-based recovery proposed in this study and the other is a recovery method based on a single clustering algorithm. Based on the given facts, software architecture module views are recovered by each method. Finally, we evaluate the results of each technique. The experiment for the case study was performed on a computer with Intel (R) Core (TM) i7-4790 3.60 GHz CPU and 16 GB RAM running on Ubuntu 14.04 LTS.

A. SUBJECT PROJECT

Table 1 shows a summary of the subject projects. These software systems are open source projects with publicly available architecture information that has been manually recovered by experts. We treat the architecture information as the ground-truth decompositions [9], [19]. The column #Clusters in the table indicates the number of clusters in each ground-truth decomposition.

B. FACT EXTRACTION

In this study, we treat source code files as software entities for clustering, which is typical in software architecture recovery. This is also due to that the available ground-truth

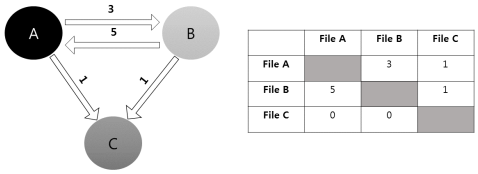
TABLE 1. A summary of subject software projects.

Name	Version	Description	Language	#Files	SLOC	#Clusters
ArchStudio [36]	4	IDE for Architecture Development	Java	592	55K	57
Bash [37]	1.14.4	Unix Shell	C	129	70K	25
Bash [37]	4.2	Unix Shell	C	321	115K	14
OODT [35]	0.2	Data Management	Java	1008	180K	217
Hadoop [34]	0.19.0	Distributed File System	Java	597	87K	67

decompositions in Table 1 consist of clusters of the source code files. Note that our method can be easily applied to other software entities such as packages and classes.

As introduced in Section 2, there are various features that can be used for software clustering. We reiterate that the purpose of this study is not to find the most effective features but to apply cluster ensembles to architecture recovery and validate its effectiveness. Therefore, the experiments are conducted using only the most common features, which are the structural relationships among source code files.

We extract the structural relationships by using Understand [40], a commercial static analysis tool supporting various programming languages such as Java and C++. The tool provides a functionality of analyzing a set of source code and exporting their structural relationships into a CSV file. Each line consists of three columns where the file shown in the first column is dependent on the file in the second column and the third column indicates the number of dependencies.



	File A	File B	File C
File A		3	1
File B	5		1
File C	0	0	

FIGURE 4. Two representations of source code files and the structural relationships between them.

We run a small script to process the CSV file and the extracted facts can be expressed in the form of a graph or a matrix, as shown in Fig. 4. The figure shows three files A, B, and C and the structural relationships between them. In the graph on the left-hand side, nodes and edges represent the files and the structural relationships between the files, respectively. An edge from a source node to a target node means that the source refers to the contents of the target. The weight of the edge indicates the number of references.

C. ARCHITECTURE RECOVERY VIA CLUSTER ENSEMBLES

This section explains the two major steps of the process of recovering a module-view of a software architecture by applying cluster ensembles.

1) GENERATION OF BASE CLUSTERINGS

In this step, a variety of decompositions are generated, and these are used as base clusterings. As described in Section 2.2.1, there are two approaches to generate diverse base clusterings; (1) using a single clustering algorithm with different parameter settings and (2) adopting a number of different clustering algorithms. In our study, we try both generation approaches to investigate how they affect the performance of the architecture recovery method.

Specifically, we use four software clustering techniques to produce base clusterings, Bunch [21], the Weighted Combined Algorithm (WCA) [20], K-means [11], and Blondel's algorithm [3]. These have been widely used in the literature and show decent performance in many studies [3], [9], [11], [19]–[21]. We have selected this set of techniques because each technique has a unique perspective on clustering.

Another consideration is the number of base clusterings we are going to generate. We conducted a preliminary study to obtain a rough guideline on the number of base clusterings. We generated the base clusterings using Bunch with different parameter settings for each execution and adopted Evidence Accumulation [8] for the consensus technique. We combined a set of the base clusterings into a consolidated result while incrementally increasing the size of the set from 1 to 100. It was observed that the quality of the consolidated results, measured by MoJoFM, gradually increased until the size of the base clustering set was less than 20. However, the quality did not change much after the size exceeded 20. While analyzing the results, we noticed that if the number of base clusterings was larger than 20, then the quality of the results became stable; hence we decided to use 50 base clusterings in the experiments, which seems to be a large enough number for the base clusterings.

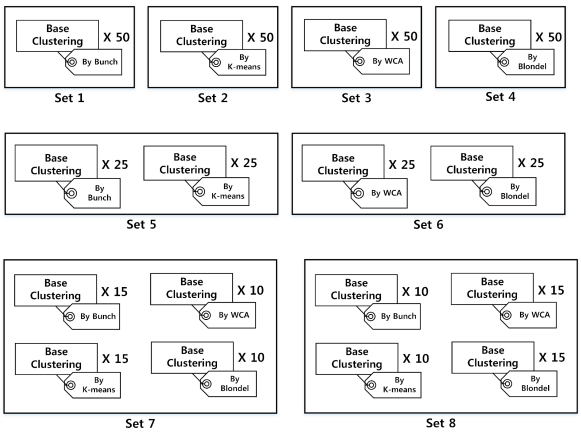


FIGURE 5. Sets of base clusterings used in our case study.

Fig. 5 shows different generation strategies for base clustering, which will be the input to the cluster ensembles. Each set 1, 2, 3, and 4 refers to a set of clustering results generated by a single clustering algorithm, Bunch, K-means, WCA, and Blondel, respectively, while changing its clustering parameters. Sets 5 and 6 include the clustering results generated by two different clustering algorithms. Sets 7 and 8 use four algorithms. We apply the consensus technique to each of these eight base clustering sets to obtain the final module-view.

The following is a brief introduction to the four software clustering algorithms. We elaborate on how to use them to generate a number of different base clusterings.

- **Bunch**

Bunch is a widely used tool that provides a set of search-based clustering algorithms for software architecture recovery. These algorithms seek a solution maximizing an objective function called Modularity Quality (MQ) [21], which calculates the sum of the modularity of all clusters. The modularity of each cluster is computed by considering both cohesion and coupling properties. Bunch includes exhaustive search, the genetic algorithm (GA), and the hill-climbing algorithm (HCA).

