Navigating complexity: comparing complexity measures with Weyuker's properties

Patrizia Schalk, Adam Burke, Robert Lorenz

Angaben zur Veröffentlichung / Publication details:

Schalk, Patrizia, Adam Burke, and Robert Lorenz. 2024. "Navigating complexity: comparing complexity measures with Weyuker's properties." In 2024 6th International Conference on Process Mining (ICPM), 14-18 October 2024, Kgs. Lyngby, Denmark, edited by Xixi Lu, Luise Pufahl, and Minseok Song, 145–52. Piscataway, NJ: IEEE. https://doi.org/10.1109/icpm63005.2024.10680655.

Navigating Complexity: Comparing Complexity Measures With Weyuker's Properties

1st Patrizia Schalk
University of Augsburg
Augsburg, Germany
patrizia.schalk@uni-a.de

2nd Adam Burke
Queensland University of Technology
Brisbane, Australia
at.burke@qut.edu.au

3rd Robert Lorenz
University of Augsburg
Augsburg, Germany
robert.lorenz@uni-a.de

***Abstract*—A good process model is expected not only to reflect the behavior of the process, but also to be as easy to read and understand as possible. Because preferences vary across different applications, numerous measures provide ways to reflect the complexity of a model with a numeric score. However, this abundance of different complexity measures makes it difficult to select one for analysis. Furthermore, most complexity measures are defined only for BPMN or EPC, but not for workflow nets.**

This paper adapts existing complexity measures to the world of workflow nets. It then compares these measures with a set of properties originally defined for software complexity, as well as new extensions to it. We discuss the importance of the properties in theory by evaluating whether matured complexity measures should fulfill them or whether they are optional. We find that not all inspected properties are mandatory, but also demonstrate that the behavior of evolutionary process discovery algorithms is influenced by some of these properties. Our findings help analysts to choose the right complexity measure for their specific use-case.

***Index Terms*—complexity, simplicity, understandability, workflow nets, Petri nets, process mining, process discovery**

I. Introduction

Process discovery concerns finding a model for a business process [15]. The goal is to automatically construct an understandable model that contains all relevant behavior, so stakeholders and process analysts can make business decisions. Especially large processes that contain many sub-processes tend to produce cluttered and complex models. This is one of the reasons why the Directly Follows Miner [6] is so popular in practice: Directly Follows Graphs have only one type of node and easy semantics for the arrows. However, this model type struggles with distinguishing concurrency from loops. BPMN, EPCs and workflow nets, on the other hand, feature easy ways to model concurrency but have several types of nodes that each have different semantics. Consequently, researchers have developed complexity measures for these model types to evaluate which combinations of nodes make models difficult to understand. These measures tend to count one type of complex structure in a model. Because there are many types of such structures, numerous complexity measures were proposed [9]. Formal properties for such measures help us understand their behavior. Creators of new measures can and should clearly state which properties the measure satisfies, and designers of discovery algorithms may prove desirable simplicity properties are always maintained. The same is true for software programs,

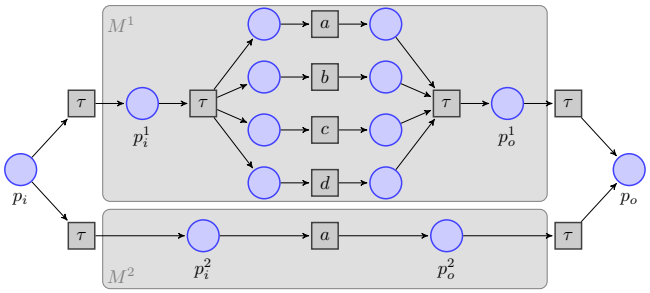


Fig. 1: A workflow net M with 14 places and 11 transitions.

so as a step towards this ideal setting, Weyuker [18] introduced nine formal properties that complexity measures for software should fulfill. We find six of these properties useful for process models as well, while three are less meaningful for this setting. We also draw on work from process mining on properties for complexity [2] and other quality dimensions [16], and propose new properties. Since process discovery algorithms benefit most from formal properties for complexity measures, we perform the analyses on workflow nets, like that in Figure 1, which have a strong theory and are therefore often returned by discovery algorithms. Our analyses hold for BPMN and EPCs as well, since workflow nets can be translated into these modeling languages [3]. The paper is structured as follows: Section II gives an overview of the analyses already performed on complexity measures, before Section III presents the basic definitions we need for our analyses. Section IV then defines the properties that we use for the comparative analysis of the complexity measures defined in Section V. In Section VI, we show and discuss the result's implications for the properties, complexity measures and evolutionary discovery algorithms. Section VII concludes the paper.

II. Related Work

There are many factors that make a process model difficult to understand. Mendling [9] identified 28 such factors and defined complexity measures for EPC models based on them. He demonstrated their relevance by showing that these complexity measures predict modeling errors in the SAP reference. Reijers et al. [11] found that some of these measures are also tied to the understandability of EPC models by conducting a case study with students of three universities. According to

their study, especially the number of nodes, the density of the network, the average degree of connectors and the cross-connectivity between nodes influence how understandable a process model is. Lieben et al. [8] analyzed if existing complexity measures often agree in their score. They found that they are not completely distinct, but essentially cover four dimensions of complexity, which they call TOKEN BEHAVIOR COMPLEXITY, NODE IO COMPLEXITY, PATH COMPLEXITY and DEGREE OF CONNECTEDNESS. They conclude that it is not necessary to use every complexity measure to adequately evaluate the complexity of a model. Instead, it is sufficient to use one complexity measure of each dimension, as well as two measures that do not fit to any dimension.

Until now, little is known about formal properties of complexity measures. For the quality dimensions fitness, precision and generalization, Syring et al. [14] use desirable properties defined by van der Aalst [16] to compare measures. Since they focus on the behavior and abstract from the representation of the model, they did not include properties for complexity measures. Weyuker [18], on the other hand, focuses on properties for software complexity measures. Because software and process models use similar control structures, these properties are a good starting point to analyze complexity measures for process models. Cardoso[2] therefore analyzed the control flow complexity measure with Weyuker's properties.

