Enhancing change mining from a collection of event logs: Merging and Filtering approaches

A Hmami\textsuperscript{1}, H Sbai\textsuperscript{2} and M Fredj\textsuperscript{1}

\textsuperscript{1}AlQualsadi research team, ENSIAS, University Mohammed V of Rabat, Morocco
\textsuperscript{2}Faculty of sciences and technology, University Hassan II of Casablanca, Mohammedia, Morocco
E-mail: hmami.asmae@gmail.com, hanae.sbai@univh2c.ma, mounia.fredj@um5.ac.ma

Abstract. An event log is the key element of all change mining and process mining approaches. Those approaches bridge the gap between conventional business process management and data analysis techniques such as machine learning and data mining. In this day, companies and business organizations usually use a family of business processes that may face different variations and adjustments. Still, those processes are widely identical, with a slight difference in specific points. Consequently, performing a process mining or a change mining for each process will be a redundant task. The use of a configurable process model is a practical solution for redundancy problem. Thus, the process mining areas such as discovering verifying the conformity of a business process and enhancing processes, are reduced considerably. However, the configurable process models and the variability concept are rarely introduced in change mining approaches. The existing methods that analyse and manage event logs do not then consider the variability issue. Therefore, the fact of using a collection of event log becomes a challenging task. Our proposed approach is to merge and filter a collection of event logs from the same family with respect to variability. Our goal is to enhance change mining from a collection of event logs and detect changes in variable fragments of the obtained event log.

Key words: Business process, variability, configurable process, collection of event logs, Filtering, Merging, variable fragments.

1. Introduction

In a competitive and rapidly changing environment, companies should be able to track changes quickly and efficiently with a minimum effort required, because the next decade is likely to see more dynamic changes. In the business process context, mining techniques are used to improve change management in business processes by learning changes from the past and recommending the best practices for future evolution. However, there are several copies of similar processes that are changed manually, which is an error-prone and time-consuming task. Against this background, with respect to variability, reuse in process mining is introduced through configurable process mining \cite{1}\cite{2}\cite{3}. The main idea is to use mining techniques to discover configurable processes directly from a collection of event logs. This can be achieved by discovering configurable fragments \cite{3}\cite{4}.

As business processes change over time to meet new needs, changes in configurable process mining should also be manage. However, existing change mining techniques \cite{5}\cite{6}\cite{7} are not suitable to be applied to a collection of event logs, because they do not support the variability concept. Furthermore, processes are mined separately, which is a redundant task. We aim then to manage a collection of
event logs by using the variability concept to create a global event log for future analysis as the first step towards change mining in configurable processes. Accordingly, time of event logs processing will be reduced, and mining algorithm is applied on variable parts, while changes in common parts can be discovered by ordinary techniques [5][6][7]. To this end, we propose a merging and filtering algorithms to generate an event log related to a collection of business process variants.

In order to implement our proposed approach, we develop a new tool that prepares a test environment with a random configurable process model and that generates process variants as well as variability specification files. The idea of developing a random configuration process model generator, is inspired by existing toolsets [8]. Our tool also implements a new merging and filtering algorithm.

The remainder of this paper is organized as follows. Section 2 includes an overview of the existing work related to managing event log and change mining in the configurable process model. In Section 3, we present our merging and filtering approaches. Section 4 presents the simulation environment and tests. Finally, Section 5 concludes the article.

2. Related work
Managing configurable process models has been widely discussed in the field of business process mining, especially in process discovery. Many research works have been proposed to deal with different issues when managing a configurable process model. For instance, we find work on variability discovery in processes [1][2][3][4], configuration discovery [9][10] and mining deviation of configurable processes [11]. In this paper, we are concerned about research related to change mining in configurable process models, especially in managing and analysing a collection of event logs to be processed and mined to extract changes that can be recommended to designers. To perform a change mining in business processes, we need a well-constructed event log. When talking about the family of business process, we need to deal with a collection of event logs. Therefore, many authors have dealt with event logs in order to remove unnecessary information, correct errors, and clean events especially in change mining works. Works on managing event logs in change mining area can be divided into two main categories:

- Single event log based approaches.
- Multiple event logs based approaches.

