Received: July 27, 2019

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Novel Parallel Business Process Similarity Methods Based on Weighted-Tree Declarative Pattern Models

Cahyaningtyas Sekar Wahyuni[1] **Kelly Rossa Sungkono[1]** **Riyanarto Sarno[1]***

[1]*Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia*
* Corresponding author's Email: riyanarto@if.its.ac.id

Abstract: Existing methods, such as Graph Edit Distance (GED) and Cosine measure, still have drawback in obtaining similarity of parallel relationships by neglecting the control-flow patterns, i.e. AND, OR, and XOR. Since AND > OR > XOR, the similarity value of AND versus OR is greater than XOR versus OR and AND versus XOR. This paper proposes two new similarity methods, Tree Declarative Pattern Edit Distance (TPED) and Cosine-Tree Declarative Pattern (Cosine-TDP). They provides value to the control-flow pattern so the value of similarity can be seen more differently. The new methods utilize tree model of the declarative pattern. The results show that the proposed methods are better at differentiating parallel relationships than the existing methods, GED and Cosine measure. In obtaining AND versus OR, XOR versus OR, and AND versus XOR, TPED obtained 0.821, 0.811, and 0.78 while Consine-TDP obtained 0.834, 0.826, and 0.693. Meanwhile, GED obtained 1 for all parallel relationships whereas Cosine measure obtained 0.02, 0.08, and 0.04.

Keywords: Cosine-tree declarative pattern, Parallel business process, Tree declarative pattern edit distance, Weighted-linear temporal logic, Weighted-tree declarative pattern.

1. Introduction

Business process models are graphical representations of activity workflows [1, 2] that can be represented by two types of relationships, which are sequence relationships and parallel relationships. There are several control-flow patterns [3] to depict operators of parallel relationships, i.e. AND, OR, and XOR [4, 5]. The control-flow patterns can be modeled in different ways depend on the kinds of the graphical representations. The first representation, imperative model [6], uses specific operators to express parallel relationships. For example, in Business Process Modeling Notation (BPMN) [7, 8], AND, OR, and XOR are denoted by three gateways, i.e. +, O, and X respectively. There are other imperative model besides BPMN, i.e. Petri Nets [9], Yet Another Workflow Language (YAWL) [10], and Event-Driven Process Chains (EPC) [11]. The second representation, declarative model [12], uses rules to declare the operators of parallel relationships. One of declarative model, Linear Temporal Logic (LTL) [13–16], constructs the rules by combining temporal operators, such as <> and _O, with logical operators, e.g. ^ and V. For examples, to construct sequence relationship, LTL uses operator _O. Further, to construct AND, OR, and XOR relationship, LTL uses operators <> ^, <> V, and _O V respectively.

Business process models can be compared to evaluate its structure [17, 18] and behavior [19, 20] that have similar functions [21]. The comparison is called a business process similarity [22]. In calculating similarity between process models, structural [23], behavioral [23], or semantic similarity can be applied [24]. In several previous works, behavioral similarity was calculated using Jaccard similarity coefficient and Transition Adjacency Relations (TARs). TARs were used to calculate the similarity in behavioral relationships between two models, while Jaccard similarity coefficient was used to compare the values of similar labels, transitions, and edges. Another method of measuring structural similarity uses Graph Edit Distance (GED). GED calculates the number of edits in the nodes and edges needed to transform one process model into another process model divided by the total number of nodes in the two process models.

Existing similarity methods, such as GED, Cosine, Jaccard, and TARs have several drawbacks. Firstly, they are unable to differentiate between the operators of parallel relationships. Secondly, when using some imperative similarity measures (i.e. the Cosine measure [25] and GED [23]) to compare two imperative models with the different operators of parallel relationships (i.e. the model containing XOR relation versus the model containing AND relation), the similarity result might be identical although the models have different operators of parallel relationships. The inability in differentiating the operators of parallel relationships is caused by only considering labels, nodes, and edges, not control-flow patterns.

In this research, we propose novel similarity methods that are able to differentiate the operators of parallel relationships: AND versus OR, OR versus XOR, and AND versus XOR by assigning weight of the control-flow patterns and edges on the declarative model [16]. The weight of edges is based on the probability of occurrences of the activities. The weighted declarative model is named Weighted-Tree Declarative Pattern (W-TDP). This research uses declarative model, i.e. Linear Temporal Logic (LTL), because the combination of its operators can be derived into logic table as a basis of our proposed weighted control-flow patterns calculation. Then, the declarative models are transformed into tree model [26] to describe activity workflows in a more structured way.

The contributions of this research are:

(1) Discovering process model based on W-TDP, which are tree models of Weighted-Linear Temporal Logic (W-LTL).

(2) Proposing two novel similarity methods based on W-TDP. The proposed similarity methods are: (a) Tree Declarative Pattern Edit Distance (TPED), a modification of GED to measure structural similarity; and (b) Cosine-Tree Declarative Pattern Similarity (Cosine-TDP), a modification of the Cosine measure to calculate behavioral similarity.

(3) TPED and Cosine-TDP are able to differentiate the similarity of the operators of parallel relationships, i.e. AND, OR, and XOR. This statement is verified in section 4.

(4) Proposing the weight of the control-flow pattern which are based on logic table as a basis of our proposed similarity methods.

(5) To prove that TPED and Cosine-TDP are able to differentiate the operators of parallel relationships compared to the existing method (GED and Cosine measure).

The paper is organized as follows. The existing methods and other related works are discussed in Section 2. The proposed methods are discussed in Section 3. Then, we discuss the experiment results in Section 4. Finally, the conclusion of this work is presented in Section 5.

2. Literature review

2.1 Existing similarity methods

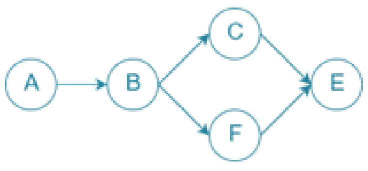
Several previous works have investigated similarity methods. One of the works [27] compared the performance of Transition Adjacency Relations (TARs) (see Eq. (1)) and Naive Algorithm to calculate behavioral similarity. The TARs were obtained from the relationships between activities, as described in Definition 1. The value of behavioral similarity was obtained from the order of the process in the model using Naive Algorithm. The TARs similarity was obtained by comparing the similar fragment between two business process models and was then divided by the total number of activities in the process order. .

