Clustering Software Developer Repository Accesses with the Cophenetic Distance

David Sánchez-Charles[1], Josep Carmona[2], Victor Muntés-Mulero[1], Marc Solè Simo[1]

[1] CA Strategic Research, CA Technologies, Spain
{David.Sanchez,Victor.Muntes,Marc.Sole}@ca.com
[2] Universitat Politècnica de Catalunya, Spain
jcarmona@cs.upc.edu

Abstract. In this paper we report a case study on the use of the *cophenetic distance* for clustering software developer behavior accessing to a repository. This distance was recently proposed for the comparison of tree-based process models. We show how hierarchical clustering techniques over the cophenetic distance are capable of detecting homogeneous clusters corresponding to well-defined software developer roles. The techniques have been evaluated on real-life data from a software development project, and the quality of the clusters over the cophenetic distance is compared with similar techniques in the literature.

1 Introduction

Globalization of large enterprises is encouraging the industry to move towards Software as a Service (SaaS) solutions and communicate in digital platforms. One of the key benefits of these SaaS solutions is that any employee can access it whenever and wherever they can, and with limited resources such as personal laptops or mobile phones. Besides, these technologies enables for decentralized collaboration with peers in completely different time zones, providing diversity and flexibility in the workplace.

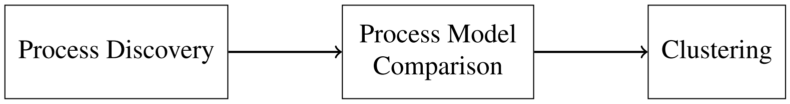
In the context of software development, Apache Subversion (SVN) and GitHub are two software versioning and revision control systems that allows software developers to collaborate in the maintenance and development of software, by monitoring changes in files such as source code, web pages and documentation. Those system have a record of actions performed by users, in order to increase visibility of developers or in case some changes need to be reverted.

The objective of this paper is to test if we are able to recover the original function (or role) of some workers of an organization, based on the actions performed in the software repository. Instead of directly analyzing sequences of actions, we propose to summarize them in the form of a process model. I.e. we will assume that a process model is capable of representing the behavior of the individuals in the platform. On the long-run, this is the first step to assess if one is able to gain some knowledge about individuals by analyzing, and comparing, how they behave in a digital platform.

In case of success, the techniques of this paper open the door for interesting applications, ranging from profiling of user behavior, detection of outlier behavior that may be suspicious of malware or fraud, user interfaces and processes that adapt to the behaviour of the user, and, in general, other scenarios in which we want to predict some attribute of the process owner. It will be of particular interest in crowdsourcing projects, in which organizations outsource particular internal processes to large corpora of virtual workers across the globe, for allowing organization to understand which type of workers are contributing to their problem resolution.

2 Methodology

In this paper we will try to group users with a similar role by analyzing the actions that they performed in a digital platform. The following diagram summarizes the approach we followed in this paper:



First, we preprocess the actions performed by the users and apply a process discovery technique over them. Following the process discovery, we will apply a process model comparison for measuring the pairwise dissimilarity of all process models and, implicitly, the dissimilarity of the human behavior that they represent. Finally, a clustering technique will be applied in order to discover groups of users with similar process models.

Section 2.1 briefly discusses the discovery of the process models that represents the behavior of the users. Section 2.2 covers the considered similarity metrics on this case study, and then Section 2.3 briefly mentions some clustering techniques appropriated for this scenario. Later in Section 3, we will evaluate such metrics and clustering techniques by applying them on a real-scenario in which the roles of the individuals are already known.

2.1 Process Discovery

The first step would be to generate a representation of the behavior of users in the digital platform. For this we have chosen to discover process models that summarizes the sequences of actions performed by users. Analyzing behavior over process models, instead of directly on the sequences, allow us to not focus on the specifics (such as, for instance, activity A has been repeated 5 times) but the general overview of how the user behaved (following the example, a process model simply states that activity A is usually repeated several times, whilst the number of times is not relevant).

