# Mining Process Variants: Goals and Issues

Chen Li
University of Twente
The Netherlands
lic@cs.utwente.nl

Manfred Reichert
University of Ulm
Germany
manfred.reichert@uni-ulm.de

Andreas Wombacher
University of Twente
The Netherlands
a.wombacher@utwente.nl

## Abstract

*Recently, Process-Aware Information Systems (PAIS) were introduced, which allow for dynamic process and service changes. This, in turn, has led to a large number of process model variants, which are difficult to maintain and expensive to configure. This paper deals with goals and issues related to the mining of process model variants. Our overall goal is to learn from process changes and to "merge" the resulting model variants into a generic process model in the best possible way. By adopting this generic process model in the PAIS, future cost of process change and need for process adaptations will decrease.*

## 1 Introduction

In today's dynamic business world success of an enterprise increasingly depends on its ability to react to internal and external changes in a quick and flexible way. In response to this need, Process-Aware Information Systems (PAIS) emerged, which support the modeling, orchestration and monitoring of business processes. Recently, a new generation of flexible PAIS (e.g. ADEPT2 [2]) was introduced, which additionally allows for dynamic process and service changes (i.e., to insert, delete or move activities).

The ability to adapt processes at different levels, results in collections of process model variants (process variants for short) created from the same process model, but slightly differing from each other. Fig. 1 depicts an example. Fig.1a shows a high-level view on a patient treatment process as it is normally executed: a patient is $admitted$ to hospital, where he first $registers$, then $receives\ treatment$, and finally $pays$. In emergency situations, however, it might become necessary to deviate from this model; e.g., by first starting treatment of the patient and allowing him to register later during treatment. To capture this behavior of the respective process instance, we need to move activity $receive\ treatment$ to a position parallel to activity $register$. This leads to an instance-specific process variant $S'$ as shown in Fig. 1b. Generally, a large number of process variants derived from the same original process model might exist [8].

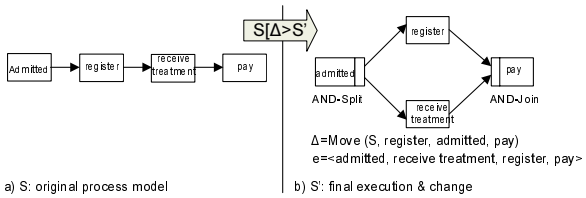


**Figure 1. Example of a process variant**

Such process variants are expensive to configure and difficult to maintain.

An interesting approach is provided by process mining [4]. Its key idea is to discover a process model by analyzing the execution log of the process instances. However, only little work considered *process variant mining* so far. In this paper we deal with related issues. Our goal is to learn from process changes applied in the past and to "merge" the resulting process variants into a generic process model which covers these variants best. By adopting this generic process model within the PAIS, cost of change and need for future process adaptations will decrease. This paper deals with the following research questions:

*Why is process variant mining needed and what are the differences between traditional process mining and the mining of process variants?*

We motivate the need for mining process variants and discuss some major challenges arising in this context. The mining algorithm itself is out of the scope of this paper (see [9]). We have developed respective techniques and utilized them for comparing variant mining with conventional process mining. Sec. 2 gives background information. In Sec. 3 we discuss why process changes should be expressed in terms of high-level change operations. Sec. 4 discusses major goals of process variant mining and shows why it is different from traditional process mining. Sec. 5 discusses related work. We conclude with a summary in Sec. 6.

## 2 Backgrounds

We first introduce basic notions needed in the following.

**Process Model**. Let $\mathcal{P}$ denote the set of all process models. A *process model* $S = (N, E, \ldots) \in \mathcal{P}$ is block-structured, where $N$ is the set of activities and $E$ constitutes

the set of precedence relations between them [2].

