Extracting Configuration Guidance Models from Business Process Repositories

Nour Assy(✉) and Walid Gaaloul

Computer Science Department, Telecom SudParis,
UMR 5157 CNRS Samovar, Évry, France
nour.assy@telecom-sudparise.eu

Abstract. *Configurable process models* are gaining a great importance for the design and development of reusable business processes. As these processes tend to be very complex, their configuration becomes a difficult task. Therefore, many approaches propose to build decision support systems to assist users selecting desirable configuration choices. Nevertheless, these systems are to a large extent *manually created* by domain experts, which is a time-consuming and tedious task. In addition, relying solely on the expert knowledge is not only error-prone, but also challengeable. In this paper, we propose to learn from past experience in process configuration in order to automatically extract a *configuration guidance model*. Instead of starting from scratch, a configuration guidance model assists analysts creating business-driven support systems.

1 Introduction

Motivated by the "Design by Reuse" paradigm, *configurable process models* are recently gaining momentum due to their capability of explicitly representing the common and variable parts of similar processes into one customizable model [1]. However, configurable process models cannot be freely configured as the derived variants have to be *correct*. Besides the structural and behavioral correctness [2], the configured variants need to be valid considering specific domain constraints. For instance, in a hotel reservation process, if the "online reservation" activity is excluded from the model, the "online payment" activity would be excluded, otherwise the derived variant would not be optimal or consistent. While automated approaches have been proposed for configuring process models in a structurally and semantically correct manner [3], existing domain-based approaches [4–7] still require a significant manual work.

Inspired by the need to integrate the users' experience in process configuration [1,8], we propose in this paper to benefit from previous experience in process configuration in order to automatically extract configuration guidance models. Our aim is to learn from the experience gained through past process configurations in order to extract useful and implicit knowledge that assist analysts deriving business-driven decision support systems. Following the requirements identified in [1] for a successful process configuration technique, a configuration guidance model targets to answer the following three questions: (1) *When* a

H.R. Motahari-Nezhad et al. (Eds.): BPM 2015, LNCS 9253, pp. 198–206, 2015.
DOI: 10.1007/978-3-319-23063-4_14

configuration decision can be taken for a configurable element? (2) *How* an element is configured given the previously selected choices? (3) *How often* a specific decision has been made?

With respect to these questions, we define a configuration guidance model as a tree-like structure with dependencies' relations and frequency information. First, the tree structure allows a "hierarchical" ordering of the configuration steps in a parent-child fashion. That is, the parent element is configured before the child element (answer of the *when*). Second, two types of dependencies relations, inclusion and exclusion, may exist between the configurable elements (i.e. tree elements), (2) between their configuration choices and (3) between the configurable elements and the configuration choices (answer of the *how*). And last, the configuration choices and dependencies relations are labeled with frequency information that reveal the probability of their presence in previous process configurations (answer of the *how often*).

The remainder of the paper is organized as follows: Section 2 introduces our configuration guidance model. In section 3, we present our automated approach to extract configuration guidance models from existing business process repositories. Related work is discussed in section 4, and we conclude in section 5.

2 Configuration Guidance Model

In this section, we give some definitions on configurable process models and introduce our configuration guidance model.

A configurable process model, is a business process model with configurable elements. A configurable element is an element whose configuration decision is made at design-time [1]. An example of a configurable process model for a simple travel booking process modeled with the configurable BPMN (c-BPMN) is illustrated in Fig. 1. The configurable elements are graphically modeled with thick lines. They are the active elements of a process modeling notation. In case of c-BPMN, configurable elements can be *activities* and/or *connectors*. A configurable activity 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. It can be configured by (1) changing its type while preserving its behavior and/or (2) restricting its incoming (respectively outgoing) branches in case of a join (respectively split) [1]. For example, the configurable "OR" can be configured to any connector's type while a configurable "AND" can be only configured to an "AND". We denote by $c_1 \sqsubseteq c_2$ iff the behavior of c_1 is subsumed by that of c_2. For example $AND \sqsubseteq OR^c$, $AND \sqsubseteq AND^c$, $Seq \sqsubseteq XOR^c$ etc.