2.1. Single event log based approaches
Change mining approaches such as clustering [7], change tree [6] or windowing approach [5] are based on analysing a single event log. The correctness of this analysis and the perfectness of their result is ensured using a well-constructed event log. In order to create well-constructed event logs, researches proposed different approaches and patterns to enhance the quality of event log generated by information systems. Some researches focused on cleaning noises of event logs [12]. Others focus on completing traces by adding the needed event in order to create a complete trace that has a starting and ending point [13][14], all those approaches are dealing with one event log. However, as it is mentioned in the introduction, companies use many business processes. Thus, the concerns have arisen on dealing with a collection of event logs.

2.2. Multiple event logs based approaches
Using a collection of event logs was initiated with the concept of cross-organization. This concept uses a collection of business processes to improve applying mining techniques in the business process, the author of [15] proposes to compare event logs of business processes from the same family to obtain possible improvements. Authors in [1] summarize four different approaches for discovering a configurable process model. These approaches take as input a collection of event logs and use either merging or individual analysis to discover the model. Authors in [3], propose a procedure to extract variable fragments from a collection of event logs. Moreover, they do not discuss any managing strategy for the collection of event logs. More recently, they propose [16] a recommendation system which is based on users’ needs. In other work, some rules are developed to construct an event log from
events collected from complementary systems [17]. These works use a collection of event logs to discover or enhance business process models. Despite the interest of managing and analysing event logs and from the best of our knowledge, these works do not deal with change mining with respect to variability and they do not use a collection of event logs.

3. Our proposed approach for managing a collection of event logs related to a configurable process

To overcome the complexity of using a collection of event logs, and at the same time benefit from the huge amount of data, we propose a new merging and filtering approach Figure 1. The purpose is to construct an input for a change mining algorithm that can handle a collection of event logs. As it is described in figure 1 our proposed approach is based on three steps:

![Figure 1. The proposed framework for Managing a collection of event logs.](image)

- The first step: concerns selecting event logs, which have similar parts with some differences.
- The second step: an ID is added to each event log in order to identify the relevance of traces, and all event logs are merged in one.
- The third step uses the variability specification file to filter events related to variation points. Finally, we obtain a sub-part of the merging event log that contains only events related to variable fragments, we called the obtained file ‘event log of variable fragments’.

To achieve all steps, our approach is based on three main concepts that we define in the following sections, such as event log, configurable process and variability specification file.

3.1. Preliminaries

3.1.1. Event log representation. An event log is a file where events are recorded. It is a set of events grouped on traces each trace is a complete execution of the business process. In order to represent an event log, several representations exist. The simplest one is the CSV file. The Standard representation is an XES (eXtensible Event Stream) adopted in 2010 [18]. The XES format is based on Markup language, and it is the most used representation for process and change mining. A more basic representation is the MXML (Mining eXtensible Markup Language) [19]. The mapping of one representation to another can be made by many existing tools [18]. Thus, in order to simplify the implementation of our solution, we choose the CSV representation and in case if a transformation to an XES representation is required, we will use the CVStoXES plugin implemented in prom [18].
3.1.2. Configurable process model (CPM). To represent a configurable process model, we need to specify the variation point and their variants. We use the symbols inspired by BPMN and C-BPMN [20]. We selected only three Gateways: AND, OR and variation points. The other types of gateways are out of the scope of this paper. Table 1 presents the list of used gateways with their representation. For each gateway, we present the adopted symbol, the acronym and an example of the use of the symbol on a model.

| Symbol        | The used name in the simulated tool | Example         |
|---------------|-------------------------------------|-----------------|
| And           | AndActy1Acty2                       | AndA1A2         |
| Or            | OrActy1Acty2                        | OrA1A2          |
| Variation     | PV_Acty1Acty2                       | PV_A1A2A3       |
| Connector     | ---                                 | ---             |

3.1.3. Variability specification file. Based on the representation chosen in [21], we assume that the variability specification file exists, and it is created with the configurable process model. In order to perform our filtering algorithm, we add two properties to the variability specification file that represent activities directly connected to the variation points. Table 2 lists the properties of the XML variability specification file. We are starting by the standard ones, variation point and variants. Followed by the updated properties, Next and Previous.

