

# Matrix Based Approach to Structural and Semantic Analysis Supporting Software Product Line Evolution

Jakub Perdek<sup>1,\*</sup>, Valentino Vranić<sup>2</sup>

<sup>1</sup>*Institute of Informatics, Information Systems and Software Engineering, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovičova 2, 842 16 Bratislava 4, Slovakia*

## Abstract

The evolution of product families is complex due to the exponential growth of new products with a modified codebase. It is caused by the introduction of new such important or restrictive features that allow transforming an existing solution into a new product as the basis for another one. Specifically, both the structure of resulting components and semantic information need to be taken into account to provide in-depth supporting materials by capturing feature interactions and their capabilities. Additionally, data from heterogeneous applications usually differ and their handling needs account for different scoring. We tackled these problems by creating various supporting views based on structural and semantic information. Related applications are modeled as graphs, also instances of their components are optionally included, and various matrix based algorithms based on similarity metrics are integrated. The final integrated and automated approach is efficient because it runs in polynomial time, is extensively focused on dependencies/connections between nodes, can be adapted to big data, and is extendable and enhanced to support different metrics. Its capability to organize analyzed parts into hierarchies by applying hierarchic clustering helps to specify the context to support comprehension or find a related position of features based on their interaction, especially their coupling. With regards to possible design issues, each updated product should be checked by an expert or more autonomously against performed predictions, especially its complexity and coupling between components. Additionally, extendability can be measured from the chosen sequence of such updates. An approach including an automated matrix based feature recognition process is presented in the analysis of modular Angular applications. Our future work will be more focused on semantic enhancements, and its applications on big data by generating and analyzing various types of fractals including extracted knowledge from them.

## Keywords

feature modeling, feature trees, matrix clustering, hierarchical clustering, graph comparison

## 1. Introduction

Software evolution, especially the evolution of software product lines, requires support decision-making with domain information about features, components, and related software artifacts. Even if the number of resulting products rises exponentially then the process is more complicated. We tackle both problems by proposing a method capable of generating various supporting views

---

*SQAMIA 2023: Workshop on Software Quality Analysis, Monitoring, Improvement, and Applications, September 10–13, 2023, Bratislava, Slovakia*

\*Corresponding author.


✉ [xperdek@stuba.sk](mailto:xperdek@stuba.sk) (J. Perdek); [vranic@stuba.sk](mailto:vranic@stuba.sk) (V. Vranić)

🌐 <http://fiit.sk/~vranic/> (V. Vranić)

🆔 0009-0003-3616-4373 (J. Perdek); 0000-0001-9044-4593 (V. Vranić)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

according to chosen metrics based on structural and semantic information. For this purpose, we integrated various matrix based algorithms which are based on similarity measures.

We addressed scalability issues by selecting fast algorithms which are capable to process graphs given by their adjacency matrices. In addition, issues related to restricted matrix size are handled by selecting only a subset from all candidates. Available approaches are mostly focused on semantics but do not directly handle structural information given by dependencies/connections between nodes. In our integrated approach, the information can be balanced with semantic one to focus on different software aspects that can support decision-making. Consequently, it is driven by calculated similarity metrics which selection depends on the chosen configuration.

The paper is focused on presenting the important aspects and applications of the used algorithms followed by their integration. Additionally, we commented on some of their parts and given decisions during their integration. Finally, We present our method applied to analyze two Angular applications, namely Puzzle To Play and Design 3D. From both of them, graphs were created. Extracted modular software fragments are mostly services or components. Connections are created based on observed coupling given by imports and occurrences in the HTML template. We made the proposed solution publicly available on GitHub<sup>1</sup>. The most of functionality is verified with unit tests and results agree with ones from the authentic papers.

The paper is organized as follows: Section 2 provides a detailed analysis of semantic methods for feature extraction and hierarchical mapping with a focus on graph merging and clustering driven by semantics. The way how various matrix based methods are integrated to support decision-making about software product line evolution is explained in Section 3. The solution is discussed in Section 4. Comparison with other available semantic-based methods is stated in Section 5. Finally, conclusions and future work are presented in Section 6.