Whether a complexity measure fulfills formal properties is important for algorithms that find process models by optimizing over quality criteria. An example of such a discovery algorithm is the Evolutionary Tree Miner (ETM) [1]. This evolutionary algorithm continuously mutates a randomly generated set of process models until a model with good fitness, precision, generalization, and simplicity is found. The importance of each dimension can be set by weights, and the concrete measures can be chosen freely. In turn, the results of the ETM depend on the chosen quality measures and their properties. Because properties of the quality dimensions fitness, precision, and generalization are already analyzed, we focus on evaluating whether Weyuker's properties [18] are useful for complexity measures of process models.

III. PRELIMINARIES

Let $\mathbb{N} := \{1, 2, \dots\}$ and $\mathbb{N}_0 := \mathbb{N} \cup \{0\}$. Further, let $\mathbb{R}$ be the set of real numbers and $\mathbb{R}_0^+ := \{r \in \mathbb{R} \mid r \geq 0\}$. To define workflow nets, we first need to define Petri nets.

Definition 1 (Petri net)**. —** *A (simple)* Petri net *is a 3-tuple $N = (P, T, F)$, where P is a finite set of* places*, T is a finite set of* transitions*, $P \cap T = \emptyset$ and F is a* flow relation *with $F \subseteq (P \times T) \cup (T \times P)$. For any place $p \in P$ of N, we call ${}^\bullet p = \{t \in T \mid (t, p) \in F\}$ the* preset of p *and $p^\bullet = \{t \in T \mid (p, t) \in F\}$ the* postset of p*. We define the pre- and postset of transitions $t \in T$ accordingly.*

Workflow nets are frequently used for the modelling of business processes. In this paper, we focus on *labeled workflow nets*—a special subclass of Petri nets—for our analyses of complexity measures.

Definition 2 (Workflow net)**. —** *A labeled* workflow net *is a 6-tuple $W = (P, T, F, \ell, p_i, p_o)$ where (P, T, F) is a Petri net, $p_i, p_o \in P$ with $p_i \neq p_o$,*

- *p_i is the only place in W for which $|{}^\bullet p_i| = 0$,*
- *p_o is the only place in W for which $|p_o^\bullet| = 0$,*
- *every node in $P \cup T$ lies on some path from p_i to p_o*

and, for a finite alphabet A with $\tau \notin A$, $\ell : T \to A \cup \{\tau\}$ is a labeling function, assigning a transition label to each transition. We call $t \in T$ with $\ell(t) = \tau$ a silent transition*.*

With $\mathcal{M}$, we denote the set of all possible workflow nets. Figure 1 illustrates a workflow net with 14 places, drawn as circles, and 11 transitions, drawn as squares. Many complexity measures take connectors for choices or parallelism into account. Therefore, to make our analyses also valid for BPMN and EPC models, we define which kinds of structures in a workflow net we understand as a connector.

Definition 3 (Connectors in workflow nets)**. —** *Take a workflow net $W = (P, T, F, \ell, p_i, p_o)$, where $t \in T$ and $p \in P$.*

- *If $|p^\bullet| > 1$, we call p an* xor-split*.*
- *If $|{}^\bullet p| > 1$, we call p an* xor-join*.*
- *If $|t^\bullet| > 1$, we call t an* and-split*.*
- *If $|{}^\bullet t| > 1$, we call t an* and-join*.*

Accordingly, we define

- *the set of xor-splits in W as $\mathcal{S}_{xor}^W := \{p \in P \mid |p^\bullet| > 1\}$,*
- *the set of xor-joins in W as $\mathcal{J}_{xor}^W := \{p \in P \mid |{}^\bullet p| > 1\}$,*
- *the set of and-splits in W as $\mathcal{S}_{and}^W := \{t \in T \mid |t^\bullet| > 1\}$,*
- *the set of and-joins in W as $\mathcal{J}_{and}^W := \{t \in T \mid |{}^\bullet t| > 1\}$.*

Note that these sets are not necessarily disjoint. The set of xor-connectors in W is $\mathcal{C}_{xor}^W := \mathcal{S}_{xor}^W \cup \mathcal{J}_{xor}^W$, the set of and-connectors in W is $\mathcal{C}_{and}^W := \mathcal{S}_{and}^W \cup \mathcal{J}_{and}^W$ and the set of all connectors is $\mathcal{C}^W := \mathcal{C}_{xor}^W \cup \mathcal{C}_{and}^W$.

In Definition 3, we do not define or-connectors for workflow nets. This is because there are multiple ways to model an or-connector in this model type. However, all of these ways use parallel splits and exclusive choices, so if a complexity measure punishes a connector representing an inclusive choice, it can punish those connectors instead. The complexity of a model may not only depend on the connector types but also on the labels, leading to the next definition.

Definition 4 (Relabeling of a Workflow net)**. —** *Let A and B be two finite alphabets. Let $W = (P, T, F, \ell, p_i, p_o) \in \mathcal{M}$ be a labeled workflow net with $\ell : T \to A \cup \{\tau\}$. We call a labeling $\ell' : T \to B \cup \{\tau\}$ a* uniform relabeling *of ℓ if for all transitions $t_1, t_2 \in T : \ell(t_1) = \ell(t_2) \leftrightarrow \ell'(t_1) = \ell'(t_2)$ and if for all $t \in T : \ell(t) = \tau \leftrightarrow \ell'(t) = \tau$. We denote the set of all uniform relabelings of ℓ by $\mathcal{R}_\ell$. For any $\ell' \in \mathcal{R}_\ell$, we call $W_{\ell'} = (P, T, F, \ell', p_i, p_o)$ a* relabeling *of W.*

For one of Weyuker's properties, we need to check if changing the order of activities impacts a complexity measure. Cardoso [2] implements this by allowing activities to change labels, and connectors to change type. We exploit the expressiveness of workflow nets and define a permutation as a net with the same transitions, but different control flow.

Definition 5 (Permutations of Workflow Nets)**.** — *Let W be a workflow net with $W = (P, T, F, \ell, p_i, p_o)$, P' be a set of places, $F' \subseteq (P' \times T) \cup (T \times P')$ be an arbitrary flow relation and $p_i', p_o' \in P'$ two places in P'. We call $Perm(W) := \{W' = (P', T, F', \ell, p_i', p_o') \mid W' \in \mathcal{M}\}$ the set of* permutations *of W.*

To keep the definition of permutations as restrictive as possible, we require all transitions to remain the same in a permuted net—including τ-transitions. As it turns out in Section VI, loosening the definition by allowing to add or remove τ-transitions would not give us new insights.

