




A Framework for Estimating Simplicity of Automatically Discovered Process Models Based on Structural and Behavioral Characteristics

Anna Kalenkova , Artem Polyvyanyy , and Marcello La Rosa 

School of Computing and Information Systems
The University of Melbourne, Parkville, VIC, 3010, Australia
{anna.kalenkova;artem.polyvyanyy;marcello.larosa}@unimelb.edu.au

Abstract. A plethora of algorithms for automatically discovering process models from event logs has emerged. The discovered models are used for analysis and come with a graphical flowchart-like representation that supports their comprehension by analysts. According to the Occam’s Razor principle, a model should encode the process behavior with as few constructs as possible, that is, it should not be overcomplicated without necessity. The simpler the graphical representation, the easier the described behavior can be understood by a stakeholder. Conversely, and intuitively, a complex representation should be harder to understand. Although various conformance checking techniques that relate the behavior of discovered models to the behavior recorded in event logs have been proposed, there are no methods for evaluating whether this behavior is represented in the simplest possible way. Existing techniques for measuring the simplicity of discovered models focus on their structural characteristics such as size or density, and ignore the behavior these models encoded. In this paper, we present a conceptual framework that can be instantiated into a concrete approach for estimating the simplicity of a model, considering the behavior the model describes, thus allowing a more holistic analysis. The reported evaluation over real-life event logs for several instantiations of the framework demonstrates its feasibility in practice.

1 Introduction

Information systems keep records of the business processes they support in the form of event logs. An event log is a collection of traces encoding timestamped actions undertaken to execute the corresponding process. Thus, such logs contain valuable information on how business processes are carried out in the real world. Process mining [1] aims to exploit this historical information to understand, analyze, and ultimately improve business processes. A core problem in process mining is that of automatically discovering a process model from an event log. Such a model should faithfully encode the process behavior captured in the log and, hence, meet a range of criteria. Specifically, a discovered model should describe the traces recorded in the log (have good *fitness*), not encode traces not present in the log (have good *precision*), capture possible traces that may stem from the same process but are not present in the log (have good *generalization*), and be “*simple*”. These quality measures for discovered process models are studied within the conformance checking area of process mining. A good discovered model is thus supposed to achieve a good balance between these criteria [3].

In [1], Van der Aalst suggests that process discovery should be guided by the Occam’s Razor principle [10,8], a problem-solving principle attributed to William of Ockham. Accordingly, “one should not increase, beyond what is necessary, the number of entities required to explain anything” [1]. Specifically to process mining, a discovered process model should only contain the necessary constructs. Various measures for assessing whether a discovered model is simple have been proposed in the literature [9,15], such as the number of nodes and arcs, density, and diameter. However, these measures address the number of constructs, i.e., the structure of the discovered models, while ignoring what these constructs describe, i.e., the process behavior.

In this paper, we present a framework that considers the model’s structure and behavior to operationalize Occam’s Razor principle for measuring the simplicity of a process model discovered from an event log. The framework comprises three components that can selectively be configured: (i) a notion for measuring the structural complexity of a process model, e.g., size or diameter; (ii) a notion for assessing the behavioral similarity, or equivalence, of process models, e.g., trace equivalence, bisimulation, or entropy; and (iii) the representation bias, i.e., a modeling language for describing models. A configured framework results in an approach for estimating the simplicity of process models. The obtained simplicity score establishes whether the behavior captured by the model can be encoded in a structurally simpler model. To this end, the structure of the model is related to the structures of other behaviorally similar process models.

To demonstrate these ideas, we instantiate the framework with the *number of nodes* [9] and *control flow complexity (cfc)* [4] measures of structural complexity, *topological entropy* [20] measure of behavioral similarity, and uniquely labeled *block-structured* [19] process models captured in the Business Process Model and Notation (BPMN) [18] as the representation bias. We then apply these framework instantiations to assess the simplicity of the process models automatically discovered from event logs by the Inductive miner algorithm [16]. This algorithm constructs *process trees*, which can then be converted into uniquely labeled *block-structured* BPMN models [14].

Once the framework is configured, the next challenge is to obtain models of various structures that specify behaviors similar to that captured by the given model, as these are then used to establish and quantify the amount of unnecessary structural information in the given model. To achieve completeness, one should aim to obtain all the similar models, including the simplest ones. As an exhaustive approach for synthesizing all such models is often unfeasible in practice, in this paper, we take an empirical approach and synthesize random models that approximate those models with similar behavior. To implement such approximations, we developed a tool that generates uniquely labeled block-structured BPMN models randomly, or exhaustively for some restricted cases, and measures their structural complexity and behavioral similarity.

The remainder of this paper is organized as follows. Section 2 discusses an example that motivates the problem of ignoring the behavior when measuring the simplicity of process models. Section 3 presents our framework for estimating the simplicity of the discovered models. In Section 4, we instantiate the framework with concrete components. Section 5 presents the results of an analysis of process models discovered from real-life event logs using our framework instantiations. Section 6 concludes the paper.

2 Motivating Example

In this section, we show that the existing simplicity measures do not always follow the Occam's Razor principle. Consider event log $L = \{\langle \text{load page}, \text{fill name}, \text{fill passport}, \text{fill expire date} \rangle, \langle \text{load page}, \text{fill name}, \text{fill expire date}, \text{fill passport} \rangle, \langle \text{load page}, \text{fill passport}, \text{fill expire date} \rangle, \langle \text{load page}, \text{fill name}, \text{fill expire date} \rangle, \langle \text{load page}, \text{fill name}, \text{fill passport} \rangle, \langle \text{load page}, \text{fill name}, \text{fill passport}, \text{fill expire date}, \text{load page} \rangle, \langle \text{load page}, \text{fill name}, \text{fill expire date}, \text{fill passport}, \text{load page} \rangle, \langle \text{load page}, \text{fill name}, \text{load page} \rangle, \langle \text{fill name}, \text{fill expire date}, \text{fill passport} \rangle, \langle \text{fill passport}, \text{fill expire date} \rangle\}$ generated by a passport renewal information system.¹ The log contains ten *traces*, each encoded as a sequence of *events*, or *steps*, taken by the users of the system. Usually, the user loads the Web page and fills out relevant forms with details such as name, previous passport number, and expiry date. Some steps in the traces may be skipped or repeated, as this is common for the real world event data [13], Fig. 1 and Fig. 2 present BPMN models discovered from L using, respectively, the Split miner [2] and Inductive miner [16] process discovery algorithm.

