Clustering Similarity Measures for Architecture Recovery of Evolving Software

Douglas E. U. Silva[1], **Roberto A. Bittencourt**[1], **Rodrigo T. Calumby**[1]

[1] UEFS – Universidade Estadual de Feira de Santana
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900

douglaseusilva@gmail.com, roberto@uefs.br, rtcalumby@uefs.br

Abstract. *Automated software architecture recovery of module views from source code is a challenging research issue. Different similarity measures are used to evaluate clustering algorithms in the software architecture recovery of module views. However, few studies seek to evaluate whether such measures accurately capture the similarities between two clusterings. This work presents an evaluation of six clustering similarity measures through the use of intrinsic quality and stability measures and the use of ground truth architectures proposed by developers. The results suggest that the MeCl metric is the most adequate to measure similarity in the context of comparison with ground truth models provided by developers. However, when the architectural models do not exist, the Purity metric shows the best results, as measured by the correlation with the intrinsic Silhouette coefficient.*

1. Introduction

Clustering algorithms are quasi-automatic techniques that seek to identify clusters of similar software entities from their features. In general, we may state that clustering software entities into modules, from a relation of similarity between them is a way to modularize a system. These modules formed by such clustering algorithms are usually called clusters.

Previous work by Wu et al. (2005) extracted monthly versions of some open source systems and applied different clustering algorithms to them, defining stability and authoritativeness metrics to evaluate the algorithms. Informally, stability indicates that when a system undergoes changes, generated clusters must reflect these changes at the architectural level. On the other hand, authoritativeness evaluates how resulting clusters resembles a clustering created by a software architect, i.e., an architectural model. Unfortunately, their work used architecture views based on file allocation in source code directories as the authority, and not a reference model of system modules explicitly defined by software architects. More recent work by Garcia et al. (2013) and Lutellier et al. (2015) evaluated several algorithms against five open source systems from the perspective of comparison with reference models and four different similarity metrics. Although those papers advance the process of evaluating clustering techniques for software architecture recovery, one needs to define the best clustering similarity metrics by taking into account appropriate clustering similarity criteria.

This work evaluates six similarity metrics, taking into account their authoritativeness and stability as well as their discriminatory power, and the comparison with intrinsic clustering metrics and with variability measures of the evolving software.

2. Background

Agglomerative hierarchical algorithms are a class of clustering algorithms that start with singleton clusters and, after $N - 1$ steps, all items are contained in a single cluster. Typically, rule of thumb cutoff points stop the clustering process before reaching the single cluster in order to generate meaningful clusterings.

2.1. Clustering Similarity Metrics

The work of Wu et al. (2005) proposed the evaluation of clustering algorithms from three criteria, from which we use two. *a)* **Authoritativeness.** Clusters generated by the algorithms should resemble some authority. Therefore, authoritativeness compares clusterings computed by an algorithm against a reference model clustering. *b)* **Stability.** Similar clusterings should be produced by similar versions of a software system. Therefore, stability assessment compares clusterings of consecutive versions of the target system.

Next, we present some clustering similarity metrics from the literature.

Precision-Recall. Defined by Anquetil and Lethbridge (1999) for the field of software architecture recovery, Precision-Recall, as a measure of similarity between two software clusterings A and B, is computed based on the comparison of entity pairs, and is defined as follows: *a)* Precision: Percentage of intra-cluster pairs in clustering A that are also intra-cluster pairs in clustering B. *b)* Recall: Percentage of intra-cluster pairs present in clustering B that are also intra-cluster pairs in clustering A.

To balance the precision and recall values, we use the $F1$ measure which is defined as the harmonic mean of precision and recall.

B-Cubed Precision-Recall. The *B-Cubed* version of Precision-Recall was created by associating precision and recall for each item in the clustering. The B-Cubed metric defines precision as the number of items in the same cluster belonging to its category, and recall as how many items in its category appear in the cluster.

As in the Precision-Recall metric, we also use the F1-measure to combine the precision and recall values computed by the B-Cubed metric. For simplicity, we will call the F1-measure of B-Cubed Precision-Recall as $B\text{-}Cubed\text{-}F1$.

Purity. Purity is a metric defined to quantify how much a cluster C_i is "pure", that is, how many items are correctly identified when comparing two clusterings A e B. This way, Purity can be used as a similarity metric to computer the similarity between two clusterings.