Definition 1. ***Transition Adjacency Relations (TAR).*** *Let $G = (N, E)$ be a graph model with a set of nodes $N = \{n_1, n_2, n_3, \dots, n_x\}$ and a set of edges $E = \{e_1, e_2, e_3, \dots, e_x\}$. We can say e_1 and e_2 are in a Transition Adjacency Relation (TAR), shown as $e_1 \overset{TAR}{\Longrightarrow} e_2$ iff there is a flow of a sequence f_s, where r_1 is executed, and $f_s \overset{r_1}{\rightarrow} f_s'$, such that r_2 is executed at f_s'.*

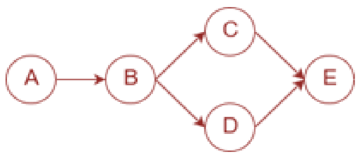
The equation to compute the TARs similarity value is shown in Eq. (1).

$$TARs\ Sim = \frac{(The\ amount\ of\ similar\ TARset)^2}{TARset_1 \, . \, TARset_2} \quad (1)$$

where $TARset$ is the similar fragment between first and second model. For example, we have two graph process models as shown in Fig. 1. Graph model 1 has TARset {AB, BC, BF, CE, FE} and Graph model 2 has TARset {AB, BC, BD, CE, DE}. TARs similarity value between graph model 1 and graph model 2 is $3^2/25$ = 0.36. We can see that the amount of similar fragment TARset between two process models is 3, which are {AB, BC, CE} and



(a)



(b)

Figure. 1 Example of two graph process models: (a) Graph model 1 and (b) Graph model 2

the amount of TARset in graph model 1 and 2 is 5 each. From the example above, we can see that TARs similarity does not differentiate the operators of parallel relationship.

Another previous work, [28, 29], attempted to extract similar fragments based on similarity measures from a scalable process model. Jaccard similarity (see Eq. (2)) and TARs were used to obtain the structural similarity value and the behavioral similarity value respectively. As shown in Definition 2, the Jaccard coefficient value was obtained from the similarity between aspects such as the number of nodes transitions in the model divided by all aspects contained in the two models under comparison.

Definition 2. Jaccard coefficient. *Given x as nodes and y as edges of a graph model. $x_1, y_1 \in G_1$; $x_2, y_2 \in G_2$, where G_1 is the first graph model and G_2 is the second graph model, Jaccard similarity is defined as:*

$$J\,(x,y) = \begin{cases} \frac{\sum(x_1 \cap x_2)+\sum(y_1 \cap y_2)}{\sum(x_1 \cup x_2)+\sum(y_1 \cup y_2)} & if\ x \cup y\ \neq \emptyset \\ 1 & if\ x \cup y = \emptyset \end{cases} \quad (2)$$

where
x is node,
y is edge,
$x_1\ \cap\ x_2$ is the similar set of nodes from two models,
$y_1\ \cap\ y_2$ is the similar set of edges from two models,
$x_1\ \cup\ x_2$ is the union of all nodes, and
$y_1\ \cup\ y_2$ is the union of all edges.

From the example above, we can see that Jaccard similarity only consider the same set of nodes and edges from two models, but does not differentiate the operators of parallel relationship.

The other similarity metric, used on the structure of the process model, is Graph Edit Distance (GED) (see Eq. (3)) [23]. As defined in Definition 3, GED measures the minimal number of edit operations between two models. The three types of measures in GED are: node insertion or deletion, edge insertion or deletion, and node label substitution (see Eq. (4)-(6)). The final result of GED similarity which is presented in Eq. (3) is obtained by subtracting 1 from the average number of node insertions or deletions, edge insertions or deletions, and node label substitutions.

Definition 3. Graph Edit Distance (GED). *Let G_1 and G_2 be the first and the second business process model respectively. The similarity of Graph Edit Distance $SimGED(G_1, G_2)$ is denoted as:*
$SimGED(G_1, G_2) = min_{(x_1,\dots,x_k)\in P(G_1,G_2)} \sum_{i=1}^{k} c(x_1)$, where $P(G_1,G_2)$ is the set of edits needed to transform G_1 into G_2 and $c(x) \geq 0$ is the cost of GED to perform operation x.

The equations to compute the value of GED similarity are shown in Eq. (3) to Eq. (6):

$$SimGED(G_1, G_2) = \ 1 - avg(snv, sev, sbv) \quad (3)$$

$$snv = \frac{|sn|}{|N1|+|N2|} \quad (4)$$

$$sev = \frac{|se|}{|E1|+|E2|} \quad (5)$$

$$sbv = \frac{2.\sum_{(n,m)\in M} 1-Sim(n,m)}{|N1|+|N2|-|sn|} \quad (6)$$

where
snv is the average number of node insertions or deletions,
sev is the average number of edge insertions or deletions,
sbv is the average number of node label substitutions,
$|sn|$ is the number of node insertions or deletions,
$|se|$ is the number of edge insertions or deletions,
$|N1|$ is the number of nodes in G_1,
$|N2|$ is the number of nodes in G_2,
$|E1|$ is the number of node edges in G_1, and
$|E2|$ is the number of node edges in G_2.

From the example above, we can see that GED similarity only consider the number of nodes and edges from two models, but does not differentiate the operators of parallel relationship.

Another similarity metric, the Cosine measure in Eq. (7), has been used to measure behavioral similarity. In [25], the Cosine measure was used to calculate the behavioral similarity between two graph models using two vectors: the activity vector and the transition vector. As described in Definition 4, the Cosine measure is the measurement of two-non zero vectors.

Definition 4. Cosine measure. Let $\vec{A}$ and $\vec{B}$ be the two vector attributes, A_i and B_i are the components of $\vec{A}$ and $\vec{B}$ respectively. The Cosine measure $\cos\theta$ is formulated as:

$$Sim\cos\theta = \frac{A.B}{||A||||B||} = \frac{\sum A_i.B_i}{\sqrt{\sum {A_i}^2 \sum {B_i}^2}} \quad (7)$$

where A_i is the component of activity in model A and B_i is the component of activity in model B.

From the example above, we can see that Cosine similarity only consider the component of activity from two models, but does not differentiate the operators of parallel relationships.

2.2 Control-flow patterns in linear temporal logic (LTL)

Linear Temporal Logic (LTL) is an official language to describe various temporal logics that refers to time [16]. In Definition 5, it is shown that LTL is constructed from the atomic proposition, logical, also temporal capital operators. This research adopts LTL to describe business process model to discover model pattern by using LTL. The previous research has constructed control-flow patterns derived from the declarative model.

Definition 5. Linear Temporal Logic (LTL). *Let $\tau \in LTL$; φ is the name of activities; $\alpha,\beta \in$ the order of activities;*
$\tau \mid= \circ\varphi$ iff $\tau \mid_\alpha\mid= \varphi$ for all $\alpha = 1$,
$\tau \mid= \square\varphi$ iff $\tau \mid_\alpha\mid= \varphi$ for all $\alpha \geq 0$,
$\tau \mid= \lozenge\varphi$ iff $\tau \mid_\alpha\mid= \varphi$ for all $\alpha = 1$, and
$\tau \mid= \varphi_1 U \varphi_2$ iff $\tau \mid_\alpha\mid= \delta$ for all $\alpha \geq 0$ && $\tau \mid_\beta\mid= \varphi$ for all $0 \leq\beta < \alpha$

To describe the relation between activities, control-flow patterns are converted to LTL. To convert the control-flow patterns, several rules are used as shown in Table 1. To construct a sequence, we use the operators `->` and `_O`. The operators `->` and `_O` are placed in front of next activity. For example, activity B occured after executing activity A. Therefore, activity B acts as next activity. Then, the operators `->` and `_O` are placed in front of B.

