Configuration Rule Mining for Variability Analysis in Configurable Process Models

Nour Assy and Walid Gaaloul

Computer Science Department, Telecom SudParis
UMR 5157 CNRS Samovar, France
{firstname.lastname}@telecom-sudparis.eu

Abstract. With the intention of design by reuse, *configurable process models* provide a way to model variability in reference models that need to be configured according to specific needs. Recently, the increasing adoption of configurable process models has resulted in a large number of configured process variants. Current research activities are successfully investigating the design and configuration of configurable process models. However, a little attention is attributed to analyze the way they are configured. Such analysis can yield useful information in order to help organizations improving the quality of their configurable process models. In this paper, we introduce *configuration rule mining*, a frequency-based approach for supporting the variability analysis in configurable process models. Basically, we propose to enhance configurable process models with configuration rules that describe the interrelationships between the frequently selected configurations. These rules are extracted from a large collection of process variants using *association rule mining* techniques. To show the feasibility and effectiveness of our approach, we conduct experiments on a dataset from SAP reference model.

1 Introduction

With the rapidly changing demands in today's business requirements, there is no doubt that new paradigms for managing enterprises' business processes turn into a pressing need. In such a highly dynamic environment, seeking reuse [1] and adaptability [2] become a strong requirement for a successful business process design. To this end, *configurable process models* [3] provide a way for modeling variability in reference models. A configurable process model is a generic model that integrates multiple process variants of a same business process in a given domain through variation points. These variation points are referred to as *configurable elements* and allow for multiple design options in the process model. A configurable process model needs to be configured according to a specific requirement by selecting one design option for each configurable element. In this way, an individual process variant is derived without an extra design effort.

Recently, several approaches addressed the problem of building configurable process models. Some of them propose to merge existing process variants [4–7], others try to mine one configurable process model from execution logs [8–10].

X. Franch et al. (Eds.): ICSOC 2014, LNCS 8831, pp. 1–15, 2014.

These research results highlight the need for means of support to derive individual variants as integrated models tend to be complex with a large number of configurable elements [11]. To fill this gap, some works propose to use questionnaires [12] or ontologies [13] in order to get business requirements and guide the configuration process. Others propose to use non functional requirements to assess configuration decisions on the process performance [14]. Although these works have made a considerable effort on process variability design and configuration, a less attention has been paid to understand the way a configurable process model is configured. That means *which configurations are frequently selected by the users* and *how configuration decisions may have an impact on others in the process model*. The configurations' frequencies and interrelationships have been identified in the requirements for a configurable modeling technique in [3].

In this work, we propose to enhance configurable process models with *configuration rules*. These rules reveal the frequency and association between the configuration decisions taken for different variation points in a configurable process model. Concretely, we propose to discover from a large collection of process variants the frequently selected configurations in a configurable process model. Then, taking advantage of machine learning techniques [15], in particular *association rule mining*, we extract configuration rules between the discovered configurations. These rules can be then used to support business analysts to develop a better understanding and reasoning on the variability in their configurable process models. For instance, business analysts can manage the complexity of existing configurable process models by removing or altering the configurations that were never or rarely selected. Moreover, the automated discovery of the interrelationships between configuration decisions can assist the configuration process by predicting next suitable configurations given the selected ones.

The remainder of the paper is organized as follows: in section 2, we present a running example used throughout the paper to illustrate our approach. Section 3 provides some concepts and definitions needed for our approach. In section 4, we detail our approach to derive configuration rules using association rule mining techniques. The validation and experimental results are reported in section 5. In section 6, we discuss related work and we conclude in section 7.

2 Running Example

Our running example is from SAP reference model for a procurement process management modeled with the Configurable Event-Driven Process Chain notation (C-EPC) [3] (see Fig. 1). The EPC notation consists of three elements: event, function and connector. An event can be seen as a pre- and/or post-condition that triggers a function. A function is the active element that describes an activity. Three types of connectors, OR, exclusive OR (XOR) and AND are used to model the splits and joins. In our example, we index connectors with numbers in order to distinguish between them. The C-EPC notation adds the configurability option for functions and connectors. A configurable function can be included or excluded from the model. A configurable connector can change its type

