Process Mining: A New Approach for Simplifying the Process Model Control Flow Visualization

Zineb Lamghari

LRIT associated unit to CNRST (URAC 29), Rabat IT Center, Faculty of Sciences, Mohammed V University in Rabat, Morocco, Email: zineb_lamghari2@um5.ac.ma

Received 18 June, 2022; Revised 25 July, 2022; Accepted 25 July,, 2022
Available online 26 July, 2022 at www.atlas-tjes.org, doi: 10.22545/2022/00193

Process Mining is a young discipline that provides a variety of techniques for extracting knowledge from recorded event logs. In this context, the process model is simulated, from event logs, to help enterprises to better understand their business processes. In this context, the discovery technique has always been a significant focus of process mining research. But despite the availability of several proposed approaches, participated in this process model visualization, we observe as the number of activities increases, the representation of the entire control flow becomes quite tedious. Indeed, event logs' complexity makes the process model visualization more difficult in terms of logical links and the number of causal relationships between activities. Therefore, we propose a new method for simplifying the process model control flow visualization. This is done by using string similarity algorithms to visually summarize the number of activities, sequence of execution, relative significance, and dependencies.

Keywords: String similarity algorithms, dynamic programming, process discovery algorithms, process mining, progressive alignment, affinity propagation.

1 Introduction

Process Mining (PMg) is a scientific discipline that falls between data mining and business process analysis as cited by Van der Aalst in [1]. The main idea of PMg is to use the execution BPs event logs that are recorded in the information system, to automatically generate, check, and enhance process models. An event log is a collection of traces that represent process instances that have been executed. Each event in an event log is assigned to an activity executed for a singular process instance (one trace). For each case, all events belonging to that case are ordered in a chronological style. In this regard, A = {a1, a2, ..., an} denotes a finite set, where ai, i:1, 2, ..., n is an activity of a case or sequence of length n. Thus, one case l can be expressed as l=<a1, a2, ..., an > and <> denotes an empty sequence. The event logs L with n cases and r repetitions can be expressed as L = [l1r, l2r, . . . , lnr]. For example, the event logs L=[<a,b,c,d>20, <a,c,b,d>15] signifies that l1=<a,b,c,d> repeated 20 times, l2=<a,c,b,d> repeated 15 times, etc.

Furthermore, PMg consists of three [1], which are: discovery, conformance, and enhancement. Discovery: An automatic process modeling methodology that takes event logs as input and produces a BP model as output. Conformance: compares the newly discovered process model with the existing process model.

The purpose is to detect deviations and identify bottlenecks. Enhancement: focuses on improving or extending the existing process model using the information recorded in event logs. In this context, the discovery technique has always been a significant focus of PMg research. The most popularly used discovery techniques typically produce node-edge graphs. In the occurrence of massive datasets, the discovered control flow model is incomprehensible. Furthermore, connecting multiple nodes can become challenging. Therefore, this paper proposes a new method for visualizing the causal relationship between the activities of a process (control flow). This method goes beyond control flow complexity because, regardless of event log complexity, it continues to generate comprehensible and simple control flow visualization.

The process model is simulated, from event logs, to represent, control and improve each company business process reality. In this context, the discovery technique has always been a major topic in PMg research. Several discovery algorithms are described with the basic representation of process models, like alpha algorithm. Other algorithms are representing different abstraction levels combined with clustering and classification techniques, to model processes from unstructured and complex events. In this regard, we are inspired by [2-4] to list the following process discovery algorithms: Alpha ++ algorithm, Heuristic Miner (HM), Inductive Miner (IM), Genetic Miner (GM), Fuzzy Miner (FM), State Based Regions (SBR), Language-Based Regions (LBR).

On one hand, the Alpha ++ detects non-free choice relations by describing activities of the selected relation that depend on other activities [5]. It cannot detect invisible tasks. Therefore, this algorithm gives unsound results. In this sense, an extended version of the alpha algorithm has been created to take into consideration the patterns' frequency. Indeed, the HM algorithm [6] can discover main behaviors and abstract exceptional and noisy ones, leaving out less important activities. This latter is unable to group traces with sub-log representation. Accordingly, the IM algorithm has been developed to treat events by grouping them into sub-logs. For each sub-log, a sub-process is generated. Then, a combination of the resulted sub-processes is released to obtain the generic process model. In this respect, the IM algorithm produces sound models [7], i.e., fewer non-conformities are detected, and it fits with many of the present logs. Furthermore, it is incapable of identifying complex and non-local process control patterns.

On the other hand, new algorithms have been developed to treat event logs in their uncertainty, for example, the GM algorithm. This algorithm uses the genetic concept in creating process models from logs. This is done randomly. For each process, the precision metric is calculated. Then, sound models are combined based on the mutation operation. The main limitation of this approach is its complexity in discovering and representing process models from real data sets [8]. From the same complexity standpoint, FM deals with unstructured processes [9]. In this sense, FM simplifies unstructured processes by preserving significant behavior, while less significant but highly correlated behaviors are aggregated into clusters, and less significant or less correlated behaviors are abstracted.