To construct AND relation, we use the operators `->`, `<>` and `^`. In AND SPLIT relation, the operators `<>` is used once before split and operators `^` is used as separator between next activities. Lastly, the operators `->` and `_O` are used in AND JOIN relation to merge split and executing activity after split. For example, activities B, C, and D are executed concurrently in split relation from activity A and are merged in activity E. Therefore, the operators `->` and `<>` are placed before split activities (B, C, and D) and the operators `^` is placed to seperate B, C, and D. Lastly, to merged the split relation, operators `->` and `_O` are placed before activity E.

To construct XOR relation, we use the operators `->`, `_O` and `V`. In XOR SPLIT relation, the operator `_O` is used once before split and operators `V` is used as separator between next activities. Lastly, the operators `->` and `_O` are used in XOR JOIN relation to merge split and executing activity after split. For example, activities B, C, and D are chosen in XOR Split relation from activity A and are merged in activity E. Therefore, the operators `->` and `_O` are placed before split activities (B, C, and D) and the operators `V` is placed to seperate B, C, and D. Lastly, to merged the split relation, operators `->` and `_O` are placed before activity E.

Table 1. LTL control-flow patterns

Control flow Patterns	LTL	Description
Sequence relation	`A->_O(B)`	A and B occur sequentially.
AND Split relation	`A-><>((B^C^D))`	AND Split occurs concurrently in B, C, and D after A.
AND Join relation	`<>((B^C^D))->_O(E)`	AND Join occurs concurrently in B, C, and D before E.
XOR Split relation	`A->_O((BVCVD))`	XOR Split occurs in B, C, and D after A.
XOR Join relation	`_O((BVCVD))->_O(E)`	XOR Join occurs in B, C, and D before E.
OR Split relation	`A-><>((BVCVD))`	OR Split occurs in B, C, and D after A.
OR Join relation	`<>((BVCVD))->_O(E)`	OR Join occurs in B, C, and D before executing E.

Received: July 27, 2019

To construct OR relation, we use the operators `->`, `<>` and `V`. In OR SPLIT relation, the operators `<>` is used once before split and operators `V` is used as separator between next activities. Lastly, the operators `->` and `_O` are used in OR JOIN relation to merge split and executing activity after split. For example, activities B, C, and D are chosen two among three in split relation from activity A and are merged in activity E. Therefore, the operators `->` and `<>` are placed before split activities (B, C, and D) and the operators `V` is placed to seperate B, C, and D. Lastly, to merged the split relation, operators `->` and `_O` are placed before activity E.

2.3 Tree model

Formally, a given tree model T has a set of nodes $N = \{n_1, n_2, n_3, \dots, n_x\}$ and a set of edges $E = \{e_1, e_2, e_3, \dots, e_{x-1}\}$. Based on [16], the first step is choosing the first W-LTL, which is the control-flow pattern that contains the first activity. `Start_Act(act)` contains the first activity of the process model. Then, the W-LTL is split into parts of W-LTL (pW-LTL) separated by commas. For example `A -> _O((B V C V D))` can be split as [`A`, `->`, `_O`, `(`, `(`, `B`, `V`, `C`, `V`, `D`, `)`, `)`]. The tree model is built after all of the pW-LTL have been processed. An open parenthesis is intended to construct a child of a node and a closing parenthesis is intended to put back the pointer position to the parent of that node. Other logical operators, i.e. `X`, `V`, `^`, and `->`, as nodes of W-LTL are used to construct XOR, OR, AND, and sequence relations respectively. The detail algorithm to construct Weighted-Tree Declarative Pattern (W-TDP) is shown in section 3 in Table 4.

3. The proposed method

The methodology to conduct the research is depicted in Fig. 2. It consists of three steps: constructing the weighted declarative pattern, constructing the weighted-tree model, and conducting the similarity analyses using the proposed methods.

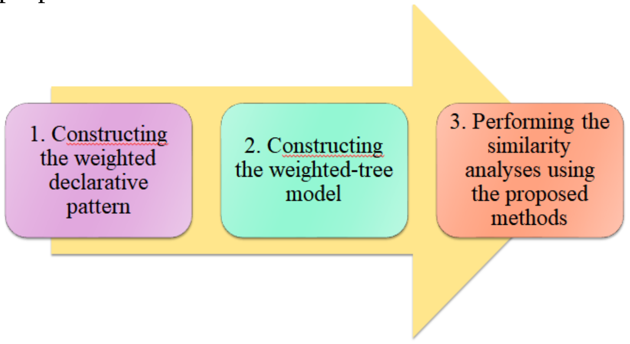

1. Constructing the weighted declarative pattern
2. Constructing the weighted-tree model
3. Performing the similarity analyses using the proposed methods

Figure. 2 The proposed methods

3.1 Weighted-linear temporal logic (W-LTL) declarative patterns

Each relation between activities is assigned by weight value. Weight w is assigned to calculate the probability of occurences of activities. The example of calculating w-value is shown in Table 2. The w-value is used as a basis to construct W-LTL. The W-LTL is constructed based on the graph model by using the algorithm as shown in Table 3. The input to construct W-LTL is a dataset of graph process model. In this paper, the graph process model is shown in Fig. 4. Then the W-LTL is shown in Table 8. If the graph model contains NEXT relation, then the constructed W-LTL is defined as `A -> _O (activity` B_w `)`. Weight w is assigned besides `activity` B_w. If the graph model contains ANDSPLIT relation, then the constructed W-LTL is defined as `A-><>((activity` B_w^C_w^D_w`))`.

Table 2. Assigning weight value to activity relation

Relation	Trace examples	w-value
Sequence Type	[A]-[B]-[C] 10x	Occurences of [A]-[B] = 10 Outgoing edge of [A] = 10 w-value of [A]-[B] = 10/10 = 1 Occurences of [B]-[C] = 10 Outgoing edge of [B] = 10 w-value of [B]-[C] = 10/10 = 1
Split Type	[A]-[B] 7x [A]-[C] 3x	Occurences of [A]-[B] = 7 Outgoing edge of [A] = 10 w-value of [A]-[B] = 7/10 = 0.7 Occurences of [A]-[C] = 3 Outgoing edge of [A] = 10 w-value of [A]-[C] = 3/10 = 0.3
Join Type	[B]-[D] 7x [C]-[D] 3x	Occurences of [B]-[D] = 7 Incoming edge of [D] = 10 w-value of [B]-[D] = 7/10 = 0.7 Occurences of [C]-[D] = 3 Incoming edge of [D] = 10 w-value of [C]-[D] = 3/10 = 0.3