MoJo and MoJoSim. Tzerpos and Holt (1999) defined a dissimilarity metric between two architectural clusterings called *MoJo*. This metric is based on the number of operations to transform one clustering into another. ***Move*** entails removing one entity from one cluster and allocating it to another, and ***Join*** implies joining two existing clusters, decreasing the number of clusters of one.

To measure similarity between two clusterings, we define MoJoSim:

$$MoJoSim(A, B) = 1 - \frac{MoJo(A, B)}{n} \tag{1}$$

where n is the number of entities to be clustered.

EdgeSim. This metric was defined by Mitchell and Mancoridis (2001a) to deal with the shortcomings of MoJo, that does not take into account the edges between entities.

Given a graph $G = (V, E)$ representing the structure of a system, V being the set of source code entities, and E the set of weighted dependencies between entities, EdgeSim counts a set of pairs of edges which are either intra-cluster or inter-cluster in both clusterings A and B. This collection of edges is called the Υ set. From the computation of the Υ set, *EdgeSim(A, B)* is defined as:

$$EdgeSim(A, B) = \frac{weights(\Upsilon)}{weights(E)} \tag{2}$$

where $weights(\Upsilon)$ is the sum of the weights of the edges of the Υ set.

MeCl. Designed by Mitchell and Mancoridis (2001) to complement EdgeSim, MeCl measures the similarity between two clusterings from a different perspective, considering both vertices and edges.

$$MeCl(A, B) = 1 - \frac{weights(\Upsilon_B)}{weights(E)} \tag{3}$$

where $weights(\Upsilon_B)$ is the sum of the weights of the set of edges that are intra-cluster in A but are inter-cluster in B, generating costs when inserting new inter-cluster edges. When the edges are not weighted, their weight is taken as equal to one. MeCl is not reflexive, i.e., $MeCl(A, B) \neq MeCl(B, A)$, thus we take the minimum of both to measure MeCl.

2.2. Intrinsic Metrics of Clustering Quality

To evaluate the metrics, we compared the values of authoritativeness to an intrinsic metric of clustering quality named *Silhouette coefficient*. This is an internal clustering evaluation method used when there is no ground truth to compare to. It produces higher values when clusters are compact and well spaced from each other. Equation 4 shows its definition.

$$S = \frac{b(i) - a(i)}{\max\left(a(i), b(i)\right)} \tag{4}$$

where $a(i)$ is the average distance of the item i to all other items in its cluster, and $b(i)$ is the average distance to all the items in the closest cluster to the given i.

The Silhouette of a cluster is the average of the Silhouettes of the cluster items. The Silhouette of a clustering is the average of the clusters' Silhouettes. The Silhouette value ranges from -1 to 1, indicating a good clustering quality as it approaches 1.

2.3. Variability Metrics for Evolving Software Systems

To evaluate the adequacy of a stability metric, we need intrinsic measures of software variability. Thus, we used information on software system variability over time. Given the current version n of a system and its next version $(n + 1)$, we may derive measures of change existence and magnitude, which we name **deltas**. We used two types of deltas: one in the number of lines of code (ΔLOC) and another in the number of classes ($\Delta Classes$), both computed between two consecutive versions of a software system.

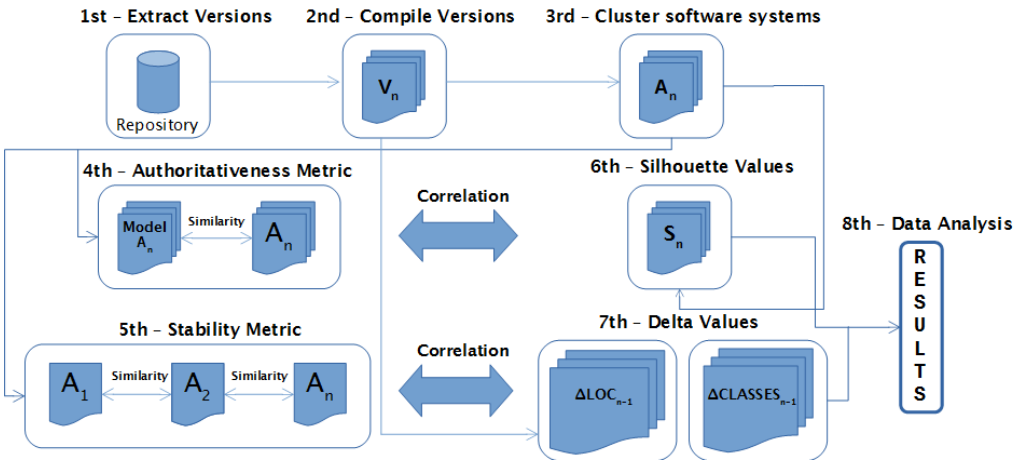

Figure 1. Experimental Design.