One of Weyuker's properties requires us to be able to combine two or more models. We take inspiration from the creation of block-structured workflow nets [7] and use the operations for sequential composition ($\rightarrow$), parallel composition ($\wedge$), exclusive choice ($\times$) and iteration ($\circlearrowleft$). Figure 2 shows how these operations combine arbitrary workflow nets. As an example, the net M of Figure 1 is obtained by combining the nets M^1 and M^2 of the same figure with an $\times$-operation. For a formal definition, we refer to the original work on block-structured workflow nets [7] or the extended version of this paper [13].

We have two reasons to focus on these operations: First, they are very well known and established due to the inductive miner [7],[15, p.222] relying on them, so many process analysts are already familiar with them. Second, the ETM heavily relies on these operations, which gives us the opportunity to directly test the impact of certain properties on a well-established discovery algorithm. The ETM does not operate directly on workflow nets, but on *process trees* [15], which can easily be converted to workflow nets with the operations shown in Figure 2. For example, the process tree T_1 in Figure 4 represents the workflow net M^1 of Figure 1, while the process tree T_2 of Figure 4 represents the workflow net M of Figure 1.

Finally, for this section, we define complexity measures. Existing complexity measures all are functions that take a model as input and return a *complexity score*—a real value that reflects how complex the net is.

Definition 6 (Complexity Measure)**.** — *A* complexity measure *C is a function $C : \mathcal{M} \rightarrow \mathbb{R}$.*

Complexity influences many properties of a process model, like its understandability, correctness, or the time needed to execute certain algorithms.

IV. Properties for Complexity Measures

In the first part of this section, we define Weyuker's properties for workflow nets. Since this set of properties is not intended to be complete, we propose simple extensions that further deepen the understanding of the analyzed complexity measures in a second part. Our analyses show that the first four of Weyuker's properties are trivially fulfilled by existing complexity measures. They state that complexity measures should be able to give more than one score (W1), give only finitely many models the same score (W2), should allow two different models to get the same score (W3) and should be independent of the model's language (W4). We therefore omit these properties in this paper and refer the interested reader to the extended version of this paper [13]. In the following, we take C as a placeholder for any complexity measure.

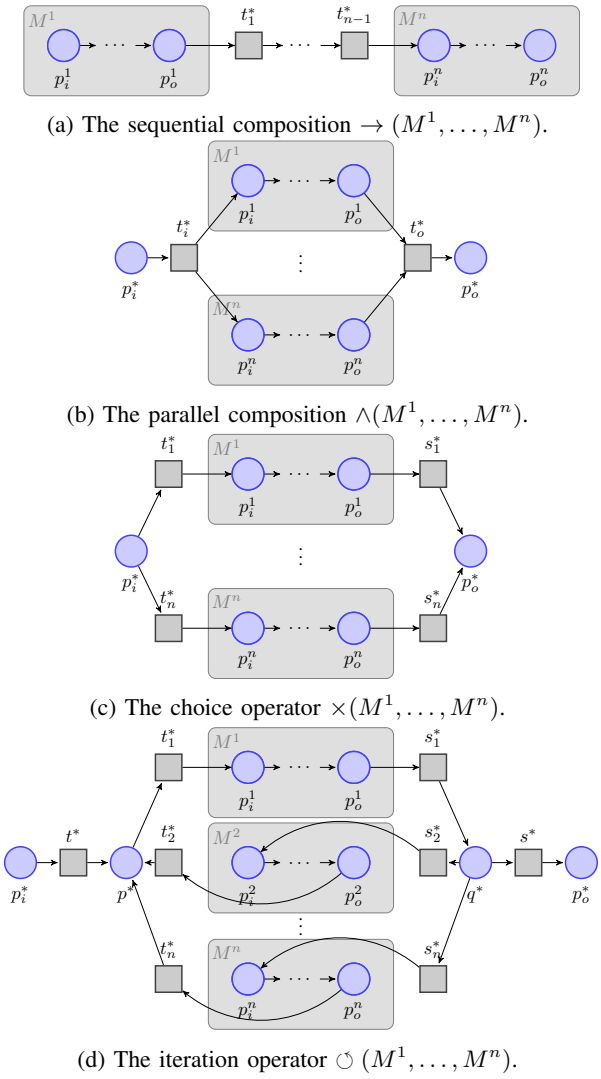


Fig. 2: The four compositional operations used in this paper.

A. Properties of Weyuker

It seems sensible that a small part of a process model should not be more complex than the entire model. For example, the complexity of the model M^1 of Figure 1 should not be higher than the complexity of the entire model, M. Weyuker therefore proposes that complexity measures should be monotone in this sense. Van der Aalst [16] agrees with this, as his properties **RecPro3**, **PrecPro1** and **GenPro3** are closely related to MON.

Property 1 (MON (W5))**.** — *C is **monotone**, i.e. the complexity score of a composed model cannot be lower than the complexity score of its parts: Let $\mathcal{O} := \{\rightarrow, \wedge, \times, \circlearrowleft\}$. For any $M_1, \ldots, M_n \in \mathcal{M}$ and $M_i \in \{M_1, \ldots, M_n\}$ we have:*

$$\forall \oplus \in \mathcal{O} : C(\oplus(M_1, \ldots, M_n)) \geq C(M_i)$$

For the next property, suppose we have two models with the same complexity score. Weyuker argues that composing each of them with a third one does not necessarily lead to the same complexity score, since one part of a model can influence the complexity of another part. We can observe this with the nets in Figure 1: The fact that the activity a of M^2 is also present in M^1 increases the perceived complexity of M. If M^1 would not contain the activity label a, the net would be simpler. It is desirable that a complexity measure finds such dependencies.

Property 2 (COMP (W6)). — *C is **composition sensitive**, i.e. there are process models of equal complexity according to C, whose complexity differ when composed with a third process model. Let $\mathcal{O} := \{\rightarrow, \wedge, \times, \circlearrowleft\}$.*

$$\forall \oplus \in \mathcal{O} : \exists M_1, M_2, M_3 \in \mathcal{M} : C(M_1) = C(M_2) \\ \wedge\, C(M_1 \oplus M_3) \neq C(M_2 \oplus M_3)$$

One way of transforming a program is to reorder its instructions. Weyuker argues that such a reordering should impact the complexity. This is also true for process models: If we add more arcs to M in Figure 1, it looks more complex.

