Enhancing the Quality of Merged Process Models by Addressing Invisible Task

Kelly Rossa Sungkono [1,*)], Riyanarto Sarno [2)], I Gusti Agung Chintya Prema Dewi [3)], and Muhammad Suzuri Hitam [4)]

[1, 2, 3)] Informatics Department, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia

[4)] Faculty Of Computer Science and Mathematics, Universiti Malaysia Terengganu
Kuala Nerus Terengganu Darul Iman, Malaysia

E-mail: kelly@its.ac.id[1)], riyanarto@its.ac.id[2)], chintyapremadewi709@gmail.com[3)], and suzuri@umt.edu.my[4)]

ABSTRACT

Model merging is a key approach for integrating multiple process model variants into a unified representation. Existing automated merging methods face challenges in handling invisible tasks, which are intentionally inserted in the process model to depict certain conditions, including stacked branching relationships. The inability to handle invisible tasks reduces the quality of the merged process models. A proposed Graph-merging method explicitly addresses sequence, branching relationships, and invisible tasks. The proposed method first identifies common activities across model variants. Furthermore, the method applies the proposed graph rules grounded in behavioral and structural aspects to combine those common activities as well as their related relationships and generate the graph-based merged process model. Behavioral rules govern the integration of sequence and branching relationships, while structural rules handle branching and invisible tasks. An evaluation against three existing approaches by Derguech, Yohanes and Graph Semantic Similarity demonstrates that the proposed Graph-merging method achieves higher precision. The Graph-merging method substantially improves the quality of merged process models.

Keywords: Behavioral analysis, graph-based model integration, process model merging, structural analysis, quality enhancement.

1. Introduction

Process models play an essential role in business process management (BPM) by providing a clear and detailed visual representation of how processes operate within an organization. The structure visualization of process models eliminates ambiguity and guesswork, ensuring that all stakeholders have a shared understanding of the process flow. Organizations can more easily identify inefficiencies by clearly outlining the sequence of activities, such as bottlenecks, redundant activities, or underutilized resources. Moreover, process models provide the standardization procedure across different divisions or locations, ensuring consistent execution and improving overall operational efficiency. Process models are applied in various domains, including streamlined production workflows in the manufacturing sector [1], patient paths in the healthcare sector [2], and a blockchain sector [3]. As such, process models are essential for organizations seeking to optimize their processes, improve decision-making, and maintain competitive advantage.

Model merging integrates multiple model variations into a consistent model. Model merging differs from process discovery, where several process discovery methods ([4],[5], and [6]) have been published by the researchers of this paper. The process discovery forms a process model based on the restored processes in the event log, while the model merging combines several process models into a consolidated process model. The model merging technique is crucial to comprehensively understanding the entire process and analyzing the relationships

* Corresponding author.
Received: September 9th, 2025. Revised: November 4th, 2025. Accepted: December 1st, 2025.
Available online: January 15th, 2026.

DOI: https://doi.org/10.12962/j24068535.v24i1.a1381

between paired models [7]. Some considerations related to the model merging process are variation in activity patterns [8], conflict resolution [9], [10], and semantic similarity in the activity description [11], [12]. In the context of business process management, process merging consolidates several variations of a business process model into one unified process model [13]. The process model merging considers behavioral and structural aspects. The behavioral aspect focuses on the relationship of activities in the process, while the structural aspect concerns the control flows between activities [14]. Behavioral relationships between activities can be analyzed through event logs, which record the sequence and time of activities [15]. Meanwhile, the structural aspect comes from the underlying process model, which captures the activity control flows [16].

There are various patterns of process model relationships, namely sequence, branching, invisible prime tasks, and invisible non-prime tasks. Sequential relationships execute a sequence of activities in a process. Branched relationships, such as XOR, OR, and AND, define the selected activity path [17]. XOR is used when only one of the choice activities is processed in all processes, OR is used when several choice activities are executed, and AND is used when all choice activities without restricted order are executed in every process [18]. However, not all conditions can be depicted by sequence or branching relationships. A process model that utilizes only sequence and branching relationships will misrepresent skipped, redo, and switch conditions, so the model needs to add invisible prime tasks. Invisible prime tasks are activities intentionally inserted into a process model to correctly describe certain conditions [4], [19]. Stacked branching cannot be described using invisible prime tasks, so that the invisible non-prime tasks are formed [5]. Some conditions need combinations of invisible tasks, such as invisible prime tasks for skipped conditions and invisible non-prime tasks for stacked branching relationships [5].

Manual merging of process models is at risk of inaccuracy and takes a long time, resulting in the emergence of automatic merging methods. There are several existing automatic merging methods. Derguech [20] proposed a technique to combine business processes by utilizing two business processes with several activities with similar names and combining these activities to create a new process. Then, the process will enter the post-processing stage, where restructuring will be carried out on the relationship and process form to produce a new process that can be configured and meet the rules of the process model used when conducting research. In addition, Zemni [21] proposes the consolidation of business processes by using process pieces (not whole processes) and utilizing the Gateway Path Matrix (GPM), which consists of relationship elements that have the same source and target of activity or can also be called a Relationship Matrix. The study divided the method into two phases, namely the merger phase and the structural reconstruction phase of the process. Both studies provide a comprehensive implementation in forming a single business process based on two different business processes. Still, the method is only compatible with process models that have sequence and branching relationships. On the other side, Yohanes [18] addressed a novel method to enumerate the resemblance between the branching relationship of identical activities from several business process event logs. The work successfully measures the accurate resemblance value between branching relationships. Those existing methods have the same weakness: the inability to merge or measure resemblance between process models obtaining invisible tasks. The inability to merge process models results in a decrease in the quality of the resulting combined process models.