3. Experimental Design

Figure 1 shows the experimental design of this study. In the first step, weekly versions of the open source software systems are extracted. In the second, these versions are compiled and their designs, i.e., simplified system graphs, are extracted from the compiled files. In the third, we apply the clustering algorithms to the extracted designs. In the fourth and fifth steps, the clusterings are evaluated through authoritativeness and stability metrics. In the sixth, authoritativeness values are correlated with Silhouette values. In the seventh, stability values are correlated with ΔLOC and $\Delta Classes$. Finally, in the eighth, the results of authoritativeness and stability are studied in terms of descriptive statistics, and the computed correlations are analyzed to compare the clustering similarity metrics.

Table 1 lists the algorithms we used in our evaluation. The agglomerative algorithms compute similarity between clusters through information retrieval (IR) techniques. The IR techniques use only the vocabulary of code identifiers (e.g., class names, method names), following a pipeline of tokenizing, stop-word removal, normalizing, stemming, tf-idf computation and LSI reduction. We used the cut-off points 75 and 90 for the agglomerative algorithms such as presented in the work of Wu et al. (2005). In our evaluation, we use the software systems listed in Table 2. All of them are open source software systems developed in Java.

Table 1. Clustering Algorithms Used in the Analysis.

Name	Type	Linkage Rule
SL75	Hierarchical	Single
SL90	Hierarchical	Single
CL75	Hierarchical	Complete
CL90	Hierarchical	Complete

Table 2. Target systems with ground truth architectures.

System	Analysis Interval	# Classes	# Modules
SweetHome3D	03/08/09 to 02/28/10	142 to 165	9
Ant	10/29/06 to 10/21/07	487 to 511	16
Lucene	03/21/10 to 03/13/11	473 to 513	7
ArgoUML	11/19/06 to 11/11/07	1388 to 1524	19

4. Results

Here we describe our evaluation in terms of authoritativeness and stability.

A good clustering similarity metric is expected to have sufficient discriminatory power to identify differences between the clustering generated by the clustering algorithm and the clustering generated by experts: small differences should produce large values of authoritativeness while large differences should produce small values of authoritativeness. In a preliminary evaluation, we computed authoritativeness for each metric.

In addition, we compared the authoritativeness values of each metric with the Silhouette values of the algorithmic clustering. To do so, we computed the Silhouette in the clusterings generated by the algorithms in each system and measured the Pearson correlation between the values of authoritativeness and Silhouette.

Since the Silhouette generates values from -1 to 1, we normalized Silhouette values through the expression $S_{normalized} = \frac{S+1}{2}$ to bring Silhouette to the metrics scale.

Figure 2 illustrates data concentration and dispersion for the authoritativeness computed with each metric. Looking at the box plots, when comparing model-based metrics with each other, MeCl had higher median values of authoritativeness with a good dispersion range, followed by EdgeSim (with the exception of ArgoUML), and then Purity, MojoSim, B-Cubed-F1, followed by F1, with greater dispersion. For all box plots in this section, the axis named as *pr* is the F1-measure of the Precision-Recall values, while the axis named as *bcubed* is the F1-measure of the B-Cubed Precision-Recall values.

Figure 3 shows the correlations of each metric with Silhouette. We noticed from them that, in the per system graph, the measures of Purity, B-Cubed-F1, and F1 more strongly correlated with Silhouette. In the per metric graph, there was a trend of reducing the correlation of each metric with Silhouette with the increase of system size.

A good clustering similarity metric is expected to have sufficient discriminatory power to identify differences between consecutive versions of systems: small changes should produce large values of stability while large changes should produce small values of stability. In a preliminary evaluation, we first computed stability for each metric.

In addition, we compared the stability values of each metric with the deltas in both lines of code and in the number of classes between consecutive system versions. To do so, we computed the clustering stability produced by the algorithms for each system and measured the Pearson correlation between the stability for each metric and the ΔLOC or the $\Delta Classes$.