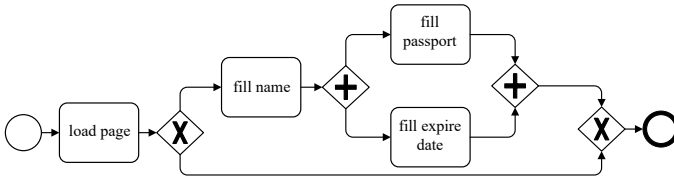


Fig. 1: A BPMN model discovered by Split miner from event log L .

It is evident from the figures that the models are different. First, they have different structures. Fig. 1 shows an acyclic model with exclusive and parallel branches. In contrast, the model in Fig. 2 only contains exclusive branches enclosed in a loop and allowing to skip any of the steps. Second, the models describe different collections of traces. While the model in Fig. 1 describes three traces (viz. $\langle \text{load page} \rangle$, $\langle \text{load page}, \text{fill name}, \text{fill passport}, \text{fill expire date} \rangle$, and $\langle \text{load page}, \text{fill name}, \text{fill expire date}, \text{fill passport} \rangle$), the model in Fig. 2 describes all the possible traces over the given steps, i.e., all the possible sequences of the steps, including repetitions.

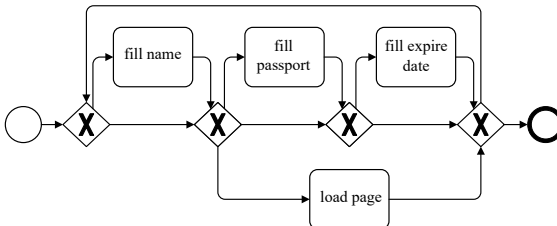


Fig. 2: A BPMN model discovered by Inductive miner from event log L .

The latter fact is also evident in the precision and recall values between the models and log. The precision and recall values of the model in Fig. 1 are 0.852 and 0.672,

¹ This simple example is inspired by a real world event log analyzed in [13].

respectively, while the precision and recall values of the model in Fig. 2 are 0.342 and 1.0, respectively; the values were obtained using the entropy-based measures presented in [20]. The values indicate, for instance, that the model in Fig. 2 is more permissive (has lower precision), i.e., encodes more behavior not seen in the log than the model in Fig. 1, and describes all the traces in the log (has perfect fitness of 1.0); the measures take values on the interval $[0, 1]$ with larger values showing better precision and fitness.

To assess the simplicity of discovered process models, measures of their structural complexity [4,9,15,17], such as the *number of nodes* and/or *edges*, *density*, *depth*, *coefficients of network connectivity*, and *control flow complexity*, can be employed. If one relies on the number of nodes to establish the simplicity of the two example models, then they will derive at the conclusion that they are equally simple, as both contain ten nodes. This conclusion is, however, naïve for at least two reasons: (i) the two models use ten nodes to encode different behaviors, and (ii) it may be unnecessary to use ten nodes to encode the corresponding behaviors.

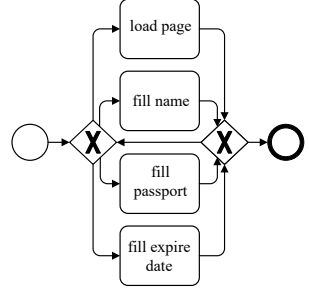


Fig. 3: A “flower” model.

The model in Fig. 3 describes the same behavior as the model in Fig. 2 using eight nodes. One can use different notions to establish similarity of the behaviors, including exact (e.g., trace equivalence) or approximate (e.g., topological entropy). The models in Figs. 2 and 3 are trace equivalent and specify the behaviors that have the (short-circuit) topological entropy of 1.0 [20]. Intuitively, the entropy measures the “variety” of traces of different lengths specified by the model. The more distinct traces of different lengths the model describes, the closer the entropy is to 1.0. The entropy of the model in Fig. 1 is 0.185. There is no block-structured BPMN model with unique task labels that describes the behavior with the entropy of 0.185 and uses less than ten nodes. Thus, we argue that the model in Fig. 1 should be accepted as such that is simpler than the model in Fig. 2.

3 A Framework for Estimating Simplicity of Process Models

In this section, we present our framework for estimating the simplicity of process models. The framework describes standard components that can be configured to result in a concrete measure of simplicity. The *simplicity framework* is a tuple $\mathcal{F} = (\mathcal{M}, \mathcal{C}, \mathcal{B})$, where \mathcal{M} is a collection of *process models*, $\mathcal{C} : \mathcal{M} \rightarrow \mathbb{R}_0^+$ is a measure of *structural complexity*, and $\mathcal{B} \subseteq (\mathcal{M} \times \mathcal{M})$ is a *behavioral equivalence* relation over \mathcal{M} .

The process models are captured using some process modeling language (representation bias), e.g., finite state machines [11], Petri nets [21], or BPMN [18]. The measure of structural complexity is a function that maps the models onto non-negative real numbers, with smaller assigned numbers indicating simpler models. For graph-based models, this can be the number of nodes and edges, density, diameter, or some other existing measure of simplicity used in process mining [17]. The behavioral equivalence relation \mathcal{B} must define an equivalence relation over \mathcal{M} , i.e., be reflexive, symmetric, and transitive. For instance, \mathcal{B} can be given by (weak or strong) bisimulation [12] or trace equivalence [11] relation over models. Alternatively, equivalence classes of \mathcal{B} can

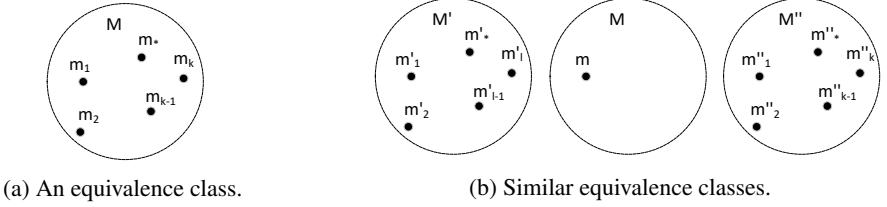


Fig. 4: Behavioral classes of model equivalence.

be defined by models with the same or similar measure of behavioral complexity, e.g., (short-circuit) topological entropy [20].

Given a model $m_1 \in \mathcal{M}$, its behavioral equivalence class per relation \mathcal{B} is the set $M = \{m \in \mathcal{M} \mid (m, m_1) \in \mathcal{B}\}$, cf. Fig. 4a. If one knows a model $m_* \in M$ with the lowest structural complexity in M , i.e., $\forall m \in M : \mathcal{C}(m_*) \leq \mathcal{C}(m)$, then they can put the simplicity of models in M into the perspective of the simplicity of m_* . For instance, one can use function $sim(m) = (\mathcal{C}(m_*)+1)/(\mathcal{C}(m)+1)$ to establish such a perspective.