In this study, we select the HCA for the generation of base clusterings [26]. Bunch can generate different results from the same input due to the nature of search-based algorithms. However, we still modify the parameters of the HCA with every execution in order to generate more diverse base clustering results. The parameter values range from 0 to 100. The behavior of the HCA becomes more like that of Nearest Ascent Hill Climbing (NAHC) or Steepest Ascent Hill Climbing (SAHC) if the parameter is set to close to zero or one hundred. We generate 50 decompositions as base clusterings by changing the parameters from 1 to 100. The experiment is conducted using the implementation available online [21].

- **Weighted Combined Algorithm (WCA)**

The WCA [20] is a special form of AHCA. The difference between the WCA and the standard AHCA is the updating rule defining a method of calculating the similarity between sub-clusters (or between sub-clusters and entities). The WCA provides a set of measures for the computation of similarity between entities, e.g., the Jaccard coefficient and the unbiased Ellenberg measure for binary features and non-binary features. On each iteration, the WCA creates a new cluster and its feature vector by combining feature vectors of entities which belong to the new cluster.

In this study, the value of the feature is set to the number of structural relationships between the source code files; hence, we choose the unbiased Ellenberg measure to calculate similarity. Unlike Bunch, the WCA is deterministic; hence, we try different parameters including the number of clusters for each execution to generate various results.

In the case of ArchStudio, OODT, and Hadoop projects, the value of the parameter is increased or decreased by five based on the number k of clusters in the ground-truth structure, and a total of 50 values are used. The projects Bash 1.14.4 and Bash 4.2 are relatively small in the number of entities. Especially the number of ground-truth clusters is smaller than 25 in each project. Therefore, the number of clusters is increased or decreased not by five but by one and a total of 50 values are used for the both projects.

We implemented the WCA in Python by utilizing the implementation of the AHCA available online.

- **K-means algorithm**

K-means [11] is one of the most well-known clustering algorithms. Given user-defined parameter K, it produces a decomposition consisting of K clusters. At the initial step, K-means randomly selects K points in the search space. Then each entity chooses the nearest among the K points and joins a cluster based on the chosen point. In this way, the initial K clusters are prepared. After this, each cluster recalculates its central point of the entities included in the cluster. Then, each entity again chooses the nearest central point and rejoins a cluster formed by the central point. The above steps are repeated until convergence. K-means computes the distance between data points in the search space, and we use the Jaccard distance to calculate the distance between the software objects.

Similar to the search-based clustering algorithm, K-means can produce different clusterings with each iteration because it selects the K points randomly at the initial stage. Due to this non-deterministic property, many studies on cluster ensembles have employed K-means for generating base clusterings [7], [8], [15], [25], [28]. In addition, we set different numbers of clusters K for every iteration to produce diverse base clusterings.

- **Blondel's algorithm**

Community detection methods for the analysis of large networks have been utilized for architecture recovery in several studies [6], [16]. We thus consider the community detection algorithm proposed by Blondel. This algorithm detects communities by optimizing Newman's Modularity [24] in a large network. The method is well known for its scalability as well as performance [3]. In our work, we run Gephi [2], which includes an implementation of Blondel's algorithm. The tool exposes a parameter for the resolution of clusters [18] that is adjusted differently at every iteration to produce diverse base clusterings. We generate a total of 50 base clusterings by increasing the cluster resolution parameter from 0.1 to 1 by steps of 0.05 and from 1 to 32 by steps of 1.

2) CONSENSUS METHOD

In this step, we apply a consensus technique to the multiple base clusterings and consolidate them into a single module-view. As described in Section 2.2.2, several consensus techniques have been proposed [10], [32]. In this study, we perform architecture recovery using five representative consensus techniques based on the objects co-occurrence approach [32]. The underlying concept of these algorithms is that entities frequently belonging to the same cluster in the base clusterings must be tied to the same cluster. The five consensus techniques selected in this study were found to be very intuitive and perform well [7], [8], [25], [28]. We provide details of the selected consensus methods in the following.

- **Cluster-based Similarity Partitioning Algorithm (CSPA)**

The number of co-occurrences between two entities appearing in all base clusterings can be interpreted as the

similarity between them. Such an approach is proposed in the CSPA [28]. The algorithm treats the data set as a graph in which nodes represent entities and the weights of the edges are set to the aforementioned similarity. Then, a graph partitioning algorithm is applied to the graph, and it produces a consensus result. The CSPA uses METIS [13] for graph partitioning.

- **HyperGraph-Partitioning Algorithm (HGPA)**

In the HGPA [28], the summarization of base clusterings is reduced to a hypergraph partitioning problem. The nodes and the hyperedges of a hypergraph are mapped to the entities of the data set and the clusters in base clusterings, respectively. A hypergraph partitioning algorithm is employed to obtain a final clustering result. We select HMETIS [14] for the hypergraph partitioning algorithm as done in study [28].

- **Meta-Clustering Algorithm (MCLA)**

The MCLA [28] employs graph formulation as well. However, a distinct feature of the algorithm is that the target to be clustered is not entities but clusters. In the algorithm, clusters in base clusterings represent nodes of the graph, and weights of edges are set to the similarity measured by the binary Jaccard coefficient between the clusters. After applying a graph partitioning algorithm to the graph such as METIS, we can obtain a clustering result consisting of meta-clusters formed by several clusters from base clusterings. Because the clusters result from multiple base clusterings in the MCLA, a meta-cluster may have several clusters sharing the same entities. It is also possible that an entity belongs to multiple meta-clusters after the partitioning process. Therefore, each entity should be reassigned so that it is included in only one meta-cluster. Note that nodes of the formulated graph in the CSPA and HGPA are entities of data sets, whereas the nodes are clusters from base clusterings in the MCLA.

- **Hybrid Bipartite Graph Formulation (HBGF)**

HBGF [7] takes both entities and clusters as nodes of its graph. In the graph of HBGF, only the edges between an entity and a cluster are allowed; i.e., there is no edge between clusters or between entities. Hence, the formulated graph is bipartite. The authors of HBGF claim that the bipartite approach takes advantage of both the entity-based and the cluster-based graph formulations. Graph partitioning algorithms such as METIS can transform the bipartite graph into a final clustering result.