Figure 4 illustrates data concentration and dispersion for the stability values computed with each metric. We noticed that most metrics maintain high stability values, which is consistent with minor weekly changes. However, we also noticed that the stability of most metrics produced outliers with less stability, which captures situations where slightly larger changes occurred in the systems. Comparing the metrics by looking at the box plots, we noticed that MeCl had higher median values of stability, followed by EdgeSim (with the exception of ArgoUML), MojoSim, Purity and B-Cubed-F1. F1, on the other hand had very low stability values.

On the other hand, it is important to compare stability oscillations with variations of the evolving software itself, measured by ΔLOC and the $\Delta Classes$. It is expected that

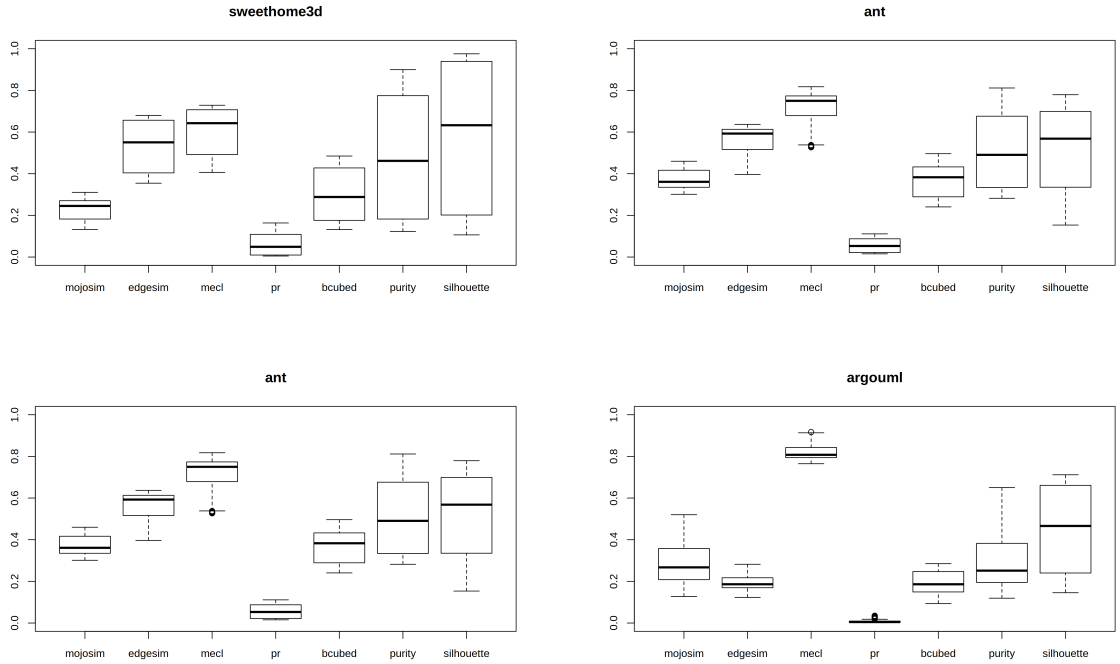

Figure 2. Authoritativeness and Silhouette per System

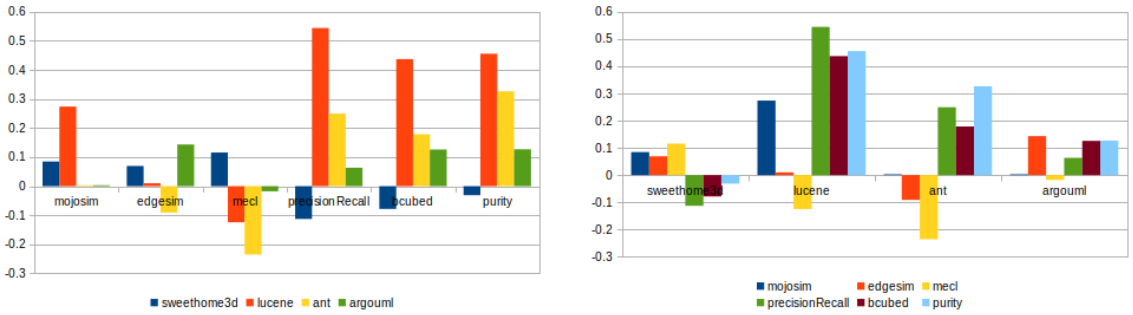

Figure 3. Correlations of Authoritativeness with Silhouette per System and per Metric.

stability correlates inversely with the deltas, i.e., the greater the changes in the software from one week to the next, the lower the stability.