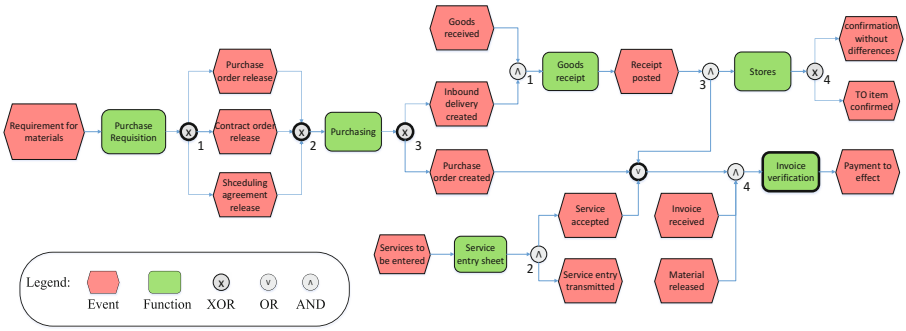

Fig. 1. An example of a configurable process model from SAP reference model

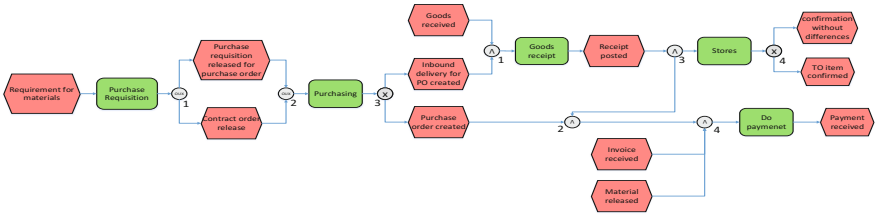

Fig. 2. A variant derived from the configurable process model in Fig. 1

(e.g. from OR to AND) or restrict it incoming or outgoing branches. Graphically, a configurable element is modeled with a thick line.

Returning to our example, the procurement process starts by detecting the need for new materials. A purchase request is sent to the corresponding provider. The purchase requisition type (for purchase order, for contract release order or for scheduling agreement schedule) is evaluated before the purchase starts. At this stage, either the goods delivery is followed until receiving goods or a purchase order is created. At the same time, a service entry sheet is created and transmitted. Last, the invoice is sent to the customer for verification and payment. We identified five configurable elements in the process: the connectors "$\times_1$", "$\times_2$", "$\times_3$", "$\vee$", and the function "invoice verification". This reference model is configured and used in a large number of companies that aim at reusing best practices for modeling their procurement processes.

Assume that a large number of configured process variants has been collected in a business repository from which we show one process variant in Fig. 2. This process is derived from the configurable process model in Fig. 1 through the following configurations:

1. Remove the outgoing branch starting with the event "Scheduling agreement release" from the configurable "$\times_1$";
2. Change the type of the configurable "$\vee$" to "$\wedge$", and remove the incoming branch ending with the event "service accepted";
3. Exclude the function "invoice verification" from the model.

In addition, some modifications have been performed on the configured model according to the company specific requirements such as renaming and/or adding events and functions. For example, the event "purchase order release" is renamed to "purchase requisition released for purchase order".

Using our proposed approach, we target to induce a set of configuration rules for each configurable element from available process variants. These rules are in the from of *if...then* and describe the combinations of the frequently selected configurations. For example, a configuration rule CR_1 for the configurable elements "$\times_1$" and "$\vee$" in the process model in Fig. 1 is:

$$\begin{aligned} CR_1 : &< \times, \{\text{purchase order release, contract order release}\} > \\ &\xrightarrow{S=0.7/C=0.65} < \wedge, \{\text{Purchase order created, } \wedge_3\} > \end{aligned} \tag{1}$$

This rule means that:

- The configurable "$\times_1$" is frequently configured to a "$\times$" with the outgoing branches starting with "purchase order release" and "contract order release";
- The configurable "$\vee$" is frequently configured to an "$\wedge$" with the incoming branches ending with "Purchase order created" and "$\wedge_3$";
- $S = 0.7$ means that in 70% of the process variants, these two configurations are selected;
- $C = 0.65$ means that in 65% of the process variants, whenever the first configuration (that of "$\times_1$") is selected, then the second configuration (that of "$\vee$") is also selected.

In the following sections, we give a formal definition for our configuration rules. Afterwards, we detail our approach for extracting the configuration rules that explain all possible combinations of the frequently selected configurations in the configurable process model.

3 Preliminaries

In this section, we present the definition of the *business process graph* and *configurable process model* enhanced with our *configuration rules* definition.

3.1 Business Process Graph

A business process model is a directed graph with labeled nodes. There exist many notations to represent a business process model such as Event-driven Process Chain (EPC), Business Process Modeling Notation (BPMN), Unified Modeling Language (UML), etc. In this work, we abstract from any specific notation and we represent a process model as a directed graph called *business process graph*. This notation is inspired from [4] in which the elements are derived from the common constructs of existing graphical process modeling notations.