Furthermore, the SBR algorithm generates a Petri net from a Transitions System (ST) based on specific abstractions, such as: Set, Multi-Set, Sequence, and other types of abstractions, in which each state of the ST can be represented by a complete or partial trace. This algorithm ensures the fitness metric, as well as the identification of complex control structures. Besides, SBR is unable to process incomplete and noisy logs [10], while the LBR algorithm can find process model places based on the language process. Indeed, the LBR algorithm uses properties derived from logs (causal relationships), to determine the final model by describing various places. Unfortunately, this algorithm is unable to process incomplete and noisy logs [1].

In summary, obtaining a quality model is the main goal of a process discovery algorithm. There are various metrics and approaches for estimating process model quality criteria [9, 11]. Indeed, we can classify PMg discovery algorithms, according to quality criteria. This is based on the logic of the final representation, i.e., Algorithms producing Petri net models are suitable with fitness (this metric quantifies how much the observed behavior is captured by the model. i.e., it quantifies the fit of a log in a model), generalization (the model should generalize the present behavior in the log. This metric quantifies how well the model explains unobserved behaviors), and precision metrics (this metric quantifies how much behavior exists in the model that is not observed). For the fuzzy miner algorithm, the output model is a fuzzy model. To evaluate the fuzzy model, two metrics are available: Node detail (describes activities displayed in the Fuzzy model, related to the aggregated or deleted activities) and conformance (a measure

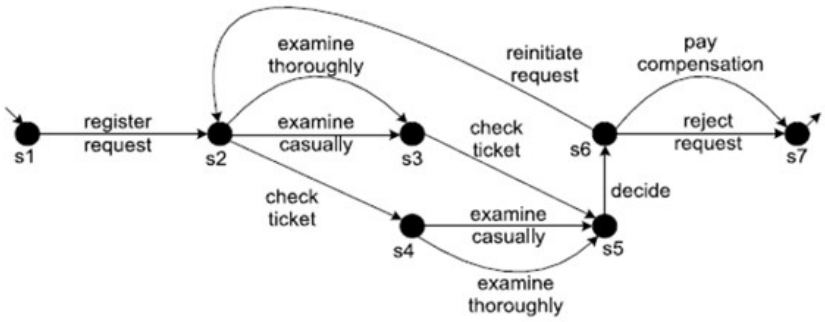

Figure 1: *Control flow modelled in Transition Net Notation.*

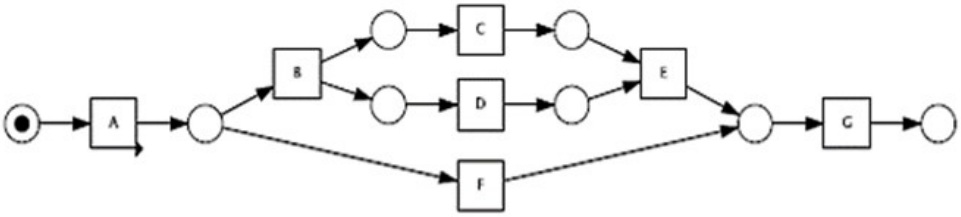

Figure 2: *Control flow modelled in Transition Net Notation.*

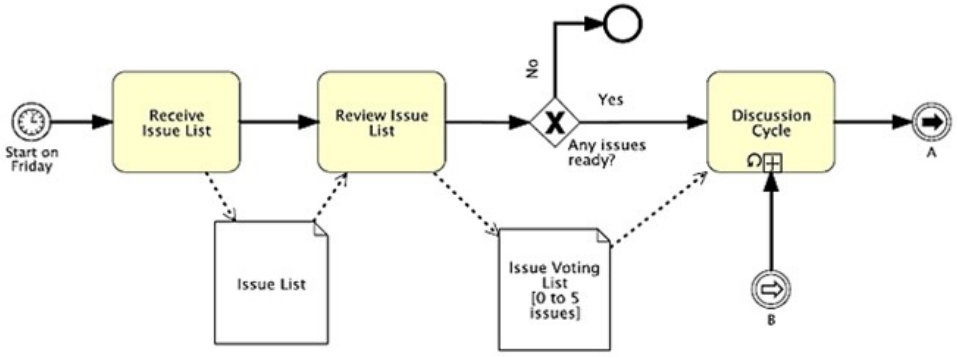

Figure 3: *Control flow modelled in BPMN.*

that describes the alignment between the Fuzzy model and the logs. Each activity in the logs that do not exist in the Fuzzy model will be counted as a deviation.

However, these traditional process discovery algorithms still have more ambiguities in terms of notation (see Figures 1, 2 and 3). For instance, the importance of the process's activities cannot be obtained from the Petri [12] notation or BPMN [13], i.e., the frequency with which an activity is carried out in comparison to other activities in the process. Besides, if the number of activities exceeds a certain threshold, it becomes extremely difficult to draw viable conclusions from them. Furthermore, it lacks a visual representation of the process's dominant path. As a result, existing control flow methods can be improved to visually include this missing information and make it more informative.

Adequately, sequences extracted from event logs can be encoded by letters. Indeed, string similarity algorithms [14] can be used to comprehend the control-flow perspective of the process more informatively. In this sense, it is primordial to 1) define a coherent set of traces, 2) calculate the similarity between events and clusters, 3) generate the aligned sequences, and 4) visualize sequences with an informative score. Therefore, we propose a new visualization method using string similarity algorithms.