## 2. Semantic Matrix Based Analysis of Features

Similarity can be measured by a lot of metrics. The main text-based approaches are shingles [1] and the cosine index. Categorical data are measured by the Jaccard index to compare two distinct sets which have evenly distributed data [2]. A few other metrics used to measure local structural information are the Hub Promoted index, Hub Depressed index, and Leicht-Holme-Newman index [3]. For problems not dependent, and thus unbiased on the number of overlapping items, but only on how many items from a smaller set are covered by a bigger one, is beneficial to use Inclusion index [4]. This index should be preferred over Cosine or the Jaccard index [4]. Another semantic approach focused on document categorization is Latent Semantic Indexing (LSI) where the similarity between two documents is measured by using cosine similarity or inner product on the vector representation of documents [5].

The farthest point algorithm (complete agglomerative clustering scheme [6]) was used as base metrics to evaluate the distance between clusters and finally during the creation of dendrogram in hierarchical clustering to get communities of manufacturing process [2]:

$$d(u, v) = \max\{dist(u[i], v[j])\}$$

Also, they combined similarity calculations by summing and making square root from the

---

<sup>1</sup><https://github.com/jperdek/matrixBasedMethodsForGraphs>

additive value of features where each feature is given by squared similarity ( $S_i$ ) of feature multiplied by given feature weight ( $k_i$ ):

$$S(x, y) = \sqrt{\sum_{i=0}^n k_i * (S_i)^2}$$

Metrics measure similarity from different representations such as sets or vectors and thus is necessary to choose also semantic attributes which should be processed. In the semantic-based extraction approach [7], authors extract/tag information from source code using encoding rules suitable for mapping into UML form. The return type, signatures, parameters, and lines of code are some of these semantic relations which can be measured and transformed into similarity scores. To extract appropriate information many metrics such as the Jaccard coefficient and cosine similarity are forming the similarity model in the next analyzed methods.

## 2.1. Matrix Based Hierarchical Graph Clustering

Connections in the graph carry semantic information fulfilled by implication that each reference from one source file to another one creates a relation between these documents [8]. Document weight can be evaluated accordingly by using non-negative input and output weights [9]. The maximal value from these two weights appropriately normalized is chosen to determine if the document is a better hub or authority [10]. Adding one to this value helps to not decrease overall weight during their multiplication. The next step consists of finding the correlation between pages by multiplying each weight on the path and with chosen correlation factor ( $0 < F < 1$ , but usually 0.5) powered by the length of the path to penalize longer paths:

$$C_{ij} = w_{i,k1} * w_{k1,k2} * w_{k2,k3} * \dots * w_{kn,j} * F^{dist(i,j)}$$

where each correlation weight  $w_{ij}$  is the maximum from weights  $w_i$  and  $w_j$  or 1 if documents match [10]. This part can be implemented using Floyd-Warshall algorithm [11] applied on adjacency matrix where maximum weights of edges in this future shortest path matrix are assigned for no connections (zeros) and self connections do not exist. The correlation weight matrix is assigned according to evaluated input and output weights and updated together with the distance matrix in each update of the shortest path matrix in this algorithm. If nodes are the same resulting correlation is one [10].

The next important part is the creation of a similarity metric to take into account the input and output contributions of links [10]:

$$sim(i, j) = \alpha_{ij} * sim_{ij}^{in} + \beta_{ij} * sim_{ij}^{out}$$

For simplicity, we refer to each row and column of the matrix as vectors. Each mentioned evaluated metric is multiplied by the coefficient consisting of the contribution of vector value for each row or column divided by the contribution of rows and columns together [10]:

$$\begin{aligned} rows_{i,j}^{contr} &= |row_i| + |row_j| & cols_{i,j}^{contr} &= |col_i| + |col_j| \\ \alpha_{ij} &= \frac{rows_{i,j}^{contr}}{rows_{i,j}^{contr} + cols_{i,j}^{contr}} & \beta_{ij} &= \frac{cols_{i,j}^{contr}}{rows_{i,j}^{contr} + cols_{i,j}^{contr}} \end{aligned}$$