- **Evidence Accumulation Algorithm (EA)**

The EA [8] is similar to the CSPA in that it uses the number of co-occurrence across base clusterings as the similarity between two entities. However, the EA constructs not a graph but a similarity matrix of which rows and columns correspond to entities. Applying the AHCA to the matrix produces a final clustering result. The algorithm is known for its robustness and stability across various data sets [25].

- **Implementations and Parameters**

MATLAB implementations of CSPA, HGPA, and MCLA algorithms are available on the website of paper [28]. We have implemented each algorithm in the Python language based on an understanding of their MATLAB code. In addition, the HBGF and EA were implemented in Python by referring to studies [7] and [8].

These five consensus techniques require the number of clusters as an input parameter. Depending on the purpose of architecture recovery, you can set the input parameter to a small number to produce a module-view from a macroscopic perspective, or a large number for a microscopic perspective. Hence, it is challenging to determine the number of clusters in our experiment. We determine the number of clusters for consensus techniques by referring to the number of clusters of the ground-truth decomposition. Table 2 shows the range of the number of clusters for the projects; the increment is set to five. The final result is chosen to have the best evaluation scale among the consensus outputs from various experiments.

TABLE 2. The number of clusters for consensus techniques.

Project	ArchStudio	Bash 1.14.4	Bash 4.2	Oodt	Hadoop
# of clusters	7, 12, 17, …, 152	5, 10, 15, …, 150	4, 9, 14, …, 149	152, 157, 162, …, 297	7, 12, 17, …, 152

D. ARCHITECTURE RECOVERY VIA AN INDIVIDUAL CLUSTERING ALGORITHM

We design a control group relying on a single clustering algorithm to compare its performance with the approaches using cluster ensembles. The control group uses only one of the baseline approaches such as Bunch, the WCA, K-means, Blondel's algorithm, or ACDC. We select the best quality recovery results among base clusterings as the control group. ACDC is a popular architecture recovery algorithm exploiting subsystem patterns [31] and its good performance is reported in the literature [9], [19]. Note that we did not utilize ACDC for generating base clustering because it was almost impossible to generate different base clusterings due to the nature of the algorithm.

E. EVALUATION

For recovery evaluation, we perform external validation, an evaluation method that compares the clustering results with the ground-truth decomposition. This method has been used in several recovery studies since it can be evaluated objectively [9], [19], [44]. However, this evaluation method can only be applied if there is a ground-truth architecture. Creating the ground-truth architecture is difficult and costly. Fortunately, several recent studies have proposed processes for building ground-truth decompositions and made the ground-truth decompositions of several open source software projects [9], [19] publicly available.

External validation measures the degree of similarity between the decomposition generated by the automatic recovery technique and the ground-truth decomposition. We use

MoJoFM [33] for similarity measurement, which is widely used in the field of architecture recovery. MoJoFM measures the number of move and merge operations required when one structure is converted to another. The definition of MoJoFM is as follows:

$$MoJoFM(D, G) = \left(1 - \frac{mno(D, G)}{max(mno(\forall D, G))}\right) \times 100\% \quad (1)$$

$mno\,(D, G)$ indicates the minimum number of moving and merging operations required to convert the decomposition D into G. D is the decomposition generated by the automatic recovery technique and G is the ground truth. A higher value is considered better.

V. EXPERIMENTAL RESULTS AND ANALYSIS

This section analyzes the experimental results of our case study while focusing on the research questions

A. RQ1

RQ1 - Do cluster ensemble-based recovery methods perform better than methods using a single clustering algorithm?

The Null Hypothesis Is: Cluster ensemble-based architecture recovery does not show better quality than single clustering algorithm-based recovery.

Table 3 summarizes the results of the case study. The leftmost two columns represent the base clustering set and the consensus methods. In the case study, we perform a total of 40 cluster ensembles by using eight base clustering sets and five consensus techniques as shown in Fig. 5. The evaluation results of each recovery technique for each target project are presented as MoJoFM values. The evaluation results of the five baseline approaches are also shown at the bottom of the table.

In the table, the MoJoFM values for the top five results for each target project are highlighted in gray. In the case of ArchStudio 4, ACDC shows the best performance and ensemble recovery techniques rank second to fifth. For Bash 1.14.4, the ensemble recovery techniques rank first, second, and fourth, while ACDC and Blondel rank third and fifth, respectively. For Bash 4.2, the first to fifth ranks are all ensemble-based recovery techniques, and Hadoop 0.19.0 shows similar results. For OODT 0.2 the ensemble-based recovery techniques rank fifth in all cases except for the WCA, where it ranks fourth. The top five of the averages are also occupied by the ensemble-based recovery techniques. From these results, it can be observed that the ensemble-based recovery technique shows better recovery results than the baseline approach in most cases. For example, when the EA is used as a consensus technique in Set 1, the average MoJoFM value is at least 3% to 10% higher than cases using the baseline techniques only.

Unfortunately, the ensemble-based recovery technique does not always outperform the baseline approach. In ArchStudio 4, ACDC shows better performance than ensemble techniques, and WCA and Blondel show decent performance with respect to MoJoFM.

Based on the experimental results and analyses, we answer RQ1: An appropriate combination of a consensus method and a set of base clusterings can lead to better performance than relying on a single clustering algorithm.

B. RQ2

RQ2 –Which consensus method performs well for software architecture recovery?

In the previous section, we observed that it is important to select and use an appropriate consensus technique to obtain good results when performing ensemble-based recovery. Table 3 shows that the MCLA and EA techniques perform well among the consensus techniques. The case showing the highest average of MoJoFM adopts the EA, and the second and third cases use the MCLA. Note that the cases in which MoJoFM values are within the top five in each project are highlighted in gray, and many use the MCLA and EA. Among the remainders, HBGF shows decent performance, while the CSPA and HGPA show relatively poor performance.

Since our experiments could not consider all the existing consensus techniques, the above analysis cannot be generalized. However, the experiment results may provide helpful hints on selecting consensus techniques in the context of cluster ensembles for architecture recovery.

C. RQ3

RQ3 – *How does the generation method for base clusterings affect the recovery performance of cluster ensemble-based recovery methods?*

Not only the consensus technique but also the base clustering generation method is an important factor in cluster ensemble-based architecture recovery. In this section, we will examine the experimental results to find out which base clustering method can lead to good performance.