Table 3. Algorithm for W-LTL Declarative Pattern

Input :Dataset of graph process model G (the example can be seen in Fig. 4)
Output :W-LTL y (the example can be seen in Table 8)
For each data in the dataset:
If the relation of the data is NEXT
Add weight *w* to the W-LTL
Print activity A -> _O (activity B_w)
If the relation of the data is ANDSPLIT
For each data in the dataset:
Add all ANDSPLIT relation to ands
For each data in dataset ands:
Search all weight values in ANDSPLIT
Print activity A-><>((activity B_w^C_w^D_w))
If the relation of the data is ANDJOIN
For each data in the dataset:
Add all ANDJOIN relation to andj
For each data in dataset andj:
Search all weight values in ANDJOIN
Print activity <>((activity B_w^C_w^D_w))->_O(E)
If the relation of the data is XORSPLIT
For each data in the dataset:
Add all XORSPLIT relation to xors
For each data in dataset xors:
Search all weight values in XORSPLIT
Print activity A->_O((activity B_wVC_wVD_w))
If the relation of the data is XORJOIN
For each data in the dataset:
Add all XORJOIN relation to xorj
For each data in dataset xorj:
Search all weight values in XORJOIN
Print activity _O((activity B_wVC_wVD_w))->_O(E)
If the relation of the data is ORSPLIT
For each data in the dataset:
Add all ORSPLIT relation to ors
For each data in dataset ors:
Search all weight values in ORSPLIT
Print activity A-><>((activity B_wVC_wVD_w))
If the relation of the data is ORJOIN
For each data in the dataset:
Add all ORJOIN relation to orj
For each data in dataset orj:
Search all weight values in ORJOIN
Print activity <>((activity B_wVC_wVD_w))->_O(E)

Weight *w* is assigned besides activity B_w, C_w, and D_w. If the graph model contains ANDJOIN relation, then the constructed W-LTL is defined as <>((activity B_w ^ C_w ^ D_w))->_O(E). Weight *w* is assigned besides activity B_w, C_w, and D_w. If the graph model contains XORSPLIT relation, then the constructed W-LTL is defined as A->_O((activity B_w V C_w V D_w)). Weight *w* is assigned besides activity B_w, C_w, and D_w. If the graph model contains XORJOIN relation, then the constructed W-LTL is defined as _O((activity B_wVC_wVD_w))->_O(E). Weight *w* is assigned besides activity B_w, C_w, and D_w. If the graph model contains ORSPLIT relation, then the constructed W-LTL is defined as A-><>((activity B_w V C_w V D_w)). Weight *w* is assigned besides activity B_w, C_w, and D_w. If the graph model contains ORJOIN relation, then the constructed W-LTL is defined as <>((activity B_w V C_w V D_w))->_O(E). Weight *w* is assigned besides activity B_w, C_w, and D_w.

Table 4. Algorithm for Weighted-Tree Declarative Pattern

Input :W-LTL y **Output** :W-TDP z
Algorithm W-TDP
While W-LTL are being modeled :
Split W-LTL into pW-LTL as part of W-LTL separated by comma
For pW-LTL in W-LTL :
If pW-LTL is “(”
Then make child of node, pointer is in the child of node
Else if pW-LTL is “)”
Then place the pointer position in the parent of node
Else if pW-LTL is “O”
Then create “O”, bracket next to this part is skipped
Else if pW-LTL is “<>”
Then create “<>”, bracket beside is skipped
Else if pW-LTL is “V”
If node is “O”
Then create node “X”
Else If node is “<>”
Then create node “V”
Else if pW-LTL is “^” or “->”
If node is “X” or “V”
Then make child node and fill in the node with “->” or “^”
Child node of “X” or “V” become child node of “->” or “^”
Else   Fill in the node name with pW-LTL ("ActNode^*w*-value"), pointer at parent

Received: July 27, 2019

3.2 Building a weighted-Tree declarative pattern (W-TDP) model

Given W-TDP z of order n with a set of nodes $N = \{n_1, n_2, n_3, \ldots, n_x\}$, and a set of edges $E = \{e_1, e_2, e_3, \ldots, e_{x-1}\}$, and weights $w = \{w_1, w_2, w_3, \ldots, w_{x-1}\}$. The tree model is built based on all discovered patterns in W-LTL. The first W-LTL to be processed is the control-flow pattern that has `Start_Act(act)` as the first activity to be executed. Then, the W-LTL is split into parts of W-LTL (pW-LTL) as explained in Subsection 2.3. To construct the W-TDP model, Table 4 is used as the modification result from the algorithm in [16] by inserting the weight after `ActNode`. The w-values from the declarative pattern are inserted into the tree model by adding operator '^' before the w -values in the declarative template. For example, `Initializing Game ->_O (Collecting Company Expenditure Data Every Month` $_{1.00}$`)` becomes `Initializing Game ->_O (Collecting Company Expenditure Data Every Month^`$_{1.00}$`)` as shown in the last line.

3.3 The proposed execution probability $p(e)$

As shown in Definition 6, this paper proposes execution probability $p(e)$ to weigh the control-flow pattern in the proposed TPED and Cosine-TDP similarity.

Definition 6. Execution probability *p(e)*. *Let* $p(e)$ *be the execution probability of activities in the relations and e is the control-flow patterns.* $p(e)$ *is ¼ iff e =* `<>` *and* `^`*;* $p(e)$ *is ½ iff e =* `<>` *and* `V`*;* $p(e)$ *is ¾ iff e =* `_O` *and* `X`*;*

The value of $p(e)$ is derived from truth table as shown in Table 5. We use uppercase letters to represent variables (in this context, we use A, B, and C). The variables are always said to have truth value. It is said that we have 2^n possible truth combination from n set of variables. Because we have 3 variables, then the possible truth combinations are $2^3 = 8$. Next, we assign truth value (either True (T) or False (F)) to AND Logic, OR Logic, XOR Logic, and Implication Logic. AND Logic has True (T) value as long as these 3 variables are all True (T). However, OR Logic has True (T) value as long as there is at least one True (T) among those 3 variables. Next, XOR Logic has True value where there is exactly one True value among 3 variables. We assign value rate to give score to relatonships types and rank them ascendingly. Value rate is the possibility of True (T) value based on the truth table.

Table 5. The truth table logic and the proposed $p(e)$

Variables			`<> ^` AND	`<> V` OR	`_O V` XOR	Sequence `(A→B)→C`
A	B	C				`-> _O`
F	F	F	F	F	F	F
F	F	T	F	T	T	T
F	T	F	F	T	T	F
T	F	F	F	T	T	T
F	T	T	F	T	F	T
T	F	T	F	T	F	T
T	T	F	F	T	F	F
T	T	T	T	T	F	T
Value Rate			3	1.714	1	0.6
rank			1	2	3	4
$p(e) = rank/n$			1/4	2/4 = ½	3/4	4/4 = 1
n is the number of relationship types (Sequence, AND, OR, and XOR)						

Then, we give the execution probability value $p(e) = rank/n$, where $rank$ is the rank of the value rate and n is the number of relationships that this research used (Sequence, AND, OR, and XOR).

The first example is `Start_Act(act) -> <> (A ^ B ^ C)`, where `Start_Act(act)` is the first activity and A, B, and C occur concurrently in AND relationship seperated by "^". The LTL rules used in this relationship are `<>` and `^`. For AND relationship type, rule "^" is used to assign True value. True value is based on AND Logic, where all variables have to be True (consider the yellow shades). Hence, the value rate is the number of True from the selected combination divided by True value in AND Logic. Therefore, the value rate of AND relationship type is $3/1 = 3$.