Some sort of preprocessing may be needed in order to generate a process model. In particular, we may need to clean event names or ensure that traces represent independent runs of the underlying process. In our case, traces must compromise actions in a single session, with a unique purpose. In case this information is not provided, some heuristics must be applied in order to artificially split those sessions. For example, a *login* event may clearly separate two different sessions.

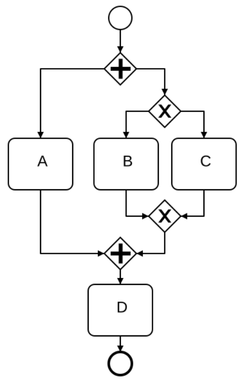
As for the process discovery itself, we will use the infrequent Inductive Miner [7]. This algorithm provides a good generalization of process models, can be efficiently applied to large event logs, and its output is a process tree – as required by some distances used in Section 2.2.

2.2 Process Model Comparison

The state of the art techniques for comparing process models can be naively split into structural and behavioral. In the former, process models are considered as labeled graphs and the comparison is regarded as edit operations over their edges, nodes or both. In contrast, behavioral techniques focus at the comparison of the execution semantics of the compared models.

If we focus on the case of structural comparison of process models, techniques based on graph edit operations have been defined [5,3,4,12]. In particular, we will consider the **graph edit distance** as defined in [5] for representing the group of structural similarity metrics. This distance counts the number of modifications (addition or removal of nodes and edges) must be performed in order to transform one process model into the other.

Analogously to structural techniques, behavioral ones compare how activities are related to each other. A *behavioral profile* of a process is a representation of the process model as an $n \times n$ matrix, where n is the number of tasks in the process [11]. An element $w_{i,j}$ in the matrix states the behavioral relation between the activity represented by the row i and the column j, which can be *causality* (typically depicted by $>$), *reverse causality* ($<$), *mutually exclusive* ($\vee$) or *co-occurrence* ($\wedge$) in the case of **causal behavioral profiles**. Behavioral profiles provides one mechanism for comparing the behavior of two process models, by counting the number of cells in which the two processes do not have the same behavioral relation.



	A	**B**	**C**	**D**
A		$\wedge$	$\wedge$	$>$
B	$\wedge$		$\vee$	$>$
C	$\wedge$	$\vee$		$>$
D	$<$	$<$	$<$	

Fig. 1. Example of a Casual Behavioral Profile for a process model. Activity A is concurrently executed ($\wedge$) with activities B and C, whilst activities B and C cannot be executed both in the same run as they are mutually exclusive ($\vee$). Finally, activity D is a consequence ($>$) of activities A, B and C.

In [9], we showed that the boundary between structural and behavioral metrics is not very clear, and some metrics capture behavioral differences even though they are purely defined in an structured manner. The **Cophenetic Distance** fall in this fuzzy boundary, and we will also consider it in this study.

In the rest of this section, we provide an informal explanation of the Cophenetic distance, whilst the rest of distances are well-known in the literature. The use of this metric limits the scope of the study to process trees in which activities are not duplicated. Nevertheless, the inductive miner algorithm already provides such type of process models, and such constraint also makes feasible the computation time of the aforementioned techniques.

Cophenetic Distance over Process Trees A **process tree** is a labeled rooted tree T in which activities are represented as leaves of the tree and internal nodes describe the control-flow of the process. For the sake of simplicity, we will label internal labels as *OR*[3], *AND*, *SEQ* and *LOOP* to represent the usual behavioural structures in a process model. We will also denote by **gateways** to these internal nodes, following the BPMN nomenclature. Children of a SEQ gateway are ordered in order to represent the sequential ordering of the subprocesses they represent. We allow silent activities by labeling them as $\emptyset$. Figure 2 depicts an example of a sequential process with a optional branch and two concurrent activities. On the right, the same process is represented as a process tree.