Definition 1. ***(Business process graph)*** *A business process graph* $P = (N, E, T, L)$ *is a labeled directed graph where:*

- N *is the set of nodes;*
- $E \subseteq N \times N$ *is the set of edges connecting two nodes;*
- $T : N \rightarrow t$ *is a function that assigns for each node* $n \in N$ *a type* t *where* t *depends on the elements' types for each standard notation. In case of the EPC notation,* $t \in \{$*event, function, connector*$\}$*; Throughout the paper, we refer to functions and events by activities.*
- $L : N \rightarrow$ *label is a function that assigns for each node* $n \in N$ *a label such that if* $T(n) =$ *event* $\vee$ *function, then* $L(n)$ *is its name, and if* $T(n) =$ *connector then* $L(n) \in \{\vee, \wedge, \times\}$ *where* $\vee = OR$*,* $\wedge = AND$ *and* $\times = XOR$*.*

Let $P = (N, E, T, L)$ be a business process graph. We define the preset and postset of a connector $c \in N$ as the set of elements in its incoming and outgoing branches respectively.

Definition 2. ***(preset*** $\bullet c$***, postset*** $c\bullet$***)*** *The preset of a connector* $c \in N$ *denoted as* $\bullet c$ *is efined as* $\bullet c = \{n \in N : (n, c) \in E\}$*. The postset of* c *denoted as* $c\bullet$ *is defined as* $c\bullet = \{n \in N : (c, n) \in E\}$*.*

A connector "c" is a **split** if $|c \bullet| > 1$; it is a **join** if $|\bullet\ c| > 1$. For example, in Fig. 2, $\times_1 \bullet = \{$purchase requisition released for purchase, contract order release$\}$; $\bullet \wedge_4 = \{\wedge_2,$ invoice received, material released$\}$. "$\times_1$" is a split connector and "$\wedge_4$" is a join connector.

3.2 Configurable Process Model

A configurable process model, is a business process graph with configurable elements. A configurable element is an element whose configuration decision is made at design-time [3]. Configurable elements can be *functions* and/or *connectors*. A configurable function can be included (i.e. *ON*) or excluded (i.e. *OFF*) from the process model. A configurable connector has a generic behavior which is restricted by configuration. A connector can be configured by changing its type while preserving its behavior and/or restricting its incoming (respectively outgoing) branches in case of a join (respectively split). Table 1 presents the set of constraints identified in [3] for the configuration of connectors' types. A configurable connector is denoted by *[label]*^{c}. Each row in the table corresponds to a configurable connector which can be configured to one or more of the connectors presented in columns. The last column (i.e. *Seq*) corresponds to a simple "sequence". For example, the configurable "$\vee$" can be configured to any connector's type while a configurable "$\wedge$" can be only configured to an "$\wedge$". These configuration constraints are formalized through the partial order $\preceq$ that specifies which concrete connector may be used for a given configurable connector.

Definition 3. ***(partial order*** $\preceq$***)*** *Let* c^c *be a configurable connector and* c *be a normal connector or a sequence (i.e. "Seq").* $c \preceq c^c$ *iff* $(L(c^c) =$ "$\vee$"$) \vee (L(c^c) =$ "$\times$" $\wedge\ L(c) =$ "*Seq*"$) \vee\ (L(c^c) = L(c))$*.*

Table 1. Configuration constraints of configurable connectors

	$\vee$	$\wedge$	$\times$	Seq
$\vee^c$	✓	✓	✓	✓
$\wedge^c$		✓		
$\times^c$			✓	✓

Formally, the configuration of a configurable element, i.e. a function or a connector, is defined as:

Definition 4. ***(Configuration Conf)*** *The configuration of a node n^c such that $T(n^c) =$ '$function$' $\vee$ '$connector$' is defined as:*

- *if $T(n^c) =$ '$function$' then $Conf(n^c) \in \{ON, OFF\}$;*
- *if $T(n^c) =$ '$connector$' then $Conf(n^c) =< n, \bullet n >$ (respectively $Conf(n^c) = < n, n \bullet >$) in case n^c is a join (respectively split) connector where:*
  1. *$n \preceq n^c$;*
  2. *$\bullet n \subseteq \bullet n^c$ (respectively $n \bullet \subseteq n^c \bullet$) in case n^c is a join (respectively split) connector*

For example, the process variant in Fig. 2 is derived from the configurable process model in Fig. 1 by configuring the "$\vee^c$" to: $Conf(\vee^c) =< \wedge_2, \{$purchase order created, $\wedge_3\} >$; the configurable function "invoice verification" to: $Conf($invoice verification$) = OFF$; etc.