Definition 1 gives the formal definition of a configuration.

Definition 1 (Configuration *Conf*). *A configuration of a configurable node n^c denoted as $Conf_{n^c}$ is defined as:*

- *if n^c is an activity then $Conf_{n^c} \in \{ON, OFF\}$;*
- *if n^c is a connector then $Conf_{n^c} \in \{(c', s) : (c', s) \in CT \times 2^S\}$ where:*

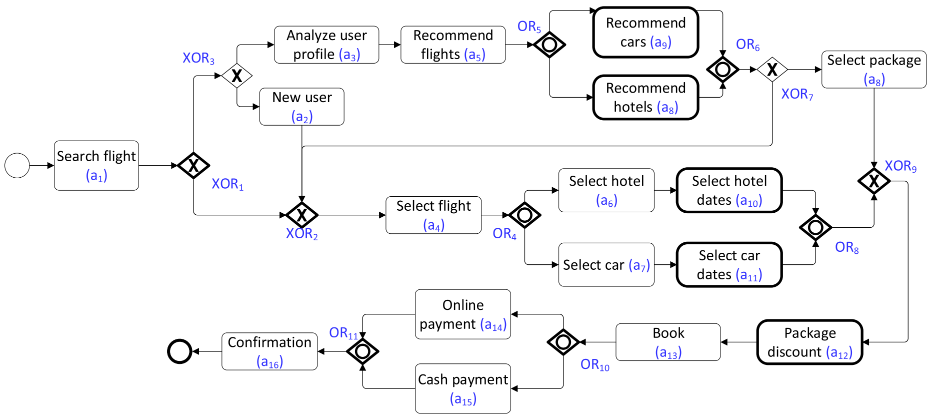

Fig. 1. An example of a configurable travel booking process

- *$CT = \{OR, AND, XOR, Seq\}$ and c' $\sqsubseteq$ c,*
- *$S = \bullet c$ (respectively $S = c\bullet$) in case c is a join (respectively split) connector where $\bullet c$ (respectively $c\bullet$) is the set of elements in the incoming branches (respectively outgoing branches) of c.*

We denote by $\mathbb{C}_{n^c}$ the set of all configurations of the configurable element n^c according to Definition 1. For example, in Fig. 1, $\mathbb{C}_{OR_4} = \{(OR, \{a_6, a_7\}), (XOR, \{a_6, a_7\}), (AND, \{a_6, a_7\}), (Seq, \{a_6\}), (Seq, \{a_7\})\}$.

A configuration guidance model is a tree-like structure with inclusion and exclusion dependencies relations. An excerpt of the configuration guidance model for the configurable process in Fig. 1 is illustrated in Fig. 2. The tree structure allows for a "hierarchical" ordering of the configurable elements of a process model in a parent-child fashion, that is the parent element is configured before the child element (see Section 3.1). The tree elements are graphically modeled with circles. Each tree element has multiple configuration choices (see Definition 1). In our approach, we compute the probability of selection of each configuration option (see Section 3). Graphically, the configuration choices are modeled with rectangles attached to their configurable elements.

The configuration guidelines represented as inclusion and exclusion relations are graphically modeled with dotted lines and have their probability of certainty. The probability of certainty expresses to which extent an inclusion or exclusion relation is valid. Three types of inclusion relations (denoted as I_R) may exist (i) between the configurable elements (denoted as I_{c-c}), (ii) between the configuration choices (denoted as I_{cf-cf}) and (iii) between the configurable elements and the configuration choices (denoted as I_{c-cf}). The same holds for the exclusion relations E_R (see section 3.2).