In this sense, the reminder of this paper is as follows: Section 2 introduces our visualization method in understanding the control-flow perspective of the process model and making it more informative. Section 3 presents two case studies to evaluate the applicability of our visualization method. Section 4 discusses the obtained results Section 5 summarizes the paper and introduces future research.

2 Methodology

In this section, we present our visualization method for comprehending the control flow perspective of the process model and making it more informative. In this context, our method consists of the following four phases (see Figure 4): 1) Data Preparation, 2) Data Selection, 3) String Similarity Algorithm Selection, and 4) Control Flow Visualization.

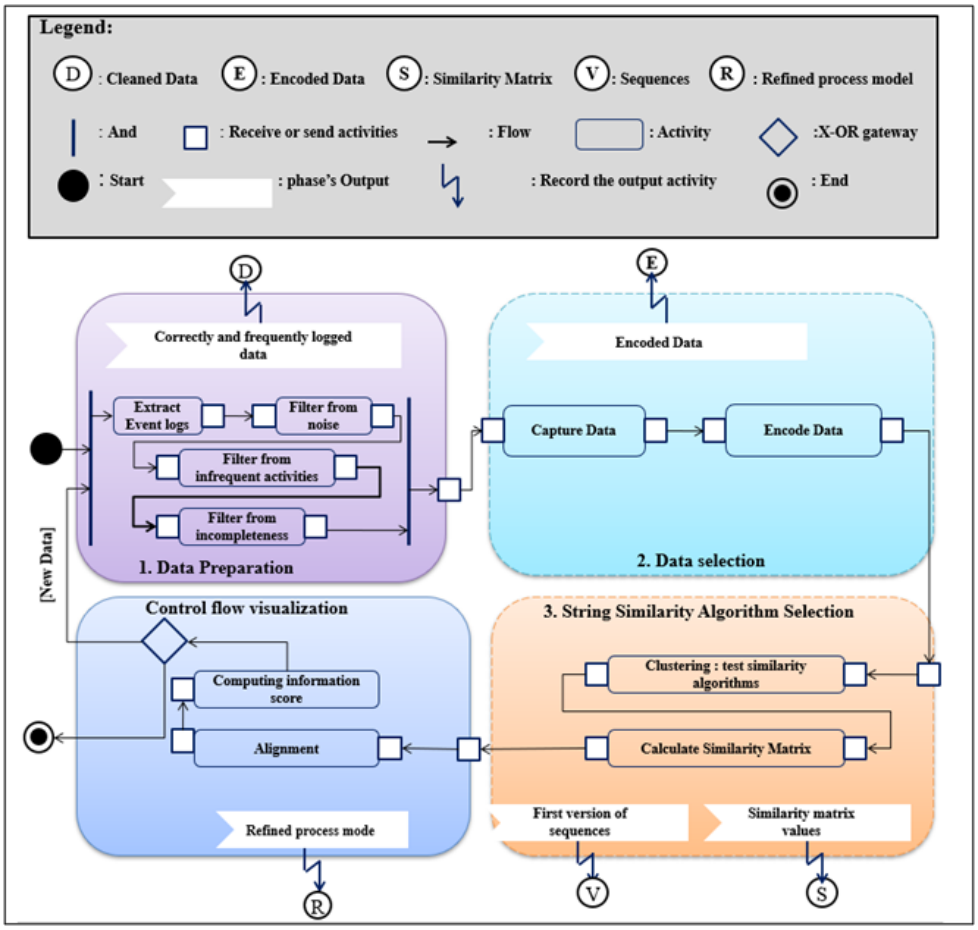

Figure 4: *Framework for simplifying the process model control flow visualization.*

3 Data Preparation

We start our method with the data preparation phase (see figure 4). This phase uses three filtering functions, to clean event data from noise, incompleteness, and infrequent behaviour. This phase's output is a set of cleaned event data (1), which is expressed as L' :

$$L' = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} = \left(a_{ij}\right)_{\substack{1 \le i \le m \\ 1 \le j \le n}} where\ m\ \le M\ and\ n\ \le N \quad (1)$$

$$(MxN\ dimensions\ of\ the\ original\ event\ data\ matrix).$$

It is required to clean event data in order to have a high-quality data set. For example, we need to do something with missing data or irrelevant (no-contribution) data, if the field has times missing value. In this context, we keep these events and ensure the field is set to Numpy.NaN. In addition, it is quite possible to record events without a case ID. Keeping them could cause mistakes; as such, we delete them. All these operations can be achieved with the PM4PY library and python libraries [33].

After obtaining cleaned data, the next step is to select events on which we try to apply specific treatment. Here, we describe the selection phase by two steps. The first step involves capturing data. The second step aims at encoding activities with Upper letters.

4 Data Selection

In this sense, we focus on cases and their corresponding activities as mentioned in figure 5. For example, L' cap = Ds(L' .case, L' .activity)= array([activity-01, . . . , activity-0n], . . . , [activity-m1,. . . , activity-mn]) as mentioned in Figure 5. Each activity is given a short name (usually a letter) to make things easier during the processing phase. For instance, the first trace in Figure 5 is transformed as follows:¡Appointment, Admit patient, Check history, Decide, Begin Treatment> = <A, B, C, D, E>.