Cosine similarity metrics are used here to measure similarity between each type of links [10]:

$$sim_{ij}^{in} = \frac{(col_i * col_i)}{|col_i| * |col_j|} \quad sim_{ij}^{out} = \frac{(row_i * row_i)}{|row_i| * |row_j|}$$

Similarity for the same nodes in the resulting similarity matrix is 1 otherwise the value of  $sim(i, j)$ . Finally, similar documents should be put together and clustered by the algorithm based on matrix partition. How closely related attributes are (togetherness) is measured by affinity metric [12]. In the Bond energy algorithm (BEA) columns of the similarity matrix are permuted to maximize the global affinity matrix defined as [13]:

$$GA = \sum_{i=1}^n \sum_{j=1}^n sim_{i,j} * (sim_{i,j-1} + sim_{i,j+1})$$

By this operation elements of greater values are put closer to each other [14]. Rows are then permuted accordingly. Firstly, the algorithm initializes the first two columns according to the first columns of the similarity matrix. Then in each iteration, a new column is inserted into the new matrix in the place where the cont value is maximized. This value is evaluated for each adjacent pair of columns by temporarily substituting a new column before, between, or behind and given as [12]:

$$cont(colLeft, colMiddle, colRight) = 2 * bond(colLeft, colMiddle) + 2 * bond(colMiddle, colRight) - 2 * bond(colLeft, colRight)$$

where bond (affinity) is given as [12]:

$$aff(i, j) = bond(i, j) = \sum_{k=1}^n sim_{i,k} * sim_{j,k}$$

The global affinity matrix is iteratively created by inserting all columns into a new global affinity matrix according to Bond (affinity) between the column before and the first column or column after and the last column is 0 [13]. For the maximum value of cont, the new column is inserted in the colMiddle position. The rows are permuted in the same order. The similarity matrix then can be recreated by only swapping values according to evaluated permutations that were required to form a global affinity matrix. Swapping is applied firstly on columns and then on rows accordingly. Permuted similarity matrix should be then partitioned into submatrices by choosing dividing point X positioned on a diagonal. Four matrices then can be evaluated using the following metric [14]:

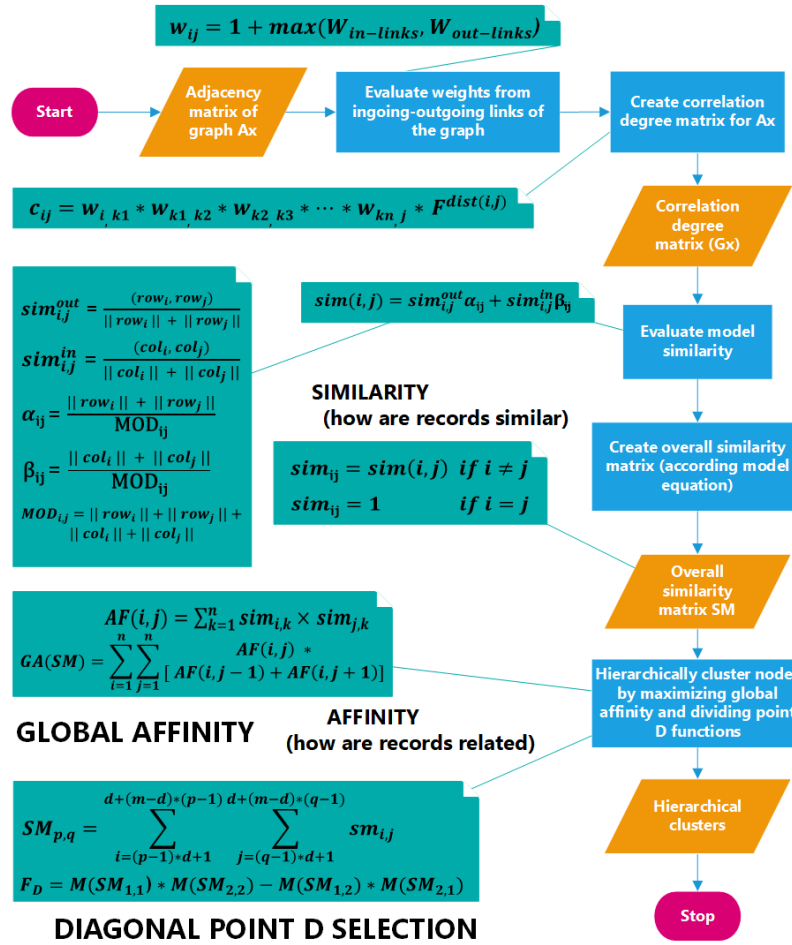
$$W_{SM_{p,q}} = \sum_{i=(p-1)*r+1}^{r+(m-1)*(p-1)} \sum_{j=(q-1)*r+1}^{r+(m-r)*(q-1)} sm_{i,j}$$