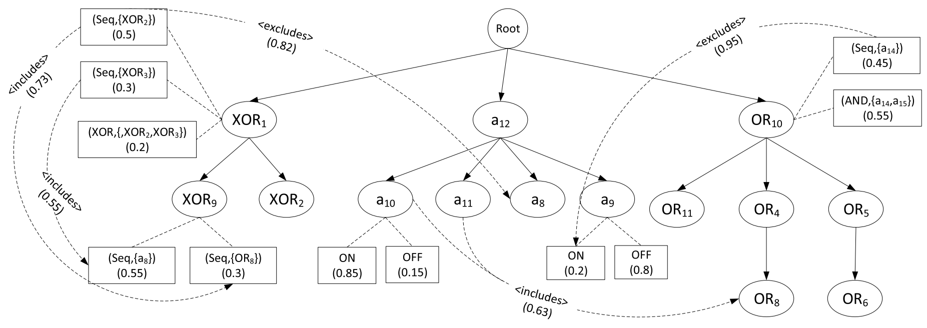

Fig. 2. An excerpt of the extracted configuration guidance model

3 Deriving Configuration Guidance Models

In this section, we present our automated approach for extracting configuration guidance models from business process repositories. Let P^c be a configurable process model and $\mathbb{P} = \{P_i : i \geq 1\}$ a set of previously derived variants from P^c. The processes in $\mathbb{P}$ can be collected by computing a similarity value (e.g. [9]). P^c and $\mathbb{P}$ are used as inputs by our algorithm (see Algorithm 1) to generate a configuration guidance model denoted as $\mathcal{G}^c_M = (\mathcal{T}^c, \mathbb{C}^*, I_R, E_R)$ where $\mathcal{T}^c$ is the tree hierarchy, $\mathbb{C}^*$ is the set of elements' configurations, I_R is the set of inclusion relations and E_R is the set of exclusion relations. The algorithm consists of a preliminary step (Line 3) then proceeds in two main steps (Lines 4-9).

Algorithm 1. Building a configuration guidance model

```
1: input: $P^c$, $\mathbb{P}$
2: output: $\mathcal{G}^c_{\mathcal{M}} = (\mathcal{T}^c, \mathbb{C}^*, I_R, E_R)$
3: get inclusion associations $\mathcal{A}_{\rightarrow}$ and exclusion associations $\mathcal{A}_{\rightarrow\neg}$
   {extract tree hierarchy}
4: derive probabilistic dependency matrix $M_P = getProbabilisticMatrix(A_{\rightarrow})$
5: derive implication graph $G_{\rightarrow} = getImplicationGraph(M^p)$
6: generate tree hierarchy $\mathcal{T}^c = getTreeHierarchy(G_{\rightarrow})$
   {derive model additional information}
7: derive configurations' probability $\mathcal{P}_{conf} = Sup(conf) : conf \in \mathbb{C}^*$
8: derive inclusion relations $I_R$ from $G_{\rightarrow}$ and $A_{\rightarrow}$
9: derive exclusion relations $E_R$ from $A_{\rightarrow\neg}$
```

In the preliminary step, the sets of positive and negative configuration associations denoted as $A_{\rightarrow}$ and $A_{\rightarrow\neg}$ respectively are extracted from $\mathbb{P}$ using *Apriori* [10], a well known algorithm for deriving association rules. This step has been elaborated in our previous work [11] and is briefly explained in the following. A positive configuration association is in the form of $conf_1 \rightarrow conf_2$

where $conf_1$ and $conf_2$ are configuration choices of different configurable elements and $conf_1 \rightarrow conf_2$ means that $conf_1$ and $conf_2$ co-occur frequently together. An example of a positive configuration association is: $Conf_{a_{10}} = ON \rightarrow Conf_{OR_8} = (Seq\{a_{10}\})$. A negative configuration association is in the form of $conf_1 \rightarrow \neg conf_2$ and means that the occurrence of the configuration $conf_1$ excludes that of $conf_2$. An example of a negative configuration association is: $Conf_{OR_{10}} = (Seq, \{a_{14}\}) \rightarrow \neg Conf_{a_9} = ON$. Well known metrics, such as *support*, *confidence* and *Conditional Probability Increment Ratio (CPIR)* are used by Apriori in order to (1) prune the set of extracted configurations to the frequently ones using a minimum support threshold, (2) generate the highly probable configuration associations using a minimum confidence threshold and (3) mine negative associations using a minimum CPIR threshold.