The second example is `Start_Act(act) -> <> (A V B V C)`, where `Start_Act(act)` is the first activity and A, B, and/or C occur in OR relationship seperated by "V". The LTL rules used in this relationship are `<>` and `V`. For OR relationshp type, the true value is based on OR Logic, where there should be at least one True among 3 variables (consider the shades area). Hence, the value rate is the number of True from the selected combination divided by True value in OR Logic. Therefore the value rate of OR relationship type is $12/7$.

The third example is `Start_Act(act) -> _O (A V B V C)`, where `Start_Act(act)` is the first activity and A, B, or C occur in XOR relationship seperated by "V". The LTL rules used in this relationship are `_O` and `V`. The rule of XOR is the combination of OR (`V`) and Sequence (`_O`). Therefore, we choose exactly one True value among 3 variables (consider the green shades). Hence, the value rate is the number of True from the selected combination divided by True value in XOR Logic. Therefore the value rate of OR relationship type is $3/3 = 1$.

The last example is `Start_Act(act->_O(A)->_O(B)->_O(C)`, where `Start_Act(act)` is the first activity and A, B, and C occur sequentially. The LTL rule used in this relationship is `_O`, where there is next activity executed sequentially. All variables have to be true (consider the yellow shades) and divided by True value in Implication logic. Hence, the value rate is $3/5 = 0.6$.

3.4 Similarity analysis

In this research, we propose two similarity methods: Tree Declarative Pattern Edit Distance (TPED) to calculate structural similarity and Cosine-Tree Declarative Pattern (Cosine-TDP) to calculate behavioral similarity. As described in Definition 7, TPED is a modification of GED and is used to differentiate the types of nodes and edges in the declarative pattern contained in W-TDPs. TPED are the absolute value of $|1-(\overline{ns}+\overline{es}+\overline{nb})|$. Meanwhile, as defined in Definition 8, Cosine-TDP is a modification of the Cosine measure and is used to compare the occurrence probability of activities between W-TDPs. $p(e)$ is added to give weight to the control-flow patterns contained in two models. The proposed algorithm for computing the similarity between W-TDPs is shown in Table 6.

Definition 7. Tree Declarative Pattern Edit Distance (TPED). *Let $M_x = (N_1, E_1, w_1, p(e)_1)$ and $M_y = (N_2, E_2, w_2, p(e)_2)$ be the first and the second weighted-tree Declarative Pattern model respectively. Let $p(e)$ be the execution probability to weigh the relationships. The similarity of Tree Declarative Pattern Edit Distance $Sim_{TPED}(M_x, M_y)$ is denoted as: $min_{((s_1,\dots,s_k),(p(e)_1,\dots,p(e)_k))\in P(M_x,M_y)} \sum_{i=1}^{k} c(s_1)$, where $P(M_x,M_y)$ is the set of edits needed to transform M_x into M_y and $c(x) \geq 0$ is the cost of TPED to perform operation s.*

The equations to compute TPED are shown in Eq. (8) to Eq. (11):

$$Sim_{TPED}(M_x, M_y) = |1-(\overline{ns}+\overline{es}+\overline{nb})| \quad (8)$$

$$\overline{ns} = \frac{|ni|+p(e)_x+p(e)_y}{|Nx|+|Ny|} \quad (9)$$

$$\overline{es} = \frac{|ei|}{|Ex|+|Ey|} \quad (10)$$

$$\overline{nb} = \frac{2.\sum(\frac{edit}{max})}{|Nx|+|Ny|-(|ni|+p(e)_x+p(e)_y)} \quad (11)$$

where

$\overline{ns}$ is the average number of **node** insertions or deletions in W-TDPs,

$\overline{es}$ is the average number of **edge** insertions or deletions in W-TDPs,

$\overline{nb}$ is the average number of nodes label substitution in W-TDPs,

$|ni|$ is the number of node insertions or deletions in W-TDPs,

$p(e)_x$ is the exection probability $p(e)$ to weigh the control-flow patterns in Model M_x,

$p(e)_y$ is the exection probability $p(e)$ to weigh the control-flow patterns in Model M_y,

$|Nx|$ is the number of nodes in the first W-TDP model,

$|Ny|$ is the number of nodes in the second W-TDP model,

$|ei|$ is the number of node insertions or deletions in W-TDPs,

$|Ex|$ is the number of nodes in the first W-TDP model,

$|Ey|$ is the number of nodes in the second W-TDP model, and

$\frac{edit}{max}$ is $\frac{|p(e)_x - p(e)_y|}{\max(p(e)_x\ ,\ p(e)_y)}$.

Definition 8. Cosine-Tree Declarative Pattern (Cosine-TDP). *Let $M_x = (N_1, E_1, w_1, p(e)_1)$ and $M_y = (N_2, E_2, w_2, p(e)_2)$ be the first and the second weighted-tree declarative pattern model respectively. Let $p(e)$ be the execution probability to weigh the relationships. The similarity of Cosine-Tree Declarative Pattern (Cosine-TDP) is denoted as: $p(e)_1N_1 \ .\ p(e)_2N_2 + p(e)_1p(e)_2 - (p(e)_1 - p(e)_2) = ||p(e)_1N_1 + p(e)_1||.||p(e)_2N_2 + p(e)_2||cos\,\theta$.*

The equation to compute Cosine-TDP is shown in Eq. (12):

$$Sim_{Cosine-TDP}(M_x, M_y) = \frac{(\sum p(e)_{i,x} \, . \, w_{at_{i,x}} \, . \, p(e)_{i,y} \, . \, w_{at_{i,y}}) + (\sum p(e)i_{,x} \, . \, p(e)i_{,y}) - edit}{\sqrt{\sum(p(e)_{i,x}{}^2 + (p(e)_{i,x}(w_{at_{i,x}}))^2) \, . \, \sum(p(e)_{i,y}{}^2 + (p(e)_{i,y}(w_{at_{i,y}}))^2)}} \quad (12)$$

Where

Table 6. Algorithm of the proposed similarity methods

```
Input: Weighted-Tree Declarative Pattern Model W-TDP
Output:
$Sim_{TPED}(M_x, M_y), Sim_{Cosine-TDP}(M_x, M_y), Sim_{all}(M_x, M_y$
Algorithm $Sim(M_x, M_y)$
Foreach W-TDP$_x$, W-TDP$_y$ do
    Compute $Sim_{TPED}(M_x, M_y)$ for structural similarity
    Compute $Sim_{Cosine-TDP}(M_x, M_y)$ for behavioral similarity
    Compute $Sim_{all}(M_x, M_y)$ for structural similarity
end for
```

w_{at_x} is the weight of an activity in Model M$_x$,
w_{at_y} is the weight of an activity in Model M$_y$, and
$edit$ is $|p(e)_x - p(e)_y|$.

