Mediterranean Smart Cities Conference (MSCC 2024)
2-4 May, 2024, Martil, Morocco

Optimizing City Services through Configurable Change Mining: Implementation of the Variability Change Tree Approach

1rd HMAMI Asmae
AlQualsadi research team
ENSIAS, University Mohammed V in Rabat
Rabat, Morocco
Asmae_hmami@um5.ac.ma
ORCID: 0000-0003-4562-9067

Abstract— **In providing services to citizens, understanding their behavior is crucial for effective adaptation. Often, the service provided to citizens is presented as a business process, which describes all the required actions that a citizen's request will go through until the citizen's needs are fulfilled. However, citizens and all individuals and systems interacting with the process may choose not to follow the prescribed actions in the process for various reasons, leading to the execution of tasks in different ways. Therefore, these changes in behavior should be taken into account to better adapt the process to their needs and provide improved services to citizens. In a previous work, we proposed a framework that can detect changes not only in a single business process but in a family of business processes issued from a configurable business process, and the detected changes are represented in a change tree. The goal of this paper is to show the applicability of our proposed approaches in the context of services provided to citizens and to implement the configurable process change tree, making the result of the configurable Change Mining more user-friendly for the end user.**

Keywords—component; Variability; Configurable business process; Change Mining; Change tree;

I. INTRODUCTION

Business processes are widely employed across various types of organizations, with the primary objective of achieving business goals that satisfy the client[1]. In an administrative context, the client is typically a citizen in need of a service [2]. This service is delivered through a series of carefully orchestrated steps that require resources, data, and strategic decision-making. To ensure efficiency, these steps are organized into a structured framework known as a business process, overseen by an administrative manager[1]. This process outlines the chronological sequence of activities related to a specific task, aimed at fulfilling the citizen's requirements.

This service can be provided on-site, where citizens visit the administration to perform tasks manually and interact face-to-face with employees, or it can be facilitated through internet platforms offering administrative services. However, in both cases, due to various circumstances, citizens may engage with the process and its activities differently, leading to potential variations in the execution of business processes compared to the initially organized structure by the business process manager. To effectively serve citizens, administrations must analyze the citizen behavior in relation to the process.

Additionally, citizens are not the alone actors influencing the process; other stakeholders can also impact its dynamics [3]. Furthermore, when various types of the same business process are executed, the challenges become more significant, and analyzing each business process separately can become a time-consuming task [4].

To address this issue, numerous efforts have been undertaken to deal with changes in business processes by using some business process reengineering [5], business process improvement [6], process mining approaches including methods such as Change Mining[7]–[9], conformance checking [8], [10], [11], and enhancement [12]. However, only a limited number have specifically tackled the challenge of business process variants, known as the process family or configurable process variants. Some notable works in this domain include change patten proposed by [13].

With the absence of a process mining solution for managing changes related to a family of business processes [12],[14], as proposed in our previous work [15], we introduce a Change Mining approach. This approach begins by gathering data from a set of event logs associated with a family of business processes. The approach includes preprocessing this data to generate the necessary input for the Change Mining algorithm, specifically designed to identify changes, especially in the variant components of the configurable process, and subsequently producing the change log [16],[15]. This solution is further enhanced by representing the changes in a structured manner, specifically as a variability change

tree[17]. In our current work, our objective is to implement this change tree and demonstrate its effectiveness in facilitating the detection and recognition of changes related to business processes, with a focus on those providing citizen services.

The structure of the remaining sections in this paper is outlined as follows: the second section provides background information. In the third section, there is a brief presentation of our variability Change Mining approach and the variability change trees designed to represent changes in configurable business processes. The fourth section focuses on presenting the implementation of the variability change tree, along with the corresponding results of the implementation. Finally, the fifth section serves as the conclusion of the paper.

II. BACKGROUND

A. Configurable business process

To define a configurable business process, we must first establish a clear understanding of two key terms: business process and variability.