**Process Change**. A process variant results from an original process model $S$ by applying a sequence of changes to it [2]. Such changes modify initial model $S$ by altering its set of activities or by changing their order relations. Thus, each change $\Delta$ of a process model $S$ consists of a sequence of change operations and results in another process model $S'$, denoted as $S[\Delta\rangle S'$. Examples of high-level change operations include *insert*, *delete*, and *move activity* as implemented in the ADEPT change framework [2]; e.g., $move(S, \texttt{A}, \texttt{B}, \texttt{C})$ means to move activity A from its current position within $S$ to the position after B and before C, while operation $delete(S, \texttt{A})$ deletes A from $S$. Issues concerning the correct use of these operations are described in [2]. Though the depicted change operations are discussed in relation to ADEPT, they are generic in the sense that they can be easily applied in connection with other process meta models (e.g., Petri Nets) as well [1]. We refer to ADEPT since it covers by far most high-level change patterns when compared to other approaches [1].

**Distance** and **Bias**. The *distance* between two process models $S$ and $S'$ is the minimal number of change operations needed for transforming $S$ into $S'$. The respective sequence of change operations is denoted as *bias* between $S$ and $S'$. Generally, the distance between two process models measures the complexity for model transformation. As example take Fig. 1. Here, distance between $S$ and $S'$ is *one*, since we only need to perform one change operation $move(S, receivetreatment, admitted, pay)$ to transform $S$ into $S'$ (see [5]).

**Trace**. A *trace* $t$ on process model $S = (N, E, \ldots)$ denotes a valid execution sequence $t \equiv < a_1, a_2, \ldots, a_k >$ of activities $a_i \in N$ on $S$ according to the control flow defined by $S$. All traces producible on $S$ are summarized in set $\mathcal{T}_S$.

## 3 On Representing Process Changes

We first discuss basic aspects concerning the representation of process changes.

**Why Do We Need a Change Log?** Change and execution logs capture different runtime information on process instances and are not interchangeable. Even if the original model of a process instance is given, it is not possible to convert its change log to its execution log or vice verse. Consider our example from Fig. 1. When applying the described change to original model $S$, we obtain process variant $S'$ (i.e. $S[\Delta\rangle S'$) with change log $\Delta =< move(S, reveive\ \ treatment, admitted, pay) >$. This process variant represents an instance-specific model and the trace of the particular instance is $\{admitted, receive\ treatment, register, pay\}$. If original model $S$ and instance trace had been the only available information, it would be not possible to determine the respective change. Note that the process model, which can produce the given trace, is not unique; e.g., a process model with the four activities contained in four parallel branches could produce this trace as well. By contrast, it is generally not possible to derive the trace of a process instance from its structure and change log, because execution behavior of $S'$ is also not unique, e.g., trace $< admitted, register, receive\ treatment, pay >$ is also producible on $S'$.

**High-level Change Operations vs. Change Primitives** We now discuss why it is beneficial to measure the distance between process models based on high level change operations. Consider process model $S$ from Fig. 2a, it comprises a parallel branching (C and D may be performed concurrently), a conditional branching (either E or F is executed), and a silent activity $\tau$ (depicted as empty node). Assume that in two different scenarios high-level change operations are applied to $S$ resulting in variants $S_1$ and $S_2$ respectively: $S[\Delta_1\rangle S_1$ with $\Delta_1 = move(S, \texttt{C}, \texttt{A}, \texttt{B})$ and $S[\Delta_2\rangle S_2$ with $\Delta_2 = move(S, \texttt{A}, \texttt{B}, \texttt{C})$. Fig. 2 additionally depicts the change primitives representing the snapshot differences between $S$ and variants $S_1$ and $S_2$ respectively. In comparison with low-level primitives, the use of high-level change operations offers several advantages:

High-level change operations with formal pre-/post-conditions (e.g., [2]), guarantee soundness (e.g., absence of deadlocks); i.e., their application to a sound process model $S$ results in another sound model $S'$ [2]. This also applies to our example from Fig. 2. By contrast, when applying single primitives, soundness cannot be guaranteed in general.