The equation to compute the overall similarity between W-TDPs is shown in Eq. (13):

$$Sim_{all}(M_x, M_y) = \alpha Sim_{Cosine-TDP}(M_x, M_y) + (1-\alpha) Sim_{TPED}(M_x, M_y) \quad (13)$$

where α is the threshold (here, we used 0.5 as a threshold because we use two similarity metrics for similarity analyses: structural similarity and behavioral similarity, therefore the total similarity value is the addition of two similarities divided by 2).

4. Results and discussion

This research was aimed at getting more accurate similarity measures between W-TDPs. The proposed methods were evaluated in a retail business process. Fig. 3 shows the real-life complex retail graph model, which has 23 activities as mentioned in Table 7.

Referring to Table 7, the activities in the process are: initializing game, collecting company expenditure data every month, updating product configuration data, choosing supplier, restocking automatically, receiving items automatically, adding items automatically, calculating market share, recording items purchase journal, selling items based on market share, delivering items automatically, calculating EOQ, calculating supplier selection, calculating ROP, calculating optimal price, recording items sales journal, making maximum round checks, calculating the highest profit item, calculating the highest income item, calculating ROA, calculating ROI, calculating ROE, and displaying dashboard.

Table 7. Activities in event log and their aliases

Activities	Aliases	Activities	Aliases
Initializing game	A	Calculating supplier selection	M
Collecting company expenditure data every month	B	Calculating ROP	N
Updating product configuration data	C	Calculating optimal price	O
Choosing supplier	D	Recording items sales journal	P
Restocking automatically	E	Making maximum round checks	Q
Receiving items automatically	F	Calculating the highest profit item	R
Adding items automatically	G	Calculating the highest income item	S
Calculating market share	H	Calculating ROA	T
Recording items purchase journal	I	Calculating ROI	U
Selling items based on market share	J	Calculating ROE	V
Delivering items automatically	K	Displaying dashboard	W
Calculating EOQ	L		

In real-life complex retail graph model, there is General Ledger department which is used as analysis material. The General Ledger department consists of the following activities: ‘Delivering items automatically (symbolized as K)’, ‘Calculating supplier selection (symbolized as L)’, ‘Calculating optimal price (symbolized as M)’, ‘Calculating ROP (symbolized as N)’, ‘Calculating EOQ (symbolized as O)’, and ‘Recording items sales journal (symbolized as P)’. Suppose that we have three types of general ledger part containing AND relation, OR relation, and XOR relation as shown in Fig. 4.

After discovering the graph model, the W-LTL declarative pattern was constructed. Table 8 shows the results of composing the W-LTL declarative pattern and Table 9 shows the result of composing the tree model from the W-LTL declarative pattern.

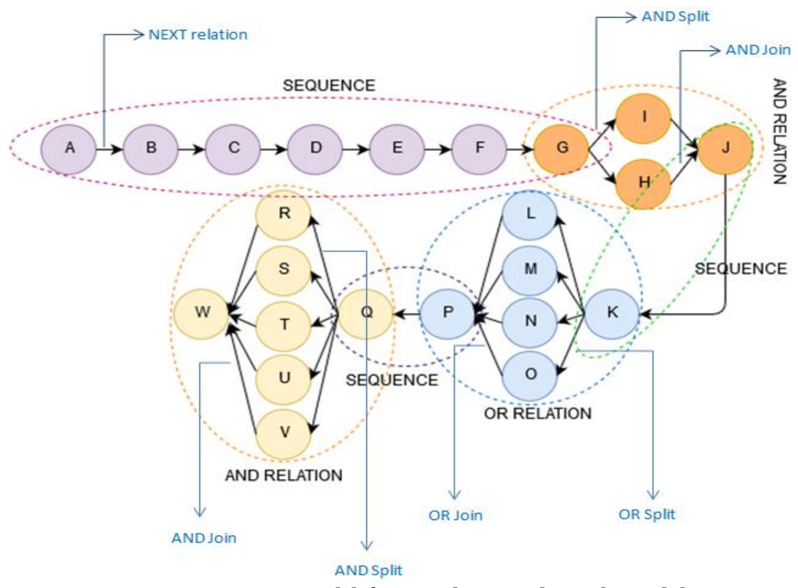


Figure. 3 Real-life complex retail graph model

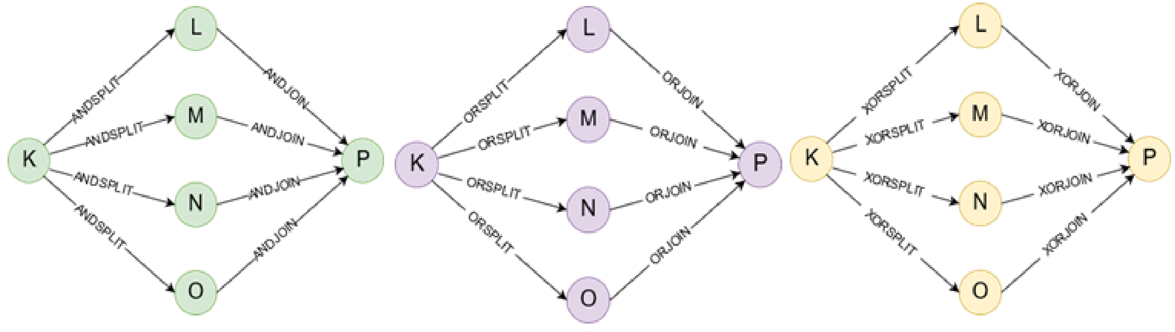


Figure. 4 General ledger part: (a) AND relations (b) OR relations, and (c) XOR relations

Table 8. Results of composing the W-LTL Declarative Pattern of general ledger

AND Split Relation | AND Join Relation	K -><> (($N_{0.26}$ ^ $O_{0.22}$ ^ $L_{0.17}$ ^ $M_{0.35}$)) <> (($M_{0.35}$ ^ $L_{0.13}$ ^ $N_{0.17}$ ^ $O_{0.35}$)) ->_O (P)
OR Split Relation | OR Join Relation	K -><> (($L_{0.23}$ V $M_{0.27}$ V $N_{0.27}$ V $O_{0.23}$)) <> (($O_{0.35}$ V $N_{0.19}$ V $L_{0.23}$ V $M_{0.23}$)) -> _O (P)
XOR Split Relation | XOR Join Relation	K ->_O (($L_{0.10}$ V $N_{0.30}$ V $O_{0.20}$ V $M_{0.40}$)) _O (($L_{0.10}$ V $M_{0.40}$ V $N_{0.30}$ V $O_{0.20}$)) ->_O (P)

Table 9. Results of composing the Weighted-Tree Declarative Pattern model of General Ledger

AND Split | AND Join	OR Split | OR Join	XOR Split | XOR Join
-> ├─── K^$_{1.00}$ ├─── ^ │ ├─── N^$_{0.26}$ │ ├─── O^$_{0.22}$ │ ├─── L^$_{0.17}$ │ └─── M^$_{0.35}$ ├─── P Elements: {->K, ->^, ^N, ^O, ^L, ^M, ->P}	-> ├─── K^$_{1.00}$ ├─── V │ ├─── L^$_{0.23}$ │ ├─── M^$_{0.27}$ │ ├─── N^$_{0.27}$ │ └─── O^$_{0.23}$ ├─── P Elements: {->K, ->V, VL, VM, VN, VO, ->P}	-> ├─── K^$_{1.00}$ ├─── X │ ├─── L^$_{0.10}$ │ ├─── N^$_{0.30}$ │ ├─── O^$_{0.20}$ │ └─── M^$_{0.40}$ ├─── P Elements: {->K, ->X, XL, XN, XO, XM, ->P}
Element values: {1, ¼, 0.26, 0.22, 0.17, 0.35, 1}	Element values: {1, ½, 0.23, 0.27, 0.27, 0.23, 1}	Element values: {1, ¾, 0.1, 0.3, 0.2, 0.4, 1}
We assigned the value of ->^, ->X, and ->V with the proposed constraint $p(e)$ as declared in Table 5.		