Configuration rule. A configuration rule describes an association among the frequently selected configurations for different configurable elements in a configurable process model. It is defined as:

$$Conf_{h_1}, ..., Conf_{h_p} \xrightarrow{S,C} Conf_{b_1}, ..., Conf_{b_q} \tag{2}$$

where $Conf_{h_i} : 1 \leq i \leq p$ is called the *rule head* and $Conf_{b_j} : 1 \leq j \leq q$ is called the *rule body*. The rule head and body represent the configurations of different configurable elements in a configurable process model. These configurations are retrieved from a business process repository. A configuration rule is parameterized by two well known metrics in association rule mining: the *Support* S and the *Confidence* C. The support is the fraction of process variants in the business process repository that contain the configurations of the rule head and body. It evaluates the usefulness of a rule. The confidence is the fraction of process variants that contain the rule body configurations among those that contain the rule head configurations. It represents the certainty of the rule. Let $\mathbb{P} = \{P_m : 1 \leq m \leq n\}$ be a business process repository. Formally:

$$\begin{aligned} S &= \frac{|\{P_{hb} : 1 \leq hb \leq n \wedge Conf_{h_i}, Conf_{b_j} \in P_{hb}\}|}{n} \\ C &= \frac{|\{P_{hb} : 1 \leq hb \leq n \wedge Conf_{h_i}, Conf_{b_j} \in P_{hb}\}|}{|\{P_h : 1 \leq h \leq n \wedge Conf_{h_i} \in P_h\}|} \end{aligned} \tag{3}$$

where $|\{P_{hb} : 1 \leq hb \leq n \wedge Conf_{h_i}, Conf_{b_j} \in P_k\}|$ is the number of process variants in $\mathbb{P}$ that contain the configurations in the rule head and body; $|\{P_h : 1 \leq h \leq n \wedge Conf_{h_i} \in P_h\}|$ is the number of process variants that contain the configurations in the rule head. The semantic of a configuration rule is: *if the configurations in the rule head are selected, then it is highly probably that the configurations in the rule body are also selected.* An example of a configuration rule is given in (1).

Definition 5. ***(Configurable process model)*** *A configurable process model is denoted as* $P^c = (N, E, T, L, B, Conf^c, CR^c)$ *where:*

- N, E, T, L *are as specified in Definition 1;*
- $B : N \rightarrow \{true, false\}$ *is a boolean function returning true for configurable nodes;*
- $Conf^c$ *is the set of valid configurations according to Definition 4;*
- CR^c *is the set of configuration rules.*

4 Configuration Rule Mining

In this section, we present our approach for mining configuration rules. Let $P^c = (N, E, T, L, B, Conf^c, CR^c)$ be a configurable process model and $\mathbb{P} = \{P_i = (N_i, E_i, T_i, L_i) : i \geq 1\}$ an existing business process repository. First, we extract from $\mathbb{P}$ the set of similar configurations for the configurable elements in P^c (see section 4.1). Then, using association rule mining techniques, we mine configuration rules from the retrieved similar configurations (see section 4.2).

4.1 Retrieving Similar Configurations

In this step, we extract from each process variant $P_i \in \mathbb{P}$ the configurations corresponding to the configurable elements in P^c. Nevertheless, retrieving exact configurations is not realistic as existing process variants may have similar but not exact parts with the configurable model. Thus, we aim at extracting *similar configurations* for the configurable elements. In order to match graph elements, we compute two similarities: the similarity Sim_A between activities and the similarity Sim_C between connectors.

Activities' similarity. Let $a \in N$ be an activity (function or event) in P^c and $a' \in N_i$ be an activity in the process variant P_i. To compute the similarity Sim_A between a and a', we use a combination of syntactic and semantic similarity metrics since they are popular for measuring the similarity between activities' labels in business process models [16]. We use a syntactic similarity based on Levenshtein distance [17] which computes the number of edit operations (i.e. insert, delete or substitute a character) needed to transform one string into another. For the semantic similarity, we use WordNet database [18] which a is lexical database for English words. The WordNet similarity package includes

a set of algorithms for returning the synonyms between two words. We use in particular the *WUP* algorithm [19] which measures the relatedness of two words by considering their depths in WordNet database. After normalizing activities' labels (i.e. put all characters in lowercase, remove stop words, etc.) the total similarity is the average of their syntactic and semantic similarities.

$$Sim_A(L(a), L_i(a')) = \frac{LD(L(a), L_i(a')) + WUP(L(a), L_i(a'))}{2} \quad (4)$$

where $0 \leq Sim_A \leq 1$, LD and WUP are functions returning the Levenshtein distance and the WordNet based similarity respectively between $L(a)$ and $L_i(a')$. We say that **a' is the best activity matching for a** iff: $Sim_A(L(a), L_i(a') \geq minSim_A \ \wedge \ \nexists a_x \in N_i : Sim_A(L(a), L_i(a_x)) > Sim_A(L(a), L_i(a'))$, where $minSim_A$ is a user specified threshold. For example, in Fig. 1 and 2, the similarity between the events "Purchase order release" and "Purchase requisition released for purchase order" is 0.735. For a $minSim_A = 0.5$,"Purchase order requisition for purchase order" is the best activity matching for "Purchase order release" as it has the highest similarity with "Purchase order release".