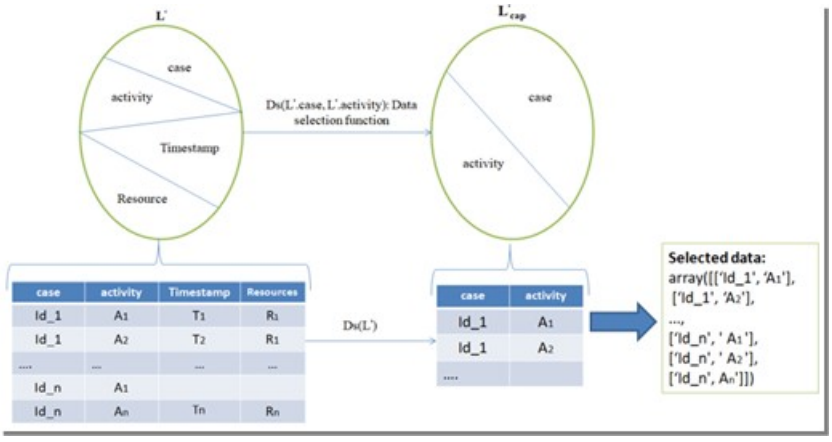

Figure 5: *Selecting data frame example.*

5 String Similarity Algorithm

This phase consists of defining the set of coherent traces (clusters) and calculating the similarity matrix using the dynamic programming technique [20] that consists of solving a problem by breaking it down into sub-problems, then solving the sub-problems from the smallest to the largest by storing the intermediate results.

In this sense, we select a suitable string similarity algorithm [14] that can be matched with the affinity propagation algorithm for clustering sequences into similar groups. Then, we apply the dynamic programming technique to generate the string similarity matrix [21]. This latter will be used in the alignment process to obtain control flow visual representation.

In statistics and data mining, affinity propagation is a clustering algorithm based on the concept of "message passing" between data points. Affinity propagation does not require the number of clusters to be determined or estimated before running the algorithm. It finds "exemplars," members of the input set that can represent clusters. As an example, we have selected the levenshtein algorithm [22] because of its simplicity of application. For our method, the algorithm for assembling sequences into clusters is based on (see Figure 6):

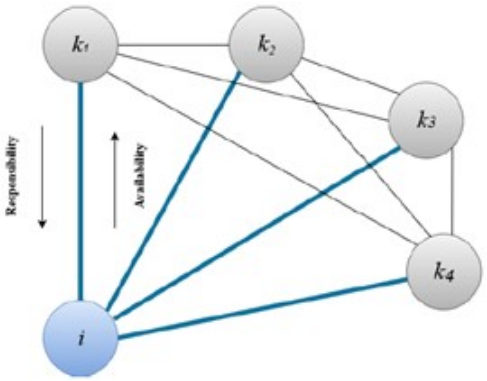

Figure 6: *A five-point data graph.*

- *Responsibility Matrix:* In this matrix R, r(i,k) defines how well point k can be used as a model for point i. *Availability Matrix:* In this matrix A, a(i,k) defines how well point I is suited to select point k as its exemplar.

  Figure 6 depicts an example of a graph with 5 data points. Every connection I and k of the graph between two points can be combined to form a matrix. In Figure 6, the data points communicate with one another and select an exemplar (based on the distance between them). The exemplar is the most key component of that cluster, and the rest of the data points are also components of that cluster. The clusters are defined by two matrices: responsibility and availability. The algorithm for grouping sequences into similar groups is shown below.
- The unaligned sequences are converted to an array to index them with a list.
- The unaligned sequences are converted to an array to be indexed by a list. The inverse of the Levenshtein distance for w1 in words to w2 in words is stored as a similarity matrix S for each word split in the array of words.
- i iterate through each row, calculating the maximum $(A + S)$ of that row for each index that is not equal to k. Where k is a function node (an example) and i is a variable node. This is the matrix of responsibilities (2).

$r(i,k) = s(i,k) - a(i,k') + s(i,k')$ (2)

The Availability matrix is determined by the responsibility at point k and sum of responsibilities that other data points assigned to k for all points, not on the diagonal of A (see equations 3 and 4).

$a(i,k) = (0, r(k,k)) + \sum \max(0, r(i',k))$ (3)

$a(k,k) = \sum \max(0, r(i',k))$ (4)

- The maximum value of A + B is used for selecting the final exemplar (see equation 5)

  $examplar(i,k) = max\{a(i',k) + b(i',k)\}$ (5)

  And the message graph is obtained by applying the above algorithm to the set of sequences defined in the Results and Experiments section.

$$\textbf{\textit{MessageGraph}} = \begin{bmatrix} 0 & -5 & -5 & -4 & -6 & -5 \\ -5 & 0 & -4 & -2 & -9 & -2 \\ -5 & -3 & 0 & -2 & -9 & -2 \\ -4 & -2 & -2 & 0 & -9 & -3 \\ -6 & -9 & -9 & -9 & 0 & -9 \\ -5 & -2 & -2 & -3 & -9 & 0 \end{bmatrix}$$

The message graph matrix is recognized as a similarity matrix, which is then used for dynamically aligning the sequences.