High-level change operations enable more effective user support when compared to low-level primitives. Generally, a high-level change operation is based on a set of primitives which collectively realize a particular change pattern. As example take $\Delta_1$ from Fig. 2. This operation is internally based on *15* primitives to delete and add edges, to delete the silent activity, and to update node types. By defining changes with high-level operations, cost of change can be reduced. As another benefit, high-level operations perform model optimizations when realizing a process change. Regarding $\Delta_1$ from Fig. 2, for example, movement of C is accompanied by deletion of silent activity $\tau$, since the parallel branching is no longer needed.

Another important aspect concerns the number of change operations needed to transform a process model $S$ into an-

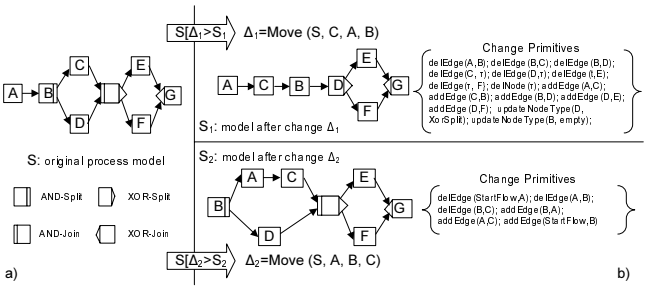


**Figure 2. High-Level Change Operations**

other model $S'$. Regarding our example, for instance, we only need *one* move operation to transform $S$ to either $S_1$ or $S_2$. When using change primitives instead, migrating $S$ to $S_1$ requires *15* primitives, while the second change $\Delta_2$ can be realized with *6* primitives. This demonstrates that the number of change primitives needed for model transformation does not provide an adequate means to express the difference between two process models.

**How Do High-level Changes Influence Process Behavior?** [3] measures similarity between two process models based on their trace sets, i.e., behavioral distance between models $S_1$ and $S_2$ is calculated as sum of the edit distances of all possible pairs of traces ($t_1$,$t_2$) with $t_i \in \mathcal{T}_{S_i}$ ($i = 1, 2$). Obviously, this method evaluates to what degree the behaviors of the two process models differ from each other rather than on what the effort for transforming one model into another is. On the one hand, application of one change operation can significantly modify execution behavior of the respective process model. On the other hand, several changes might be required to realize a smooth change of the behavior of a process model. When considering the two models from Fig. 1, we obtain behavioral distance *one*. However, when performing another change $S[\Delta_2\rangle S''$ with $\Delta_2 = move(S, pay, admitted, receivetreatment)$, behavioral distance between $S$ and $S''$ is *four*. Particularly, when a change moves or adds activities to parallel branches, the number of possible traces grows exponentially.

In our approach, we focus on the relationship between behavior of process models and biases.

## 4 Mining Process Variants

So far, we have motivated the need for representing process changes in terms of high-level change operations. This section discusses the major goal of mining process variants, namely to derive a generic process model out of a given collection of process variants. This shall be done in a way such that the different process variants can be efficiently configured out of the generic model. We measure the efforts for respective process configurations by the number of change operations needed to transform the generic model into the respective model variant. The challenge is to find a generic model such that the average number of change operations needed (i.e., the average distance) becomes minimal.

To make this more clear, we compare process variant mining with traditional process mining. Obviously, input data for process mining and process variant mining differ. While traditional process mining operates on execution logs, mining process variants is based on change logs (i.e., the process variants we can obtain from them). In principle, methods like alpha algorithm or genetic mining [4] can be applied to our problem as well. For example, we could derive all traces producible by a given collection of process variants [3] and then apply existing mining algorithms to them. To make the difference between process mining and process variant mining more evident, in the following, we consider behavioral similarity between two process models as well as structural similarity based on their bias.