Property 3 (PERM (W7)). — *C is sensitive for **permutations**, i.e. changing the start- and end-points of arcs in a workflow net can have an impact on the complexity score:*

$$\exists M \in \mathcal{M} : \exists M' \in Perm(M) : C(M) \neq C(M')$$

Weyuker states that renaming the variables of a program should not impact its complexity. Cardoso [2] translates the renaming of variables to the uniform renaming of activity names. We also use this translation for workflow nets.

Property 4 (ROB$_\ell$ (W8)). — *C is **robust** against relabelings, i.e. uniformly changing the labeling does not affect complexity:*

$$\forall M = (P, T, F, \ell, p_i, p_o) \in \mathcal{M} : \forall \ell' \in \mathcal{R}_\ell : C(M) = C(M_{\ell'})$$

The final property concerns the combination of two process models. According to Weyuker, it should be possible that the complexity of a composed model is higher than the sum of complexity scores of the parts.

Property 5 (¬SUB (W9)). — *C is **not subadditive**, i.e. the complexity score of a composed model can be greater than the sum of the complexity of its parts: Let $\mathcal{O} := \{\rightarrow, \wedge, \times, \circlearrowleft\}$.*

$$\forall \oplus \in \mathcal{O} : \exists M_1, M_2 \in \mathcal{M} : C(M_1 \oplus M_2) > C(M_1) + C(M_2)$$

The ¬SUB property is especially interesting if a measure does not fulfill it. In this case, we know that the complexity score of a combination of two nets is always less or equal to the sum of complexity scores of the parts.

B. Extensions for Weyuker's Properties

A simple but useful property for the complexity of Petri nets was introduced by Morasca [10], who states that a complexity measure should always be defined and return non-negative values. Obviously, this property is also important for complexity measures of process models, since negative complexity scores would be difficult to interpret.

Property 6 (DEF). — *The complexity measure C is **defined** for every process model M and returns non-negative values:*

$$\forall M \in \mathcal{M} : C(M) \in \mathbb{R}_0^+$$

Often, we want a complexity measure to have a minimum value to avoid generating models with peculiar structures that lower the complexity. For example, if we want to lower the density of a model, we can add a long chain of τ-transitions after the initial place. A complexity measure not having a minimum value is a hint that this can happen.

Property 7 (MIN). — *The complexity measure C has a **minimum** that can be reached by a process model:*

$$\exists m \in \mathcal{M} : \forall M \in \mathcal{M} : C(m) \leq C(M)$$

The last two properties we want to investigate are related to the ¬SUB property. If a complexity measure fulfills ¬SUB it makes sense to analyze it for superadditivity. While subadditive complexity measures can help in estimating the complexity of a composed net, superadditive measures imply that one should take care when using the operations of Figure 2.

Property 8 (¬SUP). — *C is **not superadditive**, i.e. the complexity score of a composed model can be less than the sum of the complexity scores of its parts: Let $\mathcal{O} := \{\rightarrow, \wedge, \times, \circlearrowleft\}$.*

$$\forall \oplus \in \mathcal{O} : \exists M_1, M_2 \in \mathcal{M} : C(M_1 \oplus M_2) < C(M_1) + C(M_2)$$

Similarly, if a complexity measure fulfills neither ¬SUB nor ¬SUP, it is interesting to investigate whether this is because the measure is additive. Additive measures would remove the need to recalculate the complexity for composed nets.

Property 9 (ADD). — *C is **additive**, i.e. the complexity score of a composed model is exactly the sum of the complexity scores of its parts: Let $\mathcal{O} := \{\rightarrow, \wedge, \times, \circlearrowleft\}$.*

$$\forall \oplus \in \mathcal{O} : \forall M_1, M_2 \in \mathcal{M} : C(M_1 \oplus M_2) = C(M_1) + C(M_2)$$

Of course, one could imagine more properties for complexity measures. We focus especially on Weyuker's properties and their extensions, since they are well-known for the analysis of software complexity measures and because they were already successfully used on a complexity measure for process models.

V. Complexity Measures

For our analyses, we chose complexity measures that were already categorized into complexity dimensions by Lieben et al. [8]. These measures are well-known and frequently used in practice and literature. The analyses of Reijers et al. [11] and Lieben et al. [8] form a good basis for understanding these measures, which we aim to deepen. In particular, we investigated the following complexity measures:

- Size C_{size} [9, p.118], Connector Mismatch C_{MM} [9, p.125], Connector Heterogeneity C_{CH} [9, pp.126-127], Cross Connectivity C_{CC} [17], Token Split C_{ts} [9, p.128], Separability C_{sep} [9, p.122] and Control Flow Complexity C_{CFC} [2], all belonging to the first complexity dimension: Token Behavior Complexity.

TABLE I: Complexity measures for workflow nets, sorted by the complexity dimension they belong to.

$C_{\text{size}}(W)$	$\lvert P\rvert + \lvert T\rvert$
$C_{\text{CH}}(W)$	$-\left(\frac{\lvert\mathcal{C}^W_{\text{and}}\rvert}{\lvert\mathcal{C}^W\rvert}\cdot\log_2\left(\frac{\lvert\mathcal{C}^W_{\text{and}}\rvert}{\lvert\mathcal{C}^W\rvert}\right)+\frac{\lvert\mathcal{C}^W_{\text{xor}}\rvert}{\lvert\mathcal{C}^W\rvert}\cdot\log_2\left(\frac{\lvert\mathcal{C}^W_{\text{xor}}\rvert}{\lvert\mathcal{C}^W\rvert}\right)\right)$
$C_{\text{ts}}(W)$	$\sum_{t\in T}(\lvert t^\bullet\rvert - 1)$
$C_{\text{CFC}}(W)$	$\lvert\mathcal{S}^W_{\text{and}}\rvert + \sum_{p\in\mathcal{S}^W_{\text{xor}}}\lvert p^\bullet\rvert$
$C_{\text{acd}}(W)$	$\frac{1}{\lvert\mathcal{C}^W\rvert}\cdot\sum_{x\in\mathcal{C}^W}\deg(x)$
$C_{\text{mcd}}(W)$	$\max\{\deg(x)\mid x\in\mathcal{C}^W\}$
$C_{\text{seq}}(W)$	$1-\frac{1}{\lvert F\rvert}\cdot\lvert\{(x,y)\in F\mid x,y\notin\mathcal{C}^W\}\rvert$
$C_{\text{cyc}}(W)$	$\frac{1}{\lvert P\rvert+\lvert T\rvert}\cdot\lvert\{x\in P\cup T\mid x\text{ lies on a cycle in }W\}\rvert$
$C_{\text{CNC}}(W)$	$\frac{\lvert F\rvert}{\lvert P\rvert+\lvert T\rvert}$
$C_{\text{dens}}(W)$	$\frac{\lvert F\rvert}{2\cdot\lvert T\rvert\cdot(\lvert P\rvert-1)}$
$C_{\text{dup}}(W)$	$\sum_{a\in A}(\max(\lvert\{t\in T\mid\ell(t)=a\}\rvert,1)-1)$
$C_{\emptyset}(W)$	$\lvert\{p\in P\mid {}^\bullet p\subseteq\mathcal{S}^N_{and}\wedge p^\bullet\subseteq\mathcal{J}^N_{and}\}\rvert$