and for each dividing point, the following function is maximized [14]:

$$C_X = W_{SM_{1,1}} * W_{SM_{2,2}} - W_{SM_{1,2}} * W_{SM_{2,1}}$$

Dividing point X divides the matrix into four sub-matrices and the algorithm recursively clusters the upper left matrix ( $SM_{1,1}$ ) and then the bottom right one ( $SM_{2,2}$ ) [10]. The approach

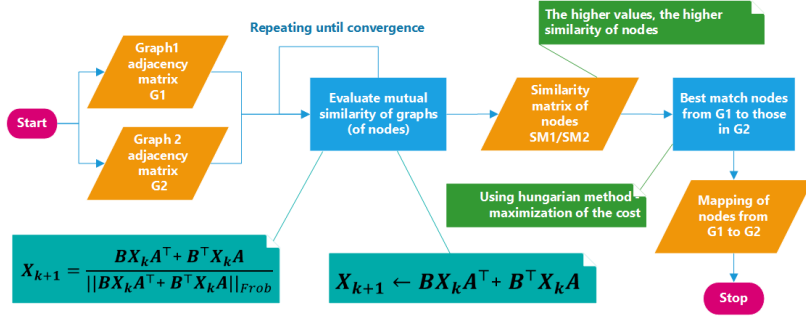
has been also extended to cluster the same document into multiple clusters in the work of Hou et al. [10]. The whole clustering process is depicted in Figure 1.



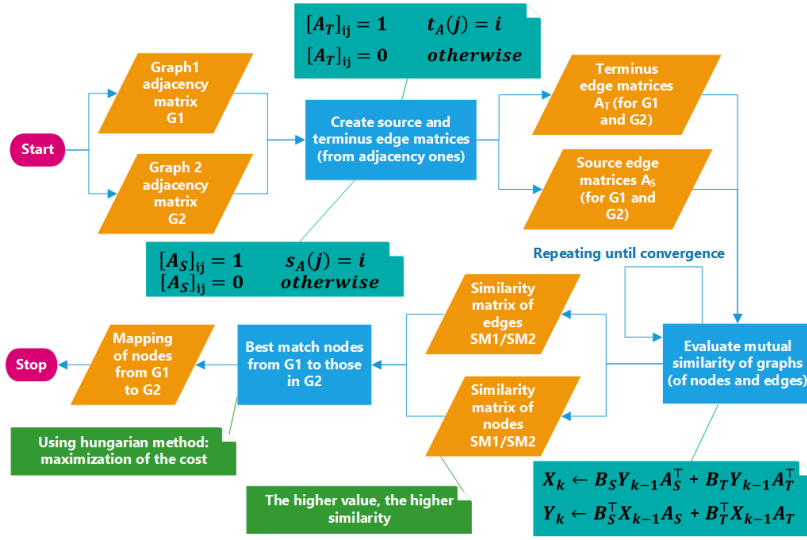
**Figure 1:** Matrix based hierarchical clustering. Source of algorithm: [10].