This research proposes a Graph-merging method as a new automated process model merging method. This research is essential because process model incorporation automatically avoids the risk of inaccuracy of results and improves the efficiency of process model incorporation. This research utilizes graph language [22], [23], [24] to conduct the proposed rules of the merging process models, including merging process models involving invisible tasks. This research considers behavioral aspects to combine process models involving sequence and branching relationships, i.e., XOR, OR, and AND, and uses structural aspects to accommodate branching relationships to produce merged process models containing invisible tasks. Thus, the contributions of this research are included the following:

1. proposing graph-based behavioral rules to merge process models containing sequence and branching relationships, i.e., XOR, OR and AND;

Table 1: Research gap compared to proposed method: graph-merging.

No	Papers	Main Focus of Paper	Research Gap with Proposed Method
1	[18]	Enumerating the resemblance between the branching relationship of identical activities from several business process event logs	Does not address structural combinations needing invisible tasks at the model level
2	[20]	Combining business processes which have several activities with similar names	Does not consider invisible tasks when merging processes
3	[11]	Merging business processes by considering semantic similarity of activities name	Does not depict invisible tasks of stacked branching relationships when combining models
4	[4]	Discovering a process model containing invisible tasks of stacked branching relationships (XOR-OR and XOR-AND) based on the event log	Forming a process model based on the event log; does not combines process models
5	[5]	Constructing a process model containing invisible tasks of stacked branching relationships (XOR-OR and XOR-AND) and non-free choice constructs	Processing event log; does not merged process models
6	[21]	Merging business processes by utilizing Gateway Path Matrix (GPM)	Only considering sequence and branching elationships
7	[19]	Calculating behavior differentiation of process variants considering invisible tasks	Using invisible tasks for behavioral comparison; does not unified process models
8	[25]	Discovering process variants by semantic discovery method	Depicting process variants, not a consolidated model; considering context semantic and does not regarding various types of relationships when merging
9	[26]	Analyzing the influence of structural distance on behavioral distance when comparing process variants	Comparing processes, not combining processes; does not focuses on invisible tasks
10	[27]	Extending Inductive Miner by discovering semantic-annotated model based on the event log	The input is an event log, not process models; does not explicitly discovering processes containing invisible tasks

2. proposing graph-based structural rules to merge process models containing branching relationships which produces invisible tasks;
3. proposing Graph-merging methods to increase the quality of merged process model.

The Graph-merging method will be evaluated by comparing the obtained merged models with those by other existing methods, i.e., Derguech [20] method and Yohanes [18] method. The evaluation uses process models from several case studies. The used quality measurements are correctness and precision. This research focuses on elevate the quality of combined process models using Graph-merging methods considering behavioral and structural aspects.

The remainder of this paper is organized as follows. Section Related Works explains the research gap and preliminaries of this research, such as business process models and graph formalization, process model relationships, behavioral and structural aspects, and the quality measurement. A detailed description of the proposed method, i.e., the Graph-merging method, is presented in Section Methods. The material, the evaluation and the analysis are described in Section Analysis and Result. The authors conclude the research in Section Conclusion.

2. Related Works

2.1. Business Processes Analysis (Process Mining) Methods

Process mining is a field of analyzing business processes by depicting the process model or gaining information based on the obtained process model [5], [15]. There are several approaches related to forming process models and analyzing configurable, consolidated, or variant process models. Based on Table 1, there is still a lack of model-to-model merging methods that consider various types of relationships, such as sequence and branching relationships, as well as additional invisible tasks. To address this gap, the proposed Graph-merging method combines process model variants into a single, consolidated process model, supporting heterogeneous relationships, including invisible tasks, within the proposed rules.

2.2. Business Process Models and Graph Formalization

A business process model structurally represents organizational activities and their flows. This model is depicted in several modeling languages, such as Business Process Model and Notation (BPMN) [28], Petri Net [29], and graph-based process model [5]. In this research, models are formalized as directed graphs $G := (Y,S)$, where Y is a collection of nodes (activities) and $S \subseteq Y \times Y \times Z$ is a collection of edges that represents Z relationships. For instance, $K :=$ ({*ky*,*ks*,*kn*},{(*ky*,*ks*,SEQUENCE),(*ks*,*kn*,SEQUENCE)}) means the process model K consists of three activities (Y) which are *ky*,*ks*,*kn* and has sequence relationships starting from *ky* , then *ks* and finally *kn* . This representation provides the formal basis required to define the rules for merging graph-based process models.