The behavior of a process model $S$ can be represented by the trace set $\mathcal{T}_S$ it can produce. Therefore, two models can be compared based on the difference between their trace sets [3, 4]. By contrast, biases is used to express (structural) distance between two process models based on model transformations [5]. While process variant mining addresses structural similarity, traditional process mining focuses on behavior. Obviously, this leads to different choices in algorithm design and also different mining results. Fig. 3 shows one example. Assume that 55 process instances are running on process variant $S_1$ and 45 instances on variant $S_2$. If we focus on behavior, like existing process mining algorithms do [4], the discovered process model will be $S$; all traces producible on $S_i$ ($i = 1, 2$) can be produced on $S$ as well, i.e. $\mathcal{T}_{S_i} \subseteq \mathcal{T}_S$ ($i = 1, 2$). However, if we adopt $S$ as reference model, all instances running on $S_1$ or $S_2$ will have non-empty bias. The average weighted distance between $S$ and $S_i$ ($i = 1, 2$) is *one*; i.e., for each process instance we need on average one high-level change operation to configure $S$ into $S_i$ ($i = 1, 2$) ($S[\Delta_1\rangle S_1$ with $\Delta_1 = move(S, \texttt{B}, \texttt{A}, \texttt{C})$ and $S[\Delta_2\rangle S_2$ with $\Delta_2 = move(S, \texttt{B}, \texttt{C}, \texttt{D})$).

By contrast, if the goal is to reduce the average distance between genetic model and process (instance) variants, we should choose $S'$ as reference model. Though $S'$ does not cover all traces $S_1$ and $S_2$ can produce, average distance between generic model and process instances becomes minimal. More precisely, the average distance between $S'$ and instances running either on $S_1$ or $S_2$ is *0.45*; while $S' = S_1$, we only need to move B for the 45 instances based on $S_2$, i.e. $S'[\Delta'\rangle S_2$ with $\Delta' = move(S', \texttt{B}, \texttt{C}, \texttt{D})$. Though $S'$ cannot cover all traces variants $S_1$ and $S_2$ can produce, adapting $S'$ rather than $S$ as new reference model requires less efforts for process configuration, since the average weighted distance between $S'$ and the instances running on both $S_1$ and $S_2$ is 55% lower than when using $S$.

Our discussions on the difference between behavioral and structural similarity also demonstrate that current process mining algorithms do not consider structural similarity based on bias and change distance. We give an example to illustrate the basic goal as well as relevant issues and chal-

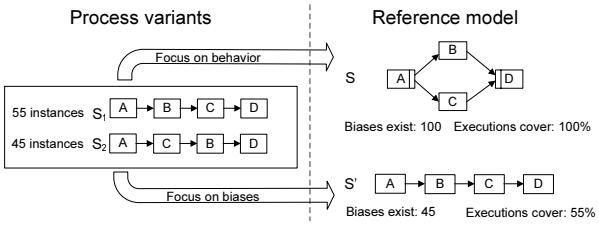


**Figure 3. Mining focusing either on behavior or minimization of biases**

lenges for mining process variants. Consider Fig. 4 and assume that process model $S$ defines a standard business process. Six process variants $S_i$ ($i = (1, \ldots, 6)$) have been configured out of $S$. Furthermore, on each of these variants several instances are running. A simple statistics is given to show the respective ratio (e.g., 30% of all instances are running on variant $S_1$ and 8% on $S_5$). Based on the relative frequency of each process variant (i.e., its weight), the weighted average number of changes required for configuring the six process variants is 1.5. This means we have to perform on average 1.5 changes on original model $S$ in order to configure these process variants out of it.

When mining the six variants by our approach [9], we obtain process model $S'$ as result (cf. Fig. 4). This model $S'$ is better in the sense that it can reduce average distance process variants have with respect to a reference process model (if using $S'$ instead of $S$ as reference model). The weighted average number of change operations (i.e., average distance between reference model and process variants) then decreases from 1.5 to 1.15 (cf. Fig. 4).

We also compared our result with the models obtained through process mining algorithms, like Alpha and Alpha++ algorithms, Heuristic mining and Genetic mining (see them at [4]). We first enumerate all traces producible on the six variants, and use these traces as the input of the process mining algorithms. Afterwards, we measure the result by computing average weighted distance between the discovered models and the 6 process variants. Evaluation results show that the process model discovered using our approach has shortest average weighted distance to the process variants (see [9] for details).