6 Process Model Refinement

Once, all the traces in the event log are converted to sequences, the suitable string similarity algorithm and the Affinity Propagation Algorithm are used to categorize similar instances into multiple clusters. Then, each cluster of traces is processed by applying dynamic programming techniques to obtain a similarity matrix.

This phase aims at refining the process model control flow visualization. To that end, we will apply the progressive alignment method to each similarity matrix and its corresponding cluster to generate the aligned sequences [15-17].

We start by computing the Longest Common Subsequence (LCS) to establish a table of similarity [15]. This step can be completed by using the dynamic programming method, which consists of solving a problem by breaking it down into sub-problems. Then, we proceed to align the traces with the highest value. Later, we align the resultant of the previous alignment with the trace that has the next maximum value. Progressively and similarly, we get the aligned sequence. Finally, we use an information score to visualize the aligned sequences. Multiple aligned sequences [17] help in understanding how the processes occur.

7 Synthesis

Our visualization approach can be summarized in Figure 7:
We start by filtering out deficiencies from event data using PM4PY. In this order, we obtain clean events (without noise, infrequent incomplete behaviors, and chaotic activities). Then, we proceed to select the data frame on which we will apply our visualization method. We target cases and their corresponding sequence

of activities. Afterward, we focus on extracting valuable features and discarding irrelevant information from the original data set. This is applied to the objective of unifying data and encoding activities by letters to apply the selected string similarity algorithm for clustering data. Thereafter, we calculate a similarity matrix in order to align the data progressively. This reduces the event data complexity. This allows acquiring simple data. Then, we must evaluate the obtained results in terms of complexity and simplicity. In the case of simple data, we proceed to calculate the score to generate informative information related to the control visualization step. In the case of complex data, we propose an optimization step. Last, we illustrate the simplified control flow.

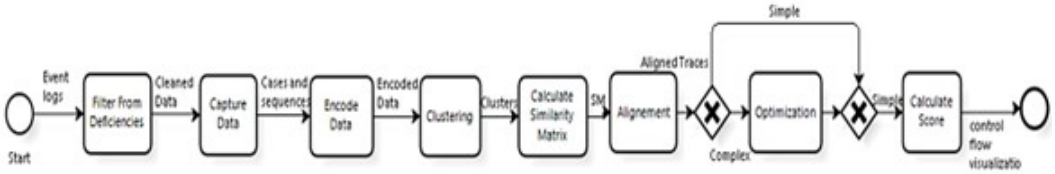

Figure 7: *A five-point data graph.*

8 Illustrative Case Study

The loan process[1] is a collection of artificial event logs that represent example of financial loan application belonging to a sample bank. The loan process was chosen for this case study for several reasons. The first reason is data availability where the process loan is among the most popular case studies at times, resulting in the availability of substantial amounts of data to analyse. The second reason is the loan process design, so that events are tracked and stored a prominent level of details, which allows insights into user behaviour and simplifies the extraction of data with. Last, the subsidiary information where information on the loan process is available, such as in [18] and [19], which serves as a basis, e.g., for conformance checking. Thus, the loan process promised to be an interesting candidate for deeper analysis.

The event log is containing a total of 100 event logs. We try to check the applicability of our proposed method for visualizing this process control flow. In this sense, we start by extracting and pre-processing event logs, to filter deficiencies (infrequent, incomplete, chaotic, and noisy event logs). For instance, figure 8.b illustrates with purple colour the existence of chaotic activities. These activities are executed arbitrarily in the process. They are correctly logged, but are still undesired, because they represent activities that are logged by the system even though they are not part of the main process flow. In this sense, we must filter them to generate cleaned data. This will help to provide coherent clusters in next stages.

For the loan Process, we have 8 activities with 100 traces (cases). We encode them with alphabetic symbols, from A to G (see Table 1 and Figure 8.a). This is done, to apply string similarity algorithms. In this context, we represent the loan encoded version in Figure 8. Here, each activity is mentioned with a specific color.

After applying our visualization method, we obtain the graph mentioned in Figure 8.b. Here, we observe the existence of two main clusters. We can classify activities from the mostly appeared to the lowest appeared (Cluster1: Actv2> Cluster1: Actv3> Cluster1: Actv5> Cluster1: Actv7> Cluster1: Actv4> Cluster1: Actv6> Cluster2: Actv1>Cluster2: Actv0).