| Properties       | Syntax | Model                                      | Syntax           |
|------------------|--------|--------------------------------------------|------------------|
| Variation point  | PV_Ni  | PV_is the acronym used to define a variation point followed by the list of variant | PV_actyMactyC   |
| Variant          | X      | Where X is an activity                     | ActyM            |
| Previous         | P_Type_Acty | The P_is the acronym for the previous activity followed by the type (OR, And, PV, S) of the activity | P_And_A1A2       |
| Next             | N_Type_Acty | The N_is the acronym for the Next activity followed by the type (OR, And, PV, S) of the activity | N_Or_A1A2       |

3.2. The merging and the filtering approaches
Our approach builds an event log that group traces related to variable fragments from different process variants. To obtain the desired event log, we propose a merging algorithm followed by an updated version of the filtering algorithm presented in [22].

3.2.1. The merging algorithm. The merging algorithm presented in Figure 2, loops through every event log and assign a unique identifier to each trace to keep the trace related to its event log. All those traces are going to be merged in one event log.

### Algorithm 1: Merging algorithm

```plaintext
01 Var X integer //while x is the number of event log to be merged
02 For i ← 1 to X
03 Var M Array // While Event_log_i is the i’th Event log and M an Array
04 Var dimM <- dim(M) //Where Dim Is the function that return the dimension of the event log
05 For j ← 1 to dimM do // This loop goes through each trace of the i’th event log.
06 Var event <- M[j] // The element M[j] is an event from the event log
07 Var New_event <- Add_identifier(event, i) // the function Add_identifier add an id i to the event as a property
08 Var Y Array // while Y is a variation point.
09 Var key <- 1 integer
10 For i <- 1 to m do // This loop goes through each event.
11   X <- M[i]
12   For j <- 1 to n do
13     Y <- M[j]
14     if (comp_v(M[i], N[j])==1) // the function comp_v verifies if M[j] is in N[i].
15       var frag <- fragments(M[i-1], M[i], M[i+1])
16       var frag_with_key <- add_key(frag, key)
17       store(frag_with_key)
18     End if
19   End for
20 End for
```

**Figure 2.** The merging algorithm.

3.2.2. The Filtering algorithm. As it is presented in previous work [22] the filtering algorithm Figure 3 selects from the merged event log variable fragments. The selection is made by following two steps: i) the first one selects all the activities directly connected to variation. ii) And the second one selects a couple of dependencies. All those selections are made by comparing the merging event log with the properties of the variability specification files. And once the fragment is selected, a key is added to its traces to label fragments.

### Algorithm 2: Filtering algorithm (Updated)

```plaintext
01 Var M, N Array //while N and M are the Merged event log and the variability file respectively
02 Var m, n integer // while m and n are the dimension of M and N respectively
03 Var X Array // while X is an event from N
04 Var Y Array // while Y is a variation point.
05 Var key <- 1 integer
06 For i <- 1 to m do // This loop goes through each event.
07   X <- M[i]
08   For j <- 1 to n do
09     Y <- M[j]
10     if (comp_v(M[i], N[j])==1) // the function comp_v verifies if M[j] is in N[i].
11       var frag <- fragments(M[i-1], M[i], M[i+1])
12       var frag_with_key <- add_key(frag, key)
13       key <- key++
14     End if
15     ElseIf (comp_d(M[i-1], N[j])==1) or (comp_d(M[i+1], N[j])==1))
16     var frag <- fragments(M[i-1], M[i], M[i+1])
17     var frag_id <- add_key(frag, key)
18     key <- key++
19     store(frag_id)
20   End elseIf
21 End for
22 End for
```

**Figure 3.** The Filtering algorithm.

4. The simulation environment tool and the approach test
In order to implement our proposed approach, we need an environment to generate a configurable process model, their variants, the variability specification file and generate event logs for process variants and finally, apply our prosed approach.

The process and logs generator toolset PLG [23] is able to generate randomly business process and their event logs and the extension of PLG proposed in [8] have added the possibility to generate randomly process variant of a configurable process model. However, we need in our environment test the possibility to generate the variability specification file. Consequently, and based on the previous existing random toolset we create our own toolset.