For Set 1 in Table 3, the ensemble-based recovery methods show overall better MoJoFM values than the baseline approach using Bunch. Specifically, the EA ensemble shows better recovery quality on average than Bunch. In the OODT 0.2 project, where Bunch performs poorly, EA shows a significant performance improvement of more than 25%. Although there are various degrees of improvement, the ensemble-based methods using Bunch for base clusterings show significant performance improvement. Also, when we use K-means for base clusterings (Set 2), the MCLA and HBGF show higher MojoFM values than K-means in all other projects except Bash 1.14.4.

In contrast to the above cases, when the Blondel algorithm is used as the base clustering generation method (Set 4), most of the ensemble-based techniques show worse performance than Blondel's approach. Even when using the WCA as a base clustering method (Set 3), no ensemble technique outperforms it. Based on the above results, we do not recommend the WCA or the Blondel algorithm for generating base clusterings. We will briefly discuss this phenomenon while pointing out some characteristics of each base clustering method in the following paragraphs.

TABLE 3. Experimental results (rounded to four digits) of our case study on the five subject projects. Eight sets of base clusterings and five consensus methods are used for a total of forty cluster ensemble-based recovery methods.

Used set of base clusterings	Consensus method	ArchStudio 4	Bash 1.14.4	Bash 4.2	OODT 0.2	Hadoop 0.19.0	Average
Set 1: 50 by Bunch	CSPA	64.70	42.86	56.49	47.18	48.57	51.96
	EA	65.95	57.14 (1st)	59.54	54.35	54.29 (3rd)	58.25 (1st)
	HBGF	65.23	48.21	59.16	45.88	48.75	53.45
	HGPA	64.34	46.43	55.34	43.88	46.07	51.21
	MCLA	66.67	50.89 (4th)	59.54	42.94	48.57	53.72
Set 2: 50 by K-means	CSPA	65.23	40.18	60.69 (4th)	47.76	51.43	53.06
	EA	62.54	41.96	53.05	54.24	50.89	52.54
	HBGF	68.46	42.86	59.54	56.71 (1st)	54.82 (1st)	56.48 (5th)
	HGPA	65.59	41.96	55.73	53.06	52.14	53.70
	MCLA	67.74	42.86	61.83 (2nd)	56.35 (2nd)	54.82 (1st)	56.72 (3rd)
Set 3: 50 by WCA	CSPA	62.19	37.50	53.44	51.88	47.68	50.54
	EA	68.82	40.18	58.02	55.41 (4th)	52.32	54.95
	HBGF	65.59	40.18	56.87	53.76	51.61	53.60
	HGPA	67.74	40.18	54.20	52.12	49.82	52.81
	MCLA	67.03	41.07	58.78	55.53 (3rd)	51.43	54.77
Set 4: 50 by Blondel	CSPA	65.23	45.54	57.25	43.06	47.14	51.64
	EA	67.20	54.46 (2nd)	57.25	49.06	51.96	56.00
	HBGF	64.34	44.64	55.73	34.82	43.57	48.62
	HGPA	64.70	46.43	58.78	45.06	46.07	52.21
	MCLA	66.85	49.11	54.58	38.24	47.14	51.18
Set 5: 25 by Bunch 25 by K-means	CSPA	64.87	43.75	59.16	52.59	51.25	54.32
	EA	66.49	48.21	56.49	53.88	52.14	55.44
	HBGF	68.82	43.75	61.45 (3rd)	54.12	54.64 (2nd)	56.56 (4th)
	HGPA	67.38	41.07	59.54	51.88	50.89	54.15
	MCLA	69.71 (5th)	46.43	62.21 (1st)	52.94	54.64 (2nd)	57.19 (2nd)
Set 6: 25 by WCA 25 by Blondel	CSPA	65.59	39.29	54.96	48.24	49.29	51.47
	EA	69.00	39.29	58.02	54.71	52.5	54.7
	HBGF	66.67	41.07	55.34	50.71	50.36	52.83
	HGPA	66.85	45.54	54.58	52.71	50	53.94
	MCLA	70.79 (2nd)	45.54	58.78	55.06	51.79	56.39
Set 7: 15 by Bunch 15 by K-means 10 by WCA 10 by Blondel	CSPA	65.95	40.18	58.4	52.71	50.89	53.63
	EA	65.05	46.43	58.4	53.88	51.96	55.14
	HBGF	67.38	41.96	56.11	54.24	52.32	54.4
	HGPA	66.67	40.18	54.2	52.35	48.21	52.32
	MCLA	69.89 (4th)	44.64	60.69 (4th)	53.88	53.04	56.43
Set 8: 10 by Bunch 10 by K-means 15 by WCA 15 by Blondel	CSPA	66.31	40.18	55.73	49.53	49.46	52.24
	EA	67.38	44.64	55.34	53.88	53.75	55
	HBGF	67.38	41.07	53.82	52.94	50.36	53.11
	HGPA	64.16	40.18	53.44	51.53	49.82	51.83
	MCLA	70.25 (3rd)	41.96	58.02	54.24	51.43	55.18
Bunch	-	64.34	47.32	57.63	28.71	42.32	48.01
K-means	-	63.08	44.64	59.16	51.41	51.25	53.91
WCA	-	68.82	40.18	58.78	55.41 (4th)	52.32	55.10
Blondel	-	67.56	50.00 (5th)	58.02	46.94	51.96	54.90
ACDC	-	74.91 (1st)	51.79 (3rd)	52.67	43.76	40.00	52.63