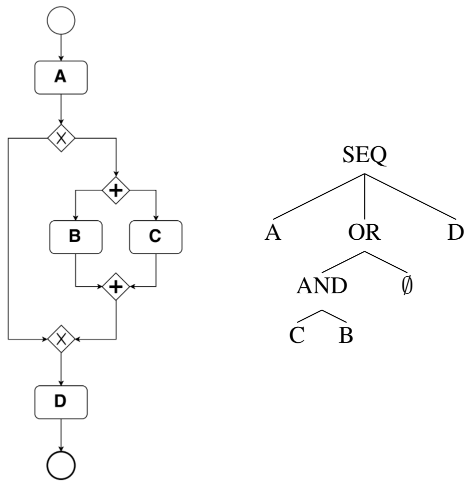


Fig. 2. Example of a structured process and its translation into a process tree. Activities B and C are concurrently executed, but are completely optional. The process always starts with activity A and ends with activity D.

In any process tree T, the deepest common ancestor of two nodes u and v–denoted by $[u, v]_T$–holds the direct causal relationship of the two nodes, and the depth of the common ancestor denotes the complexity of the process structure up to this behavioural decision. The depth of $[u, v]_T$ is known as **Cophenetic value**, simply denoted by $Depth_T$

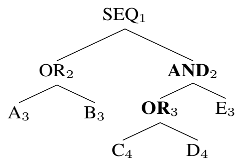
[3] Following the semantics of block-structured models in [1], only exclusive ORs are modeled.

$([u, v]_T)$, and the **Cophenetic vector** is the collection of such Cophenetic values for every possible pair of nodes u and v. Authors in [2] show that the Cophenetic vectors are enough to discern the structure of a certain class of labeled trees, which includes process trees without activity repetitions. And therefore, we could define a structural distance between process trees based on their cophenetic values.

Definition 1. *Let T and T' be two process trees, and S the set of activities of the two trees. Their* ***cophenetic distance*** *is*

$$d_\varphi(T, T') = \sum_{i,j \in S} |\mathit{Depth}_T([i, j]_T) - \mathit{Depth}_{T'}([i, j]_{T'})|$$

For the sake of simplicity $\mathit{Depth}([i, j]_T)$ is zero if either i or j are not present in the process tree T.



	A	B	C	D	E
A	3	2	1	1	1
B		3	1	1	1
C			4	3	2
D				4	2
E					3

	A	B	C	D	E
A	3	2	1	1	1
B		3	1	1	1
C			4	3	2
D				4	2
E					3

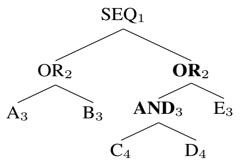
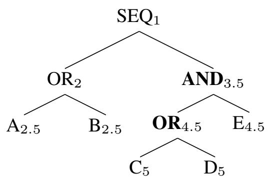


Fig. 3. Example of process trees and their cophenetic vector (in matrix representation), assuming the depth of the root is 1. For simplicity, we included node's depth as a subscript of the label.

Unfortunately, cophenetic vectors are not enough for determining behavioral similarity: for instance, Figure 3 depicts two similar processes where two internal nodes have been interchanged, but share the same cophenetic vector due to havin identical structures. In [9], we presented a new approach to compare process trees using the cophenetic distance that, by modifying the notion of tree depth, enables us to overcome such issue and leverages the traditional structural comparison with the behavioural information hard-coded into the new depth. Let's take the left model of Figure 4 to illustrate some of the rules defined in [9]. Activities A and B originally had depth 3, but 2.5 considering the proposed depth definition: the depth of their parent, which is 2, plus 0.5 given the dichotomy of the exclusive choice they are representing. The *AND* gateway was originally at depth 2 since it was a direct child of the root, whilst we are now positioning it at depth 3.5 as if it was a direct consequence of the two activities A and B. These consecutive depths are triggered by the behavioral function of its parent node – a sequential construct. One can check [9] for more details on how this new depth is defined.