1) Business process

In literature, the business process is defined in different ways. We can cite the most referenced definition, which is "a business process is a set of logically related tasks that, when executed, achieve a defined business outcome" [18]. Also it is define as "a network of connected activities, with well-defined boundaries and priority relationships, that use resources to transform inputs into outputs, in order to meet customer requirements" [19]. These two definitions highlight the multidimensional nature of business processes, emphasizing their role in achieving specific business objectives and satisfying customer needs.

2) Variability

The concept of variability is used in different domains, for example, in 'Product Line Engineering' the variability allows for product differentiation and diversification. It refers to the ability of an artifact to be configured, customized, extended, or modified for use in a specific context[20]. In the context of the business process, variability has a similar objective as in the Product Line, it refer to the diversity of variation of the manufacturing processes for producing the product variants in the product family[21].

In the specific case of business process engineering, variability is characterized by three essential elements[22]:

- Variation points: These are the elements that are likely to change.
- Variants: These are the options that each variation point can have.
- Configuration: This allows you to create a process variant from a configurable process by selecting a variant for each variation point. This choice has three options: choose, reject, or choose subject to a condition.

Furthermore, the variability concept helps providing a customizable process model[4], that is known as a configurable business process.

3) Configurable business process

The configurable business process is based on the concept of variability and the reusing existing business process[23] in order to create a references model which is a generic conceptual models that formalize recommended practices for a reference models that capture reusable state-of-the-art practices. It is motivated by «design by reuse » [24]. Thus, a configurable business process is an integrated representation of multiple process variants. [23].

In managing business processes, including configurable ones, Business Process Management (BPM) offers a set of approaches for identifying, designing, executing, documenting, measuring, monitoring, and controlling business processes[25]. An example of intelligent approach of Business Process Management is Process Mining, which utilizes data mining approaches to achieve specific BPM objectives.

B. Process mining and Change Mining

1) Process mining

Process mining is a discipline that bridges the gap between machine learning and data mining on the one hand, and process modeling on the other. It is a set of algorithms and techniques that aim to discover, improve, and verify business processes by analyzing process execution data recorded in event logs as traces [26].

The event logs play an import role in process mining, as all process mining algorithms take them as input [26]. In the early stages of process mining, well-known algorithms were primarily dedicated to discovery, verifying conformity, or enhancing business processes[27]. However, contemporary process mining encompasses various approaches, including those used for prediction and analyses and more[27]. Change Mining emerges as one such approach within process mining, specifically designed for analyzing and understanding changes in processes[9].

2) Change Mining

Change Mining is an approach that enables the detection and analysis of changes by using process mining approaches.

In the field of data science, Change Mining is defined as a data mining solution that allows the study of data associated with time. Its objective is the discovery, modeling, monitoring, and interpretation of changes in the models that describe an evolution [28].

In the context of business processes, Change Mining was first introduced by [8], and it refers to the application of process mining techniques on change logs, to discover the changes that the process has experienced during its execution. By the time Change Mining was made from event logs and approaches from machine learning in the context of business process.

However, changes mining in the context of configurable business process was not widely addressed. Thus we investigated in this research area and we proposed in previous work a changes mining approaches dedicated to configurable business process that deal with the variability changes in business process[15], [16].

While change mining is a valuable tool for business process management, it hasn't been extensively applied to configurable processes. To address this gap, our previous work [15] [16] introduced a novel change mining approach specifically designed for configurable business processes.

This approach deals with the challenge of identifying and understanding variability changes in configurable business processes.

In the following section, we summarize our Change Mining approaches and the variability change tree designed to represent the outcomes of the Change Mining approach.

III. VARIABILITY CHANGE MINING AND CHANGE TREE

A. Variability Change Mining

In our earlier work, we proposed a framework to detect changes in a family of business processes [15], [16]. This framework involves i) preprocessing event logs of processes of the same family in order to create an event log of variable fragments of the configurable process, ii) identifying changes from the created event logs, and the iii) recording the detected changes in a change log.