2.3. Relationships in the Process Models

Process models define fundamental relationships, such as sequence and branching (XOR, AND and OR) relationships. An XOR relationship indicates that exactly one activity from a given set is executed in a process instance. In contrast, an AND relationship denotes that all activities in the set must be executed. For example, a graph-based process model $G :=$ ({*kh*,*kj*,*kl*,*kn*,*kp*,*kt*,*kr*},{(*kh*,{*kj*,*kl*},AND),({*kj*,*kl*},*kn*,AND),(*kn*,{*kp*,*kt*},XOR), ({*kp*,*kt*},*kr*,XOR)}) is obtained based on four traces: ($kh \rightarrow kj \rightarrow kl \rightarrow kn \rightarrow kp \rightarrow kr$) , ($kh \rightarrow kl \rightarrow kj \rightarrow kn \rightarrow kp \rightarrow kr$) , ($kh \rightarrow kj \rightarrow kl \rightarrow kn \rightarrow kt \rightarrow kr$) , and ($kh \rightarrow kl \rightarrow kj \rightarrow kn \rightarrow kt \rightarrow kr$). An OR relationship specifies that one or more activities from the set are executed within a process. For instance, $G :=$ ({*kh*,*ki*,*kj*,*kl*,*km*}, {(*kh*, {*ki*,*kj*,*kl*},OR),({*ki*,*kj*,*kl*},*km*,OR)} represents traces containing activities *kh*, *km* and the combination of *ki*,*kj*,*kl*. In the graph-based process model, branching relationships are captured by the SPLIT and JOIN gateways, which are encoded as labeled edges within the model.

In this paper, the proposed method addresses invisible tasks. Invisible tasks are added in the process model, capturing skipped, redo, switch conditions, and stacked branching relationships. Invisible tasks of stacked branching relationships (XOR-AND) are denoted by gray circles in Table 2.

2.4. Behavioral and Structural Aspects of Process Merging

The behavior of a process model refers to the characteristics of the flow that a process model has. The characteristic of the flow in question is the sequence of possible execution of activities that exist in the process. The causality between activities in the process model can be illustrated by the execution flow that occurs [4], [30]. The structure of the process model refers to the dependencies between activities based on the relationships associated with each activity [14], [17], [31]. The example of merging process models based on behavioral and structural aspects is shown in Table 2.

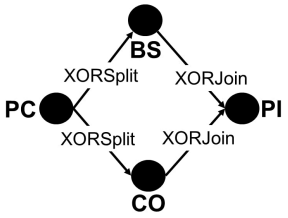
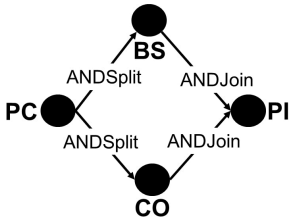
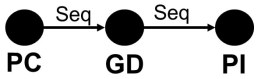
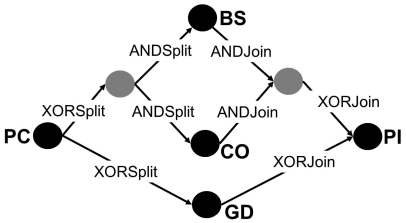
In the behavioral aspect example, the first variant model has a sequence relationship between activity PC and BS, then between activity BS and PI. The second variant model has a sequence relationship between PC and CO then between CO and PI. When merging those variant models, activity PC branches to BS and CO, so PC has XOR relationships to BS and CO. Then, activity PI has branches from BS and CO, so PI has XOR relationships from BS and CO. The resulting merged model is B' := ({PC,BS,CO,PI},{(PC,{BS,CO},XOR),({BS,CO},PI,XOR)}), as visualized in Table 2 where XOR relationships are denoted as XORSplit and XORJoin gateways in the graph model.

In the structural aspect example, the first variant process model applies AND relationship from PC (symbolized by ANDSplit gateway) and AND relationship to PI (symbolized by ANDJoin gateway); whereas, the second variant employs SEQUENCE relationships for the same activities. Merging these structures yields stacked branching (XOR-AND) relationships [5], necessitating the insertion of invisible tasks. The resulting merged model is B' := ({PC,invisibleTask,BS,CO,GD,invisibleTask1,PI},{(PC,{invisibleTask,GD},XOR),(invisibleTask, {BS,CO},AND),({BS,CO},invisibleTask1,AND),({invisibleTask1,GD},PI,XOR}) which is visualized in Table 2. In Table 2, invisible tasks (invisibleTask and invisibleTask1) are represented as gray circles and AND relationships are denoted as ANDSplit and ANDJoin.

2.5. Quality Dimension of Merged Models

The quality of merged models is evaluated by correctness and precision. In this paper, the fitness matrix is used to assess correctness. The correctness and precision equations are defined in Eq.(1) and Eq.(2).

Table 2: An example of merging models according to behavioral and structural aspects.

Aspect Types	Variation models	Expected Merged Models
Behavioral	First variant Trace = {(PC → BS → PI)} Second variant Trace = {(PC → CO → PI)}	
Structural	First variant  Second variant 	

$$Correctness_G = \frac{Tr(GVar) \cap Tr(GCon)}{Tr(GVar)} \quad (1)$$

$$Precision_G = \frac{Tr(GVar) \cap Tr(GCon)}{Tr(GCon)} \quad (2)$$

where *Tr(GVar)* is traces of all model variants and *Tr(GCon)* is traces of the consolidated model.