Suppose that \mathcal{M} is the set of all block-structured BPMN models with four uniquely labeled tasks, \mathcal{B} is the trace equivalence relation, and \mathcal{C} is the measure of the number of nodes in the models. Then, it holds that $sim(m_1) = 1.0$ and $sim(m_2) = 9/11 = 0.818$, where m_1 and m_2 are the models from Fig. 1 and Fig. 2, respectively, indicating that m_1 is simpler than m_2 . To obtain these simplicity values, we used our tool and generated all the block-structured BPMN models over four uniquely labeled tasks, computed all the behavioral equivalence classes over the generated models, and collected statistics on the numbers of nodes in the models.

For some configurations of the framework, however, such exhaustive analysis may yield intractable. For instance, the collection of models of interest may be infinite, or finite but immense. Note that the number of block-structured BPMN models with four uniquely labeled tasks is 2,211,840, and is 297,271,296 if one considers models with five uniquely labeled tasks (the number of models grows exponentially with the number of allowed labels). In such cases, we suggest grounding the analysis in a representative subset $\mathcal{M}' \subset \mathcal{M}$ of the models.

Suppose that one analyzes model $m \in \mathcal{M}$ that has no other (or only a few) models in its equivalence class M , refer to Fig. 4b. Then, model m can be compared to models of lowest structural complexities m'_* and m''_* from some other equivalence classes M' and M'' which contain models that describe the behaviors “similar” to the one captured by m . To this end, one needs to establish a measure of “similarity” between the behavioral equivalence classes of models.

In the next section, we exemplify the discussed concepts by presenting example instantiations of the framework.

4 Framework Instantiations

In this section, we instantiate our framework $\mathcal{F} = (\mathcal{M}, \mathcal{C}, \mathcal{B})$ for assessing the simplicity of process models discovered from event logs and define the set of models (\mathcal{M}), structural complexity (\mathcal{C}), and the behavioral equivalence relation (\mathcal{B}) as follows:

- \mathcal{M} is a set of block-structured BPMN models with a fixed number of uniquely labeled tasks. BPMN is one of the most popular process modeling languages. Besides, block-structured uniquely labeled process models are discovered by Inductive miner — a widely used process discovery algorithm;
- \mathcal{C} is either the *number of nodes* or the *control flow complexity* measure. These measures were selected among other simplicity measures, because, as shown empirically in Section 5, there is a relation between these measures and the behavioral characteristics of process models; and
- \mathcal{B} is the behavioral equivalence relation induced by the notion of (short-circuit) *topological entropy* [20]. The entropy measure is selected because it maps process models onto non-negative real numbers that reflect the complexity of the behaviors they describe; the greater the entropy, the more variability is present in the underlying behavior. Consequently, models from an equivalence class of \mathcal{B} describe behaviors with the same (or very similar) entropy values.

For BPMN models the problem of minimization is still open and only some rules for local BPMN models simplification exist [14,22]. Although, NP-complete techniques [5] synthesizing Petri nets with minimal regions (corresponding to BPMN models [14] with minimal number of routing contracts) from the sets traces can be applied, there is no general algorithm for finding a *block-structured* BPMN model that contains a minimal possible number of nodes or has a minimal control flow complexity for a given process behavior. In this case, it may be feasible to generate the set of all possible process models for the given behavioral class (see the general description of this approach within our framework Section 3, Fig. 4a). However, due to the combinatorial explosion, the possible number of block-structured BPMN models grows exponentially with the number of tasks. While it is still possible to generate all block-structured BPMN models containing 4 or less tasks, for larger number of tasks this problem is computationally expensive and cannot be solved in any reasonable amount of time. In this work, we propose an approach that approximates the exact solutions by comparing analyzed models with only some randomly generated models that behave similarly. This approach implements a general approximation idea proposed within our framework (Section 3, Fig. 4b). Section 4.1 introduces basic notions used to describe this approach. Section 4.2 describes the proposed approach, discusses its parameters and analyzes dependencies between structural and behavioral characteristics of block-structured BPMN models.

4.1 Basic Notions

In this subsection, we define basic notions that are used later in this section.

Let X be a finite set of elements. By $\langle x_1, x_2, \dots, x_k \rangle$, where $x_1, x_2, \dots, x_k \in X$, $k \in \mathbb{N}_0$, we denote a finite *sequence* of elements over X of length k . X^* stands for the set of all finite sequences over X including the empty sequence of zero length.

Given two sequences $x = \langle x_1, x_2, \dots, x_k \rangle$ and $y = \langle y_1, y_2, \dots, y_m \rangle$, by $x \cdot y$ we denote *concatenation* of x and y , i.e., the sequence obtained by appending y to the end of x , i.e., $x \cdot y = \langle x_1, x_2, \dots, x_k, y_1, y_2, \dots, y_m \rangle$.

An *alphabet* is a nonempty finite set. The elements of an alphabet are its *labels*. A (formal) *language* L over an alphabet Σ is a (not necessarily finite) set of *sequences*,

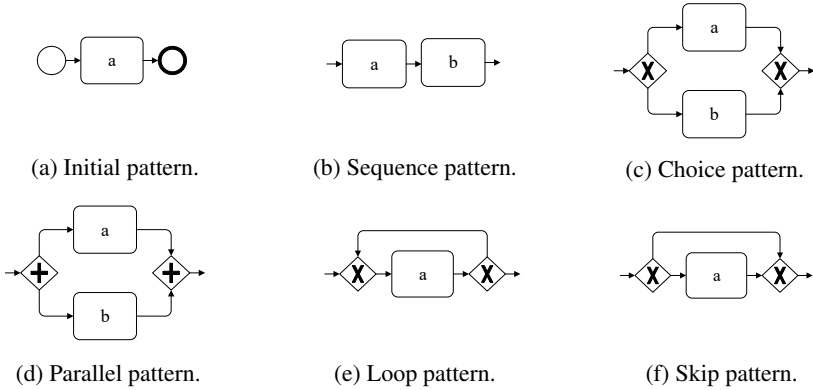


Fig. 5: Patterns of block-structured BPMN models.

over Σ , i.e., $L \subseteq \Sigma^*$. Let L_1 and L_2 be two languages. Then, $L_1 \circ L_2$ is their concatenation defined by $\{l_1 \cdot l_2 \mid l_1 \in L_1 \wedge l_2 \in L_2\}$. The language L^* is defined as $L^* = \bigcup_{n=0}^{\infty} L^n$, where $L^0 = \{\langle \rangle\}$, $L^n = L^{n-1} \circ L$.