1) Preprocessing event logs

The suggested preprocessing involves selecting event logs to form a collection where each log corresponds to a process variant of the same configurable process. From this collection, a merged event log is created. Then, the event log is filtered to obtain an event log associated with the variable fragments of the configurable process[16].

The event logs include all events associated with the execution of elements constituting the variable fragments. Consequently, we can identify events aligned with the process, as well as those that do not align with the process, which is our focal point for change detection.

2) Detecting changes

To detect changes in the filtered event log, we used an incremental and supervised algorithm inspired by the STAGGER Algorithm [29]. Our algorithm identifies events associated with changes by using an updated version of the variability files which contain information about the variable fragments of the configurable process[30]. The result of this algorithm is a list of events related to changes.

3) Creating change logs

As the list of changes is not labeled and not easy to read and analyze, we generated, following rules adapted to the needs of Change Mining, an XML file called a change log that contained the detected changes in a well-structured form.

However, even if an XML document is well-formatted, it is a large file, making it challenging for non-experts to derive benefits from it. Therefore, we decided to represent the change log as a change tree.

B. Change tree

The main idea of the change tree is to organize changes in a way that makes it easy to detect changes related to each element of the variable fragment. The structure as a tree makes it easy to read and select the appropriate part. The benefit of this tree is that it organizes changes following their appearance in time, allowing us to identify how a change persists on an element and whether it is continuous or discontinuous.

In Figure 1, we can observe a graphic design of the tree structure. The root is the configurable process, connected to the variable fragments it contains. Each fragment is connected to its elements, and each element is connected to changes that affected that specific element.

The basic rules used to create the variability change tree are primarily based on the variability files. These files help us initially create a tree that represents the configurable process and its variable fragments (first part of Figure 1). For adding changes to each element, we analyze each instance in the change log chronologically (second part of Figure 1).

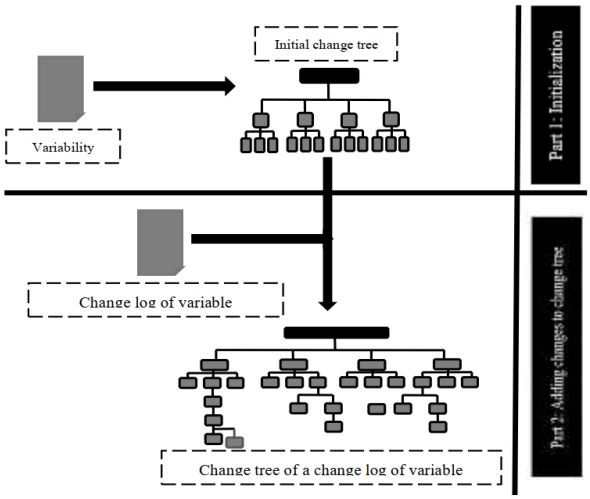

Figure 1: the variability change tree framework [17]

To verify the corresponding elements on the tree, we check if similar changes already exist, considering the following:

- The change instance will be added to the tree as a leaf connected directly to the concerned element if no similar changes occur before.
- The change instance will increment the leaf if a similar change instance is already connected to the change, and it is the last recorded change.

More specific situations demand particular rules, which are elaborated in[17].

In the next section, we will present our implementation of the variability change tree and we highlight how this representation can help in providing better service to citizen.