3.1 Extracting Tree Hierarchy

The tree hierarchy $\mathcal{T}^c$ consists of parent-child relations between the configurable elements. An element n_1^c is a candidate parent of a child element n_2^c if the configuration of n_2^c highly depends on that of n_1^c. The dependencies relations between the configurable elements can be derived from their configuration choices. In fact, the more are their configuration choices dependent, the more are the configurable elements dependent. The dependency of a configuration choice $conf_2$ on another configuration choice $conf_1$ corresponds to their conditional probability $P(conf_2|conf_1)$ which can be derived from the confidence of their positive configuration association $conf_1 \rightarrow conf_2 \in A_{\rightarrow}$. It is computed as:

$$P(conf_2|conf_1) = \frac{P(conf_1 \cap conf_2)}{P(conf_2)} = \frac{Sup(conf_1 \cup conf_2)}{Sup(conf_2)} = C(conf_1 \rightarrow conf_2) \quad (1)$$

where $P(conf_1 \cap conf_2)$ is the probability of co-occurrence of $conf_1$ and $conf_2$; $P(conf_2)$ is the probability of occurrence of $conf_2$. The probabilities are derived from the support metric computed by Apriori. Having the dependencies probabilities between the configuration choices, the conditional probability between two configurable elements n_1^c and n_2^c is computed as:

$$P(n_2^c|n_1^c) = \frac{\sum_j P(conf_{n_{2j}^c}|n_1^c)}{\#conf_{n_{2j}^c}} = \frac{\sum_j \frac{\sum_i P(conf_{n_{2j}^c}|conf_{n_{1i}^c})}{\#conf_{n_{1i}^c}}}{\#conf_{n_{2j}^c}} \quad (2)$$

where $P(n_2^c|n_1^c)$ is the average of the conditional probabilities between the configuration choices of n_1^c and n_2^c. $\sum_j P(conf_{n_{2j}^c}|n_1^c)$ is the sum of the conditional probabilities between each configuration choice $conf_{n_{2j}^c}$ of n_2^c and the configurable element n_1^c. The probability $P(conf_{n_{2j}^c}|n_1^c)$ is in turn defined as the average of the conditional probabilities between the configuration choice $conf_{n_{2j}^c}$ and each configuration choice $conf_{n_{1i}^c}$ of n_1^c. It can be computed by dividing the sum of the conditional probabilities between $conf_{n^c_{2_j}}$ and each $conf_{n^c_{1_i}}$ of n^c_1 by the number of $conf_{n^c_{1_i}}$ such that $P(conf_{n^c_{2_j}}|conf_{n^c_{1_i}}) \neq 0$; $\#conf_{n^c_{2_j}}$ is the number of the configuration choices of n^c_2 such that $P(conf_{n^c_{2_j}}|n^c_1) \neq 0$.

The conditional probabilities between each pair of configurable elements are computed and stored in a *dependency probabilistic matrix* denoted as M_P. M_P is a $m \times m$ matrix where m is the number of configurable elements. An entry (i, j) in M_P corresponds to the conditional probability $P(n^c_j|n^c_i)$ where n^c_j is the element in the j^{th} column and n^c_i is the element in the i^{th} row. We say that a configurable element n^c_2 depends on another element n^c_1 denoted as $n^c_1 \rightarrow n^c_2$ iff $P(n^c_2|n^c_1) \geq minP$ where $minP$ is a given threshold.

The derived dependencies' relations with their probabilities are modeled in a graph, called *implication graph* $G_{\rightarrow}$ [12]. The nodes in $G_{\rightarrow}$ correspond to the configurable elements. A weighted edge exists from a node n^c_1 to n^c_2 iff $n^c_1 \rightarrow n^c_2$; the edge's weight is the probability $P(n^c_2|n^c_1)$. An excerpt of $G_{\rightarrow}$ derived from a set of dependencies relations is illustrated in Fig. 3a. Having $G_{\rightarrow}$, the tree