2.3 Clustering of Process Models

At this point, we already have a process model summarizing the behavior of each individual as explained in section 2.1. We have computed a matrix D of distances between



	A	B	C	D	E
A	2.5	2	1	1	1
B		2.5	1	1	1
C			5	**4.5**	3.5
D				5	3.5
E					**4.5**

	A	B	C	D	E
A	2.5	2	1	1	1
B		2.5	1	1	1
C			5	**4**	3.5
D				5	3.5
E					**4**

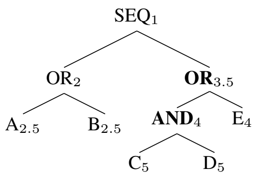


Fig. 4. Example of process trees and their cophenetic vector (in matrix representation), using the depth described in [9]. For simplicity, we included node's depth as a subscript of the label.

all process models, i.e. cell $D_{i,j}$ includes the distance between the process models of users i and j. This distance may be difned by the graph edit distance [5], differences in their causal behavioral profiles [11] or the cophenetic distance [9] as briefly discussed in section 2.2.

In this final step, our objective is to create groups of individuals that are close to each other with respect to the distance matrix D. The standard de-facto technique for clustering based on a distance matrix is Hierarchical clustering.

Hierarchical clustering Hierarchical clustering [8] follows a bottom-up approach, in which every individual starts within their own cluster and then iterates by merging the two most similar clusters into a bigger cluster. One tackles the issue of finding the two most similar clusters by averaging the distances of the individuals within the clusters. The only drawback of this technique is that the number of groups must be manually fixed. In our experiments, we will run the hierarchical clustering for all the possible number of groups.

3 Evaluation

In this section, we test the methodology explained in Section 2 with a real industrial dataset. First, we describe the provided behavioural data of 200 individuals and their organizational roles in the company. Process discovery is used to summarize the behavioural data from each individual, and dissimilarities between those processes are meant to measure differences in the behaviour of the individuals. Then, we test some clustering approaches with three different similarity metrics to measure how good they approximate the original organizational rules.

3.1 Framework of the evaluation

Apache Subversion (SVN) is a software versioning and revision control system. Software developers use SVN software to collaborate in the maintenance and development of software, by monitoring changes in files such as source code, web pages and documentation. All accesses to a SVN repositroy are done through HTTP/S, as specified in the WebDAV/DeltaV protocol. It turns out [10] that those read and write requests over HTTP/S can be translated to human-friendly SVN commands such as *svn update* or *svn* *commit*. Continuining the work done by Li Sun et.al. [10], we model the behaviour of developers by using process discovery techniques. First, SVN commands are retrieved from the system and considered as events of a system that represents the developer, and then a trace is defined as all commands executed during a complete business day. As already mentioned in Section 2.1, we discover Process Trees using the default settings of the Inductive Miner Plugin (ProM 6.5.1).

This industrial dataset contains all the accesses of more than 200 individuals to one repository of CA Technologies in production for three years. After pruning users with few accesses to the repository, 83 individuals were kept in the study and their organizational roles were retrieved at the end of the monitoring phase. In particular, 37 *Forward Engineering*, 19 *Quality Assurance Engineers*, 16 *Sustaining Engineer*, 5 *Support*, 2 *Services*, 1 *SWAT Engineer*, 1 *Infrastructure*, 1 *Technical Writers*. The following list summarizes the responsibilities for each role.

– *Forward Engineers* (R1) are in charge of the implementation of new features.
– *Quality Assurance Engineers* (R2) plan, run and design use cases or tests.
– *Sustaining Engineers* (R3) are in charge of solving defects, as well as ensuring that software successfully passes all tests.
– *SWAT engineers* (R4) are in charge of implementing custom integrations.
– *Support* (R5), *Services* (R6) and *Infrastructure Engineers* (R7) interact with internal and external customers with respect to defect detection and solution, software installation and configuration, and maintenance of the infrastructure of Software as a Service solutions provided by the company. *Support Engineers* might push some quick fixes into products.
– *Technical Writers* (R8) collaborate with Forward, Sustaining and Quality Assurance Engineers for creating helpful Knowledge Base and User Guides. Technical Writers are asked to use the source code repository to maintain different versions of the documentation.