Connectors' similarity. Let $c \in N$ be a connector in P^c and $c' \in N_i$ be a connector in P_i. The similarity between connectors cannot be done in the same way as activities since connectors' labels do not have linguistic semantics. Hence, in order to compute the similarity Sim_C between c and c', we rely on (1) the partial order $\preceq$ (see Definition 3) which orders the connectors' labels based on their behavior and (2) the postset (respectively preset) similarities in case of split (respectively join) connectors. The similarity Sim_C between split connectors is computed as:

$$Sim_C(c, c') = \begin{cases} \frac{\#BM(c\bullet, c'\bullet)}{|c\bullet|} & \text{if } c' \preceq c \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where $\#BM(c\bullet, c'\bullet)$ returns the number of best elements' matching in $c'\bullet$ that correspond to those in $c\bullet$. The join connectors' similarity is computed in the same way but with the consideration of their preset instead of postset. We say that **c' is the best connector matching for c** iff: $Sim_C(c, c') \geq minSim_C \ \wedge \ \nexists c_x \in N_i : Sim_C(c, c_x) > Sim_C(c, c')$ where $minSim_C$ is a user specified threshold. For example, in Fig. 1 and 2, the similarity between "$\times_1$" in the first process model and "$\times_1$" in the second one is $Sim_C(\times_1, \times_1) = \frac{2}{3} = 0.67$. For a $minSim_C = 0.5$, "$\times_1$" in the second process model is the best connector matching for "$\times_1$" in the first one.

Similar functions'/connectors' configurations. Having defined the similarity metrics for activities and connectors, we show how we retrieve for configurable elements in P^c the similar configurations from each process variant $P_i \in \mathbb{P}$.

A configurable function $f^c \in N$ can be configured to ON or OFF. A configuration $Conf(f^c) = ON$ is retrieved from a process variant P_i, if there exists a function $f' \in N_i$ such that f' is the best activity matching for f^c. Otherwise,

the configuration $Conf(f^c) = OFF$ holds. For example, in our running example, for a $minSim_F = 0.5$, the configurable function "Invoice verification" in the configurable process model in Fig. 1 does not have any best activity matching in the process variant in Fig. 2. Thus from this process variant, we retrieve the configuration OFF.

A configurable split (respectively join) connector $c^c \in N$ can be configured w.r.t. its type and postset (respectively preset) (see Definition 4). A configuration $Conf(c^c) =< c, c\bullet >$ [1] is retrieved from a process variant P_i:

- if there exists a connector c' $\in N_i$ such that:
  1. c' is the best connector configuration for c^c,
  2. $L(c') = L(c)$ and
  3. $c\bullet$ is the set of elements in $c^c\bullet$ that have best element' matching in $c'\bullet$.
- else if $|c\bullet| = 1$ and there exists an element e' $\in N_i$ such that e' is the best element matching for $e \in c\bullet$. In this case, c^c is configured to a "*sequence*", i.e. $c = Seq$.

For example, for a $minSim_C = 0.5$, the configuration $Conf(\times_1) =< \times, \{$Purchase order release, contract order release$\}$ for the configurable connector "$\times_1$" in the process model in Fig. 1 is retrieved from the process model in Fig. 2 since (1) "$\times_1$" in the second model is the best connector matching for "$\times_1$" in the first model, (2) $L(\times_1) = L(\times_1)$ and (3) "Purchase order release" has "Purchase requisition released for purchase order" as the best activity matching; and "contract order release" has "contract order release" in the second model as the best activity matching.

4.2 Deriving Configuration Rules

In the previous section (section 4.1), we retrieved for each configurable element in P^c the set of similar configurations found in each process variant in $\mathbb{P}$. In this section, we use these configurations to mine our configuration rules using association rule mining techniques.

Association rule mining [20] is one of the most important techniques of data mining. It aims to find rules for predicting the occurrence of an item based on the occurrences of other items in a transactional database or other repositories. It has been first applied to the marketing domain for predicting the items that are frequently purchased together. Thereafter, it has manifested its power and usefulness in other areas such as web mining [21] and recommender systems [22]. The Apriori algorithm [23] is one of the earliest and relevant proposed algorithms for association rule mining.

In our work, we also use the Apriori algorithm for deriving our configuration rules. In order to be able to apply the Apriori algorithm, we store our retrieved configurations in a **configuration matrix**. The configuration matrix is a $n \times m$ matrix where n is the number of process variants in $\mathbb{P}$ (i.e. $n = |\mathbb{P}|$) and m is the number of configurable elements in P^c. A row in the configuration matrix

[1] We show the case for a split connector.

corresponds to one process variant in $\mathbb{P}$. A column corresponds to one configurable element in P^c. The entry for the row i and the column j contains the configuration retrieved from $P_i \in \mathbb{P}$ for the j^{th} configurable element. An example of the configuration matrix for the configurable process model in Fig. 1 is presented in Table 2. For example, the second row corresponds to the configurations retrieved from the process variant in Fig. 2. For clarification purpose, we refer to the configurations by their identifiers denoted as $C[nb]$. Table 3 contains the retrieved configurations' identifiers for each configurable element.