Structural Representation. The class of process models considered in this work are *block-structured* BPMN models that are often used for the representation of processes discovered from event logs, e.g., these models are discovered by the Inductive mining algorithm [16].

Block-structured BPMN models are constructed from the following basic set of elements: *start* and *end events* represented by circles with thin and thick borders respectively and denoting beginning and termination of the process; *tasks* modeling atomic process steps and depicted by rounded rectangles with labels; routing *exclusive* and *parallel gateways* modeling exclusive and parallel executions and presented by diamonds; and *control flow arcs* that define the order in which elements are executed.

The investigated class of *block-structured* BPMN models consists of all and only BPMN models that:

- 1) can be constructed starting from the initial model presented in Fig. 5a and inductively replacing tasks with the patterns presented in Figs. 5b to 5f;
- 2) have *uniquely* labeled tasks, i.e., any two tasks have different labels;
- 3) only patterns other than loop can be applied to the nested task of the loop; only patterns other than skip and loop can be applied to the nested task of the skip pattern;

When constructing a model, the number of tasks increases if the patterns from Figs. 5b to 5d are applied, the patterns Figs. 5e and 5f can be applied no more than twice in a row, and the pattern Fig. 5a is applied only once. Hence, if we fix the number tasks (labels) in the investigated models, the the overall set of these models is finite.

After constructing the collection of models, local minimization rules are applied [14]. These rules merge gateways without changing the model semantics. An example of local reduction of gateways is presented in Fig. 6a, Fig. 6b illustrates merging of loop and skip constructs. For the detailed description of local minimization rules refer to [14].

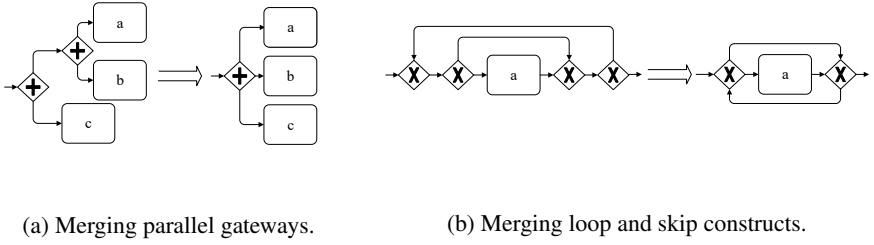


Fig. 6: Examples of applying local minimization rules.

We focus on the following complexity measures of block-structured BPMN models: (1) \mathcal{C}_n – the *number of nodes* (including start and end events, tasks, and gateways); (2) \mathcal{C}_{cfc} – the *control flow complexity* measure, which is defined as a sum of two numbers: the number of all splitting parallel gateways and the total number of all outgoing control flows of all splitting exclusive (choice) gateways [4].

Sequences of labels are used to encode executions of business processes. The ordering of tasks being executed defines the ordering of labels in a sequence. We say that a process model encodes or *accepts* a formal language if and only if this language contains all possible sequences of labels corresponding to the orderings of tasks being executed within the model and only them. Fig. 7a presents an example of a block-structured model m_1 that accepts language $L_1 = \{\langle a, b, c \rangle, \langle a, c, b \rangle, \langle b, a, c \rangle, \langle b, c, a \rangle, \langle c, a, b \rangle, \langle c, b, a \rangle\}$. A block-structured BPMN model m_2 accepting an infinite language of all sequences starting with a in alphabet $\{a, b, c\}$ is presented in Fig. 7b.

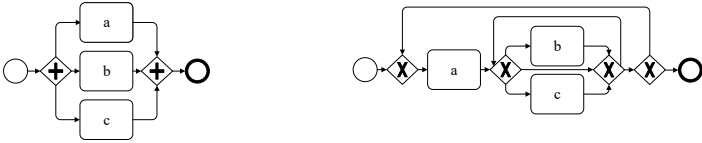
(a) Block-structured BPMN model m_1 .(b) Block-structured BPMN model m_2 .

Fig. 7: Examples of block-structured BPMN models.

Behavioral Representation. Next, we recall the notion of entropy which is used for the behavioral analysis of process models and event logs [20]. Let Σ be an alphabet and let $L \subseteq \Sigma^*$ be a language over this alphabet. We say that language L is *irreducible regular* language if and only if it is accepted by a strongly-connected automata model (for details refer to [6]). Let $C_n(L)$, $n \in \mathbb{N}_0$, be the set of all sequences in L of length n . Then, the *topological entropy* that estimates the cardinality of L by measuring the ratio of the number of distinct sequences in the language to the length of these sequences is defined as [6]:

$$ent(L) = \limsup_{n \rightarrow \infty} \frac{\log |C_n(L)|}{n}. \quad (1)$$

The languages accepted by block-structured BPMN models are *regular*, because they are also accepted by corresponding automata models [11]. But not all of them are *irreducible*, so the standard topological entropy (Eq. (1)) cannot be always calculated. To that end, in [20], it was proposed to construct an irreducible language $(L \circ \{\langle \chi \rangle\})^* \circ L$, where $\chi \notin \Sigma$, for each language L , and use so-called *short-circuit* entropy $\text{ent}\bullet(L) = \text{ent}((L \circ \{\langle \chi \rangle\})^* \circ L)$. Monotonicity of the short-circuit measure follows immediately from the definition of the short-circuit topological entropy and Lemma 4.7 in [20]:

Corollary 4.1 (Topological entropy). *Let L_1 and L_2 be two regular languages.*

1. *If $L_1 = L_2$, then $\text{ent}\bullet(L_1) = \text{ent}\bullet(L_2)$;*
2. *If $L_1 \subset L_2$, then $\text{ent}\bullet(L_1) < \text{ent}\bullet(L_2)$.*

Note that the opposite is not always true, i.e., different languages can be represented by the same entropy value. Although the language (trace) equivalence is stricter than the entropy-based equivalence, in the next section, we show that entropy is still useful for classifying the process behavior.

In this paper, we use the notion of *normalized* entropy. Suppose that L is a language over alphabet Σ , then the normalized entropy of L is defined as: $\overline{\text{ent}}(L) = \frac{\text{ent}\bullet(L)}{\text{ent}\bullet(\Sigma^*)}$, where Σ^* is the language containing all words over alphabet Σ . The normalized entropy value is bounded, because, for any language L it holds that $L \subseteq \Sigma^*$, and hence, by Corollary 4.1 $\text{ent}\bullet(L) \leq \text{ent}\bullet(\Sigma^*)$, consequently $\overline{\text{ent}}(L) \in [0, 1]$. Obviously, Corollary 4.1 can be formulated and applied to the normalized entropy measure, i.e., for two languages L_1 and L_2 over alphabet Σ , if $L_1 = L_2$, then $\overline{\text{ent}}(L_1) = \overline{\text{ent}}(L_2)$, and if $L_1 \subset L_2$, it holds that $\overline{\text{ent}}(L_1) < \overline{\text{ent}}(L_2)$.