Among all the engineers, and fairly distributed among roles, 9 individuals are Managers of a team. Besides, one agent is labeled as a bot, although the purpose of such bot is unknown to the authors of this paper. Notice that one has the possibility of advancing in their career and change to another department, and, therefore, some individuals might have been *misclassified* as their latest role. Infrastructure and Service Engineers are not supposed to access the repository in their usual pipeline and, therefore, might have been promoted during the project. Nevertheless, clustering may help us to deduce their original roles in the organization.

During the rest of the evaluation we plan to answer the following question in regard of this scenario:

– How good is clustering of process models for approximating the original role of the individuals?
– Which is the expected role of the bot? And what about the role of other anomalies?

3.2 Homogeneity of roles in process-based Clustering

In order to measure the quality of the clustering, we will use the **purity** as the metric for measuring the homogeneity of the discovered groups. Let $C = \{C_1, \ldots, C_m\}$ be a clustering of the process models, and $R_i(C_j)$ be the number of individual in cluster C_j with the role R_i, then the purity is defined as

$$\text{Purity}(C, R) = \frac{1}{\text{Number of processes}} \sum_{j \in C} \max_{R_i} R_i(C_j)$$

In other words, the purity computes accuracy as if we label all individuals inside a group with the most popular role inside it. In particular, very heterogeneous groups of individuals will lead to a poor purity.

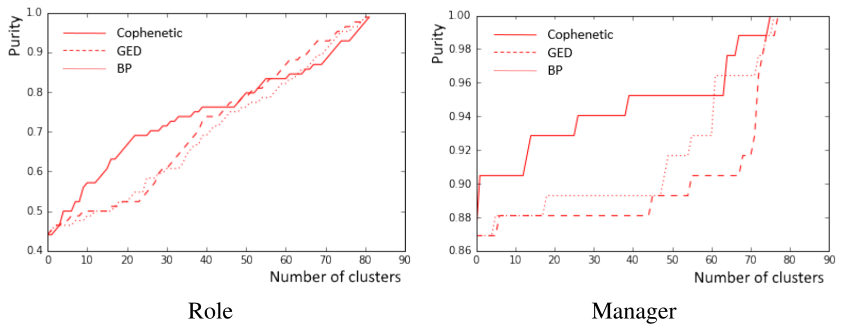


Fig. 5. The solid line depicts the evolution in the purity of the Cophenetic-based hierarchical clustering as the number of clusters increase, whilst the dashed (resp. dotted) line depicts the graph edit distance (resp. Behavioural Profiles). Two different experiments were performed for detecting individuals' role and their status as managers.

Figure 5 depicts the purity obtained from the SVN-repository dataset with respect to the number of clusters using hierarchical clustering. The X-axis represents the number of clusters considered, and its respective purity is represented by the Y-axis. The solid line depicts the purity of the clustering obtained based on the Cophenetic distance, and one can notice that its purity if significantly bigger when the number of clusters is low. On the right graphic, only the status as a Manager has been considered. Due to the low number of Managers in the dataset, these results denotes the ability of efficiently filter out abnormal behaviour. Starting from 50 clusters, the Cophenetic distance starts to classify worse than the Graph Edit Distance. Nevertheless, at that stage, clusters have 1 or 2 individuals and, therefore, are not representative of the use of clustering techniques.