- Average Connector Degree C_{acd} [9, pp.120-121], Maximum Connector Degree C_{mcd} [9, p.121] and Sequentiality C_{seq} [9, p.123], all belonging to the second complexity dimension: NODE IO COMPLEXITY.
- Depth C_{depth} [9, pp.124-125], Diameter C_{diam} [9, p.119] and Cyclicity C_{cyc} [9, pp.127-128], all belonging to the third complexity dimension: PATH COMPLEXITY.
- Coefficient of Network Connectivity C_{CNC} [9, p.120] and Density C_{dens} [9, p.120], belonging to the fourth complexity dimension: DEGREE OF CONNECTEDNESS.
- Number of Duplicate Tasks C_{dup} [12] and Number of Empty Sequence Flows $C_{\emptyset}$ [4] that both do not belong to any of the aforementioned dimensions.

These complexity measures are defined for BPMN or EPCs, so in Table I, we translate them to workflow nets. Since there are algorithms that translate free-choice workflow nets to BPMN [3], most of our analyses are also valid for BPMN and EPC models. However, our focus on workflow nets leads us to omit the structuredness measure from our analyses. This measure relies on reducing the size of the model [9, pp.123-124]. There are algorithms that reduce the size of Petri nets, but translating them to workflow nets while ensuring to use similar techniques as the original structuredness measure, is out of scope for this paper. In the rest of this section, we show the translations of measures that do not fit into Table I.

1) Connector Mismatch: For BPMN, the connector mismatch measure subtracts the number of joins of a specific type from the number of splits of that same type. We first define

$$MM^W_{\text{and}} := \left|\sum_{t\in\mathcal{S}^W_{\text{and}}}|t^\bullet| - \sum_{t\in\mathcal{J}^W_{\text{and}}}|{}^\bullet t|\right| \tag{1}$$

and M_{xor} accordingly for exclusive choice connectors. Workflow nets have no semantics for inclusive choices, so we leave them out and define C_{MM} as follows:

$$C_{\text{MM}}(W) = MM^W_{\text{and}} + MM^W_{\text{xor}} \tag{2}$$

2) Cross Connectivity: The Cross Connectivity measure quantifies how difficult it is to understand the connection between two nodes in the graph of a process model. Since BPMN, EPCs and workflow nets can all be interpreted as graphs, we can directly translate this measure to workflow nets. Thus, let $W = (P, T, F, \ell, p_i, p_o)$ be a workflow net. For a node $v \in P \cup T$, let $\deg(v)$ be the number of arcs entering or leaving v. For any node v, we define its weight as

$$w_W(v) := \begin{cases} \frac{1}{\deg(v)} & \text{if } v\in\mathcal{C}^W_{\text{xor}} \\ 1 & \text{if } v\in\mathcal{C}^W_{\text{and}} \\ 1 & \text{otherwise.}\end{cases} \tag{3}$$

We define the weight of an edge $(u, v) \in F$ as the product of the weights of its end points: $w_W((u, v)) = w_W(u) \cdot w_W(v)$. Next, we need a notion for paths. For $k \in \{2, \ldots, |P| + |T|\}$, $\rho = v_1, \ldots, v_k$ is a path if $(v_i, v_{i+1}) \in F$ for all indices i with $i \in \{1, \ldots, k-1\}$ and all nodes $v_1, \ldots, v_k$ are pairwise disjoint. The weight of such a path is defined as the product of all edge weights: $w_W(\rho) = w_W((v_1, v_2)) \cdot \ldots \cdot w_W((v_{k-1}, v_k))$. Using $\mathcal{P}_{v_i,v_j}$ as the set of all paths from $v_i \in P \cup T$ to $v_j \in P \cup T$ that visit at least two different nodes, we define the value of a connection as follows:

$$V_W(v_i, v_j) := \max(\{w_W(\rho) \mid \rho \in \mathcal{P}_{v_i,v_j}\} \cup \{0\}) \tag{4}$$

This value is ≤ 1 for each pair of vertices. Pairs of nodes for which there is no path of length at least 2 receive the value 0. Normalizing the sum of values for all pairs leads to the cross connectivity measure.

$$C_{\text{CC}}(W) = 1 - \frac{\sum_{n_1,n_2\in P\cup T} V_W(n_1, n_2)}{(|P| + |T|) \cdot (|P| + |T| - 1)}. \tag{5}$$

In a workflow net, there is no path from any node v to the input place p_i, so we divide by $(|P|+|T|)\cdot(|P|+|T|-1)$. In contrast to the original definition, we subtract the result from 1, so a low complexity score corresponds to low complexity. This way, it is easier to compare C_{CC} with other measures.