We define the relation of behavioral equivalence \mathcal{B} using the normalized entropy. Let Σ be an alphabet and let $L_1, L_2 \subseteq \Sigma^*$ be languages accepted by models m_1 and m_2 respectively, $(m_1, m_2) \in \mathcal{B}$ if and only if $\overline{\text{ent}}(L_1) = \overline{\text{ent}}(L_2)$.

Normalized entropy not only allows to define the notion of behavioral equivalence, but also to formalize the notion of behavioral similarity. For a given parameter Δ , we say that two models m_1 and m_2 are *behaviorally similar* if and only if $|\overline{\text{ent}}(L_1) - \overline{\text{ent}}(L_2)| < \Delta$, where L_1 and L_2 are the languages these models accept.

4.2 Estimating Simplicity of Block-Structured BPMN Models

In this subsection, we devise a method for assessing the simplicity of uniquely labeled block-structured BPMN models. As no analytical method for synthesizing a “minimal” block-structured BPMN model in terms of number of nodes or control flow complexity for a given behavior is known, and no computationally feasible approach for generating all possible models with a given behavior exists, we propose an approach that investigates the dependencies between the structural and behavioral model characteristics empirically, and reuse these dependencies to measure the simplicity of models.

As the set of all models \mathcal{M} cannot be exhaustively constructed, we generate its subset $\mathcal{M}' \subset \mathcal{M}$ and relate analyzed models from \mathcal{M} with behaviorally similar models from \mathcal{M}' , producing an approximate solution. We then estimate the simplicity of the

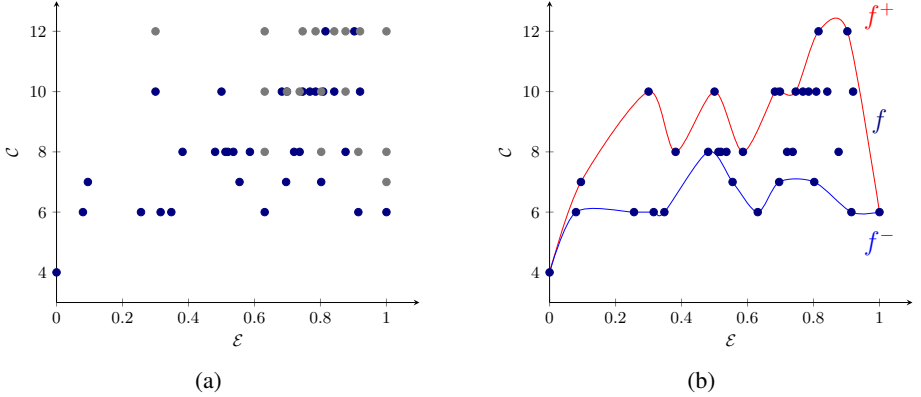


Fig. 8: Structural (\mathcal{C}) and behavioral (\mathcal{E}) characteristics of process models: (a) for all models in \mathcal{M}' and (b) filtered models with upper and lower envelopes for $\Delta = 0.05$.

given model by comparing its structural complexity with that of the simplest behaviorally similar models, with the complexity of these models being in a certain interval from “the best case” to “the worst case” complexity. In order to define this interval and relate it to entropy values, we construct envelope functions f^- and f^+ that approximate “the best case” and “the worst case” structural complexity of the simplest process models for a given entropy value. Below we give an approach for constructing these envelope functions and define the simplicity measure that relates the model complexity to an interval defined by these functions.

Let $\mathcal{E} : \mathcal{M}' \rightarrow [0, 1]$ be a function that maps process models in \mathcal{M}' onto the corresponding normalized entropy values. Fig. 8a presents an example plot relating structural characteristics $\mathcal{C}(m)$ and entropy $\mathcal{E}(m)$ values for each process model $m \in \mathcal{M}'$; the example is artificial and does not correspond to any concrete structural and behavioral measures of process models. Once such data points are obtained, we filter out all the models $m \in \mathcal{M}'$ such that $\exists m' \in \mathcal{M}' : \mathcal{E}(m) = \mathcal{E}(m')$ and $\mathcal{C}(m) > \mathcal{C}(m')$. In other words, we filter out a model, if its underlying behavior can be described in a structurally simpler model. This means that only the structurally “simplest” models remain. The process models that were filtered out are presented by gray dots in Fig. 8a.

Once the set \mathcal{M}'' of the models remaining after filtering the models in \mathcal{M}' is obtained, it defines the partial function $f : [0, 1] \dashrightarrow \mathbb{R}_0^+$, such that $f(e) = c$ if and only if exists a model $m \in \mathcal{M}''$, where $e = \mathcal{E}(m)$ and $c = \mathcal{C}(m)$ (Fig. 8b). This function relates the behavioral and structural characteristics of the remaining models. Then, we construct envelope functions that define intervals of the structural complexities of the remaining “simplest” models. The upper envelop $f^+ : [0, 1] \rightarrow \mathbb{R}_0^+$ is a function going through the set of data points $D^+ = \{(e, f(e)) \mid \forall e' \in \text{dom}(f) : (|e - e'| < \Delta \Rightarrow f(e) \geq f(e'))\}$, where Δ is a parameter that defines classes of behaviorally similar models. Less formally, f^+ goes through all the data points that are maximum in a Δ -size window. Similarly, the lower envelop is defined as a function $f^- : [0, 1] \rightarrow \mathbb{R}_0^+$ going through $D^- = \{(e, f(e)) \mid \forall e' \in \text{dom}(f) : (|e - e'| < \Delta \Rightarrow f(e) \leq f(e'))\}$. In

general, the envelope can be any smooth polynomial interpolation or a piecewise linear function, the only restriction is that it goes through D^+ and D^- data points.

Parameter Δ defines the measure of similarity between classes of behaviorally equivalent models. In each case, Δ should be selected empirically, for instance, too small Δ will lead to local evaluations, that may not be reliable because they do not take into account global trends, while setting too large Δ results in a situation when we do not take into account entropy, relating our model with all other models from the set. In Fig. 8b, the upper and lower envelopes were constructed for $\Delta = 0.05$.