The proposed similarity methods were validated by comparing the similarity results of structural similarity (the proposed TPED in Eq. (8)-(11)), behavioral similarity (the proposed Cosine-TDP in Eq. (12), and overall similarity using Eq. (13), to those of several existing methods, based on General Ledger part.

4.1 Structural similarity

We compare each W-TDPs in Table 9 (tree model containing AND relation versus OR relation, OR relation versus XOR relation, and AND relation versus XOR relation) by inputting the elements into the proposed structural similarity method, TPED. The calculation example below is to show the calculation result of W-TDP in AND relation versus OR relation in structural aspect based on Eq. (8) to Eq. (11).

$$\overline{ns} = \frac{|ni| + p(e)_x + p(e)_y}{|Nx| + |Ny|} = \frac{1 + \frac{1}{4} + \frac{1}{2}}{8 + 8} = 0.109$$

$$\overline{es} = \frac{|ei|}{|Ex| + |Ey|} = \frac{0}{7 + 7} = 0$$

$$\overline{nb} = \frac{2.\sum\left(\frac{edit}{max}\right)}{|Nx| + |Ny| - (|ni| + p(e)_x + p(e)_y)}$$

$$= \frac{2.\frac{\frac{1}{2}-\frac{1}{4}}{\frac{1}{2}}}{8 + 8 - 1.75} = 0.0702$$

$$Sim_{TPED}(M_{\text{AND}}, M_{\text{OR}}) = \left|1 - \left(\overline{ns} + \overline{es} + \overline{nb}\right)\right|$$
$$= |1 - (0.109 + 0 + 0.0702)|$$
$$= 0.821$$

4.2 Behavioral similarity

The next step is to compute behavioral similarity using the proposed method, Cosine-TDP. The calculation example below is to show the calculation result of W-TDP in AND relation versus OR relation in behavioral aspect. We compare each element and element value between W-TDP in AND relation and OR relation based on Eq. (12).

$$\boldsymbol{Sim_{Cosine-TDP}(M_{AND}, M_{OR})} =$$
$$\frac{(1x1)+(1x1)+\left(\frac{1}{4}x\frac{1}{2}\right)-\left(\frac{1}{2}-\frac{1}{4}\right)}{\sqrt{(1^2+1^2+(\frac{1}{4})^2+\left(\frac{0.26}{4}\right)^2+..+\left(\frac{0.35}{4}\right)^2)(1^2+1^2+(\frac{1}{2})^2+(\frac{0.23}{2})^2+..+\left(\frac{0.23}{2}\right)^2)}}$$

$$Sim_{Cosine-TDP}(M_{AND}, M_{OR}) = 0.834$$

As shown in Table 10, it was proved that the proposed methods are able to produce better results in differentiating the operators of parallel relationships. Meanwhile, the existing methods gave the same result (Jaccard, TARs, and GED). The existing Cosine measure produced the lowest value for AND versus OR relations, followed by AND versus XOR relations. OR versus XOR relations had the highest value. However, the proposed methods were able to produce a better similarity result (see the justification in Table 5) compared to the existing methods. AND versus OR relations had the highest value, followed by OR versus XOR and AND versus XOR. Based on Table 5, AND relation has the highest value rate followed by OR relation. Meanwhile XOR relation has the lowest value rate. Therefore, matching two model containing AND versus OR is greater than OR versus XOR and greater than AND versus XOR.

5. Conclusion

Two new similarity methods to calculate structural similarity and behavioral similarity were proposed and evaluated in this paper: Tree Declarative Pattern Edit Distance (TPED) and Cosine-Tree Declarative Pattern (Cosine-TDP). They are based on Weighted-Tree Declarative Pattern models (W-TDP) using Weighted-Linear Temporal Logic (W-LTL). The proposed similarity methods were evaluated by comparing their results with the existing methods Graph Edit Distance (GED) and Cosine measure, which were applied to a complex graph model.

Since AND > OR > XOR, the similarity value of AND versus OR is greater than that of XOR versus OR and AND versus XOR, while the similarity value of XOR versus OR lies between AND versus OR and AND versus XOR, and that of AND versus XOR is the lowest. The proposed methods, TPED and Cosine-TDP, were able to differentiate between AND, OR, and XOR relations more accurately compared to the existing methods. For structural similarity, the results of the proposed method (TPED) were 0.821, 0.811, and 0.78 for AND versus OR, XOR versus OR, and AND versus XOR, respectively, while the result of the existing method (GED) is 1 for all relations. For behavioral similarity, the results of the proposed method were 0.834, 0.826, and 0.693, while the results of the existing methods were 0.02, 0.08, and 0.04 for AND versus OR, XOR versus OR, and AND versus XOR, respectively. Lastly, the results of overall similarity of the proposed methods were 0.828, 0.819, and 0.737, while the overall results of the existing methods were 0.51, 0.54, and 0.52 for AND versus OR, XOR versus OR, and AND versus XOR, respectively.

Table 10. Result of comparison between the proposed methods and existing methods

Operators of Parallel Relation	Existing Methods				Proposed Methods		
	Jaccard [28], [29]	TARs [27]	Cosine [25]	GED [23]	TPED	Cosine-TDP	Overall Similarity
AND versus AND	1	1	1	1	1	1	1
OR versus OR	1	1	1	1	1	1	1
XOR versus XOR	1	1	1	1	1	1	1
AND versus OR	1	1	0.02	1	0.821	0.834	0.828
OR versus XOR	1	1	0.08	1	0.811	0.826	0.819
AND versus XOR	1	1	0.04	1	0.78	0.693	0.737

It can be concluded that the proposed methods are able to give better result in differentiating the operators of parallel relationships (AND > OR > XOR). Meanwhile, the existing methods give result where OR > XOR > AND.

Acknowledgments

The authors would like to sincerely thank Institut Teknologi Sepuluh Nopember, Ministry of Research, Technology and Higher Education of Indonesia, *Direktorat Riset dan Pengabdian Masyarakat*, and *Direktorat Jenderal Penguatan Riset dan Pengembangan Kementerian Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia* for supporting this research.