[1]https://data.4tu.nl/articles/dataset/Loan_application_example_configuration_1/12715685

Table 1: Excerpt of loan process Event logs

Study	*Time Period Study*	*Data*
1	< register application, check credit, calculate capacity, check system, accept, random_chaotic, send decision e-mail>	A, B, C, D, E, F, G (Exemplar for cluster 1)
2	< register application, check credit, random_chaotic, calculate capacity, check system, accept, send decision e-mail>	A, B, F, C, D, E, G
3	< register application, random_chaotic, check credit, calculate capacity, check system, accept, send decision e-mail>	A, F, B, C, D, E, G
4	< register application, calculate capacity, check credit, reject, send decision e-mail, random_chaotic >	A, C, B, H, G, F
…	…	…
97	< register application, calculate capacit, check system, random_chaotic, check credit, reject, send decision e-mail>	A, C, D, F, B, H, G
98	< register application, check credit, calculate capacity, check system, accept, random_chaotic, send decision e-mail>	A, F, E, B, H, G, F
99	< random_chaotic, register application, calculate capacity, check credit, reject, send decision e-mail>	A, B, C, D, E, F (Exemplar of cluster 2)

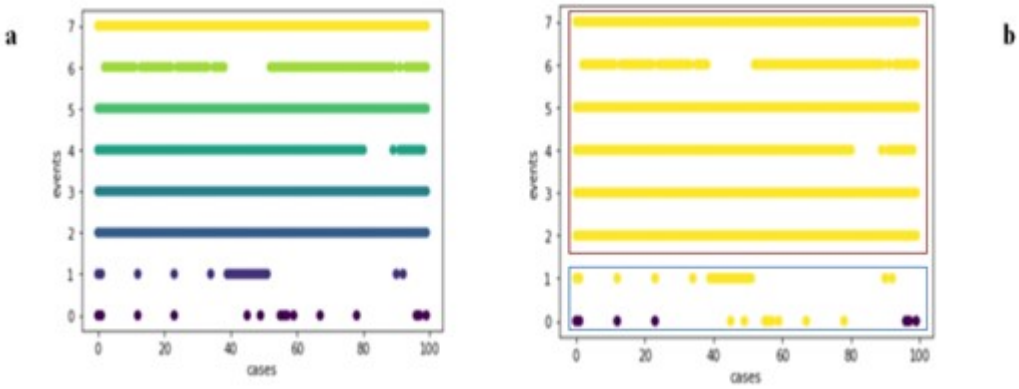

Figure 8: *Control Event logs diagnosis.*

After the clustering step (including the exemplar points), we can apply the programming dynamic technique (LCS) to obtain the matrix similarity. An excerpt of the similarity matrix is shown in Figure 9. This similarity matrix is deduced from 6 traces. This will be useful for aligning traces. Next, we can match the result of the two clusters. Clusters 1 and 2 are combined into a single output. This aids in determining how the various clustered visualizations interact with one another (see Figure 10).

Trace	1	2	3	4	5	6
1	0	4	4	5	7	4
2	0	0	3	4	4	3
3	0	0	0	3	4	4
4	0	0	0	0	4	3
5	0	0	0	0	0	4
6	0	0	0	0	0	0

A	B	-	-	-	C	-	D	E	F	C	D	G
A	B	C	D	E	C	B	D	E	B			I
A		-	-	-	-	-	-	-	-	C	D	G
A	-	-	-	-	C	-	-	-	-	B	D	G
A	-	-	-	-		-	-	-	F	C	D	I

Figure 9: *Similarity matrix and Aligned Traces.*

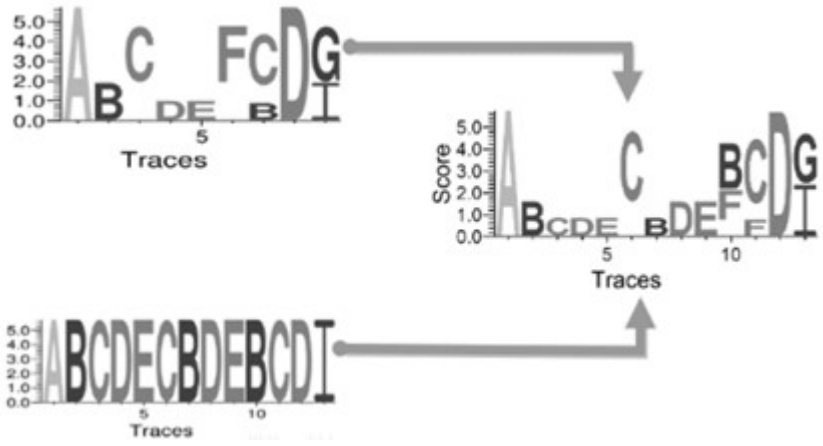

Figure 10: *Progressive Alignment by merging cluster 1 and cluster 2.*

In this sense, the score of each activity provides a streamlined representation of the control flow such as a related list of activities, their order, information gain/score, major event, etc. Further, multiple aligned Traces can be leveraged to render Optimized visualization. This representation helps in understanding how the processes occur.

9 Discussion

In this work, the approach for visualizing the control flow perspective of the process model and making it more informative hands over four main results. These results provide a visualized support for the user and enable to comprehend a process model that represents the user's behavior.

The first result represents the correct and the frequent event. This phase is like the studies of [2 and 3]; the authors applied several filtering mechanisms to extract data. In this approach, we use three filtering functions, to clean event data from noise, incompleteness, chaotic and infrequent behavior.

The second result is about selecting events on which we try to apply specific treatment. The first step involves capturing data. The second step aims at encoding each activity with a short name (usually a letter) to make things easier during the processing phase.