Using the upper and lower envelope functions we can estimate simplicity of process models from \mathcal{M} . The simplicity measure is defined in Eq. (2), where $sim(m)$ is the simplicity of model $m \in \mathcal{M}$ with an entropy value $e = \mathcal{E}(m)$; α and β are parameters, such that $\alpha, \beta \in [0, 1]$ and $\alpha + \beta \leq 1$.

$$sim(m) = \begin{cases} \alpha \cdot \frac{f^+(e)}{\mathcal{C}(m)} & \mathcal{C}(m) \geq f^+(e) \\ \alpha + (1 - \alpha - \beta) \cdot \frac{(f^+(e) - \mathcal{C}(m))}{(f^+(e) - f^-(e))} & f^-(e) \leq \mathcal{C}(m) < f^+(e) \\ 1.0 - \beta \cdot \frac{\mathcal{C}(m)}{f^-(e)} & \mathcal{C}(m) < f^-(e) \end{cases} \quad (2)$$

According to Eq. (2), $sim(m)$ is in the interval between zero and one. Parameters α and β are used to adjust the measure. Parameter α shows the level of confidence that some complexity values can be above the upper envelope. If the complexity $\mathcal{C}(m)$ of model m is above the upper envelope $f^+(e)$, m is more complex than “the worst case” model, then $sim(m)$ is less than or equal to α and tends to zero as $\mathcal{C}(m)$ grows. If the model complexity $\mathcal{C}(m)$ is between the envelopes $f^-(e)$ and $f^+(e)$, then $sim(m) \in (\alpha, 1 - \beta]$ and the higher $\mathcal{C}(m)$ is (the closer the model is to “the worst case”), the closer the simplicity value to α . Parameter β shows the level of confidence that some data points may be below the lower envelope. If it is guaranteed that there are no models in \mathcal{M} with data points below the lower envelope it is feasible to set β to zero. Otherwise, if $\mathcal{C}(m)$ is lower than the lower envelope $f^-(e)$, then $sim(m)$ belongs to the interval $[1 - \beta, 1]$ and tends to one as $\mathcal{C}(m)$ approaches zero.

Next, we apply the proposed approach to construct upper and lower envelope functions for the number of nodes and control flow complexity measures of block-structured BPMN models with a fixed number of uniquely labeled tasks. To analyze the relations between the structural and behavioral characteristics, we generated all block-structured BPMN models with three tasks. Fig. 9 contains plots with data points representing the behavioral and structural characteristics of these process models. These data points, as well as the upper and lower envelopes, are constructed using the general technique described above with window parameter $\Delta = 0.01$. Additionally, to make the envelope functions less detailed and reflect the main trends, we construct them only through some of the data points from D^+ and D^- sets in such a way that the second derivative of each envelope function is either non-negative or non-positive, i.e., envelope functions are either convex or concave.

In both cases, the upper envelope functions f_n^+ and f_{cfc}^+ grow monotonically, starting at the sequential model (a sequence of three tasks) with the entropy of zero, reach the maximum, and then drop to the “flower” model (see an example of a “flower” model

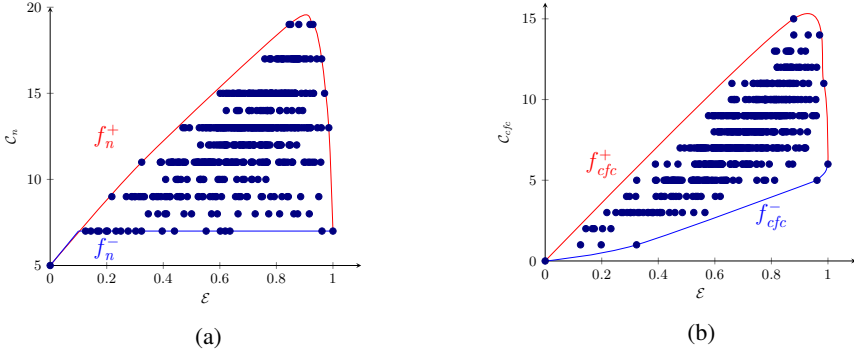


Fig. 9: Dependencies between entropy and structural complexity of block-structured BPMN models containing three tasks for the (a) number of nodes and (b) control flow complexity structural complexity measures.

in Fig. 3) with the entropy of one. These results show that, as the number of nodes (Fig. 9a) and complexity of the control flow (Fig. 9b) increase, the minimum possible entropy values also increase. As the model structure becomes more complex, more behavior is allowed and the lower bound of the entropy increases.

While the lower envelope for the number of nodes measure f_n^- (Fig. 9a) is flat and does not reveal any explicit dependency, the lower envelope for the control flow complexity f_{cfc}^- (Fig. 9b) grows. This can be explained by the fact that the number of nodes is reduced during the local model minimization [14], while the control flow complexity measure takes into account the number of outgoing arcs of exclusive splitting gateways and is not affected by the local minimization.

The empirical results and observations presented in this section reveal the main dependencies between the structural complexity and the behavioral complexity of block-structured BPMN models and can be generalized for an arbitrary number of tasks. They also show that this approach relies on the quality of the model generator, i.e., the set of generated models should be dense enough to reveal these dependencies. In the next section, we apply the proposed approach to assess the simplicity of process models discovered from real world event logs.

5 Evaluation

In this section, we use the approach from Section 4 to evaluate the simplicity of process models discovered by Inductive miner (with noise threshold 0.2) from industrial Business Process Intelligence Challenge (BPIC) event logs² and an event log of a booking flight system (BFS). Before the analysis, we filtered out infrequent events that appear less than in 80% of traces using the “Filter Log Using Simple Heuristics” Process Mining Framework (ProM) plug-in [7].³ The Inductive miner algorithm discovers uniquely

² BPIC logs: https://data.4tu.nl/repository/collection:event_logs_real.

³ The filtered logs are available here: <https://github.com/jbpt/codebase/tree/master/jbpt-pm/logs>.

labeled block-structured BPMN models. As structural complexity measures, we used the number of nodes and the control flow complexity (*cfc*).

To estimate the upper and lower bounds of the structural complexity of the models, for each number of tasks (each size of the log alphabet), we generated 5,000 random uniquely labeled block-structured BPMN models. For all the models, data points representing the entropy and structural complexity of the models were constructed. We then constructed the upper and lower envelopes using the window of size 0.01, refer to Section 4 for the details. The envelopes were constructed as piecewise linear functions going through all the selected data points.