Table 2. An excerpt of a configuration matrix

P_{id}	invoice verification	$\times_1$	$\times_2$	$\times_3$	$\vee$
P_1	$C2$	$C3$	$C3$	$C20$	$C27$
P_2	$C1$	$C4$	$C4$	$C21$	-
P_3	$C1$	$C4$	$C4$	-	$C28$
...	...	...	...	...	...

Table 3. An excerpt of the retrieved configurations associated to unique identifiers

N^c	$Conf$	$Conf_{id}$
invoice verification	OFF	$C1$
	ON	$C2$
$\times_1$	$< \times, \{$purchase order release, contract order release$\} >$	$C3$
	$< \times, \{$purchase order release, scheduling agreemenr release$\} >$	$C4$
	...	...
$\times_2$	$< \times, \{$purchase order release, contract order release$\} >$	$C8$
	$< \times, \{$purchase order release, scheduling agreemenr release$\} >$	$C9$
	...	...
$\times_3$	$< \times, \{$Inbound delivery created, Purchase order created$\} >$	$C20$
	$< Seq, \{$purchase order release$\} >$	$C21$
	...	...
$\vee$	$< \wedge, \{$Purchase order created, $\wedge_3\} >$	$C27$
	$< \times, \{$purchase order created, Service accepted$\} >$	$C28$
	$< \times, \{$Service accepted, $\wedge_3\} >$	$C29$
	...	...

The configuration matrix along with a user specified support and confidence thresholds are used as inputs by the Apriori algorithm. As output, the Apriori algorithm returns the set of configuration rules having a support and confidence above the user's thresholds. An example of a configuration rule returned by Apriori for a support $S = 0.5$ and a confidence $C = 0.5$ is given in (1).

5 Experimental Results

In order to evaluate the usefulness and effectiveness of our proposed approach, we conduct experiments on a dataset from SAP reference model which contains 604 process models in EPC notation [24]. These models are not configurable. To create configurable process models for our experiments, we merge the similar processes in SAP models into configurable EPC (C-EPC) models using the merging approach proposed in [4]. To do so, we cluster similar process models with the Agglomerative Hierarchical Clustering (AHC) algorithm using the similarity approach presented in [4]. We obtain 40 clusters of similar process models having a similarity higher than 0.5. Then, each cluster is merged into one configurable process model. The characteristics of each obtained cluster and the corresponding configurable models are reported in Table 4.

Table 4. The size statistics of the clusters and the configurable process models

	size			# configurable nodes		
	min	max	avg.	min	max	avg.
cluster	20.55	25.625	23	0	0	0
configurable model	2	162	34.175	0	36	5.575

In the first experiment, we calculate the amount of reduction in the number of allowed configurations for a configurable process model using our proposed approach. Since the exponential growth in the number of allowed configurations is a source of complexity in a configurable process model, reducing and linking configuration decisions to the frequently selected ones have a significant impact on the variability understanding in configurable process models. Therefore, for each configurable process model, we mine the configuration rules. Then, we compute the amount of reduction which is one minus the ratio between the number of valid configurations using our configuration rules and the total number of valid configurations. The amount of reduction is defined as:

$$R = 1 - \frac{\#C_R}{\#C} \tag{6}$$

where $\#C_R$ is the number of configurations using our configuration rules and $\#C$ is the total number of valid configurations. The results reported in Table 5 show that in average we save up to 70% of allowed configurations which are either infrequent configurations or never selected in existing process models. Note that this amount of reduction may vary depending on the selected *minSupport* and *minConfidence* thresholds which are set to 0.5 in our experiments.

In the second experiment, we evaluate the mined configuration rules in order to extract useful characteristics for the configuration decision. Since configuration rules can be represented as a graph where each node represents a rule

Table 5. The amount of reduction

	Size	$\#C_R$	$\#C$	R
min	2	2	6	0.6
max	162	10.5×10^4	70×10^4	0.85
Avg.	34.175	1.5×10^3	5×10^3	0.7