IV. IMPLEMENTING THE CHANGE TREE

To implement our change tree, we will follow four steps. The first step involves creating the initial change tree, representing fragments and elements that compose

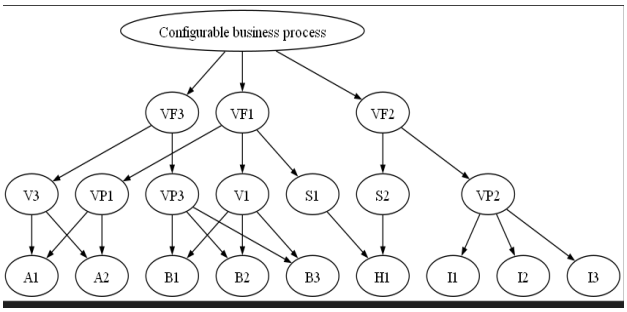

Figure 2 : An example of the obtained initial variability change tree.

Table 1: scripts used for creating the variability change tree

Steps	Scripts				
	Name	***Input***	***DESCRIPTION***	***Output***	***links***
1	step1-initial-change-tree	Variability file(VF)	Creating the intial variability change tree	Visual representation of the intial variability change tree	https://github.com/hasmae/Variability_Change_Tree/blob/main/Step1-initial-change-tree.ipynb
2	Step2-selection1_variable_fragment	Change log+Variability file	Converting the XML file into a data frame that contains changes related to a specific variable fragment	Data frame containing all changes that affect the selected element	https://github.com/hasmae/Variability_Change_Tree/blob/main/Step2-selection1_variable_fragment.ipynb
	Step(2-3)- selection2-specific-element_and grouping changes		Converting the XML file into a data frame that contains changes related to a specific element		https://github.com/hasmae/Variability_Change_Tree/blob/main/Step(2-3)-%20selection2-specific-element_and%20grouping%20changes.ipynb
	Step2-selection3-specific-element-and-role		Converting the XML file into a data frame that contains changes related to a specific element and role		https://github.com/hasmae/Variability_Change_Tree/blob/main/Step2-selection3-specific-element-and-role.ipynb
3	Step(2-3)- selection2-specific-element_and grouping changes	Data from the previous step	Grouping changes and assigning weights to them	A smaller data frame	https://github.com/hasmae/Variability_Change_Tree/blob/main/Step(2-3)-%20selection2-specific-element_and%20grouping%20changes.ipynb
4	Step4_adding_change_to_change_tree	Data frame with change instance and weight	Adding changes to the variability change tree	Visual representation of the variability change tree	https://github.com/hasmae/Variability_Change_Tree/blob/main/Step4_adding_change_to_change_tree.ipynb

the configurable business process. The second step includes selecting and formatting the data into a form that can be easily analyzed. In the third step, we will analyze the obtained data to collect information about changes affecting each element and we will give them a weight. Finally, in the fourth step, we will incorporate these changes into the initial change tree.

Each step will be implemented in one or more Python scripts utilizing functions from data science libraries. The proposed codes will respect rules described in the previous section and detailed in our paper [17].

All the scripts utilized in this paper are available on my GitHub, accessible through the following link[1]. Details regarding the scripts used for each step are outlined in Table 1.

1) Step 1: Generating the initial changes tree

To generate the initial variability change tree, we will use the updated version of the variability file [15] as input to create an organizational chart representation.

The data used in this step is obtained by transforming the content of the variability file, which is a CSV, into a data frame. From this data frame, we generate an organizational chart using the 'graphviz' library. Figure 2 shows an example of the obtained initial variability change tree. For more details about the codes used in these steps, refer to the script named 'step1-initial-change-tree'.

2) Step 2: Selecting data from the Change log

Our variability change tree is based on variable fragments and the elements that compose each variable fragment. Each variable fragment is associated with a variation point, a previous element, and a next element.

To identify changes that affect each element, we need to select from the change log the changes corresponding to the specific element. Thus, we will have different selection possibilities depending on the element for which we are looking at changes.

a) The selection based on the variable fragment.

In this case, we will use a script to extract changes from the change log for a specific variable fragment. The result will be a data frame containing only changes related to this specific variable fragment. This type of selection can be useful when we need information about only one variable fragment, and we don't require a global view of changes that affect other fragments. It will help in the next step to visualize only the needed part of the variability change tree. The corresponding code is in the first part of our Python script named 'Step2: selection1-variable-fragment'.

b) The selection based on a specific element.