## 2.2. Matrix Based Comparison of Graphs

Similarity can be measured also between graphs and thus be applied in many fields. Feature extraction, graph isomorphism, and iterative methods are the main categories for graph similarity measurement [17]. In feature analysis of many applications, the main task is to find node correspondence. Information about connections can be used to characterize the most important nodes by finding if such a node is a good authority or hub. HITS [9] is a known iterative algorithm used to evaluate the importance of web pages. Authoritative and hub scores thus can be used as similarity scores of graph vertices (for given vertex  $j$  and vertex authority/hub) and further generalized for different graphs [15]. This can be used to match given graphs in an



**Figure 2:** Graph matching based on node similarity. Source of algorithm: [15].



**Figure 3:** Graph matching based on node-edge similarity scores. Source of algorithm: [16].

iterative updating process until convergence. This approach has been used for web searching and synonyms extraction [15] where similarity matrix  $X$  is updated according to the following equation:

$$X_{k+1} = B * X_k * A^T + B^T * X_k * A, \quad k = 0, 1, 2, \dots$$

where  $A$  and  $B$  are adjacency matrices. Obtained similarity matrix from even iteration [15] can be further used to find maximum weight matching between vertices of two graphs in graph matching approaches. The process of matching the vertices based on the adjacency matrix is depicted in Figure 2. Many authors [18, 19, 16] used Hungarian method [20] mainly for its asymptotic polynomial upper bound  $O(n^3)$  [21, 22, 23] to solve similar problems.

According to Zager and Verghese [16], this iterative process can be extended on edges as well. First, each adjacency matrix needs to be transformed into the source-edge matrix and

terminus-edge matrix by determining each source node of edge in the first one ( $A_S$  and  $B_S$ ) and terminal node edge in the second one ( $A_T$  and  $B_T$ ) by setting ones in given matrices [16]. In comparison with the first method oriented only on nodes, the update of edges (matrix  $Y$ ) and nodes (matrix  $X$ ) are intertwined:

$$\begin{aligned} Y_k &= B_S^T * X_{k-1} * A_S + B_T^T * X_{k-1} * A_T \\ X_k &= B_S * Y_{k-1} * A_S^T + B_T * Y_{k-1} * A_T^T \end{aligned}$$

After each phase, matrices are normalized by the Frobenius norm. The steps of the whole process applied to graph matching are depicted in Figure 3.

### 3. Integrating Matrix Based Methods for Structural and Semantic Analysis

Product variants from the same product family share similar code fragments. Their aggregation according to their interconnections and semantic content can potentially express domain knowledge which can be organized into variability or other models. Used scores for connections between nodes indicate the relation of a given software part with another, but mainly its coupling and importance. Consequently, is important to take context, which is given by semantics, into account. Semantic similarity scores must ensure balance of semantically related software fragments in each group.

Identification of features from the given source code should help analyze and compose different applications and even their parts on different hierarchy levels. Appropriate metrics for different types of documents need to be selected to capture their semantic information and relatedness. The most used metrics are various similarity scores. They are used in iterative algorithms. Web components have not only methods and classes but also template code. Modules in some environments such as Angular require different strategies to extract information and represent their structure as a graph. All dependencies are represented as connections between nodes and can be weighted. The remaining part is focused on the integration of matrix-based algorithms and decisions applied during this process.

Firstly, the application code is parsed to extract and collect imports, links, and optionally in the case of template of web components also elements that reference other ones. For future analysis, relevant semantic information is collected as well such as method and class names, variables, parameters, comments, attributes, and classes of markup language elements. Secondly, the graph is constructed according to potential references between components and optionally filled with semantic attributes. In a graph database given elements can be analyzed directly, but mainly it allows transformation from the selected part of the graph to the adjacency matrix.