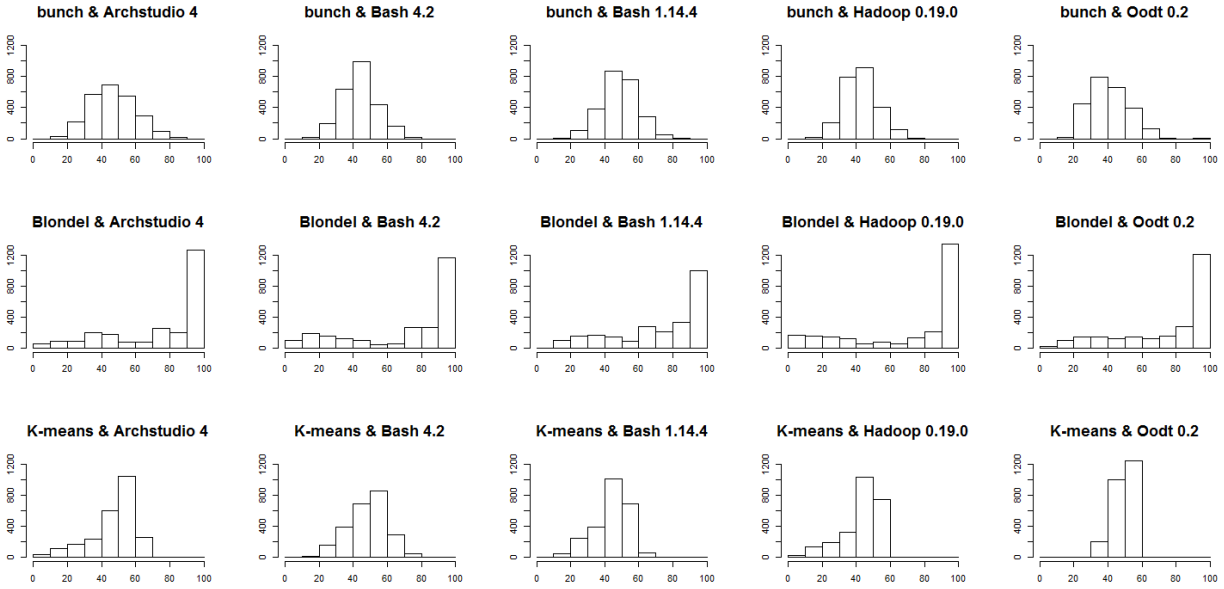


FIGURE 6. The histograms show the distributions of pairwise similarity (MoJoFM) of three sets of base clusterings, Set 1 (by Bunch), Set 2 (by Kmeans), and Set 4 (by Blondel).

First, we discuss why the WCA is not appropriate as a technique for base clustering generation. Table 3 shows that the MoJoFM values of the baseline WCA and EA-based ensemble recovery (Set 3) are almost the same. Their performance is almost the same across all projects except Bash 4.2, where the difference is only 0.76%. This implies that applying the EA consensus technique to the base clustering results generated by the WCA is not fruitful. Note that both the WCA and EA are based on the AHCA whose salient property is to group the closest pair of entities or sub-clusters until all entities are merged in a single cluster. Specifically, a pair of two entities first grouped together by the WCA must be in the same cluster in all the base clusterings by the WCA and the decision is never revoked. Similarly, the EA treats the number of co-occurrences between two entities across all the base clusterings as similarity. Therefore, the pair of objects that are initially grouped in WCA-based clustering is the same in the EA. This pattern continues in the rest of the clustering process. To sum up, the base clusterings by the WCA cannot give a variety of object co-occurrence information to the consensus methods.

As discussed in Section 2.2.1, the diversity of the base clusterings is critical to the performance of cluster ensembles. Hence, we decided to measure the similarity among base clusterings generated by the Blondel algorithm to investigate why cluster ensemble-based recovery methods were inferior when using Blondel for the generation of base clusterings. In Fig. 6, the horizontal axis of the histograms indicates the pairwise MoJoFM score. The vertical axis represents the frequency. The number of base clusterings is 50 per project; hence, the total frequencies in each case is $(50^2) * 2$. The reason for multiplying 50^2, the number of pairwise combinations, by two is that MoJoFM is not symmetric, so there are two MoJoFM values in a pair of structures. Fig. 6 includes the results for Bunch and K-means as well as those for Blondel for the purpose of comparison. In the case of the Blondel algorithm, most pairwise MoJoFM values range from 90 to 100. In the Bunch and K-means cases, on the contrary, much wider distributions are observed. The pairwise comparison results indicate that the Blondel algorithm fails to generate diverse base clusterings; hence, applying the consensus methods to such base clusterings cannot produce quality results.

By summarizing the above discussion, we can answer the following to RQ3: When creating base clusterings, we should use clustering techniques with nondeterministic characteristics such as Bunch or K-means.

Another interesting point observed in Table 3 is that the recovery quality is generally better when the ensemble is performed on Bunch or K-means (Sets 1, 2, and 5) than when the ensemble is applied to various combinations of algorithms (Sets 7 and‘8). It is supported by the data highlighted in gray, which are concentrated in Sets 1, 2, and 5. Our experiment suggests we should choose a clustering algorithm that provides diversity in results instead of using a large set of different clustering algorithms that does not guarantee diverse results.

D. RQ4

RQ4 - How does the number of base clusterings affect the recovery performance of cluster ensemble-based recovery methods?

Recall that the experiments shown in Table 3 use base clusterings with a fixed number of 50. To answer RQ4, we perform the cluster ensemble-based recovery experiments while changing the number of base clusterings. We create ten subsets of base clusterings whose numbers range from five to 50 with a step of five by randomly selecting the set of base clusterings shown in Fig. 5 without replacement.

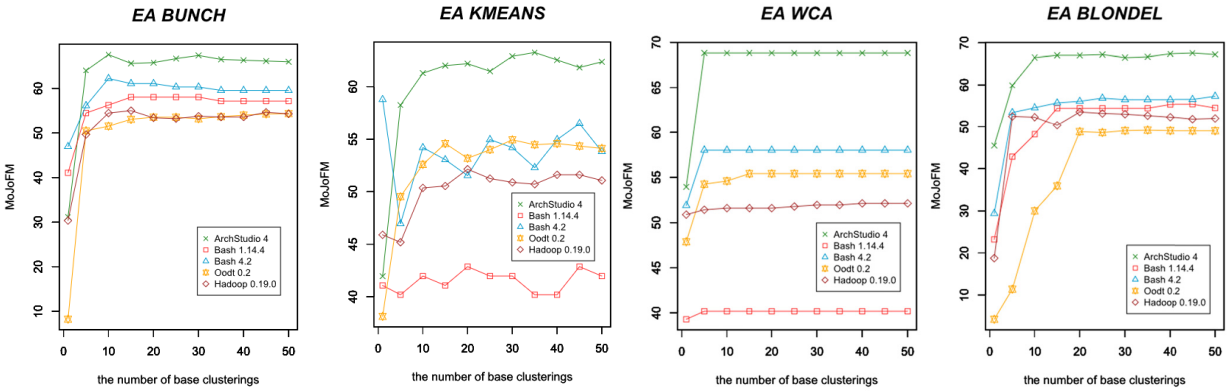


FIGURE 7. MoJoFM results of cluster ensemble-based recovery methods on varying numbers of base clusterings. The used sets of base clusterings are sets 1 to 4 in Fig. 5 and the applied consensus method is the EA.