4.1. Our proposed tool: Random configurable process model (CPM) generator

In order to prepare our test environment, we create the Random configurable process generator tool. The representation of the most functionalities offered by our proposed tool is made by a use case presented in Figure 4.

- Generating a random configurable process model: the user can choose the number of variation point (N different from zero) and the number of variants for each variation point(n(i) should be greater than one).

- Creating the variability specification file: the tool automatically generates a variability specification file based on the description presented in Table 2.

- Generating process variants: for each configurable process model, we have P possibilities: \( P = \prod_{i=1}^{N} n(i) \) while N, a strictly positive integer, is the number of process variants. And n(i) the total number of variants for variation point (i). The user chooses the desired number of process variants and randomly, the tool selects process variants from all existing possibilities.

- Generating event log: for each process variant, we can generate a random event log with the desired number of traces.

- Merging event log: the collection of event logs can be merged automatically in one event log using the algorithm presented in Figure 2.

\[
\text{Figure 4. Use case of the Random configurable process generator tool.}
\]
• Filtering: the proposed filtering algorithm Figure 3 is applied in order to get the variable fragments event log from the collection of event logs.

4.2. Graphical user interfaces and implementation scenarios
In this section, we will simulate our proposed merging and filtering algorithm on random configurable process generated by our tool Figure 5. The user interface Figure 5 gives the user the possibility to switch between the possible functionalities available on the tool.

The workspace is divided into three main sections. The first section is a table where the generated models will be listed. The second part is the drawing space, where models can be viewed on graphic representation. The third section is dedicated to event logs.

4.2.1. The configurable process model, process variants, and the variability file. In order to generate a configurable process model, we choose a simple model with one variation point and three variants. The configurable process model is presented in Figure 6.

The possible process variants, in this case, will be three possibilities. From these three possibilities, we select only two process variants in this simulation. Figure 6 shows the two process variants.

• The syntax of the first process variant is [1, And_A1A2, OR_B1B2, Single_C1, Single_D1, PV_E2, Single_F1, And_G1G2, And_H1H2].

Figure 5. User interface of the Random configurable process model Generator tool.

Figure 6. The configurable process model and its process variants.
The syntax of the second process variant is \([2, \text{And}_A1A2, \text{OR}_B1B2, \text{Single}_C1, \text{Single}_D1, \text{PV}_E1, \text{Single}_F1, \text{And}_G1G2, \text{And}_H1H2]\).

With the same tool, we generate the variability specification file obtained in the form of a CSV file, while we have one variation point, the CSV file contains one line. The Figure 7a is the CSV and the XML representation respectively.

### 4.2.2 Generating and merging event log

For each process variant, we can generate an event log with random events. The Figure 8. (a) and (b) are subparts of event log of process variant 1 and process variant 2.

After generating event log, we can call the merging function in order to merge event logs. The function adds an ID for each event log before merging them, Figure 8 (c) is an example of the merging event log.

### 4.2.2 Filtering variable events from the merged event log

In this step, the filtering algorithm provided in Figure 3 is performed by our tool, and we get as a result the list of events related to variable fragments.
The file in Figure 9 is the event log of variable fragments which contains all events related to variable fragments. In each line, we have two keys: the first key is a key that identifies the process variants, and the second key is related to the variable fragment.

5. Conclusion
In this paper, we present a merging and filtering approach. This approach intends to create an event log of variable fragments from a collection of event logs. The motivation behind constructing such event log is the need to perform a change mining in a collection of event logs by using a change mining algorithm that uses only one event log as input.

The proposed approach is based on the variability specification file that contains information about variation points and variants. By using this file, we can select variable parts from the event log.

In order to test the proposed approach, we developed a tool, especially to prepare the input of the merging and filtering approach. Therefore, this tool can be used as a test environment for any other approach. The concept of the tool is to generate a random configurable process model, process variants, event logs, and variability specification file.

After the simulation, the obtained result shows that the event log of variable fragments is a subsection of the merging event log that contains only events related to variation points. In this approach, only the control flow perspective is considered, but potential improvement can be made in order to take more perspectives such as data and resources.