Many related applications from the same domain are required for deeper analysis of features and further variability model creation such as a feature tree. Adjacency matrices from graph representation allow the grouping of similar nodes together according to the relation between software parts in each similar application modeled as a graph. In marginal cases, this grouping can be strengthened by merging nodes into the new one. This provides a way to analyze, process, and cluster unconnected graphs together. The methods for graph matching from Section 2.2 are adapted here. The simplest way is to the algorithm by Blondel et al. [15] that

evaluates similarity among nodes/documents only. Following this one, the application matrix with similarities between each pair of nodes/documents is produced. The higher the value, the higher the similarity of nodes is. These values can be used further to decide if nodes should be only grouped according to the introduced new node or merged into one. For analysis purposes only the intersection of nodes represented as the mapping between them can be used to only reinforce the existing application structure by grouping more related information. The second node-edge graph matching algorithm [16] which is performing the update of both node and edge similarity matrices in each iteration is due to the additional edge matrix being more flexible. The similarity between each pair of edges can be directly used in the clustering algorithm in time when document weight is evaluated according to input, and then output connections. Also in the process of grouping or merging nodes this information is useful to not lose edge weight determining the similarity of connection between nodes (if this value is low, the probability that nodes are semantically related in the given context is low).

The next important step is to extract given features from prepared sources. Connections amongst files are represented as an adjacency matrix with associated labels. To semantically enhance clustering additional information about indexed nodes is collected. From this matrix page weight and correlation matrix are created according to Section 2.1 which is based on Hou et al. [10]. If a similarity matrix of edges is available then these values are useful (depending on configuration) to use while evaluating document/node weight. In such cases mainly those which are analyzing graph intersection weights are updated according to the following additional change. During counting the number of edges, the adjacency matrix value referring to the given connection of nodes is multiplied by the appropriate edge similarity value increased by about one (to make weight at least one). Then it is possible to propagate the information of similarity of nodes during graph matching to the clustering process and thus make connections that appear in many applications stronger (the higher edge weight value) and match amongst each other as much as possible as those connected only in one or few applications or those partially matched according to their lower edge similarity score. Clustering of results, during which the additive value (for each node similarity score) of connections is enhanced by this scoring, are more precisely evaluated to support timeless compression of use cases and final feature detection. This is important to partially capture domain knowledge and what the system is.

In the following step, the similarity matrix used for clustering purposes is created. The connections of documents are evaluated using cosine similarity metric according to Hou et al. [10] and multiplied by a special coefficient given for each node pair as a ratio where the value of rows (columns) is divided into whole value. If we create additional matrices for each similarity metric and cluster them in case of equal values of this base similarity matrix then our model will not be easily scalable. To prevent this issue we extended the previous similarity equation about values by adding other semantic metrics multiplied by the given coefficient which value should be found during model tuning:

$$sim(i, j) = \alpha_{ij} * sim_{ij}^{in} + \beta_{ij} * sim_{ij}^{out} + \gamma_{ij} * sim_{ij}^{sem1} + \dots + \epsilon_{ij} * sim_{ij}^{sem2}$$

Other graphs, especially those created from component instances, can be processed and evaluated similarly. Final graph structure and results from clustering support further decision-making which is based on structural and semantic similarity of analyzed graphs. The process



flow of our integrated approach is displayed in Figure 4.



**Figure 4:** Matrix based structural/semantic analysis of features enhanced about node scores. Sources of integrated algorithms: [10, 15, 16].

## 4. Discussion

We apply integrated functionality on two Angular applications that significantly differ from each other but are tied up by the same domain in general. Thus they are characterized by overlap of the same components, operations, and problems. Our approach enables merging these two applications and then cluster source files. It allows us to analyze visualized results of the likely feature names or similarly coupled nodes. We decided to use the Neo4j graph database due to the possibility to perform fast evaluation and visualization of the aggregated results. It allows us to realize another analysis with inserted data. The same process has been applied to the Design 3D application. Finally, a hierarchy was created from given clusters.

Results show that semantic metrics are necessary, otherwise similarly coupled nodes will be clustered together in the clustering phase. Two tested applications are not enough to test full strength and tune algorithm parameters. For this purpose, we propose another fractal-based software product line capable to produce an extensive number of fractal products. Additionally, the evolution process will be automated and driven by knowledge handled by this integrated method. Information and component instances from each application will be merged, clustered,

and then used for classification purposes as input especially to graph neural network (GNN) or naive Bayes (NB) models.