Figures 5 and 6 show the correlations of each stability metric with ΔLOC and $\Delta Classes$. Since all correlation values were negative, we inverted their signs to facilitate understanding. The graphs show that, except for F1, all metrics obtained high correlations with both deltas in the graphs per system. However, with smaller systems, correlations were higher with software engineering metrics, and, with larger systems, they were higher with the classification metrics. In the per metric graph, there was a trend of correlation increase from small (SweetHome3D) to medium systems (Lucene), with later decrease with medium (Ant) and large systems (ArgoUML).

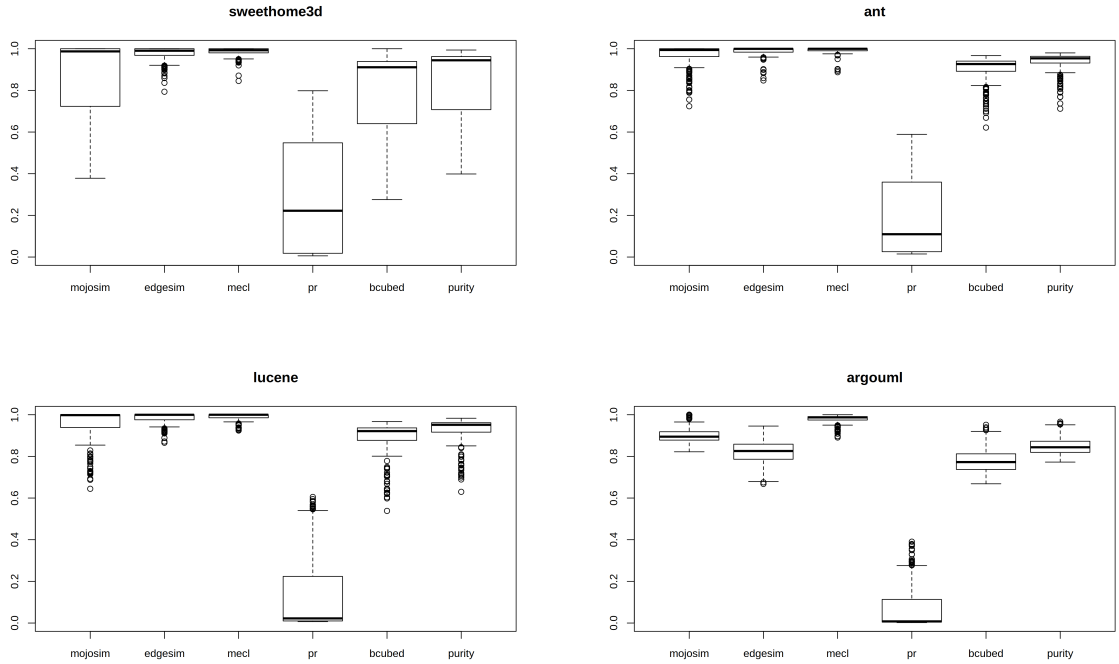

Figure 4. Stability per System

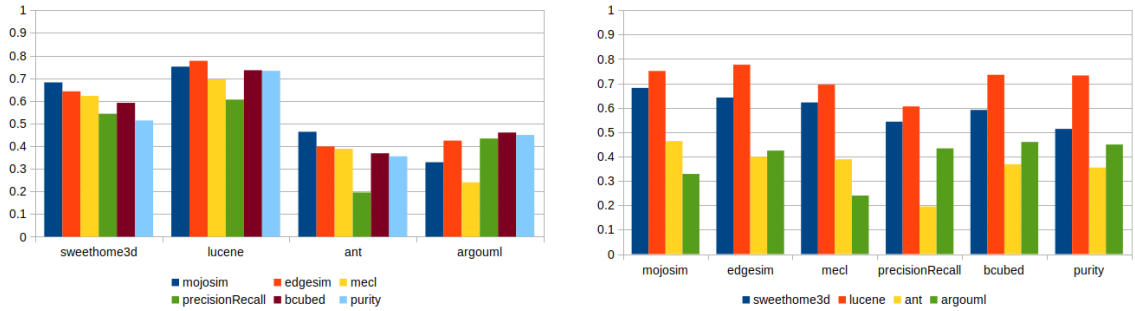

Figure 5. Correlations of Stability with ΔLOC per System and per Metric.