When focusing on specific elements, it is important to highlight that a particular element can appear not only in one variable fragment but in many variable fragments with different roles pervious element, a next element, or a variation point. Thus, there are two possibilities:

- Selecting changes related to a specific element, regardless of its roles in variable fragments. The corresponding code is in the first part of our Python script named 'Step(2-3)- selection2-specific-element_and grouping changes'.
- Selecting changes related to a specific element by considering its role in the variable fragments. Information about the role of a specific element for a particular variable fragment can be found in the variability file. The corresponding code is in the first part of our Python script named 'Step2-selection3-specific-element-and-role'.

[1] https://github.com/hasmae/Variability_Change_Tree

The structure of the obtained data frame is Ix("...", "element", "change", "…") where Ix represents a change instance. The "..." corresponds to parameters needed for more detailed analyses, "element" corresponds to the elements involved in the selection, and "change" corresponds to changes that affect the element.

3) Step 3: Weighting of changes

In this step, we will try to summarize the results of the data selection made in the previous section by creating a smaller data frame that groups similar changes, following the guidelines outlined in the article for creating the variability change tree [17].

However, to provide a more general overview and address the most common scenario, we will concentrate only on the data generated for a specific element. The other selections cases are considered specific cases of this more general one.

As the detected changes must be added as a leaf or they will increment a leaf on the variability changes tree, we need to assign a weight to each change instance. This weight is initially set to 1, and each time the algorithm finds a similar change, it will increment the weight, except if the change is not continuous.

To illustrate the Weighting step, let's take an example of the obtained data frame for the element A1:I1('…', A1, W, '…') I2('…', A1, W,'…') I3('…', A1, W, '…') I4('…', A1, Z, '…') I5('…', A1, Z, '…') I6('…', A1, W, '…').

The change instance I1 change element A1 to W and a similar change appears three times consecutively, so the weight of the change W will be 3 (W(3)), and the change Z appears two times consecutively (Z(2)), and finally, the change W reappears (W(1)). The result for element A1 A would be: W(3), Z(2), W(1).

4) Step 4 : adding changes to the initial changes tree

The aim of this final step is to add the detected changes for each element into the change tree. Since the code provided in the previous section generates a data frame of

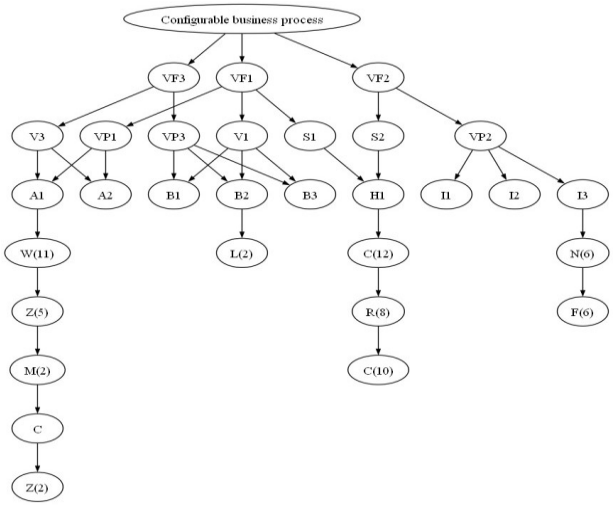

Figure 3: An example of the obtained variability change tree

weighted changes for a specific element, to obtain changes related to all elements, we will iterate through elements on the variability file and for each element we will apply the same code. This process results in creating a data frame for each element. Finally, we add lines of each data frame as consecutive leaves to the change tree.

Figure 3 is an example of the obtained variability change tree. As we can observe, this tree is not extensive. However, due to its complexity, it requires more concentration to extract information. For this reason, we implemented different types of selection in Step 2 to generate, if needed, only a specific part of the variability change tree.