3) Separability: The separability measure quantifies how easy parts of a process model can be investigated in isolation from the rest of the model. To achieve this, the measure determines the number of cut-vertices in the model. A cut-vertex in an undirected graph is a vertex whose removal decomposes the graph into more than one connected component. We use this definition for directed graphs by ignoring arc directions. Since the input- and output place can never be cut-vertices, we can normalize the number of cut-vertices as follows:

$$C_{\text{sep}}(W) = 1 - \frac{|\{v \in P \cup T \mid v \text{ is a cut-vertex in } W\}|}{|P| + |T| - 2} \tag{6}$$

4) Depth: The depth of a node $v \in P \cup T$ is defined by its in-depth and its out-depth. The in-depth counts how many split nodes precede v that were not closed by a join, while the out-depth counts how many join nodes succeed v that are not opened by a split after v. For the in-depth, we inspect all paths from p_i to v. Let $\mathcal{P}_{p_i,v}$ be the set of all such paths. Further, let $\mathcal{S}^W := \mathcal{S}^W_{\text{and}} \cup \mathcal{S}^W_{\text{xor}}$ be the set of all split nodes in

W and $\mathcal{J}^W := \mathcal{J}^W_{\text{and}} \cup \mathcal{J}^W_{\text{xor}}$ the set of all join nodes in W. For $\rho = (v_1, \ldots, v_n) \in \mathcal{P}_{p_i,v}$, we define inductively:

$$\lambda_W(v_1) = \lambda_W(p_i) := 0 \tag{7}$$

$$\lambda_p(v_n) := \begin{cases} \lambda_W(v_{n-1}) + 1 & \text{if } v_{n-1} \in \mathcal{S}^W \wedge v_n \notin \mathcal{J}^W \\ \lambda_W(v_{n-1}) & \text{if } v_{n-1} \in \mathcal{S}^W \wedge v_n \in \mathcal{J}^W \\ \lambda_W(v_{n-1}) & \text{if } v_{n-1} \notin \mathcal{S}^W \wedge v_n \notin \mathcal{J}^W \\ \lambda_W(v_{n-1}) - 1 & \text{if } v_{n-1} \notin \mathcal{S}^W \wedge v_n \in \mathcal{J}^W \end{cases} \tag{8}$$

$$\lambda_W(v) := \max\left\{0, \max_{\rho \in \mathcal{P}_{p_i,v}} \lambda_\rho(v)\right\} \quad \text{(for any } v \neq p_i\text{)} \tag{9}$$

For the out-depth, we set $\overleftarrow{W} := (P, T, \overleftarrow{F}, \ell, p_o, p_i)$ where $\overleftarrow{F} := \{(u, v) \mid (v, u) \in F\}$ as the workflow net with all arcs reversed. Now, we can reuse the definition of the in-depth to define the out-depth and define:

$$C_{\text{depth}}(W) := \max\{\min\{\lambda_W(v), \lambda_{\overleftarrow{W}}(v)\} \mid v \in P \cup T\}. \tag{10}$$

5) Diameter: The diameter of a workflow net W is the maximal length of a way from p_i to p_o where no arc is used more than once. A way is a path where it is allowed to revisit nodes. Let $\mathcal{W}_{p_i,p_o}$ be the set of all ways that use no arc more than once. Then:

$$C_{\text{diam}}(W) = \max\{|k| \mid v_1, \ldots, v_k \in \mathcal{W}_{p_i,p_o}\}. \tag{11}$$

After defining the propositions and the complexity measures we want to investigate, we are ready to show our results in the next section of this paper.

VI. Discussion of the Results

Table II shows the results of our analyses. For properties that depend on the chosen operation of Figure 2, we added a more detailed analysis in Table III. All of our analyses for each pair of property and complexity measure can be found in the extended version of this paper [13]. Instead of repeating the analyses here, we focus on a discussion of the results. We split the discussion into two parts: First evaluating the properties from Section IV, then comparing complexity measures based on which properties they fulfill. This way, we earn an understanding of both the properties and the measures.

A. Interpreting the Properties

Weyuker proposes that all complexity measures for software programs should fulfill her properties [18]. For complexity measures of process models, we disagree with that. In the following, we classify the properties of Section IV as *normative* or *descriptive*. In our setting, normative means that a complexity measure should fulfill the respective property to be useful. Descriptive properties feature interesting insights, but complexity measures are not required to fulfill them.

An immediate observation is that almost all complexity measures fulfill PERM and ROB_ℓ. Regarding PERM, it makes sense for complexity measures to react to permuting the net. Permutations can rearrange the entire structure of a process model, which should be captured by a complexity measure. We therefore argue that PERM is a normative property. However, we do not get to the same conclusion for ROB_ℓ. Even though most measures are robust to relabeling the transitions, uniformly changing the labels can have an impact on the understandability of a net, as different labels may have different semantics. We therefore see ROB_ℓ as descriptive.

Weyuker's property COMP is designed to test whether complexity measures are sensitive to dependencies between different parts of the net, but needs to be taken carefully. For example, the measure C_{MM} can "repair" existing connector mismatches by introducing more of them to the net, which leads to COMP being fulfilled. Since it is possible to fulfill COMP in this way, we propose to take this property as descriptive. We advise taking care if a measure fulfills this property and check whether it does so for the intended reasons.

The properties MON and ¬SUB are also descriptive: A composed model should be allowed to receive a lower complexity when the composition introduces more structure to the net. For example, M^1 of Figure 1 is quite dense, while the net M is less dense. Not fulfilling property ¬SUB, on the other hand, can be advantageous for certain use-cases: If a measure does not fulfill ¬SUB, we cannot increase the complexity of two nets beyond the sum of their complexity scores when combining them with one of the operations shown in Figure 2. Our new properties ¬SUP and ADD deepen these insights, but are also intended as descriptive properties.

Finally, fulfilment of the MIN property hints on whether simple models do not contain unnecessary parts, but MIN is descriptive, as it is not mandatory for a mature complexity measure. Property DEF, on the other hand, is normative, since undefined complexity scores are difficult to interpret. Table IV summarizes our classifications.

B. Comparison of Complexity Measures

It comes to no surprise that the complexity measure based on the size of a workflow net fulfills almost all of Weyuker's properties. Its simplicity and strong connection to understandability [11] further emphasize that the size of a workflow net is an important factor for model complexity. However, C_{size} is superadditive, so combining two nets with one of the operations of Figure 2 is guaranteed to return a net with higher complexity than the sum of the input nets. In contrast to C_{size}, C_{MM} and C_{CH} are not monotone, since adding certain structures to a workflow net can lower the scores of these complexity measures. If users favor monotonicity, they should avoid these measures and consider C_{CFC} instead, which also takes the different connectors in a net into account. Furthermore, users of C_{CH} should be aware that it is undefined for workflow nets that do not contain any connectors. C_{ts} is additive for all operations except $\wedge$, but when using the operator $\wedge$ to combine n nets, the complexity score of the result is exactly $n - 1$ higher than the sum of complexities, which can often be tolerated in practice. C_{ts} has the most of what we would call desirable features in the Token Behavior Complexity dimension. C_{sep} is not monotone, since all operations of Figure 2, except $\times$, introduce new cut-

TABLE II: The results of our analyses. The colums are grouped by complexity dimension, the rows by whether they were defined by Weyuker (top) or not (bottom). Entries with an asterisk imply that the answer depends on the used operation.