The third result is about defining the set of coherent traces (clusters) and calculating the similarity matrix using the dynamic programming technique [20]. In this context, the message graph matrix is recognized as a similarity matrix, which is then used for dynamically aligning the sequences. The novelty, here, is to introduce string similarity algorithm with process model visualization.

The fourth result aims at refining the process model control flow visualization. To that end, we will apply the progressive alignment method to each similarity matrix and its corresponding cluster to generate the aligned sequences. The added value over existing studies [23 and 24] is the application in visualizing process models and making it more informative.

There is one field of research that addresses a comparable problem, and it is user activity analysis. In this field, human-computer interaction is recorded to evaluate the user with respect to specific research interest. The approach is used in several areas such as e-commerce and online social networks research to create services like recommendation systems [25 and 26] or to analyze social behavior. There is no case of user activity analysis related to event logs using the visualization method in combination with string similarity algorithms. Available approaches are used in the academic digital environment as e-learning in [27], digital libraries in Shiri (2018) [28] and in the healthcare domain. These scientific studies have been developed between 2008 and 2020.

Firstly, the work of Diamantini in [29] treats event logs in the domain of health. This paper introduces the Behavioural PMg approach, which is used to identify significant sub-processes from unstructured ones. This approach is based on the use of hierarchical clustering algorithms.

Second, the work published in [30] uses extended methods, such as analysis sequence, to identify student processes in online courses. The objective of this work is to identify the most significant students' behaviors, to help professors, to adapt their teaching strategies to different student populations. Likewise, in [27] demonstrates how to leverage from PMg techniques in obtaining smart mobility and higher education. Third, the authors of [30] present approaches based on classification techniques, to compare process discovery algorithms. Fourth, the authors of [32] demonstrate the ability of PMg techniques to examine challenges that nonconfidential data poses against process discovery and conformance checking techniques.

These studies focus on analyzing recorded traces in the context of representing users' behaviors, where the execution represents many repeated or semi-similar traces. While the process visualization is a very particular approach. This demonstrates the originality of this approach that tackles precise process model representation and indeed, advanced analysis is applied, and more metrics are taken into consideration.

10 Conclusion

Traditional control-flow modeling and representation methods employ directed-graphs of nodes and edges (Petri Nets). With the increase in the number of activities involved in the information system, these approaches are prone to becoming elaborate and tedious in representation. The paper proposes a simplified visualization method, which depicts the control flow model in a simplified manner, including the display of parameters such as the activity list, its order, information benefit, and significant event. Our visualization method outperforms classical approaches in terms of the amount of information visually represented for a variety of activities.

Our method is based on a biological sequence alignment technique known as multiple-sequence alignment, which we applied to real-life process sequences. Though the discussed approach appears appropriate and efficient for control flow representation for a wide range of information system activities, it is limited by the number of available alphabets and the availability of event logs.

In future work, we intend to evaluate the capability of our approach with other real-world data sources such to develop PMg plug-ins for recognizing chaotic activities. Indeed, Future development in this field may result in the visualization of 3D models, where the results will be displayed to understand deeply

frequent processes.

Acknowledgments

This work is supported by the National Center for Scientific and Technical Research (CNRST) in Rabat, Morocco.

Funding: There is no funding provided to prepare the manuscript.

Conflicts of Interest: The author declares that there is no conflict of interest regarding the publication of this paper.

References

[1] Van der Aalst, W. (2016). *Process mining: data science in action.* 2016, Springer.

[2] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F.M., Marrella, A. (2018). Automated discovery of process models from event logs : Review and benchmark. *IEEE Transactions on Knowledge and Data Engineering,* 31(4), 686-705.

[3] Mans, R.S, Schonenberg, M.H, Song, M., van der Aalst, W., Bakker, P.J. (2008). *Application of process mining in healthcare–a case study in a dutch hospital.* In International joint conference on biomedical engineering systems and technologies, Springer.

[4] Pegoraro, M., Van der Aalst, W. (2019). *Mining uncertain event data in process mining.* In 2019 International Conference on Process Mining (ICPM). IEEE.

[5] Wen, L., Van Der Aalst, W., Wang, J., Sun, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery,* 15(2), 145-180.

[6] Weijters, A.J., Van der Aalst, W. (2003). Rediscovering workflow models from event-based data using little thumb. *Integrated Computer-Aided Engineering,* 10(2), 151-162.

[7] Bogarín, A, Cerezo R., Romero, C. (2018). Discovering learning processes using inductive miner: A case study with learning management systems (LMSs). *Psicothema,* 30(3), 322-329.

[8] Vanden Broucke, S.K., De Weerdt, J. (2017). Fodina: A robust and flexible heuristic process discovery technique. *Decision support systems,* 100, 109-118.

[9] Günther, C.W., Van Der Aalst, W. (2007).*Fuzzy mining–adaptive process simplification based on multi-perspective metrics.* International conference on business process management. Springer, Berlin, Heidelberg.

[10] Van der Werf, J. M. E., Van Dongen, B. F., Hurkens, C. A., Serebrenik, A. (2008). *Process discovery using integer linear programming.* International conference on applications and theory of petri nets. Springer.