3. Methods

The proposed Graph-merging method integrates both behavioral and structural aspects for consolidating business process models. In this approach, merging relationships are denoted by straight lines, while the resulting constructs are represented by dotted lines (Fig. 1). Behavioral rules prescribe introducing XOR branches when a sequence relationship is merged with an XOR relationship, and the insertion of invisible tasks when a sequence relationship is merged with an AND or OR relationships. Structural rules generate invisible tasks, especially invisible non-prime tasks, when merging branching relationships.

The detailed rules of the proposed Graph-merging method are formulated in Algorithm 1. The inputs consist of the first graph-based model variant (*G*) and the second graph-based model variant (*H*). The first variant contains a set of activities (*A1*) and a set of edges (*r1*), while the second variant has *A2* as a set of activities and *r2* as a set of edges. According to lines 1-3 in Algorithm 1, the approach first checks whether an activity in model *G* (*g*) has the same name as an activity in model *H* (*h*). When common activities are identified, the proposed graph rules are applied to merge the corresponding relationships.

The line 4 collects a set of activities that have relationships from activity *g* (denoted as *g1*), and the line 5 defines a list of activities having relationships from activity *h* (denoted as *h1*). Then, the approach checks the relationship type of *g* and *g1* (denoted as *l1*) and the relationship type of *h* and *h1* (denoted as *l2*). If the relationship types are different (line 7), the proposed rules define how different relationships are merged. The first rule is that when SEQUENCE and XOR relationships are merged, the SEQUENCE relationship between *g* and *g1* is transformed into XOR relationship (lines 8-9). The second rules is deleting the AND relationship between the activity *h* and its

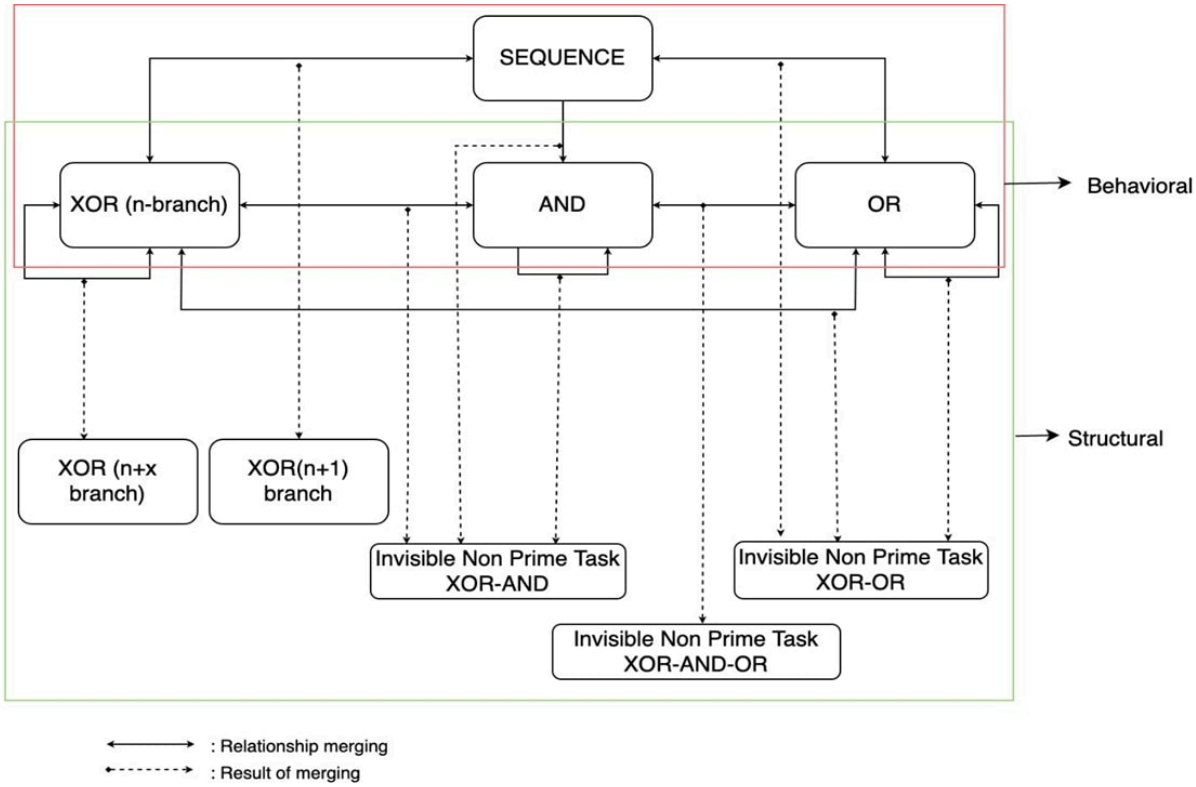


Fig. 1: Hierarchy of dependence in graph-merging method.