	C_{size}	C_{MM}	C_{CH}	C_{CC}	C_{ts}	C_{sep}	C_{CFC}	C_{mcd}	C_{seq}	C_{acd}	C_{depth}	C_{diam}	C_{cyc}	C_{CNC}	C_{dens}	C_{dup}	$C_{\emptyset}$
MON	✓	×	×	×	✓	×	✓	✓	×	×	✓	×*	×	×	×	✓	✓
COMP	×	✓	✓	✓	×	✓	×	✓	✓	✓	×*	×	✓	✓	✓	✓	×
PERM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓
ROB$_\ell$	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
¬SUB	✓	×	✓	✓	×*	×*	×*	×	×*	×	✓	✓	×*	×	×	✓	×
DEF	✓	✓	×	✓	✓	✓	✓	×	✓	×	✓	✓	✓	✓	✓	✓	✓
MIN	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	✓	✓
¬SUP	×	✓	✓	✓	×	✓	×	✓	✓	✓	✓	×*	✓	✓	✓	×	×
ADD	×	×	×	×	×*	×	×*	×	×	×	×	×	×	×	×	×	✓

TABLE III: A more detailed overview for measures where it depends on the operation whether a property is fulfilled.

	C_{CFC}	C_{sep}	C_{ts}	C_{seq}	C_{depth}	C_{diam}	C_{cyc}
MON	✓	×	✓	×	✓	→: ✓ ∧ : ✓ × : ✓ ↻: ×	×
COMP	×	✓	×	✓	→: ✓ ∧: × ×: × ↻: ×	×	✓
¬SUB	→: × ∧: ✓ ×: ✓ ↻: ✓	→: × ∧: ✓ ×: ✓ ↻: ✓	→: × ∧: ✓ ×: × ↻: ×	→: × ∧: ✓ ×: ✓ ↻: ✓	✓	✓	→: × ∧: × ×: × ↻: ✓
¬SUP	×	✓	×	✓	✓	→: × ∧: ✓ ×: ✓ ↻: ✓	✓
ADD	→: ✓ ∧: × ×: × ↻: ×	×	→: ✓ ∧: × ×: ✓ ↻: ✓	×	×	×	×

vertices. It is also not monotone regarding ×, but it is possible to slightly tweak MON, so C_{sep} is also monotone for ×. Like the control flow complexity measure, C_{sep} is subadditive only for →. C_{CFC}, however, is also additive for this operation. The measure C_{CC} fulfills COMP for the operator → because the weight of xor-connectors lower when they are p_i or p_o. Furthermore, C_{CC} has no minimum value.

Regarding NODE IO COMPLEXITY, only C_{mcd} is monotone. This measure is also subadditive for workflow nets that have at least one connector. For workflow nets without connectors, however, this measure is undefined and setting the score to 0 in these cases would destroy subadditivity. The same is true for C_{acd}. If ¬SUB needs to be fulfilled, C_{seq} is a good choice in this dimension.

With two exceptions, C_{diam} and C_{depth} of the PATH COMPLEXITY dimension have similar properties: Regarding the → operator, C_{diam} is superadditive and C_{depth} is sensitive to compositions. More importantly, C_{diam} is not monotone for compositions with the ↻ operator, so if this operator is to be used, one should favor C_{depth} over C_{diam}.

The dimension labeled DEGREE OF CONNECTEDNESS contains no monotone measures. This could hint to a gap for monotone complexity measures that take the connectedness

TABLE IV: A table classifying which properties of Section IV should be fulfilled by a complexity measure (N: normative) or are optional for a complexity measure (D: descriptive).

Property	MON	COMP	PERM	ROB$_\ell$	¬SUB
Classification	D	D	N	D	D
Property	DEF	MIN	¬SUP	ADD	
Classification	N	D	D	D	

of the net into account. Furthermore, all measures are subadditive, so if this property is undesired, this dimension offers no alternatives. C_{CNC} and C_{dens} fulfill similar properties, but C_{dens} offers no minimum value, while C_{CNC} does.

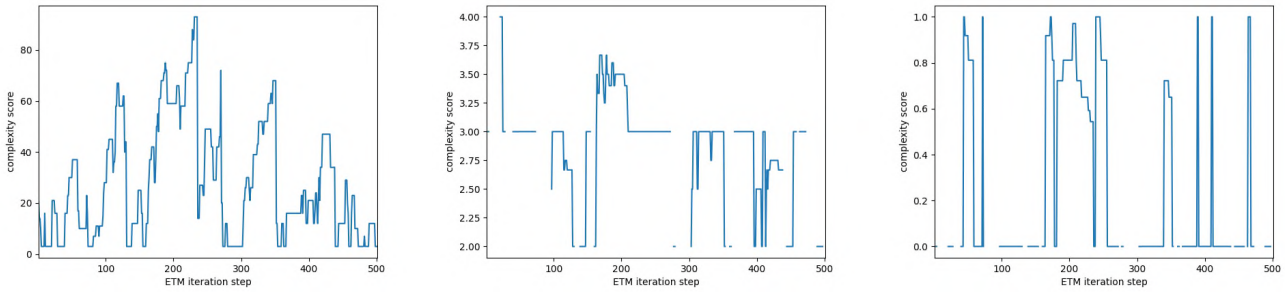
C. Experiments with an evolutionary discovery algorithm

We conducted a small case study to observe how the properties of Section IV drive the evolution of models in the ETM. We picked three complexity measures with different properties: C_{size}, C_{acd} and C_{CH}. In each iteration, the ETM chooses a random mutation for a tree in its population. Figure 4 shows one such mutation that adds an operator node.