[11] Buijs, J.C., Van Dongen, B., Van Der Aalst, W. (2012). *On the role of fitness, precision, generalization, and simplicity in process discovery.* On the Move to Meaningful Internet Systems Confederated International Conferences. Springer, Berlin, Heidelberg.

[12] Petri, C.A. (1962).*Kommunikation mit automaten.* Phd thesis, Institut fur Instrumentelle Mathematik, Universitat Bonn.

[13] Kalenkova, A.A., Van der Aalst, W., Lomazova, I.A., Rubin, V.A. (2017). Process mining using BPMN: relating event logs and process models. *Software & Systems Modeling,* 16(4), 1019-1048.

[14] Prasetya, D.D., Wibawa, A.P, Hirashima, T. (2018). The performance of text similarity algorithms. *International Journal of Advances in Intelligent Informatics,* 4(1), 63-69.

[15] Hunt, J.W., Szymanski, T.G. (1977). A fast algorithm for computing longest common subsequences. *Communications of the ACM,* 20(5), 350-353.

[16] Dumas, M., Van der Aalst, W., Hofstede, A.H.T. (2005). *Process-aware information systems: bridging people and software through process technology.* John Wiley & Sons.

[17] Higgins, D.G., Sharp, P.M. (1988).CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene,* 73(1), 237-244.

[18] Moreira, C., Haven, E., Sozzo, S., Wichert, A. (2018). Process mining with real world financial loan applications: Improving inference on incomplete event logs. *PLoS One,* 13(12).

[19] Srivastava, S. (2012). Process Mining Techniques for Detecting Fraud in Banks: A Study. *Turkish Journal of Computer and Mathematics Education,* 12, 3358-3375.

[20] Chan, W., Saharia, C., Hinton, G., Norouzi, M., Jaitly, N. (2020). Imputer: *Sequence modelling via imputation and dynamic programming.* International Conference on Machine Learning, PMLR, 1403-1413.

]

[21] Frey, B.J., Dueck, D. (2007). Clustering by passing messages between data points. *science,* 315(5814), 972-976.

[22] Behara, K.N., Bhaskar, A., Chung, E. (2020). A novel approach for the structural comparison of origin-destination matrices: Levenshtein distance. *Transportation Research Part C: Emerging Technologies,* 111, 513-530.

[23] Detro, S.P., Portela, E., Rocha, E.L., Panetto, H., Lezoche M. (2017). Configuring process variants through semantic reasoning in systems engineering. *International Council on Systems Engineering,* 20(4), 36-39.

[24] Toledo P.D., Joppien, C., Sesmero, M.P., Drews, P. (2019). *Mining Disease Courses across Organizations: A Methodology Based on Process Mining of Diagnosis Events Datasets.* The 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 354-357.

[25] Plumbaum, T., Stelter, T., Korth, A. (2009). *Semantic web usage mining: Using semantics to understand user intentions.* In International Conference on User Modeling, Adaptation, and Personalization, 391-396.

[26] Dang, A., Moh'd, A. ,Milios, E., Minghim, R. (2016). *What is in a rumour: Combined visual analysis of rumour flow and user activity.* In Proceedings of the 33rd Computer Graphics International,17-20.

[27] Jadrić, M., Ninčević Pašalić, I., Ćukušić, M. (2020). Process Mining Contributions to Discrete-event Simulation Modelling. Business Systems Research. *International Journal of the Society for Advancing Innovation and Research in Economy,* 11(2), 51-72.

[28] Shiri, A. (2018). *Methodological Considerations in Developing Cultural Heritage Digital Libraries: A Community-driven Framework.* In Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries. ACM.

[29] C Diamantini, C, Genga, L., Potena, D. (2016). Behavioral process mining for unstructured processes. (2016). *Journal of Intelligent Information Systems,* 47(1), 5-32.

[30] Liu, S., Hu, Z., Peng, X., Liu, Z., Cheng, H.N., Sun, J. (2017). Mining learning behavioral patterns of students by sequence analysis in cloud classroom.*International Journal of Distance Education Technologies,* 15(1), 15-27.

[31] Song, M., Günther, C.W, Van der Aalst, W. (2008). *Trace clustering in process mining.* In International conference on business process management, Springer.

[32] Pérez-Alfonso, D., Yzquierdo-Herrera, R., Lazo-Cortés, M. (2013). Recommendation of process discovery algorithms: a classification problem. *Research in Computing Science,* 61, 33-42.

[33] Berti, A., van Zelst, S.J, Van der Aals, W. (2019). *Process mining for python (PM4Py): bridging the gap between process-and data science.* arXiv preprint arXiv:1905.06169.

About the Author

Zineb LAMGHARI, Doctor in software engenireeing. received her technical university degree in software engineering at the Higher School of Technical Education, Mohammed V-Souissi University, Morocco, and her license degree in information systems management from polydisciplinary faculty then her master's degree in the computer system and network from the faculty of Technical Sciences, ABDELMALEK ESSAADI University, Morocco. She prepares her Ph.D. in business process improvement in the Computer Science and Telecommunication Research Laboratory (LRIT) at the faculty of sciences, Mohammed V University. Her interests cover mainly business process management, software engineering, and process mining techniques.