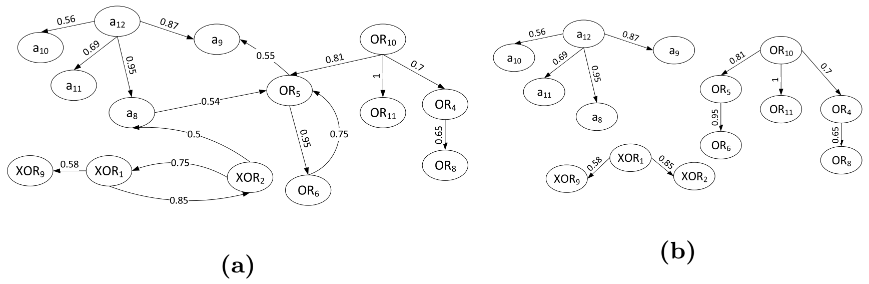

Fig. 3. (a) An implication graph and (b) its derived optimal spanning tree

hierarchy corresponds to extracting a spanning tree (called arborescence for directed graphs) [12]. Since, there exist multiple possible spanning trees, we aim at deriving the optimal hierarchy that maximizes the dependencies' relations weights. The problem can be mapped to finding the *minimal spanning tree* which can be solved using existing algorithms such as Edmonds' algorithm [13] and efficient implementations such as [14]. Figure 3b illustrates an excerpt of the optimal spanning tree extracted from the implication graph in Fig. 3a which contains multiple trees. In this case, an artificial root node is added and connected to them in order to obtain the tree hierarchy in Fig. 2.

3.2 Deriving Additional Model Information

In this section, we complete the remaining configuration guidance model information, i.e. the configuration choices probabilities and the inclusion/exclusion relations and their probabilities. The configuration choices $\mathbb{C}^*$ are the set of configurations extracted by Apriori and and their probabilities are equal to the Apriori computed support. For example, in Fig. 2, the configuration $(Seq, \{XOR_2\}) \in \mathbb{C}^*$ has a probability $P = 0.5$. The three types of inclusion relations and their probabilities are defined as follows [1]

- $I_{c-c} = \{n_1^c \rightarrow n_2^c\}$: $n_1^c, n_2^c \in N^c \wedge n_1^c \rightarrow n_2^c \in G_{\rightarrow} \setminus \mathcal{T}^c$, i.e. the inclusion relations between the configurable elements are those that are present in the implication graph but have been excluded when deriving the tree hierarchy. The probability $\mathcal{P}_R(n_1^c \rightarrow n_2^c) = P(n_2^c | n_1^c)$.
- $I_{cf-cf} = \{conf_1 \rightarrow conf_2\}$: $conf_1 \in \mathbb{C}_{n_x^c}, conf_2 \in \mathbb{C}_{n_y^c} \wedge (conf_1 \rightarrow conf_2 \in A_{\rightarrow}) \wedge (\exists conf_1' \in \mathbb{C}_{n_x^c}, conf_2' \in \mathbb{C}_{n_y^c} : conf_1' \rightarrow conf_2' \notin A_{\rightarrow})$, i.e. the inclusion relations between the configuration choices are those that appear in the positive configuration associations but whose configurable elements are not fully dependent. $\mathcal{P}_R(conf_1 \rightarrow conf_2) = P(conf_2 | conf_1)$.
- $I_{c-cf} = \{n^c \rightarrow conf\} : n^c \in N^c, conf \in \mathbb{C}^* \wedge \forall conf' \in \mathbb{C}_{n^c} : conf' \rightarrow conf \in A_{\rightarrow}$. The probability $\mathcal{P}_R(n^c \rightarrow conf) = P(conf | n^c)$, i.e. an inclusion relation exists between a configurable element n^c and a configuration choice $conf$ iff each configuration choice of n^c has a dependency relation to $conf$. The same holds for the relations $conf \rightarrow n^c \in I_{c-cf}$. The probability $P_R(n^c \rightarrow conf) = P(conf | n^c)$.