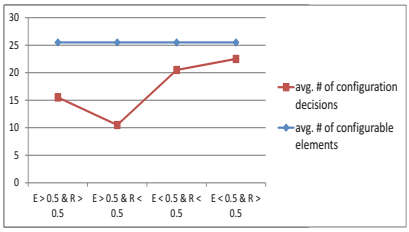

Fig. 3. The # of configuration decisions with varied emission and reception values

head or body and edges represent the implication relation between rules' head and body [25], we analyze this graph structure in order to derive interesting hypothesis for the configuration decision. We borrow the *emission* and *reception* metrics from the social network analysis domain [26] which measure the ratio of the outgoing and incoming relations respectively of a node in the graph. The reason for choosing these two metrics in particular is justified by the fact that a configuration node with a high emission have an impact on a large number of configurations in the process model. Therefore starting by its configuration may save the number of configuration decisions that should be taken by the user. Whereas a configuration node with a high reception depends on a large number of configurations. Therefore it may be useful to delay the selection of such configuration. The emission E_C and reception R_C ratios of a configuration node are computed as:

$$E_C = \frac{\#out_C}{max_i(\#out_{C_i})} \qquad R_C = \frac{\#in_C}{max_i(\#in_{C_i})} \tag{7}$$

where $\#out_C$ (respectively $\#in_C$) is the number of outgoing (respectively incoming) relations of the node C and $\#max_i(out_{C_i})$ (respectively $\#max_i(in_{C_i})$) is the maximal number of outgoing (respectively incoming) relations among the configuration nodes C_i in the graph. Using these two metrics, we select the models having more than 10 configurable nodes. Then for each configuration node, we compute its emission, reception and the number of configuration decisions that must be taken when starting with such configuration. Then, we organized these nodes in four groups based on their high (> 0.5) or low (< 0.5) reception and emission. The obtained results are illustrated in Fig. 3. The straight line represents the average number of configurable elements and the curve line represents the average number of configuration decisions that must be taken when starting with a specific group of configuration nodes. These results show that selecting the configurations with a high reception and a low emission reduce the number of configuration decisions to 10 while there exist in average $25,5$ configuration decisions (i.e. configurable elements) in the model.

6 Related Work

The limitation and rigid representation of existing business process models have led to the definition of flexible process models [27]. In this paper, we rely on the work presented in [3] where configurable process models are introduced. In their work, the authors define the requirements for a configurable process modeling technique and propose the configurable EPC notation. They highlight the need for configuration guidelines that guide the configuration process. These guidelines should clearly depict the interrelationships between configuration decisions and can include the frequency information. In our work, we demonstrate how using association rule mining techniques, we induce frequency-based configuration rules from existing process variants. These rules describe the association between the frequently selected configurations.

In order to match existing process models for merging, La Rosa et al. [4] use the notion of graph edit distance [28]. They compute the score matching using syntactical, semantic and contextual similarities identified in [16]. In our work, we propose to use existing process variants in order to analyze the variability in a configurable process model. This analysis can be used to improve the design and configuration of the configurable process model. We also use similar metrics for process model matching. However, instead of matching entire process models, we only search a matching for configurable elements.

To manage the variability in configurable process models, the researchers have been inspired from variability management in the field of Software Product Line Engineering [29]. La Rosa et al. [12] propose a questionnaire-driven approach for configuring reference models. They describe a framework to capture the system variability based on a set of questions defined by domain experts and answered by designers. Their questionnaire model includes order dependencies and domain constraints represented as logic expressions over facts. The main limitation of this approach is that it requires the knowledge of a domain expert to define the questionnaire model. In addition, each change in the configurable process model requires the update of the questionnaire model by the domain expert. This task manually performed may affect the configuration framework performance. While in our work, we propose an automated approach to extract the knowledge resulted from existing configurations using the well know concept of *association rules*. Our configuration rules can serve as a support for domain experts in order to define and update their configuration models.

Huang et al. [13] propose an ontology-based framework for deriving business rules using Semantic Web Rule Language (SWRL). They use two types of ontologies: a business rule ontology which is specified by a domain expert, and a process variation points ontology based on the C-EPC language. Using these ontologies, they derive SWRL rules that guide the configuration process. Different from them, we map the configuration process to a machine learning problem and use association rule mining instead of SWRL based rules in order to derive configuration rules. Our approach does not require any extra expert's effort and can be extended in order to classify the learned configuration rules w.r.t. specific business requirements.

7 Conclusion

In this paper, we present a frequency-based approach for the variability analysis in configurable process models. We propose to enhance the configurable process models with *configuration rules*. These rules describe the combination of the frequently selected configurations. Starting from a configurable process model and an existing business process repository, we take advantage of association rule mining techniques in order to mine the frequently selected configurations as *configuration rules*. Experimental results show that using our configuration rules, the complexity of existing configurable process models is reduced. In addition, metrics such as *emission* and *reception* applied to our configuration rules help in identifying the configurations that save users' decisions.

Actually, we are integrating our approach in an existing business process modeling tool, namely Oryx editor. In our fututre work, we target to define most sophisticated rules for retrieving similar connector' configurations. Instead of relying only on the connectors's direct preset and postset, we aim at looking for k-backward and k-forward similar elements. This in turn, would improve our preprocessing step and therefore our mined configuration rules. Moreover, we look for enhancing our configuration rules, besides the frequency, with other useful information such as the configuration performance, ranking, etc.