For both structural complexity measures, parameter α , cf. Eq. (2), was set to 0.5, i.e., models represented by the data points above the upper envelope are presumed to have simplicity characteristic lower than 0.5. In turn, parameter β was set to 0.0 and 0.1 for the *number of nodes* and *cfc* measure, respectively. In contrast to *cfc*, the lower envelope for the *number of nodes* measure is defined as the minimal possible number of nodes in any model, and this guarantees that there are no data points below it. The final equations for the *number of nodes* and *cfc* simplicity measures, where $e = \mathcal{E}(m)$ is the topological entropy of model m , $\mathcal{C}_n(m)$ is the *number of nodes* in m , $\mathcal{C}_{cfc}(m)$ is the *cfc* complexity of m , f_n^+ and f_{cfc}^+ are the upper envelopes for the *number of nodes* and *cfc* measures, and f_n^- and f_{cfc}^- are the corresponding lower envelopes, are given below.

$$sim_n(m) = \begin{cases} 0.5 \cdot \frac{f_n^+(e)}{\mathcal{C}_n(m)} & \mathcal{C}_n(m) \geq f_n^+(e) \\ 0.5 + 0.5 \cdot \frac{(f_n^+(e) - \mathcal{C}_n(m))}{(f_n^+(e) - f_n^-(e))} & f_n^-(e) \leq \mathcal{C}_n(m) < f_n^+(e) \\ 1.0 & \mathcal{C}_n(m) < f_n^-(e) \end{cases} \quad (3)$$

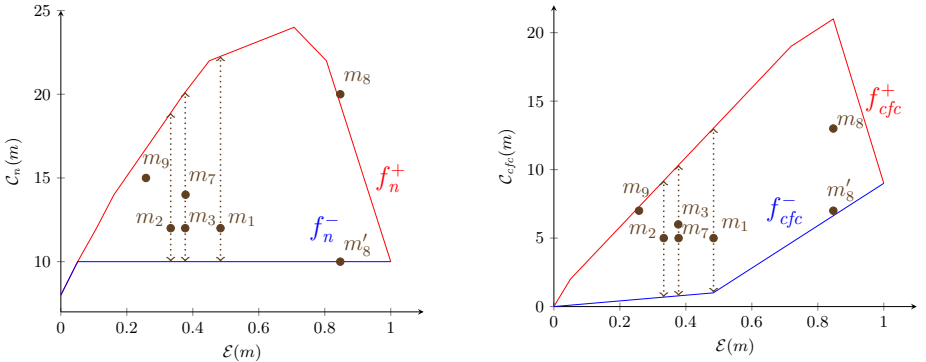
$$sim_{cfc}(m) = \begin{cases} 0.5 \cdot \frac{f_{cfc}^+(e)}{\mathcal{C}_{cfc}(m)} & \mathcal{C}_{cfc}(m) \geq f_{cfc}^+(e) \\ 0.5 + 0.4 \cdot \frac{(f_{cfc}^+(e) - \mathcal{C}_{cfc}(m))}{(f_{cfc}^+(e) - f_{cfc}^-(e))} & f_{cfc}^-(e) \leq \mathcal{C}_{cfc}(m) < f_{cfc}^+(e) \\ 1.0 - 0.1 \cdot \frac{\mathcal{C}_{cfc}(m)}{f_{cfc}^-(e)} & \mathcal{C}_{cfc}(m) < f_{cfc}^-(e) \end{cases} \quad (4)$$

Table 1 presents the original and adjusted (proposed in this paper) simplicity measures, induced by the *number of nodes* and *cfc* structural complexity measures, for the process models discovered from the evaluated event logs. Models were discovered from the filtered event logs and their sublogs that contain only traces appearing in the filtered event logs at least two or four times. Model m_8 (refer to Fig. 11a) is an automatically discovered process model with redundant nodes. The value of $sim_n(m_8)$ is less than 0.5 because the corresponding data point is above the upper envelope (Fig. 10a). Note that the manually constructed “flower” model m'_8 (Fig. 11) that accepts the same traces as m_8 has better structural complexity and, consequently, the corresponding adjusted simplicity measurements relate as follows: $sim_n(m'_8) = 1.0 > sim_n(m_8) = 0.485$, and $sim_{cfc}(m'_8) = 0.890 > sim_{cfc}(m_8) = 0.723$. The difference between $sim_{cfc}(m_8)$ and $sim_{cfc}(m'_8)$ simplicity values is not as significant as the difference between $sim_n(m_8)$ and $sim_n(m'_8)$, as despite m'_8 has only two gateways the total number of outgoing sequence flows from these gateways is rather high.

Model	Event log	#Traces	#Labels	Entropy	\mathcal{C}_n	sim_n	\mathcal{C}_{cfc}	sim_{cfc}
m_1	BPIC'2019	3,365	6	0.484	12	0.923	5	0.767
m_2	BPIC'2019	614	6	0.333	12	0.887	5	0.697
m_3	BPIC'2019	302	6	0.377	12	0.901	6	0.714
m_4	BPIC'2018	15,536	8	0.800	25	0.684	20	0.690
m_5	BPIC'2018	1,570	7	0.432	16	0.802	10	0.680
m_6	BPIC'2018	618	7	0.638	18	0.813	15	0.676
m_7	BFS	279	6	0.378	14	0.754	5	0.723
m_8	BFS	70	6	0.847	20	0.485	13	0.723
m_9	BFS	29	6	0.258	15	0.630	7	0.516

Table 1: Simplicity of uniquely labeled block-structured BPMN models discovered from industrial event logs and their sublogs; the number of unique traces ($\#Traces$), number of distinct labels in the discovered models ($\#Labels$) and their entropy values ($Entropy$) are specified.

Models m_1 , m_2 , and m_3 have the same number of nodes, but different entropy values. Model m_1 is considered the simplest in terms of the number of nodes among the three models because it is located further from the upper envelope than the other two models. Hence, its sim_n value is the highest. Models m_1 and m_2 , which in addition have the same control flow complexity values, are shown in Fig. 12a and Fig. 12b, respectively. Model m_1 is considered more simple than model m_2 because it merely runs all the tasks in parallel (allowing “Record Service Entry Sheet” task to be skipped or executed several times). In contrast, model m_2 adds additional constraints on the order of tasks (leading to lower entropy) and, thus, should be easier to test. Note that m_1 models a more diverse behavior which in the worst case, according to the upper envelope, can be modeled with more nodes. These results demonstrate that when analyzing the simplicity of a discovered process model, it is feasible and beneficial to consider both phenomena of the structural complexity of the model’s diagrammatic representation and the variability/complexity of the behavior the model describes. This way, one can adhere to the Occam’s Razor problem-solving principle.



(a) Number of nodes.

(b) Control flow complexity (cfc).

Fig. 10: Structural complexity of block-structured BPMN models over six labels.