As we can observe, the implementation of the variability change tree, along with its capabilities to provide graphical visualizations of variability change management results, can offer a clear understanding of how the process is executed in the real world. Specifically, in the context of services provided to citizens, incorporating this change tree visualization can assist managers in identifying points where citizen needs may not be fulfilled as initially described during the creation of the business process. An in-depth investigation into the variability change tree allows us to detect the impact of a change instance on other elements in the process. Thus, if a modification is recommended for the business process, an initial understanding of the potential impact on other elements can be obtained.

V. CONCLUSION

In this paper, we propose an implementation of the variability change tree using the Python programming language and some data science functions.

The goal is to provide an implementation of our approach that detects changes occurring during the execution of process variants, especially for business processes that offer services to citizens. This implementation highlights the capability to apply our Change Mining approach in the context of citizen services.

Our implementation is an open-source code available on GitHub for various uses or updates.

In future work, our aim is to enhance our implementation by adding more functions that facilitate the analysis of changes and provide recommendations for future business process enhancement.

REFERENCES

[1] R. G. Lee and B. G. Dale, "Business process management: a review and evaluation," *Bus. Process Manag. J.*, vol. 4, no. 3, pp. 214–225, 1998.

[2] B. Laila, "La transformation digitale des banques au Maroc Digital transformation of banks in Morocco," vol. 3, no. 2021, pp. 27–38, 2022.

[3] W. Song and H. A. Jacobsen, "Static and Dynamic Process Change," *IEEE Trans. Serv. Comput.*, vol. 11, no. 1, pp. 215–231, 2018, doi: 10.1109/TSC.2016.2536025.

[4] L. Rosa *et al.*, "Business Process Variability Modeling: A Survey," *BodyNets Int. Conf. Body Area Networks*, 2017, doi: 10.1145/0000000.0000000.

[5] V. Grover, S. R. Jeong, W. J. Kettinger, and J. T. C. Teng, "The implementation of business process reengineering," *J. Manag. Inf. Syst.*, vol. 12, no. 1, pp. 109–144, 1995, doi: 10.1080/07421222.1995.11518072.

[6] C. Management, B. P. Management, B. P. Management, and B. P. Management, *13 Business Process Improvement*. 2010.

[7] M. Maisenbacher and M. Weidlich, "Handling Concept Drift in Predictive Process Monitoring," *Proc. - 2017 IEEE 14th Int. Conf. Serv. Comput. SCC 2017*, pp. 1–8, 2017, doi: 10.1109/SCC.2017.10.

[8] C. W. Günther, S. Rinderle, M. Reichert, and W. Van Der

Aalst, “Change mining in adaptive process management systems,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4275 LNCS, pp. 309–326, 2006, doi: 10.1007/11914853_19.

[9] B. F. A. Hompes, J. C. A. M. Buijs, W. M. P. Van Der Aalst, P. M. Dixit, and J. Buurman, “Detecting change in processes using comparative trace clustering,” *CEUR Workshop Proc.*, vol. 1527, pp. 95–108, 2015.

[10] A. Adriansyah, B. F. Van Dongen, and W. M. P. Van Der Aalst, “Conformance checking using cost-based fitness analysis,” *Proc. - IEEE Int. Enterp. Distrib. Object Comput. Work. EDOC*, pp. 55–64, 2011, doi: 10.1109/EDOC.2011.12.

[11] D. Borrego and I. Barba, “Conformance checking and diagnosis for declarative business process models in data-aware scenarios,” *Expert Syst. Appl.*, vol. 41, no. 11, pp. 5340–5352, 2014, doi: 10.1016/j.eswa.2014.03.010.

[12] F. A. Yasmin, F. A. Bukhsh, and P. De Alencar Silva, “Process enhancement in process mining: A literature review,” *CEUR Workshop Proc.*, vol. 2270, pp. 65–72, 2018.