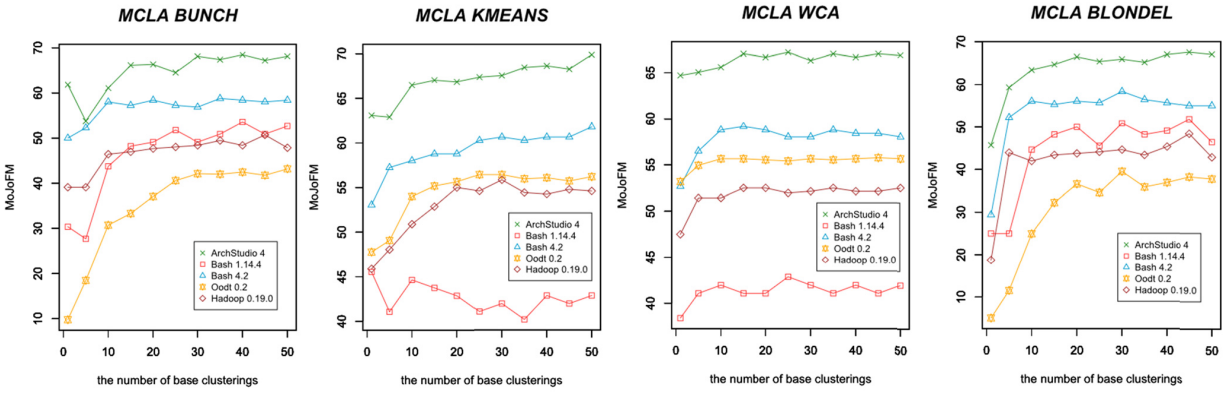


FIGURE 8. MoJoFM results of cluster ensemble-based recovery methods on varying numbers of base clusterings. The used sets of base clusterings are sets 1 to 4 in Fig. 5 and the applied consensus method is the MCLA.

We then apply consensus methods to the subsets and evaluate the results.

In designing the experiments for RQ4, we decided to focus on the consensus algorithms EA and MCLA, the two best-performing algorithms, while switching their base clusterings to Set 1, 2, . . . , 4 in Fig. 5.

Fig. 7 shows the experimental results when the EA is used as a consensus technique. The title on each graph represents the combination of an applied consensus method and a software clustering technique for generating base clusterings. For example, ''EA BUNCH'' means that we use Bunch for generating base clusterings and apply EA to produce a consolidated decomposition. The vertical axis of the graph represents the MoJoFM value, and the horizontal axis represents the number of base clusterings.

In the graph titled ''EA BUNCH,'' the MoJoFM score continuously increases for all five projects until the number of base clusterings reaches 10 and becomes stable. In the case of ''EA KMEANS,'' the performance on Bash 1.14.4 and Bash 4.2 fluctuates with varying numbers of base clusterings; however, in the remaining three projects, the MoJoFM value also increases as the number of base clusterings increases. Similar to the case of ''EA BUNCH,'' this increasing trend remains until the number of base clusters reaches 10. The MoJoFM values in the case of ''EA WCA'' become stable when the number of base clusterings is larger than five in all projects. This is due to the fact that, as we have mentioned in RQ3, the WCA fails to produce diverse base clustering sets even if the number of base clusterings increases. In the case of ''EA BLONDEL,'' it is observed that the MoJoFM value increases as the number of base clusters increases for all five projects. Also, similar to the cases for ''EA BUNCH'' and ''EA KMEANS,'' the MoJoFM value increases rapidly from 10 to 20 base clusterings, but there is no significant change thereafter.

Fig. 8 shows the result of using the MCLA instead of the EA as a consensus technique. Similar to Fig. 7, in the cases of ''MCLA BUNCH'' and ''MCLA BLONDEL,'' the MoJoFM values increase rapidly until the number of base clusters reaches 10 or 20 and then remain steady. The case for ''MCLA KMEANS'' is similar to the cases of ''MCLA BUNCH'' and ''MCLA BLONDEL'' except for the Bash 1.14.4 project. In the case of ''MCLA WCA,'' the result is almost the same as that of ''EA WCA,'' which also shows that changing the number of base clusters does not contribute to producing diverse base clusterings. The results shown in Fig. 7 and Fig. 8 suggest that at least ten different base clusterings are needed to benefit from cluster ensemble techniques for software architecture recovery as an answer to RQ4.

VI. THREATS TO VALIDITY

In this section, we discuss various threats to validity in our case study.

Construct Validity: Construct validity refers to whether the tools and methods used in the study have effectively measured what they intended to measure. The core question we are trying to answer is how well ensemble-based decomposition techniques can recover architecture. To do this, the similarity between the result of the proposed recovery method and the ground-truth structure is computed by MoJoFM, using a single ground-truth decomposition for each subject project. However, there can be several versions of acceptable decompositions for a software system. Hence, the evaluation method we adopted in this work may have caused a latent bias. To minimize this threat, we have chosen a set of ground truths that are widely accepted in the literature [9], [19]. MoJoFM is a measure of the similarity of two decompositions commonly used in recovery studies; hence we adopt this general measure that is familiar to most researchers in the software architecture recovery area.

Internal Validity: Internal validity is the question of whether there is a causal relationship between independent and dependent variables. In this study, the accuracy of ensemble-based recovery is measured while we are changing kinds of base clustering sets and consensus techniques. In addition, the number of base clusterings is considered with respect to RQ4. We have tried to minimize the threat to internal validity by controlling factors other than independent variables, but the threat still remains.

The four software clustering technologies used for creating the base clustering set support various parameter settings. As shown in Section 4.3.1, all the experiments are performed with default configuration except for the parameters adjusted to increase the diversity of the base clustering results. This default configuration might benefit or adversely affect a particular technique. However, note that we focus on applying cluster ensembles to architecture recovery, rather than trying to optimize the parameters of each technique. We believe that the default configuration should also be beneficial in reproducing our research.

Among the software clustering techniques used to generate base clusterings, Bunch and K-means are characterized by producing various outputs for each execution. Therefore, even if we repeat our case study again, the results may be different from previous experiments. To mitigate this threat, we set the cardinality of the base clusterings to 50, which is large enough to have stable results in the current experimental environment.