4 Related Work

Business process variability modeling [15] is an emergent topic that is being increasingly addressed by academic and industrial researchers for enabling design-time process flexibility. Our work is based on configurable process models proposed in [1]. The authors in [1] define the requirements for a configurable process modeling technique. They highlight the need for configuration guidelines that may include the configuration steps order, the interrelationships between the configuration decisions and the frequency information that come from system users. In our work, we follow these requirements and propose an automated approach to learn a configuration guidance model depicting such information.

La Rosa et al. [4] 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. Asadi et al. [7] and Gröner et al. [6] propose to use feature models for modeling the variability and the configuration constraints. The constraints are defined by experts and formalized in Description Logic expressions. Templates and configuration rules are used by Kumar et al. [5] in order to configure a reference process template using configuration rules which are defined and validated by experts.

In summary, existing approaches for assisting the configuration of process models require an expensive manual work from experts. These approaches are only based on the expert knowledge while, as highlighted in [1,8], a successful process configuration has to integrate the experience gained through previous

[1] The exclusion relations can be derived in the same way using $A_{\rightarrow\neg}$.

configurations. Therefore, in this paper, we address this research gap by proposing an automated approach for assisting the configuration of process models using previously configured processes.

5 Conclusion and Future Works

In this paper, we proposed an automated approach for extracting configuration guidance models from process model repositories. Our work is motivated by the need of (1) automated approaches on the one hand and (2) information originating from previous process configurations on the other hand in the creation of configuration decision support systems. Experimental results show that we generate accurate configuration guidance models.

The current limitation of our approach lies in the the lack of an empirical evaluation. In this regard, we are currently conducting experiments in order to evaluate the accuracy of our extracted configuration guidance models. In parallel, we are working with industrial partners and our team members in order to validate the approach from a business perspective.

References

1. Rosemann, M., van der Aalst, W.M.P.: A configurable reference modelling language. Inf. Syst. **32**(1), 1–23 (2007)
2. van der Aalst, W.M.P.: The application of petri nets to workflow management. Journal of Circuits, Systems, and Computers **8**(1), 21–66 (1998)
3. van der Aalst, W.M.P., Lohmann, N., Rosa, M.L.: Ensuring correctness during process configuration via partner synthesis. Inf. Syst. **37**(6), 574–592 (2012)
4. La Rosa, M., van der Aalst, W., Dumas, M., ter Hofstede, A.: Questionnaire-based variability modeling for system configuration. Software & Systems Modeling **8**(2), 251–274 (2009)
5. Kumar, A., Yao, W.: Design and management of flexible process variants using templates and rules. Comput. Ind. **63**(2), 112–130 (2012)
6. GröNer, G., BošKović, M.: Silva Parreiras, F., GašEvić, D.: Modeling and validation of business process families. Information Systems **38**(5), 709–726 (2013)
7. Asadi, M., Mohabbati, B., Gröner, G., Gasevic, D.: Development and validation of customized process models. Journal of Systems and Software **96**, 73–92 (2014)
8. Gottschalk, F.: Configurable Process Models. Ph.D thesis, Eindhoven University of Technology, December 2009
9. Weidlich, M., Mendling, J., Weske, M.: Efficient consistency measurement based on behavioral profiles of process models. IEEE Trans. Softw. Eng. **37**(3), 410–429 (2011)
10. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: VLDB, pp. 487–499 (1994)
11. Assy, N., Gaaloul, W.: Configuration rule mining for variability analysis in configurable process models. In: Franch, X., Ghose, A.K., Lewis, G.A., Bhiri, S. (eds.) ICSOC 2014. LNCS, vol. 8831, pp. 1–15. Springer, Heidelberg (2014)

12. She, S., Lotufo, R., Berger, T., Wasowski, A., Czarnecki, K.: Reverse engineering feature models. In: ICSE (2011)
13. Edmonds, J.: Optimum Branchings. Journal of Research of the National Bureau of Standards **71B**, 233–240 (1967)
14. Gabow, H.N., Galil, Z., Spencer, T., Tarjan, R.E.: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica **6**(2), 109–122 (1986)
15. Rosa, M.L., van der Aalst, W.M., Dumas, M., Milani, F.P.: Business process variability modeling: A survey. ACM Computing Surveys (2013)