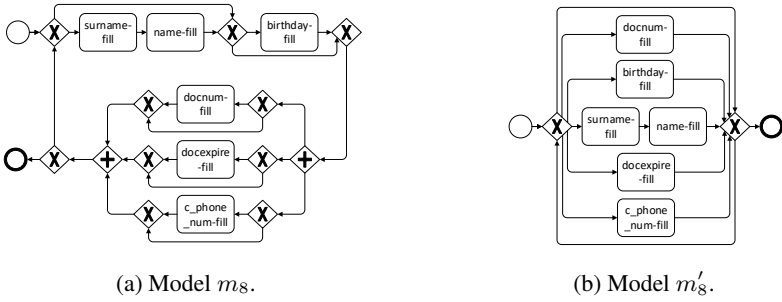


Fig. 11: Models m_8 and m'_8 discovered from the BFS event log.

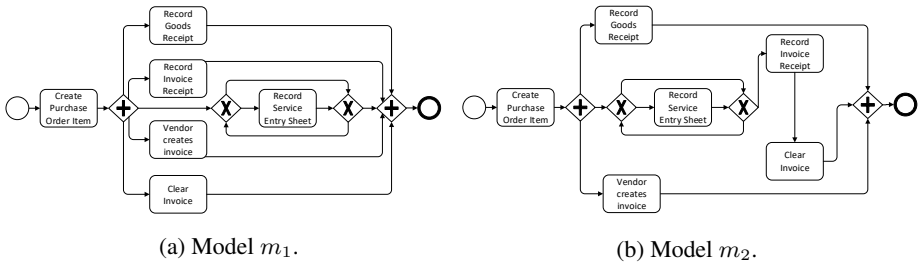


Fig. 12: Models m_1 and m_2 discovered from the BPIC'2019 event log.

6 Conclusion

This paper presents a framework that can be configured to result in a concrete approach for measuring the simplicity of process models discovered from event logs. In contrast to the existing simplicity measures, our framework accounts for both a model's structure and behavior. In this paper, the framework was implemented for the class of uniquely-labeled block-structured BPMN models using topological entropy as a measure of process model behavior. The experimental evaluation of process models discovered from real-life event logs shows the approach's ability to evaluate the quality of discovered process models by relating their structural complexity to the structural complexity of other process models that describe similar behaviors. Such analysis can complement existing simplicity measurement techniques showing the relative aspects of the structural complexity of the model.

We identify several research directions arising from this work. First, we acknowledge that the proposed instantiation of the framework is approximate and depends on the quality of the randomly generated models. The analysis of other structural complexity measures as well as more sophisticated random model generation algorithms can lead to a more precise approach. Second, the framework described in this paper can be instantiated with other classes of process models to extend its applicability to models discovered by a broader range of process discovery algorithms. Finally, we believe that this work can give valuable insights into the improvement of existing, and the development of new process discovery algorithms.

Acknowledgments. This work was supported by the Australian Research Council Discovery Project DP180102839. We sincerely thank the anonymous reviewers whose suggestions helped us to improve this paper.

References

1. van der Aalst, W.: *Data Science in Action*, pp. 3–23. Springer Berlin Heidelberg (2016)
2. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. *Knowl. Inf. Syst.* **59**(2), 251–284 (2019). <https://doi.org/10.1007/s10115-018-1214-x>
3. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: On the role of fitness, precision, generalization and simplicity in process discovery. In: *On the Move to Meaningful Internet Systems: OTM 2012*. pp. 305–322. Springer, Berlin (2012)
4. Cardoso, J.: How to Measure the Control-flow Complexity of Web processes and Workflows, pp. 199–212 (01 2005)
5. Carmona, J., Cortadella, J., Kishinevsky, M.: A region-based algorithm for discovering petri nets from event logs. In: Dumas, M., Reichert, M., Shan, M.C. (eds.) *Business Process Management*. pp. 358–373. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
6. Ceccherini-Silberstein, T., Machì, A., Scarabotti, F.: On the entropy of regular languages. *Theor. Comp. Sci.* **307**, 93–102 (2003)
7. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: *Applications and Theory of Petri Nets 2005*. pp. 444–454. Springer Berlin Heidelberg (2005)
8. Garrett, A.J.M.: Ockham’s Razor, pp. 357–364. Springer Netherlands (1991)
9. Gruhn, V., Laue, R.: Complexity metrics for business process models. In: *9th International Conference on Business Information Systems (BIS 2006)*. pp. 1–12 (2006)
10. Grünwald, P.D.: *The Minimum Description Length Principle (Adaptive Computation and Machine Learning)*. The MIT Press (2007)
11. Hopcroft, J.E., Motwani, R., Ullman, J.D.: *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA (2006)
12. Jančar, P., Kučera, A., Mayr, R.: Deciding bisimulation-like equivalences with finite-state processes. In: *Automata, Languages and Programming*. pp. 200–211. Springer (1998)
13. Kalenkova, A.A., Ageev, A.A., Lomazova, I.A., van der Aalst, W.M.P.: E-government services: Comparing real and expected user behavior. In: *Business Process Management Workshops*. pp. 484–496. Springer, Cham (2018)
14. Kalenkova, A., Aalst, W., Lomazova, I., Rubin, V.: Process mining using BPMN: Relating event logs and process models process mining using BPMN. Relating event logs and process models. *Software and Systems Modeling* **16**, 1019–1048 (01 2017)
15. Kluza, K., Nalepa, G.J., Lisiecki, J.: Square Complexity Metrics for Business Process Models, pp. 89–107. Springer International Publishing, Cham (2014)
16. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: *Application and Theory of Petri Nets and Concurrency*. pp. 311–329. Springer Berlin Heidelberg (2013)
17. Lieben, J., Jouck, T., Depaire, B., Jans, M.: An improved way for measuring simplicity during process discovery. In: *EOMAS*. pp. 49–62. Springer (2018)
18. OMG: *Business Process Model and Notation (BPMN), Version 2.0.2* (Dec 2013), <http://www.omg.org/spec/BPMN/2.0.2>
19. Polyvyanyy, A.: *Structuring Process Models*. Ph.D. thesis, University of Potsdam (2012), <http://opus.kobv.de/ubp/volltexte/2012/5902/>

20. Polyvyanyy, A., Solti, A., Weidlich, M., Ciccio, C.D., Mendling, J.: Monotone precision and recall measures for comparing executions and specifications of dynamic systems. *ACM Trans. Softw. Eng. Methodol.* **29**(3) (Jun 2020). <https://doi.org/10.1145/3387909>
21. Reisig, W.: *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*. Springer Publishing Company, Incorporated (2013)
22. Wynn, M.T., Verbeek, H.M.W., van der Aalst, W.M.P., ter Hofstede, A.H.M., Edmond, D.: Reduction rules for yawl workflows with cancellation regions and or-joins. *Inf. Softw. Technol.* **51**(6), 1010–1020 (Jun 2009)