As a future work, we aim to test our approach on real data and use the event log of variable fragments in a change mining algorithm to get changes related to variability.

References
[1] Buijs J, van Dongen B and van der Aalst 2013 Mining configurable process models from collections of event logs Business process management, Springer, Berlin, Heidelberg 33–48.
[2] Li C, Reichert M and Wombacher A 2010 The Minadept Clustering Approach for Discovering Reference Process Models out of Process Variants International Journal of Cooperative Information Systems 19(03n04) 159–203.
[3] Pourmasoumi, A, Kahani M and Bagheri E 2017 Mining variable fragments from process event logs. Information Systems Frontiers, 19(6) 1423–43.
[4] Bolt A, van der Aalst W and De Leoni M 2017 Finding process variants in event logs, OTM Confederated International Conferences” On the Move to Meaningful Internet Systems 45–52.
[5] Martjushev J, Bose R, and van der Aals W 2015 Change Point Detection and Dealing with Gradual and Multi-order Dynamics Process Mining, Perspectives in Business Informatics Research: 14th International Conference, BIR 2015, Tartu, Estonia 161.
[6] Kaes G and Rinderle-Ma S 2015 Mining and querying process change information based on change trees International Conference on Service-Oriented Computing 269–84.

[7] Hompes B, Buijs J, van der Aalst W, Dixit P and Buurman H 2015 Detecting Change in Processes Using Comparative Trace Clustering, SIMPDA 95–108.

[8] Pourmasoumi A, Kahani M, Bagheri E and Asadi, M 2015 On Business Process Variants Generation, CAiSE Forum 179–88.

[9] La Rosa M, Dumas M, and ter Hofstede 2009 Modeling Business Process Variability for Design-Time Configuration Journal 204–28.

[10] Hallerbach A, Bauer T and Reichert M 2010 Configuration and management of process variants, Handbook on Business Process Management 1 237–55.

[11] Arriagada-Benitez M, Sepulveda M, Munoz-Gama J and Buijs J 2017 Strategies to automatically derive a process model from a configurable process model based on event data, Applied sciences, 7(10) 1023

[12] Suriadi S, Andrews R, ter Hofstede A and Wynn M 2017 Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs Information Systems, 64 132–150.

[13] Günther C and van der Aalst W 2006 A generic import framework for process event logs International Conference on Business Process Management, 81–92.

[14] Perez-Castillo R, Weber B, Pinggera J, Zugal S, de Guzman I and Piattini M. 2011 Generating event logs from non-process-aware systems enabling business process mining Enterprise Information Systems, 5(3) 301–335.

[15] Buijs J, van Dongen B and van der Aalst W 2011 Towards cross-organizational process mining in collections of process models and their executions International Conference on Business Process Management 2–13.

[16] Pourmasoumi A, Kahani M and Bagheri E 2019 The evolutionary composition of desirable execution traces from event logs, Future Generation Computer Systems 78–103.

[17] Claes J and Poels G 2014 Merging event logs for process mining: A rule based merging method and rule suggestion algorithm Expert Systems with Applications, 7291–7306.

[18] Verbeek H, Buijs J, Van Dongen B and Van Der Aalst W 2010 XES, XESame, and ProM 6, International Conference on Advanced Information Systems Engineering CAiSE 60–75.

[19] van Dongen B, and Van der Aalst W 2005 Meta Model for Process Mining Data, EMOI-INTEROP 160 30.

[20] Zhang H, Han W and Ouyang C 2014 Extending BPMN for Configurable Process Modeling, ISPE, 317–30.

[21] Sikal R, Sbai H and Kjiri L 2018 Configurable process mining: variability Discovery Approach 2018 IEEE 5th International Congress on Information Science and Technology (CiSt) , 137–142.

[22] Hmami A, Sbai H and Fredj M 2020 A new Framework to improve Change Mining in Configurable Process Proceedings of the 3rd International Conference on Networking, Information Systems Security 1–6.

[23] Burattin A and Sperduti A 2010 PLG: A framework for the generation of business process models and their execution logs International Conference on Business Process Management, 214–19.