successor *h1* (line 11), introducing an invisible task in the activity set *A2* (line 12), establishing the XOR relationship between *h* and the invisible task then the AND relationship between the invisible task and *h1* (lines 13-14), and changing the SEQUENCE relationship from *g* to *g1* into the XOR relationship (line 15) when merging SEQUENCE-AND relationships. The rule of merging SEQUENCE-OR relationships (lines 16-19) follows the same principle as SEQUENCE-AND merging, with the distinction that an SEQUENCE-OR merge creates an OR relationship as the next relationship of the invisible task (line 18). Similarly, the step-by-step procedure for merging XOR-AND relationships follows that of SEQUENCE-AND merging (lines 20-21), while XOR-OR relationships follow the same rules as SEQUENCE-OR merging (line 22-23).Finally, merging AND-OR relationships generates invisible tasks (lines 27-28), creates XOR relationships between *g* and the first invisible task and between *h* and the second invisible task (line 29-30), and produces AND relationship between the first invisible task and *g1* (line 31) also OR relationship between the second invisible task and *h1* (line 32) of Algorithm 1.

The proposed rules described in lines 4-32 of Algorithm 1 address merging relationships involving split gateways. The same rules are applied for join gateways, but with a reversed perspective (line 33-37). For example, when merging an SEQUENCE relationship with a XOR relationship at a XOR-SPLIT gateway, the rule modifies the preceding relationship of the common activity, whereas merging at a XOR-JOIN gateway modifies the succeeding relationship. Finally, the variants are merged (line 38), resulting in a consolidated graph-based process model.

4. Result and Analysis

4.1. Result

The proposed method was evaluated using three case studies: two case studies from e-commerce system and a case study from the solid medical waste handling process [5]. The proposed Graph-merging method is compared with other existing methods, i.e., Derguech [20], Yohanes [18], and Graph Semantic Similarity [11]. The representative variants of the first e-commerce case study are shown in Fig. 2. Both variants share two common activities, namely “Checkout” and “Choose Payment”, which are the basis of applying the proposed merging rules. The first variant has two traces, i.e., {(CheckOut → Choose Payment), (CheckOut → Select Voucher → Choose Payment)} and the second variant only has one trace: {(CheckOut → Fill Information → Choose Payment)}. In the

Algorithm 1: Graph-merging method.

Input: G := the 1st graph-based process model, H := the 2nd graph-based process model
$G = (A1,r1)$, $H = (A2,r2)$
for $g \in A1$ and $h \in A2$ do
  if name(g) = name(h) then
    $g1 = \{a \in A1 \mid (g,a,l1) \in r1\}$
    $h1 = \{b \in A2 \mid (h,b,l2) \in r2\}$
    for $g1$ and $h1$ do
      if $r1$=(g,$g1$,$l1$) and $r2$=(h,$h1$,$l2$) and $l1 \neq l2$ then
        if $l1$ = SEQUENCE and $l2$ = XOR then
          (g,$g1$,SEQUENCE) $\mapsto$ (g,$g1$,XOR)
        else if $l1$ = SEQUENCE and $l2$ = AND then
          $r2 := r2 \setminus \{(h,h1)\}$
          $A2 := A2 \cup \{\text{invisibleTask}\}$
          $r2 := r2 \cup \{(h,\text{invisibleTask},\text{XOR})\}$
          $r2 := r2 \cup \{(invisibleTask,h1,\text{AND})\}$
          (g,$g1$,SEQUENCE) $\mapsto$ (g,$g1$,XOR)
        else if $l1$ = SEQUENCE and $l2$ = OR then
          do step 11 until 13
          $r2 := r2 \cup \{(invisibleTask,h1,\text{OR})\}$
          (g,$g1$,SEQUENCE) $\mapsto$ (g,$g1$,XOR)
        else if $l1$ = XOR and $l2$ = AND then
          do step 11 until 15
        else if $l1$ = XOR and $l2$ = OR then
          do step 17 until 19
        else if $l1$ = AND and $l2$ =OR then
          $r1 := r1 \setminus \{(g,g1)\}$
          $r2 := r2 \setminus \{(h,h1)\}$
          $A1 := A1 \cup \{\text{invisibleTask}\}$
          $A2 := A2 \cup \{\text{invisibleTask1}\}$
          $r1 := r1 \cup \{(g,\text{invisibleTask},\text{XOR})\}$
          $r2 := r2 \cup \{(h,\text{invisibleTask1},\text{XOR})\}$
          $r1 := r1 \cup \{(\text{invisibleTask},g1,\text{AND})\}$
          $r2 := r2 \cup \{(\text{invisibleTask1},h1,\text{OR})\}$
    $g1 = \{a \in A1 \mid (a,g,l1) \in r1\}$
    $h1 = \{b \in A2 \mid (b,h,l2) \in r2\}$
    for $g1$ and $h1$ do
      if $r1 = (g1,g,l1)$ and $r2 = (h1,h,l2)$ and $l1 \neq l2$ then
        do step 8 until step 32 with the reverse relationship
    GH := $G \cup H$
Output: GH := the consolidated graph-based process model

first variant, the activity is associated with an XOR relationship. When SEQUENCE-XOR relationships are merged, the SEQUENCE relationships are replaced by XOR relationships, enabling the integration of the two variants. The consolidated process model by Graph-merging and comparison methods is visualized in Fig. 3 and represents three traces, which are {(CheckOut → Choose Payment), (CheckOut → Select Voucher → Choose Payment), (CheckOut → Fill Information → Choose Payment)}.