References

1. Fettke, P., Loos, P.: Classification of reference models: a methodology and its application. Information Systems and eBusiness Management (2003)
2. Schonenberg, H., et al.: Towards a taxonomy of process flexibility. In: CAiSE Forum, pp. 81–84 (2008)
3. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. (2007)
4. Rosa, L., et al.: Business process model merging: An approach to business process consolidation. ACM Trans. Softw. Eng. Methodol. (2013)
5. Derguech, W., Bhiri, S.: Merging business process variants. In: Abramowicz, W. (ed.) BIS 2011. LNBIP, vol. 87, pp. 86–97. Springer, Heidelberg (2011)
6. Gottschalk, F., Aalst, W.M., Jansen-Vullers, M.H.: Merging event-driven process chains. In: OTM 2008 (2008)
7. Assy, N., Chan, N.N., Gaaloul, W.: Assisting business process design with configurable process fragments. In: IEEE SCC 2013 (2013)
8. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: Mining configurable process models from collections of event logs. In: Daniel, F., Wang, J., Weber, B. (eds.) BPM 2013. LNCS, vol. 8094, pp. 33–48. Springer, Heidelberg (2013)
9. Gottschalk, F., Aalst, W.M.P.v.d., Jansen-Vullers, M.H.: Mining Reference Process Models and their Configurations. In: EI2N08, OTM 2008 Workshops (2008)
10. Assy, N., Gaaloul, W., Defude, B.: Mining configurable process fragments for business process design. In: Tremblay, M.C., VanderMeer, D., Rothenberger, M., Gupta, A., Yoon, V. (eds.) DESRIST 2014. LNCS, vol. 8463, pp. 209–224. Springer, Heidelberg (2014)
11. Dijkman, R.M., Rosa, M.L., Reijers, H.A.: Managing large collections of business process models - current techniques and challenges. Computers in Industry (2012)

12. Rosa, M.L., et al.: Questionnaire-based variability modeling for system configuration. Software and System Modeling 8(2), 251–274 (2009)
13. Huang, Y., Feng, Z., He, K., Huang, Y.: Ontology-based configuration for service-based business process model. In: IEEE SCC, pp. 296–303 (2013)
14. Santos, E., Pimentel, J., Castro, J., Sánchez, J., Pastor, O.: Configuring the variability of business process models using non-functional requirements. In: Bider, I., Halpin, T., Krogstie, J., Nurcan, S., Proper, E., Schmidt, R., Ukor, R. (eds.) BPMDS 2010 and EMMSAD 2010. LNBIP, vol. 50, pp. 274–286. Springer, Heidelberg (2010)
15. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc (2005)
16. Dijkman, R.M., et al.: Similarity of business process models: Metrics and evaluation. Inf. Syst. 36(2), 498–516 (2011)
17. Levenshtein, V.I.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady (1996)
18. Pedersen, T., Patwardhan, S., Michelizzi, J.: Wordnet: Similarity - measuring the relatedness of concepts. In: AAAI, pp. 1024–1025 (2004)
19. Wu, Z., Palmer, M.: Verbs semantics and lexical selection. In: ACL 1994 (1994)
20. Agrawal, R., Imielinski, T., Swami, A.N.: Mining association rules between sets of items in large databases. In: ACM SIGMOD 1993, pp. 207–216 (1993)
21. Fu, X., Budzik, J., Hammond, K.J.: Mining Navigation History for Recommendation. In: IUI 2000, pp. 106–112 (2000)
22. Lin, W., Alvarez, S.A., Ruiz, C.: Collaborative recommendation via adaptive association rule mining. In: Data Mining and Knowledge Discovery (2000)
23. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB, pp. 487–499 (1994)
24. Keller, G., Teufel, T.: Sap R/3 Process Oriented Implementation, 1st edn. Addison-Wesley Longman Publishing Co., Inc., Boston (1998)
25. Ertek, G., Demiriz, A.: A framework for visualizing association mining results. In: Levi, A., Savaş, E., Yenigün, H., Balcısoy, S., Saygın, Y. (eds.) ISCIS 2006. LNCS, vol. 4263, pp. 593–602. Springer, Heidelberg (2006)
26. Scott, J.P.: Social Network Analysis: A Handbook. SAGE Publications (2000)
27. Bhat, J., Deshmukh, N.: Methods for Modeling Flexibility in Business Processes. In: BPMDS 2005 (2005)
28. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Systems Science and Cybernetics 4(2), 100–107 (1968)
29. Clements, P.C.: Managing variability for software product lines: Working with variability mechanisms. In: SPLC, pp. 207–208 (2006)