Table 1 summarizes the precision and recall of the hierarchical clustering of Figure 5 when we set the number of clusters to 6. Again, we consider the classification as if the predicted label of all elements inside a cluster is the role of the more popular role inside the cluster. These results show again the capabilities of the Cophenetic distance to highlight the *Manager* role and provides, in general, better results for all roles, except the *Sustaining Engineer*. On the other hand, GED and Behavioral Profiles tend to group all individuals in a very big, and heterogeneous, cluster and, therefore, we obtain those roles with high recall but low precision (and those with high precision but very low recall).

	Cophenetic		GED		Behavioral Profiles	
	Precision	Recall	Precision	Recall	Precision	Recall
Manager	0.71	0.55	No individual was labelled			
Forward Engineer	0.64	0.76	0.46	0.97	0.45	1.00
Sustaining Engineer	0.00	0.00	0.57	0.26	0.00	0.00
Quality Assurance	0.34	0.57	No indiv.		1.00	0.05
Support	0.50	0.40	No individual was labelled			
Others	No individual was labelled					

Table 1. Precision and Recall for each of the roles in the organization by considering a hierarchical clustering with 6 groups. In some cases, none of the groups had enough representation of a role.

As for the bad performance with respect to Sustaining Engineers, notice that responsibilities of the Sustaining ($R2$), Quality Assurance ($R3$), SWAT ($R4$) and Support engineers ($R5$) are all related to defects and bug fixing, and, therefore, they may share some common behaviour and practices. Besides, the number of Sustaining Engineers is slightly below the number of Quality Assurance Engineers, and, hence, it is more likely to label users as Quality Assurance Engineers in case of grouping them together.Table 2 summarizes the precision and recall for a clustering of 6 groups. Notice that precision and recall of the Forward Engineer category are not significantly affected in the case of the cophenetic distance, indicating the existence of groups with a strong presence of Forward Engineers. On the other hand, precision and recall are very affected in both GED and Behavioral Profiles cases. The results provided by the GED are an indication of one or more small groups groups of Forward Engineers (perfect precision, but low recall), and a big group in which half of the developers have a role in $R2345$ and the rest are Forward Engineers or other minor roles. As for behavioral profiles, results are slightly worse than the cophenetic distance, but still incapable of detecting the group of Managers.

We have run the same experiments using DBSCAN [6] as the clustering method. The key benefit of DBSCAN is that the number of clusters is not fixed prior to the clustering, as it defines clusters as groups of individuals that are densely together[4]. Unfortunately, results are significantly worse than the provided by the hierarchical clustering – with purity not surpassing 0.5 across several hyperparameter of the DBSCAN algorithm.

Inducing the real role of outliers Some role *anomalies* were present in the dataset. For instance, two individuals were classified as Service Engineers ($R6$) although accessing to the source code repository is not part of their responsibilities. As we have already mentioned, the role data was obtained during the finalization of the project and, hence, the worker may have changed from one department to another. In this case, one service engineer ($R6$) is more close to Quality Assurance Engineers ($R3$), and the other is close to a group of Forward Engineers ($R1$). The three distances are consistent with

[4] I.e. for every process model in the cluster, there must be at least k process models at distance less or equal than d. Both k and d are manually fixed.

	Cophenetic		**GED**		**Behavioral Profiles**	
	Precision	Recall	Precision	Recall	Precision	Recall
Manager	0.71	0.55	No individual was labelled			
Forward Engineer (R1)	0.60	0.78	1.00	0.05	0.65	0.35
R2345	0.72	0.63	0.50	1.00	0.56	0.85
Others	No individual was labelled					

Table 2. Precision and Recall for each of the roles in the organization by considering a hierarchical clustering with 6 groups after merging *Sustaining*, *Quality Assurance*, SWAT and *Support* Engineers into a unique role *R2345*. In some cases, none of the groups had enough representation of a role.

these results. With respect to the Infrastructure Engineer ($R7$), the cophenetic distance and behavioural profiles map this user close to Sustaining Engineers ($R2$) whilst the graph edit distance relate him to Forward engineers ($R1$). Finally, with respect to the agent labeled as a BOT, the cophenetic and the graph edit distance group it with other Quality Assurance Engineers ($R3$). This might be a hint that the bot is indeed an automatic testing system. Nevertheless, Behavioural Profiles are less accurate and relate this agent close to a mixed group of Forward ($R1$), Sustaining ($R2$) and Quality Assurance engineers ($R3$).