In the second case: e-commerce system case, the traces of first variant (Fig. 4(a)) and the second variant (Fig. 4(b)) are {(CheckOut → Fill Information → Select Insurance → Choose Payment), (CheckOut → Select Insurance → Fill Information → Choose Payment)} and {(CheckOut → Choose Payment), (CheckOut → Select Voucher → Choose Payment)}, respectively. Both of them has two traces. Depicting those traces, the first variant has AND relationships, while the second variant has XOR relationship and adds an invisible task. Graph-merging method produces the consolidated model (Fig. 5) which represents four traces: {(CheckOut → Choose Payment), (CheckOut → Select Voucher → Choose Payment),(CheckOut → Fill Information → Select Insurance → Choose Payment), (CheckOut → Select Insurance → Fill Information → Choose Payment)}. Graph Semantic Similarity unifies two variants by using AND relationships (Fig. 6) and its obtained model reflects six traces: {(CheckOut → Select Voucher → Fill Information → Select Insurance → Choose Payment), (Check Out → Select Voucher → Select Insurance → Fill Information → Choose Payment), (Check Out → Fill Information → Select Voucher → Select Insurance → Choose Payment), (Check Out → Fill Information → Select Insurance → Select Voucher → Choose Payment),(Check Out → Select Insurance → Select Voucher → Fill Information → Choose Payment), (Check Out → Select Insurance → Fill Information → Select Voucher → Choose Payment)}. Derguech and Yohanes combine two variants by using OR relationships and Fig. 7 is the result produced by those methods. Because using OR relationships, the consolidated models by Derguech and Yohanes generate sixteen traces respectively. The traces are {(CheckOut → Choose Payment), (CheckOut → Select Voucher → Choose Payment), (CheckOut → Fill Information → Choose Payment), (CheckOut → Select Insurance → Choose Payment), (CheckOut → Select Voucher → Fill Information → Choose Payment), (CheckOut → Select Voucher → Select Insurance → Choose Payment), (CheckOut → Fill Information → Select Voucher → Choose Payment), (CheckOut → Fill Information → Select Insurance → Choose Payment), (CheckOut → Select Insurance → Select Voucher → Choose Payment), (CheckOut → Select Insurance → Fill Information → Choose Payment), (CheckOut → Select Voucher → Fill Information → Select Insurance → Choose Payment), (Check Out → Select Voucher → Select Insurance → Fill Information → Choose Payment), (Check Out → Fill Information → Select Voucher → Select Insurance → Choose Payment), (Check Out → Fill Information → Select Insurance → Select Voucher → Choose Payment),(Check Out → Select Insurance → Select Voucher → Fill Information → Choose Payment), (Check Out → Select Insurance → Fill Information → Select Voucher → Choose Payment)}.

The third case (solid medical waste handling processes) has two variants. The first variant (Fig. 8(a)) forms four traces: {(Sort Waste → Put Pathological Waste → Confirm Waste), (Sort Waste → Put Infectious Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Put Infectious → Confirm Waste), (Sort Waste → Put Infectious → Put Pathological Waste → Confirm Waste)}, while the second variant (Fig. 8(b)) visualizes one trace: {(Sort Waste → Put Cytotoxic Waste → Confirm Waste)}. The proposed method produces a process model containing invisible non-prime tasks for XOR-OR relationships (Fig. 9) to combine those variants. The traces generated from the consolidated model of Graph-merging are {(Sort Waste → Put Cytotoxic Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Confirm Waste), (Sort Waste → Put Infectious Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Put Infectious → Confirm Waste), (Sort Waste → Put Infectious → Put Pathological Waste → Confirm Waste)}. In contrast, the other methods change the relationships into OR relationships (Fig. 10) and produces fifteen traces: {(Sort Waste → Put Cytotoxic Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Confirm Waste), (Sort Waste → Put Infectious Waste → Confirm Waste), (Sort Waste → Put Cytotoxic Waste → Put Pathological Waste → Confirm Waste), (Sort Waste → Put Cytotoxic Waste → Put Infectious Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Put Cytotoxic Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Put Infectious Waste → Confirm Waste), (Sort Waste → Put Infectious Waste → Put Cytotoxic Waste → Confirm Waste), (Sort Waste → Put Infectious Waste → Put Pathological Waste → Confirm Waste), (Sort Waste → Put Cytotoxic Waste → Put Pathological Waste → Put Infectious Waste → Confirm Waste), (Sort Waste → Put Cytotoxic Waste → Put Infectious Waste → Put Pathological Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Put Cytotoxic Waste → Put Infectious Waste → Confirm Waste), (Sort Waste → Put Pathological Waste → Put Infectious Waste → Put Cytotoxic Waste → Confirm Waste), (Sort Waste →