## 5. Related Work

The most similar approach to ours is proposed in the AMPLE project [24]. Features and their dependencies are mined from documentation by using latent semantic analysis and feature models are also created by hierarchic clustering [25]. The significant difference with our approach is in its restriction to given semantics related to requirements. Our approach evaluates connections and node importance given by them and is additionally enhanced by semantic information from different metrics. Also merging graphs, and adding or removing nodes is done according to resulting scores and thus extend analytic possibilities that can be even automated. Our integrated approach can process big data and be reused for general problems.

## 6. Conclusions and Future Work

The exponential growth of information from related and mostly heterogeneous products makes decision-making about software product line evolution more complex. Specifically, both the structure of resulting components and semantic information need to be taken into account to provide in-depth supporting materials by capturing feature interactions and their capabilities. Consequently, this process needs to handle big data and incorporate fast techniques into the resulting solution. We address these problems by creating various semantic and structural views. These views consist of related knowledge that is organized into hierarchies or groups. The process is based on various matrix based algorithms with hierarchic clustering that all process graph data represented as adjacency matrices. Thanks to used algorithms and no introduced bottleneck, it runs in polynomial time which flexibly makes it suitable for processing big data, but due to restrictions on matrix size (very large matrices cannot be processed at once), further adaptations are required. Except for integrated algorithms designed for graph matching, other ones, especially algorithms used to solve a graph isomorphism problem, are NP-hard, or are not designed for massive amounts of data because of no directed selections from a large number of nodes. Integrated algorithms extensively use similarity metrics that are based on input and output connections. These can be easily replaced or adapted to other metrics by slightly modifying algorithms or just calculating single similarity values which will be used across all of them. This makes the integrated solution extendable. We enhanced it by calculating Jaccard's index from class-based categoric semantic variables such as class names or attributes from HTML templates. Still, for such a supporting similarity-based tool verification is required.

The resulting information can be merged or grouped on different abstraction levels such as merging whole graphs by merging or grouping the most similar nodes and optionally connecting (or disconnecting) remaining unmatched nodes. Nodes are grouped also in hierarchic structures and new more significant connections can be introduced even to newly and optionally created representative nodes for each group called medoids. Visualized results (mostly in a graph database) help to discover the related context which is necessary for making predictions and recognizing features. The overall process is based on similarity measures and thus results can be

different for their various settings and calculated metrics. This opens space for further analysis and possibilities in directing software product line evolution in an automated way. Created variability models and associated knowledge support decision-making about software product line evolution. An approach is presented in the analysis of modular Angular applications.

In future work, we will focus more on semantic enhancements, and their applications on big data by generating and analyzing various types of fractals including extracted knowledge from them. Possible future improvement is to calculate different metrics from software systems such as complexity scores or using various models to calculate semantic similarity. Integration into automated software product line evolution based on fractals will provide more information on which similarity measures should be used and in which way are the most beneficial to implement. This can be fulfilled by checking products against the best ones or using thresholds for measured metrics, and then omitting poor ones. Large space of generated products according to structural/semantic configuration and possibilities to insert given features needs to be handled.

## Acknowledgments

The work reported here received funding from the Operational Programme Integrated Infrastructure for the project: Support of Research Activities of Excellence Laboratories STU in Bratislava (ITMS code: 313021BXZ1), co-funded by the European Regional Development Fund (ERDF).