[13] H. Sbai, M. Fredj, and L. Kjiri, “Towards a process patterns based approach for promoting adaptability in configurable process models,” *ICEIS 2013 - Proc. 15th Int. Conf. Enterp. Inf. Syst.*, vol. 3, pp. 382–387, 2013, doi: 10.5220/0004565503820387.

[14] A. Hmami, H. Sbai, and M. Fredj, “Change Mining in Business Process Variability: A Comparative Study,” 2019, doi: 10.1109/ICoCS.2019.8930792.

[15] A. Hmami, M. Fredj, and H. Sbai, “Handling Sudden and Recurrent Changes in Business Process Variability: Change Mining based Approach,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 12, no. 4, pp. 632–640, 2021, doi: 10.14569/IJACSA.2021.0120479.

[16] A. Hmami, H. Sbai, and M. Fredj, “Enhancing change mining from a collection of event logs: Merging and Filtering approaches,” *J. Phys. Conf. Ser.*, vol. 1743, no. 1, 2021, doi: 10.1088/1742-6596/1743/1/012020.

[17] A. HMAMI, H. SBAI, and M. FREDJ, “Applying change tree method in configurable process mining,” 2022.

[18] T. H. Davenport, *Process innovation: reengineering work through information technology*. 1993.

[19] M. Laguna and J. Marklund, *Business process modeling, simulation and design*. Chapman and Hall/CRC, 2018.

[20] J. Bosch, J. Bosch, M. Svahnberg, J. Bosch, and J. Bosch, “Variability Issues in So ware Product Lines Related papers,” 2001.

[21] B. Chakir and M. Fredj, “V3R: un framework pour la gestion de la variabilité de services SV3R: a Framework for Services Variability Management.,” 2014.

[22] L. El Faquih, H. Sbai, and M. Fredj, “Semantic variability modeling in business processes: A comparative study,” *2014 9th Int. Conf. Internet Technol. Secur. Trans. ICITST 2014*, pp. 131–136, 2014, doi: 10.1109/ICITST.2014.7038792.

[23] M. La Rosa, “Managing Variability in Process-Aware Information Systems,” *Management*, no. March, pp. 1–186, 2009, [Online]. Available: eprints.qut.edu.au/20531/1/Marcello_La_Rosa_Thesis.pdf.

[24] M. Rosemann and W. M. P. van der Aalst, “A configurable reference modelling language,” *Inf. Syst.*, vol. 32, no. 1, pp. 1–23, 2007, doi: 10.1016/j.is.2005.05.003.

[25] W. M. P. Van Der Aalst, “Business process management: A personal view,” *Bus. Process Manag. J.*, vol. 10, no. 2, 2004, doi: 10.1108/bpmj.2004.15710baa.001.

[26] W. Van Der Aalst *et al.*, “Process mining manifesto,” *Lect. Notes Bus. Inf. Process.*, vol. 99 LNBIP, no. PART 1, pp. 169–194, 2012, doi: 10.1007/978-3-642-28108-2_19.

[27] W. Van der Aalst, *Process mining: Data science in action*, no. April 2014. 2016.

[28] M. Böttcher, F. Höppner, and M. Spiliopoulou, “On exploiting the power of time in data mining,” *ACM SIGKDD Explor. Newsl.*, vol. 10, no. 2, pp. 3–11, 2008, doi: 10.1145/1540276.1540278.

[29] J. C. Schlimmer and R. Granger, “Beyond incremental processing: Tracking concept drift,” *Proc. Fifth Natl. Conf. Artif. Intell.*, vol. 1, pp. 502–507, 1986.

[30] R. Sikal, H. Sbai, and L. Kjiri, “Configurable Process Mining: Variability Discovery Approach,” *Colloq. Inf. Sci. Technol. Cist*, vol. 2018-Octob, pp. 137–142, 2018, doi: 10.1109/CIST.2018.8596526.