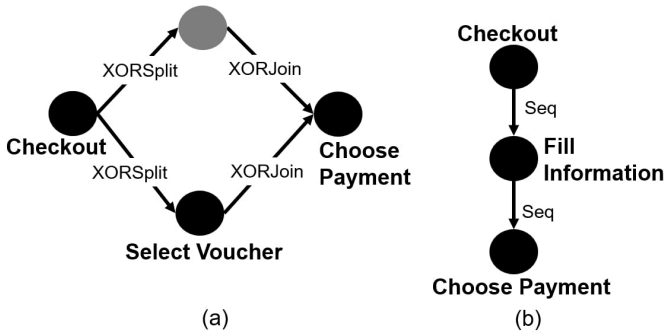


Fig. 2: Snippet model variants of first case: E-Commerce System

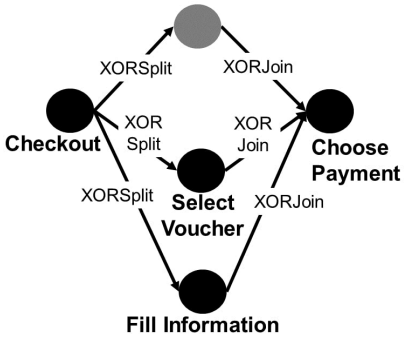


Fig. 3: The merged process model of first case produced by Graph-merging, Derguech, Yohanes and Graph Semantic Similarity methods.

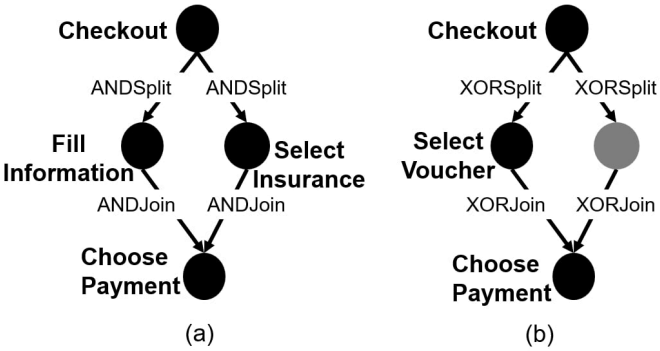


Fig. 4: Snippet model variants of second case: E-Commerce System.

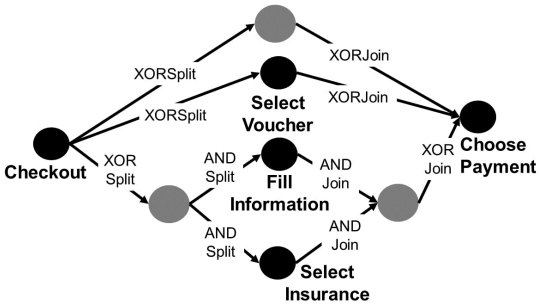


Fig. 5: The merged process model of second case produced by Graph-merging.

Put Infectious Waste → Put Cytotoxic Waste → Put Pathological Waste → Confirm Waste), (Sort Waste → Put Infectious Waste → Put Pathological Waste → Put Cytotoxic Waste → Confirm Waste),}

This research uses correctness ((1)) and precision ((2)) dimensions to assess the quality of the obtained merged methods. For example, based on the third case, the correctness of merged models produced by methods is 1.0 (5/5), because five traces (four traces from the first variant and one trace from the second variant) are successfully visualized in those merged models. The Graph-merging method achieves a precision value of 1.0 (5/5) because the obtained merged model represents five traces, and all traces are the same as those from the first and second

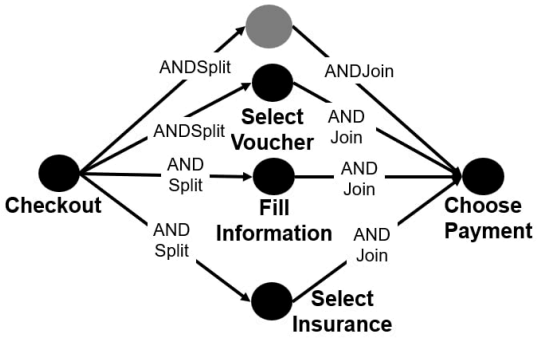


Fig. 6: The merged process model of second case produced by Graph Semantic Similarity.

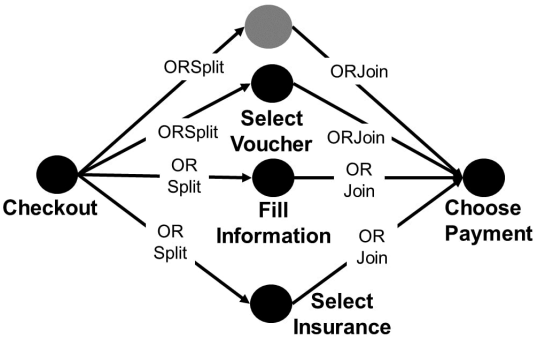


Fig. 7: The merged process model of second case produced by Derguech and Yohanes methods.

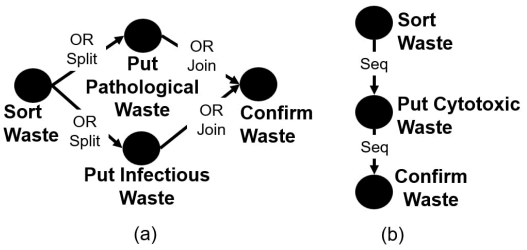


Fig. 8: Snippet model variants of third case: Solid Waste Handling Process.