## Bibliography

- [1] B. Biegel, Q. D. Soetens, W. Hornig, S. Diehl, S. Demeyer, Comparison of similarity metrics for refactoring detection, in: Proceeding of the 8th working conference on Mining software repositories - MSR '11, ACM Press, Waikiki, Honolulu, HI, USA, 2011, p. 53.
- [2] K. Li, W. Z. Bernstein, Developing a capability-based similarity metric for manufacturing processes, in: Volume 3: Manufacturing Equipment and Systems, American Society of Mechanical Engineers, Los Angeles, California, USA, 2017, p. V003T04A015.
- [3] T. Zhou, L. Lu, Y.-C. Zhang, Predicting missing links via local information, *The European Physical Journal B* 71 (2009) 623–630.
- [4] C. Sternitzke, I. Bergmann, Similarity measures for document mapping: A comparative study on the level of an individual scientist, *Scientometrics* 78 (2009) 113–130.
- [5] A. Kuhn, S. Ducasse, T. Gîrba, Semantic clustering: Identifying topics in source code, *Information and Software Technology* 49 (2007) 230–243.
- [6] D. Müllner, Modern hierarchical, agglomerative clustering algorithms, *ArXiv* (2011).
- [7] R. Kadar, S. M. Syed-Mohamad, N. Abdul Rashid, Semantic-based extraction approach for generating source code summary towards program comprehension, in: 2015 9th Malaysian Software Engineering Conference (MySEC), IEEE, Kuala Lumpur, Malaysia, 2015, pp. 129–134.
- [8] K. Bharat, M. R. Henzinger, Improved algorithms for topic distillation in a hyperlinked environment, in: Proceedings of the 21st Annual International ACM SIGIR Conference on

- Research and Development in Information Retrieval, SIGIR '98, Association for Computing Machinery, New York, NY, USA, 1998, pp. 104–111.
- [9] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, *J. ACM* 46 (1999) 604–632.
- [10] J. Hou, Y. Zhang, J. Cao, Web page clustering: A hyperlink-based similarity and matrix-based hierarchical algorithms, in: *Web Technologies and Applications*, volume 2642, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 201–212.
- [11] R. W. Floyd, Algorithm 97: Shortest path, *Commun. ACM* 5 (1962) 345.
- [12] M. T. Özsu, P. Valduriez, *Principles of Distributed Database Systems*, Springer International Publishing, Cham, 2020.
- [13] D. B. Crouch, C. J. Crouch, G. Andreas, The use of cluster hierarchies in hypertext information retrieval, in: *Proceedings of the second annual ACM conference on Hypertext - HYPERTEXT '89*, ACM Press, Pittsburgh, Pennsylvania, United States, 1989, pp. 225–237.
- [14] Jitian Xiao, Yanchun Zhang, Xiaohua Jia, Tianzhu Li, Measuring similarity of interests for clustering web-users, in: *Proceedings 12th Australasian Database Conference. ADC 2001*, IEEE Comput. Soc, Gold Coast, Qld., Australia, 2001, pp. 107–114.
- [15] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, P. Van Dooren, A measure of similarity between graph vertices: Applications to synonym extraction and web searching, *SIAM Review* 46 (2004) 647–666.
- [16] L. A. Zager, G. C. Verghese, Graph similarity scoring and matching, *Applied Mathematics Letters* 21 (2008) 86–94.
- [17] D. Koutra, A. Parikh, A. Ramdas, J. Xiang, *Algorithms for graph similarity and subgraph matching* (2011) 50.
- [18] S. Umeyama, An eigendecomposition approach to weighted graph matching problems, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10 (1988) 695–703.
- [19] H. Almohamad, S. Duffuaa, A linear programming approach for the weighted graph matching problem, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (1993) 522–525.
- [20] H. W. Kuhn, The hungarian method for the assignment problem, *Naval Research Logistics Quarterly* 2 (1955) 83–97.
- [21] N. Tomizawa, On some techniques useful for solution of transportation network problems., *Networks* 1 (1971) 173–194.
- [22] J. Edmonds, Theoretical improvements in algorithmic efficiency for network flow problems (1972) 17.
- [23] E. Munapo, Development of an accelerating hungarian method for assignment problems, *Eastern-European Journal of Enterprise Technologies* 4 (2020) 6–13.
- [24] A. Rashid, J.-C. Royer, A. Rummler, *Aspect-Oriented, Model-Driven Software Product Lines: The AMPLE Way*, first edition ed., Cambridge University Press, New York, 2011.
- [25] K. Chen, W. Zhang, H. Zhao, H. Mei, An approach to constructing feature models based on requirements clustering, in: *13th IEEE International Conference on Requirements Engineering (RE'05)*, 2005, pp. 31–40.