External Validity: External validity is related to the generalization of case study results. Our experiments were conducted on only open source software projects. Obviously, they cannot represent all software systems and our work did not cover proprietary software projects. The projects in our experiment, however, have diverse characteristics such as different domains, sizes, and implementation languages, which can mitigate the limitation.

VII. CONCLUSION AND FUTURE WORK

In this paper, we have proposed utilizing cluster ensemble techniques for software architecture recovery. The techniques are applied to five open source software projects and the results validate the effectiveness of cluster ensembles for software architecture recovery. Through the experiments, we have found that the MCLA and EA among the consensus techniques yield relatively quality recovery results. Moreover, it is observed that non-deterministic clustering algorithms such as Bunch and K-means are suitable for generating diverse base clusterings, which is critical to cluster ensembles. Our experimental results also indicate that at least 10 different base clusterings are recommended for cluster ensembles for software architecture recovery.

For future work, we will explore more diverse cluster ensemble algorithms for architecture recovery. We have considered only cluster ensembles based on the object co-occurrence approach in this work. In addition, we will apply the proposed recovery method to proprietary software projects as well as additional open source software projects.

ACKNOWLEDGMENT

A preliminary version of this paper appeared in Choongki Cho's Master thesis, "Software Architecture Module-View Recovery via Cluster Ensembles," Department of Computer Science and Engineering, Chung-Ang University, 2017.

REFERENCES

[1] P. Andritsos and V. Tzerpos, "Information-theoretic software clustering," *IEEE Trans. Softw. Eng.*, vol. 31, no. 2, pp. 150–165, Feb. 2005.

[2] M. Bastian, S. Heymann, M. Jacomy, and Gephi, "An open source software for exploring and manipulating networks," in *Proc. 3rd Int. AAAI Conf. Weblogs Social Media (ICWSM)*, San Jose, CA, USA, 2009, pp. 361–362.

[3] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech., Theory Exp.*, vol. 2008, Oct. 2008, Art. no. P10008.

[4] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little, *Documenting Software Architectures: Views and Beyond*. London, U.K.: Pearson, 2002.

[5] M. D'Ambros, M. Lanza, and R. Robbes, "On the relationship between change coupling and software defects," in *Proc. 16th Work. Conf. Reverse Eng. (WCRE)*, Washington, DC, USA, 2009, pp. 135–144.

[6] U. Erdemir, U. Tekin, and F. Buzluca, "Object oriented software clustering based on community structure," in *Proc. 18th Asia–Pacific Softw. Eng. Conf. (APSEC)*, Ho Chi Minh City, Vietnam, 2011, pp. 315–321.

[7] Z. Z. Fern and C. E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2004, pp. 36–43.

[8] A. Fred and A. Jain, "Combining multiple clusterings using evidence accumulation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 6, pp. 835–850, Jun. 2005.

[9] J. Garcia, I. Krka, C. Mattmann, and N. Medvidovic, "Obtaining ground-truth software architectures," in *Proc. Int. Conf. Softw. Eng. (ICSE)*, Piscataway, NJ, USA, 2013, pp. 901–910.

[10] J. Ghosh and A. Acharya, "Cluster ensembles," *Wiley Interdiscipl. Rev., Data Mining Knowl. Discovery*, vol. 1, no. 4, pp. 305–315, Jul/Aug. 2011.

[11] J. A. Hartigan and M. A. Wong, "A K-means clustering algorithm," *Appl. Stat.*, vol. 28, no. 1, pp. 100–108, 1979.

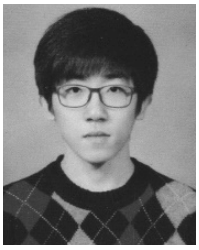
[12] A. Ibrahim, D. Rayside, and R. Kashef, "Cooperative based software clustering on dependency Graphs," in *Proc. 27th IEEE Can. Conf. Elect. Comput. Eng. (CCECE)*, May 2014, pp. 1–6.

[13] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 359–392, Aug. 1998.