## 5 Related Work

A variety of techniques for process mining has been suggested in literature [4]. As illustrated in this paper, traditional process mining is different from process variant mining due to its different goal and inputs. A few techniques have been proposed to learn from process variants by mining change primitives [6]. However, this approach does not consider important features of process meta model; e.g., it is unable to deal with silent activities or loop backs, and does also not differentiate AND- and OR-splits. Similar techniques for mining change primitives exist in the fields of association rule mining and maximal sub-graph mining [7]; here common edges between different nodes are discovered to construct a common sub-graph from a set of graphs. None of the discussed approaches aims at creating a generic process model, which allows for easy and optimized configuration for process variants.

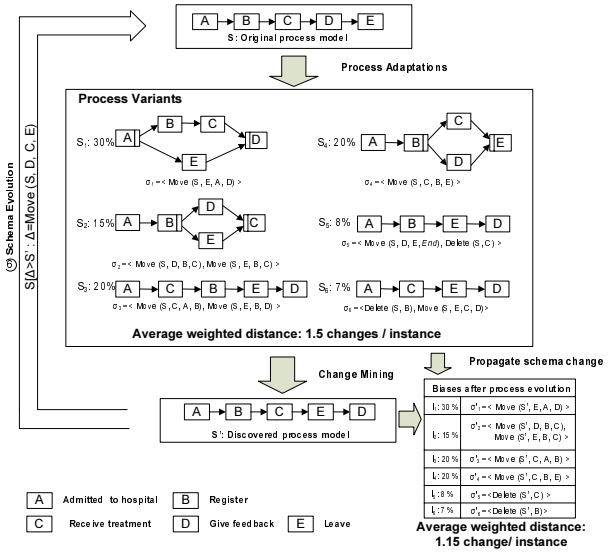


**Figure 4. One example**

## 6 Summary and Outlook

We have motivated the need for process variant mining, discussed its major goals as well as relevant issues, and elaborated its differences when compared to traditional process mining. Basically, as input our approach takes a collection of process variants, and then produces a generic process model as output which covers these variants best; i.e., the generic model has minimal average distance to process variants. Note that this will reduce adaptation and configuration costs as well. When comparing our approach with traditional mining techniques, we can see that the latter do not satisfy the need for deriving a process model which is easy configurable. This justifies the efforts for designing specific algorithms for process variant mining, which we will present in other papers.

## References


[1] B. Weber, S. Rinderle, M. Reichert: *Change Patterns and Change Support Features in Process-Aware Information Systems*. CAiSE'07, LNCS 4495, 2007, pp. 574-588

[2] M. Reichert and P.Dadam. *ADEPTflex - Supporting Dynamic Changes of Workflows Without Losing Control*. Journal of Intelligent Information Systems, 10(2):93–129, 1998.

[3] A.Wombacher, M.Rozie: Evaluation of Workflow Similarity Measures in Service Discovery. Service Oriented Electronic Commerce 2006: 51-71

[4] http://ga1717.tm.tue.nl/wiki/

[5] C.Li, M.Reichert, A.Wombacher. *Process Similarity Based on High Level Change Operations*. CTIT Technical Report, University of Twente, The Netherlands.

[6] J.Bae, L. Liu, J. Caverlee, W. B. Rouse: *Process Mining, Discovery, and Integration using Distance Measures* ICWS06, pp. 479-488, 2006

[7] P. Tan, M. Steinbach, V. Kumar: *Introduction to Data Mining*. Addison Wesley, 2006.

[8] R. Lu, S.W. Sadiq: *Managing Process Variants as an Information Resource*. BPM'06, LNCS 4102, pp 426-431, 2006

[9] C. Li, M. Reichert, A. Wombacher: *Discovering Process Reference Models from Process Variants Using Clustering Techniques*. TR-CTIT-08-30. University of Twente. 2008.