5. Discussion

From our experiments, most metrics generally show good authority and stability results except for F1 as shown in the box plots (Figures 2 and 4). However, when we look closely at the dispersion and concentration of stability and authoritativeness, we find that MeCl values are better when compared to the other metrics for both dimensions.

For authoritativeness, MeCl presents a range of values between 0.6 and 0.8, larger than the other metrics. This leads us to believe that, in the existence of reference models and from the data that we had available, MeCl would be the best metric to be used. However, if we consider Silhouette as the best intrinsic measure of clustering quality, Purity, B-Cubed-F1 and F1 show stronger correlation between authoritativeness and Silhouette.

For stability, all metrics showed high stability values. Even so, MeCl showed the highest median values, close to 1, in all systems, in addition to also showing outliers. With these results, MeCl would be the best candidate for stability. When looking at the

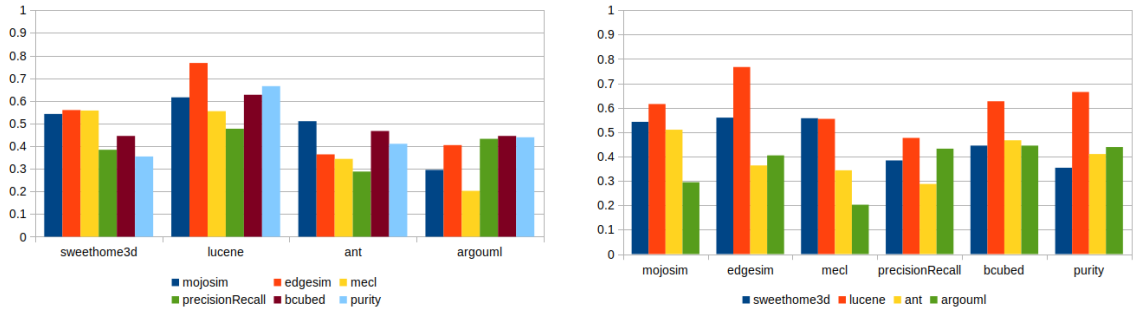

Figure 6. Correlations of Stability with $\Delta Classes$ per System and per Metric.

correlations with ΔLOC and $\Delta Classes$, we identify a trend of higher correlations for B-Cubed-F1, Purity, and F1 metrics as systems grow. Initially, all metrics show a high correlation around 0.7. However, the traditional classification metrics manage to maintain values somewhat larger with ArgoUML when compared to the software engineering metrics (e.g, MojoSim, EdgeSim and MeCl) with ArgoUML.

6. Conclusions

This work evaluated six clustering similarity metrics as candidates for measuring the quality of clusterings generated by clustering algorithms. For authoritativeness, MeCl stands out from the other metrics followed by EdgeSim (except for ArgoUML), then by Purity (greater dispersion), MojoSim, B-Cubed-F1, and F1 when the metrics are compared to each other. However, in the absence of reference models, the correlation with the Silhouette is stronger with Purity, B-Cubed-F1, and F1 metrics. For stability, MeCl also stands out with high median values in all cases. Correlations with ΔLOC and $\Delta Classes$ increased from Sweethome3D to Lucene and were lower for both Ant and ArgoUML.

As future work, we intend to evaluate agglomerative algorithms based on structural dependencies and to compare them with those based on information retrieval.

References

Anquetil, N. and Lethbridge, T. C. (1999). Experiments with clustering as a software remodularization method. In *6th Working Conference on Reverse Engineering*.

Garcia, J., Ivkovic, I., and Medvidovic, N. (2013). A Comparative Analysis of Software Architecture Recovery Techniques. In *Int'l Conf. Automated Software Engineering*.

Lutellier, T., Chollak, D., Garcia, J., Rayside, D., Kroeger, R., Tan, L., Rayside, D., Medvidovic, N., and Kroeger, R. (2015). Comparing Software Architecture Recovery Techniques Using Accurate Dependencies. In *37th Int'l Conf. Software Engineering*.

Mitchell, B. S. and Mancoridis, S. (2001). Comparing the decompositions produced by software clustering algorithms using similarity measurements. In *International Conference on Software Maintenance*.

Tzerpos, V. and Holt, R. C. (1999). MoJo: a distance metric for software clusterings. In *6th Working Conference on Reverse Engineering*.

Wu, J., Hassan, A. E., and Holt, R. C. (2005). Comparison of Clustering Algorithms in the Context of Software Evolution. In *21st International Conf. on Software Maintenance*.