[14] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar, "Multilevel hypergraph partitioning: Applications in VLSI domain," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 7, no. 1, pp. 69–79, Mar. 1999.
[15] R. Kashef and M. S. Kamel, "Cooperative clustering," *Pattern Recognit.*, vol. 43, no. 6, pp. 2315–2329, Jun. 2010.
[16] K. Kobayashi, M. Kamimura, K. Kato, K. Yano, and A. Matsuo, "Feature-gathering dependency-based software clustering using dedication and modularity," in *Proc. 28th IEEE Int. Conf. Softw. Maintenance (ICSM)*, Trento, Italy, Sep. 2012, pp. 462–471.
[17] A. Kuhn, S. Ducasse, and T. Gîrba, "Semantic clustering: Identifying topics in source code," *Inf. Softw. Technol.*, vol. 49, no. 3, pp. 230–243, Mar. 2007.
[18] R. Lambiotte, J.-C. Delvenne, and M. Barahona, "Laplacian dynamics and multiscale modular structure in networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 1, no. 2, pp. 76–90, Dec. 2015.
[19] T. Lutellier, D. Chollak, J. Garcia, L. Tan, D. Rayside, N. Medvidovic, and R. Kroeger, "Comparing software architecture recovery techniques using accurate dependencies," in *Proc. 37th Int. Conf. Softw. Eng. (ICSE)*, Florence, Italy, 2015, pp. 69–78.
[20] O. Maqbool and H. A. Babri, "The weighted combined algorithm: A linkage algorithm for software clustering," in *Proc. 8th Eur. Conf. Softw. Maintenance Reeng. (CSMR)*, Tampere, Finland, 2004, pp. 15–24.
[21] S. Mancoridis, B. S. Mitchell, Y. Chen, and E. R. Gansner, "Bunch: A clustering tool for the recovery and maintenance of software system structures," in *Proc. IEEE Int. Conf. Softw. Maintenance (ICSM)*, Oxford, U.K., 1999, pp. 50–59.
[22] S. Muhammad, O. Maqbool, and A. Q. Abbasi, "Evaluating relationship categories for clustering object-oriented software systems," *IET Softw.*, vol. 6, no. 3, pp. 260–274, Jun. 2012.
[23] R. Naseem, O. Maqbool, and S. Muhammad, "Cooperative clustering for software modularization," *J. Syst. Softw.*, vol. 86, no. 8, pp. 2045–2062, Aug. 2013.
[24] M. E. J. Newman, "Analysis of weighted networks," *Phys. Rev.*, vol. 70, no. 5, Nov. 2004, Art. no. 056131.
[25] N. Nguyen and R. Caruana, "Consensus clusterings," in *Proc. 7th IEEE Int. Conf. Data Mining (ICDM)*, Omaha, NE, USA, Oct. 2007, pp. 607–612.
[26] A. Shokoufandeh, S. Mancoridis, T. Denton, and M. Maycock, "Spectral and meta-heuristic algorithms for software clustering," *J. Syst. Softw.*, vol. 77, no. 3, pp. 213–223, Sep. 2005.
[27] M. Shtern and V. Tzerpos, "Clustering methodologies for software engineering," *Adv. Softw. Eng.*, vol. 2012, Jan. 2012, Art. no. 1.
[28] A. Strehl and J. Ghosh, "Cluster ensembles—A knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, Dec. 2003.
[29] A. Topchy, A. K. Jain, and W. Punch, "A mixture model for clustering ensembles," in *Proc. 4th SIAM Int. Conf. Data Mining*, Orlando, FL, USA, 2004, pp. 379–390.
[30] V. Tzerpos and R. C. Holt, "MoJo: A distance metric for software clusterings," in *Proc. 6th Work. Conf. Reverse Eng. (WCRE)*, Atlanta, GA, USA, 1999, pp. 187–193.
[31] V. Tzerpos and R. C. Holt, "ACCD: An algorithm for comprehension-driven clustering," in *Proc. 7th Work. Conf. Reverse Eng. (WCRE)*, Brisbane, QLD, Australia, 2000, pp. 258–267.
[32] S. Vega-Pons and J. Ruiz-Shulcloper, "A survey of clustering ensemble algorithms," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 25, no. 3, pp. 337–372, May 2011.
[33] Z. Wen and V. Tzerpos, "An effectiveness measure for software clustering algorithms," in *Proc. Int. Workshop Program Comprehension (IWPC)*, Bari, Italy, 2004, pp. 194–203.
[34] *Apache Hadoop*. Accessed: Jun. 2019. [Online]. Available: http://hadoop.apache.org/
[35] *Apache OODT*. Accessed: Jun. 2019. [Online]. Available: https://oodt.apache.org/
[36] *ARCHSTUDIO*. Accessed: Jun. 2019. [Online]. Available: http://isr.uci.edu/projects/archstudio/
[37] *GNU Bash*. Accessed: Jun. 2019. [Online]. Available: https://www.gnu.org/software/bash/bash.html/
[38] *Matlab*. Accessed: Jun. 2019. [Online]. Available: https://www.mathworks.com/products/matlab/
[39] *Python*. Accessed: Jun. 2019. [Online]. Available: https://www.python.org/
[40] *Understand++*. Accessed: Jun. 2019. [Online]. Available: https://scitools.com/
[41] J. Wu, A. E. Hassan, and R. C. Holt, "Comparison of clustering algorithms in the context of software evolution," in *Proc. Int. Conf. Softw. Maintenance (ICSM)*, Budapest, Hungary, 2005, pp. 525–535.
[42] J.-C. Um, K.-S. Lee, and C.-G. Lee, "Adaptive weighting of structural dependency and textual similarity in software architecture recovery," *IEICE Trans. Inf. Syst.*, vol. 99-D, no. 3, pp. 756–759, 2016.
[43] R. Naseem, M. B. M. Deris, and O. Maqbool, "Software modularization using combination of multiple clustering," in *Proc. Int. Multi Topic Conf. (INMIC)*, Karachi, Pakistan, 2014, pp. 277–281.
[44] J. Garcia, I. Ivkovic, and N. Medvidovic, "A comparative analysis of software architecture recovery techniques," in *Proc. 28th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Palo Alto, CA, USA, 2013, pp. 486–496.
[45] J. Garcia, D. Popescu, C. Mattmann, N. Medvidovic, and Y. Cai, "Enhancing architectural recovery using concerns," in *Proc. 26th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Washington, DC, USA, 2011, pp. 552–555.
[46] A. M. Saeidi, J. Hage, R. Khadka, and S. Jansen, "A search-based approach to multi-view clustering of software systems," in *Proc. 22nd IEEE Int. Conf. Softw. Anal., Evol. Reeng. (SANER)*, 2015, pp. 429–438.
[47] A. B. Belle, G. E. Boussaidi, and S. Kpodjedo, "Combining lexical and structural information to reconstruct software layers," *Inform. Softw. Technol.*, vol. 74, pp. 1–16, 2016.

CHOONGKI CHO was born in Seoul, South Korea. He received the B.S. degree in computer science and astronomy and space science from Chungnam National University, Daejeon, in 2015, and the M.S. degree in computer sciences from Chung-Ang University, Seoul, in 2017. Since 2017, he has been a Software Engineer with Samsung Electronics. His research interests include software engineering, deep/machine learning, natural language processing, and embedded systems.

KI-SEONG LEE was born in Seoul, South Korea, in 1978. He received the B.S. degree in Korean literature in classical Chinese from Sungkyunkwan University, Seoul, in 2005, and the M.S. and Ph.D. degrees in computer science and engineering from Chung-Ang University, Seoul, in 2011 and 2015, respectively, where he has been an Assistant Professor with the Da Vinci College of General Education, since 2017. From 2006 to 2008, he was a Software Engineer at an Internet service company, OnNet, Seoul. His research interests include software architecture, deep/machine learning, and natural language processing.



MINSOO LEE was born in Seoul, South Korea, in 1995. He received the B.S. degree in computer science and engineering from Chung-Ang University, Seoul, in 2019, where he is currently pursuing the M.S. degree in computer science and engineering. His research interests include natural language processing, deep learning, and verification and validation of artificial intelligence software.

CHAN-GUN LEE was born in Seoul, South Korea, in 1972. He received the B.S. degree in computer engineering from Chung-Ang University, Seoul, in 1996, the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, in 1998, and the Ph.D. degree in computer science from The University of Texas at Austin, Austin, TX, USA, in 2005. From 2005 to 2007, he was a Senior Software Engineer with Intel, Hillsboro, OR, USA. Since 2007, he has been a Professor with the Department of Computer Science and Engineering, Chung-Ang University. He is the author of more than 30 articles and conference papers. His research interests include software engineering and real-time systems. He was a recipient of the Korea Foundation of Advanced Studies (KFAS) Fellowship, from 1999 to 2005.