For C_{size}, this would mean a significant increase in complexity from 16 to 25, while the complexity according to C_{acd} would slightly decrease from 5 to 3.5. According to C_{CH}, the complexity would drop from its highest value 1 to its lowest value 0. This is because C_{acd} is subadditive, while C_{size} is superadditive. C_{CH} is neither sub- nor superadditive, so it is prone to high fluctuation. The fact that C_{size} is superadditive can be problematic, since the ETM must more often choose a deletion step to lower the complexity. Thus, for superadditive complexity measures, ETM needs a higher probability for removing nodes to find less complex models.

To confirm this theory, we implemented a simple version of the ETM[1] with a population size of 10 and an elite size of 1. For the input, we generated a random process tree and derived an event log from it. Since the ETM expects all quality scores to be in $[0, 1]$, we used this process tree as a reference with a fixed simplicity score of 0.75 and calculated simplicity in accordance to it. We monitored the evolution of a tree in the population with respect to its complexity, leading to the results shown in Figure 3. This way, we can witness the effect described earlier, as C_{size} gets much higher values and needs

[1] see https://github.com/Pati-nets/Navigating-Complexity



(a) Changes of the C_{size}-score. (b) Changes of the C_{acd}-score. (c) Changes of the C_{CH}-score.

Fig. 3: Development of different complexity scores during a run of the Evolutionary Tree Miner.

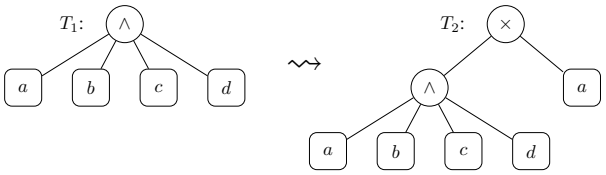


Fig. 4: A possible mutation of a process tree T_1 into T_2.

to perform several deletion steps to lower the complexity. C_{acd} has a smaller range and a more stable complexity score, but its graph is not continuous, since C_{acd} is not always defined. Finally, C_{CH} fluctuates as strongly between its highest and lowest value as expected.

VII. Conclusion

In this paper, we used and extended the properties defined by Weyuker [18] to analyze and compare popular complexity measures for process models [9]. We discussed which of the inspected properties are normative and which are descriptive for process models to give a sense of their importance. Furthermore, we compared complexity measures that are in the same complexity dimension discovered by Lieben et al. [8] and highlighted when to prefer which measure. Defining and analyzing the structuredness measure was out of scope of this paper, so possible future work involves the analysis of this measure. Moreover, one can think of more properties that highlight interesting characteristics of complexity measures. More composition rules than those used by the ETM would be interesting to analyze. Furthermore, investigating how to define permutations of Petri nets is interesting for a more expressive PERM property. The analysis could also be extended to other workflow net complexity measures, such as those in [5]. For a more in-depth case study with the ETM, general rules on how to express complexity with a value in $[0, 1]$ are needed.

We are confident that our analyses and discussions shed a new light on popular complexity measures that will help analysts and algorithm designers to choose measures fitting to their needs. We also hope to start a discussion on complexity and simplicity measures, which are often overlooked during the evaluation of process models.

References

[1] J.C.A.M. Buijs, B.F. van Dongen, and W.M.P. van der Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *Int. J. Cooperative Inf. Syst.*, 23(1), 2014.

[2] J. Cardoso. Control-flow complexity measurement of processes and Weyuker's properties, 2018.

[3] C. Favre, D. Fahland, and H. Völzer. The relationship between workflow graphs and free-choice workflow nets. *Inf. Syst.*, 47:197–219, 2015.

[4] V. Gruhn and R. Laue. Reducing the cognitive complexity of business process models. In *ICCI 2009*, pages 339–345. IEEE Computer Society, 2009.

[5] K.B. Lassen and W.M.P. van der Aalst. Complexity metrics for workflow nets. *Inf. Softw. Technol.*, 51(3):610–626, 2009.

[6] S. Leemans, E. Poppe, and M. Wynn. Directly follows-based process mining: Exploration & a case study. In *ICPM 2019*, pages 25–32. IEEE, 2019.

[7] S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering block-structured process models from event logs - A constructive approach. In *PETRI NETS 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer, 2013.

[8] J. Lieben, T. Jouck, B. Depaire, and M. Jans. An improved way for measuring simplicity during process discovery. In *EOMAS 2018*, volume 332 of *Lecture Notes in Business Information Processing*, pages 49–62. Springer, 2018.

[9] J. Mendling. *Metrics for Process Models*, volume 6 of *Lecture Notes in Business Information Processing*. Springer, 2008.

[10] S. Morasca. Measuring attributes of concurrent software specifications in petri nets. In *6th IEEE International Software Metrics Symposium METRICS 1999*, pages 100–110. IEEE Computer Society, 1999.

[11] H.A. Reijers and J. Mendling. A study into the factors that influence the understandability of business process models. *IEEE Trans. Syst. Man Cybern. Part A*, 41(3):449–462, 2011.

[12] M. La Rosa, P. Wohed, J. Mendling, A.H.M. ter Hofstede, H.A. Reijers, and W.M.P. van der Aalst. Managing process model complexity via abstract syntax modifications. *IEEE Trans. Ind. Informatics*, 7(4):614–629, 2011.

[13] P. Schalk, A. Burke, and R. Lorenz. Exploring complexity: An extended study of formal properties for process model complexity measures, 2024.

[14] A.F. Syring, N. Tax, and W.M.P. van der Aalst. Evaluating conformance measures in process mining using conformance propositions. *Trans. Petri Nets Other Model. Concurr.*, 14:192–221, 2019.

[15] W.M.P. van der Aalst. *Process Mining - Data Science in Action*. Springer, 2016.

[16] W.M.P. van der Aalst. Relating process models and event logs - 21 conformance propositions. In Wil M. P. van der Aalst, Robin Bergenthum, and Josep Carmona, editors, *ATAED 2018*, volume 2115 of *CEUR Workshop Proceedings*, pages 56–74. CEUR-WS.org, 2018.

[17] I.T.P. Vanderfeesten, H.A. Reijers, J. Mendling, W.M.P. van der Aalst, and J. Cardoso. On a quest for good process models: The cross-connectivity metric. In *CAiSE 2008*, volume 5074 of *Lecture Notes in Computer Science*, pages 480–494. Springer, 2008.

[18] E.J. Weyuker. Evaluating software complexity measures. *IEEE Transactions on Software Engineering*, 14(9):1357–1365, 1988.