References

[1] Y. A. Effendi and R. Sarno, "Non-Linear Optimization of Critical Path Method", In: *Proc. of International Conf. on Science in Information Technology (ICSITech)*, pp. 90–96, 2017.

[2] Y. A. Effendi and R. Sarno, "Discovering Process Model from Event Logs by Considering Overlapping Rules", In: *Proc. of International Conf. on Electrical Engineering, Computer Science and Informatics (EECSI 2017)*, pp. 19–21, 2017.

[3] N. Russell, A. H. M. Hofstede, W. M. P. Van Der Aalst, and N. Mulyar, "Workflow Control-Flow Patterns: A Revised View", Netherlands, 2006.

[4] R. Sarno and K. R. Sungkono, "Coupled Hidden Markov Model for Process Discovery of Non-Free Choice and Invisible Prime Tasks", In: *Proc. of the 4th Information Systems International Conference*, pp. 134–141, 2017.

[5] R. Sarno and K. R. Sungkono, "Hidden Markov Model for Process Mining of Parallel Business Processes", *International Review on Computers and Software (IRECOS)*, Vol. 11, No. 4, pp. 290–300, 2016.

[6] M. Baumann, "Comparing Imperative and Declarative Process Models with Flow Dependencies", In: *Proc. of 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pp. 63–68, 2018.

[7] K. R. Sungkono, R. Sarno, and N. F. Ariyani, "Refining Business Process Ontology Model with Invisible Prime Tasks using SWRL Rules", In: *Proc. of International Conf. on Information & Communication Technology and System (ICTS)*, pp. 215–220, 2017.

[8] R. Flowers and C. Edeki, "Business Process Modeling Notation", *International Journal of Computer Science and Mobile Computing*, Vol. 2, No. 3, pp. 35–40, 2013.

[9] M. P. Cabasino, A. Giua, and C. Seatzu, "Introduction to Petri Nets", *(eds) Control of Discrete-Event Systems. Lecture Notes in Control and Information Sciences*, Vol. 433, pp. 191–211, 2013.

[10] W. M. P. Van Der Aalst and A. H. M. Hofstede, "YAWL : Yet Another Workflow Language", *Information System*, Vol. 30, No. 4, pp. 245–275, 2005.

[11] A. Amjad, F. Azam, M. W. Anwar, W. H. Butt, M. Rashid, and A. Naeem, "UMLPACE for Modeling and Verification of Complex Business Requirements in Event- driven Process Chain (EPC)", *IEEE Access*, Vol. 6, pp. 76198–76216, 2018.

[12] P. Pichler, B. Weber, S. Zugal, J. Pinggera, J. Mendling, and H. A. Reijers, "Imperative versus Declarative Process Modeling Languages : An Empirical Investigation", In: *Proc. of International Conf. on Business Process Management*, pp. 383–394, 2012.

[13] V. V. Rybakov, "Non-transitive Linear Temporal Kogic and Logical Knowledge Operations", *Journal of Logic and Computation*, Vol. 26, No. 3, pp. 945–958, 2016.

[14] R. Marah and A. E. L. Hibaoui, "Formalism of Self-Stabilization with Linear Temporal Logic and its Verification", In: *Proc. of Third World*

Conf. on Complex Systems (WCCS), pp. 1–5, 2015.

[15] M. M. Maryamah, R. Sarno, and M. A. Nurlaili, "Determining Linear Temporal Logic Formula for Decomposed Process Model", In: *Proc. of International Conf. on Information and Communications Technology (ICOIACT)*, pp. 466–470, 2018.

[16] K. R. Sungkono and R. Sarno, "Constructing Control-Flow Patterns Containing Invisible Task and Non-Free Choice Based on Declarative Model", *International Journal of Innovative Computing, Information and Control (IJICIC)*, Vol. 14, No. 4, pp. 1285–1299, 2018.

[17] N. M. Mahmod and W. Y. Chiew, "Structural Similarity of Business Process Variants", In: *Proc. of IEEE Conf. on Open Systems (ICOS 2010)*, pp. 17–22, 2010.

[18] N. Syukriilah, D. S. Kusumo, and S. Widowati, "Structural Similarity Analysis of Business Process Model Using Selective Reduce Based on Petri Net", In: *Proc. of International Conf. on Information and Communication Technology (ICoICT) Structural*, pp. 1–5, 2015.

[19] H. Huang, R. Peng, and Z. Feng, "A time-aware method to process behavioral similarity calculation", In: *Proc. of IEEE International Conf. on Services Computing*, pp. 395–402, 2016.

[20] J. Sun, T. Gu, and J. Qian, "A Behavioral Similarity Metric for Semantic Workflows Based on Semantic Task Adjacency Relations With Importance", *IEEE Access*, Vol. 5, pp. 15609–15618, 2017.

[21] E. Kuss, H. Leopold, H. Van Der Aa, H. Stuckenschmidt, and H. A. Reijers, "A Probabilistic Evaluation Procedure for Process Model Matching Techniques", *Data & Knowledge Engineering*, Vol. 117, pp. 1–14, 2018.

[22] M. J. Amiri and M. Koupaee, "Data-driven Business Process Similarity", *IET Software*, Vol. 11, No. 6, pp. 309–318, 2017.

[23] R. Dijkman, M. Dumas, B. Van Dongen, K. Reina, and J. Mendling, "Similarity of business process models : Metrics and evaluation", *Information System*, Vol. 36, pp. 498–516, 2011.

[24] K. G. Saranya and G. S. Sadasivam, "Modified Heuristic Similarity Measure for Personalization using Collaborative Filtering Technique", *Applied Mathematics & Information Sciences*, Vol. 11, No. 1, pp. 307–315, 2017.

[25] J. Y. Jung, J. Bae, and L. Liu, "Hierarchical Clustering of Business Process Models", *International Journal of Innovative Computing, Information and Control (IJICIC)*, Vol. 5, No. 12, pp. 613–616, 2009.

[26] M. Arriagada-benítez, M. Sepúlveda, and J. C. A. M. Buijs, "Strategies to Automatically Derive a Process Model from a Configurable Process Model Based on Event Data", *Applied Science*, Vol. 7, pp. 1–28, 2017.

[27] D. Rahmawati, L. N. Aini, R. Sarno, C. Fatichah, and D. Sunaryono, "Comparison of Behavioral Similarity use TARs and Naïve Algorithm for Calculating Similarity in Business Process Model", In: *Proc. of International Conf. on Science in Information Technology (ICSITech)*, pp. 115–120, 2017.

[28] A. C. Fauzan, R. Sarno, M. A. Yaqin, and A. Jamal, "Extracting Common Fragment based on Behavioral Similarity using Transition Adjacency Relations for Scalable Business Processes", In: *Proc. of International Conf. on Science in Information Technology (ICSITech)*, pp. 131–136, 2017.

[29] A. C. Fauzan, R. Sarno, and M. A. Yaqin, "Performance Measurement Based on Coloured Petri Net Simulation of Scalable Business Processes", In: *Proc. of International Conf. on Electrical Engineering, Computer Science and Informatics (EECSI)*, pp. 1–6, 2017.