4 Conclusions and Future Work

In this paper, we have applied the cophenetic distance for process models to recover groups of individuals with similar behaviours, and hence similar roles and responsibilities. Our approach is based on comparing the behaviour of process models discovered on real logs, instead of comparing them directly on the logs, allowing us to compare a generalization of the behaviour instead of falling into the specificness of traces. For instance, our approach allowed us to realize that a bot was working for a specific team, as this bot behaved as the other team members. We compared our cophenetic approach with three other process similarity metrics and we have seen that our approach consistently provides better role retrieval, as well as detecting a small group of individuals acting as Managers of a team.

As future work, we would like to investigate the possibility of discovering process models for each cluster such that the trace-fitness within the individuals in the cluster is high, whilst significantly lower when applied to individuals outside the cluster. That would help in understanding the behavior of new users into the system.

Acknowledgements. This work is partially funded by Secretaria de Universitats i Recerca of Generalitat de Catalunya, under the Industrial Doctorate Program 2013DI062, and the Spanish Ministry for Economy and Competitiveness (MINECO), the European Union (FEDER funds) under grant COMMAS (ref. TIN2013-46181-C2-1-R).

References

1. J. C. A. M. Buijs, B. F. van Dongen, and W. M. P. van der Aalst. A genetic algorithm for discovering process trees. In *2012 IEEE Congress on Evolutionary Computation*, 2012.
2. Gabriel Cardona, Arnau Mir, Francesc Rosselló, Lucía Rotger, and David Sánchez. Cophenetic metrics for phylogenetic trees, after sokal and rohlf. *BMC Bioinformatics*, 14(1):1–13, 2013.
3. Remco M. Dijkman, Marlon Dumas, and Luciano García-Bañuelos. Graph matching algorithms for business process model similarity search. In *BPM 2009, Ulm, Germany, September 8-10*, pages 48–63, 2009.
4. Remco M. Dijkman, Marlon Dumas, Luciano García-Bañuelos, and Reina Käärik. Aligning business process models. In *EDOC 2009, 1-4 September 2009, Auckland, New Zealand*, pages 45–53, 2009.
5. Remco M. Dijkman, Marlon Dumas, Boudewijn F. van Dongen, Reina Käärik, and Jan Mendling. Similarity of business process models: Metrics and evaluation. *Inf. Syst.*, 36(2):498–516, 2011.
6. Martin Ester, Hans-Peter Kriegel, Jrg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.
7. Sander J. J. Leemans, Dirk Fahland, and Wil M. P. van der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops 2013*, pages 66–78, 2013.
8. Fionn Murtagh and Pierre Legendre. Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? *J. Classif.*, 31(3):274–295, October 2014.
9. David Sánchez-Charles, Victor Muntés-Mulero, Josep Carmona, and Marc Solé. Process model comparison based on cophenetic distance. In *Business Process Management Forum - BPM Forum 2016, Rio de Janeiro, Brazil, September 18-22, 2016, Proceedings*, pages 141–158, 2016.
10. Li Sun, Serdar Boztas, Kathy Horadam, Asha Rao, and Steven Versteeg. Analysis of user behaviour in accessing a source code repository. Technical report, RMIT University and CA Technologies, 2013.
11. Matthias Weidlich, Jan Mendling, and Mathias Weske. Efficient consistency measurement based on behavioral profiles of process models. *IEEE Tr. Soft. Eng.*, 37(3):410–429, 2011.
12. Zhiqiang Yan, Remco M. Dijkman, and Paul W. P. J. Grefen. Fast business process similarity search. *Distributed and Parallel Databases*, 30(2):105–144, 2012.