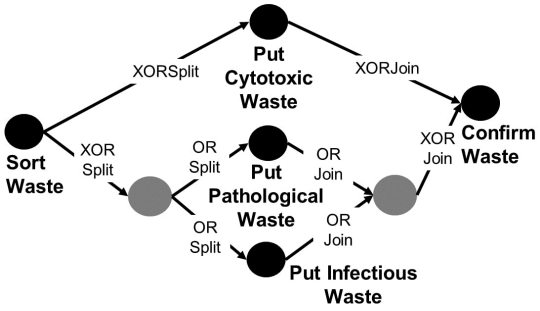


Fig. 9: The merged process model of third case produced by Graph-merging.

variants. However, the precision values of other methods are low because each of those merged models represents fifteen traces, and only five traces out of those traces are the same as the traces of the first and second variants. The precision value of those methods is 0.33 (5/15). The detailed quality measurements of the obtained models are described in Table 3. The mean correctness of obtained models by all methods except Graph Semantic Similarity are the same (according to Table 3); however, the mean precision of merged models obtained by the Graph-merging method is higher than that of the three benchmark methods.

4.2. Analysis

The proposed Graph-merging method provides automatic consolidation of process models containing sequences, branching structures, and invisible tasks by considering both behavioral and structural aspects. Across

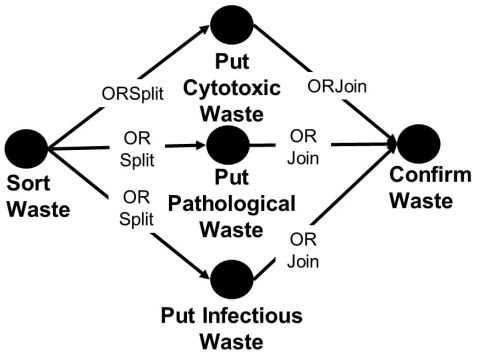


Fig. 10: The merged process model of third case produced by Derguech, Yohanes and Graph Semantic Similarity methods.

Table 3: Quality measurements of obtained models.

Case	Algorithms	Correctness (0.0 - 1.0)	Precision (0.0 - 1.0)
1st Case	Graph-Merging	1.0	1.0
	Graph Semantic Similarity	1.0	1.0
	Derguech	1.0	1.0
	Yohanes	1.0	1.0
2nd Case	Graph Merging	1.0	1.0
	Graph Semantic Similarity	0.0	0.0
	Derguech	1.0	0.25
	Yohanes	1.0	0.25
3rd Case	Graph Merging	1.0	1.0
	Graph Semantic Similarity	1.0	0.33
	Derguech	1.0	0.33
	Yohanes	1.0	0.33
$\bar{x} \pm \sigma$	Graph Merging	1.0 ± 0.0	1.0 ± 0.0
	Graph Semantic Similarity	0.67 ± 0.47	0.44 ± 0.42
	Derguech	1.0 ± 0.0	0.53 ± 0.34
	Yohanes	1.0 ± 0.0	0.53 ± 0.34

all case studies, the Graph-merging method consistently produced merged models that represented all traces of variants. In contrast, the comparison approaches, i.e., Derguech, Yohanes, and Graph Semantic Similarity, yielded models consisting of different traces from those of the variants. In the 2nd case of Table 3, Graph Semantic Similarity achieved a 0.0 correctness value because the obtained merging model cannot depict variant traces. As shown in the 3rd case of Table 3, those three methods achieved a correctness value of 1.0, because their consolidated process models contain all variant traces.

When all methods are evaluated using the precision dimension, the precision values of the models by the Graph Semantic Similarity, Derguech, and Yohanes methods dropped significantly, as the merged models introduce extra traces that are not present in the original variants. The ability of Graph-based merging to generate the invisible tasks when merging XOR-AND and XOR-OR relationships improves the precision value of consolidated process models.

According to precision values, this research applies an one-way ANOVA to compare the Graph-merging method with other methods. The analysis denotes the difference is not statistically significant at the 0.05 level ($F(3,8) = 1.289$, $p = 0.34$). However, the Graph-merging method obtains high precision values (1.0) in all case studies, while the comparison methods, i.e., Derguech and Yohanes, receive an average precision of 0.53 ± 0.34 and

Graph Semantic Similarity has 0.44 ± 0.42. The effect size is large ($\eta^2 = 0.326$), indicating a substantial practical impact, despite the limited statistical power due to the small sample size.

5. Conclusion

Manual incorporation of process models risks model inaccuracies and takes a long time. The existing merging method has not succeeded in combining process models considering invisible tasks, thereby reducing the quality of the combined process model. This study developed a proposed graph method called the Graph-merging method to combine process models utilizing behavioral and structural aspects.

This study tested the merging graph by measuring the correctness and precision of the results of the merging model using the Graph-merging method compared to the model results from the Graph Semantic Similarity, Derguech and Yohanes methods. The evaluation results demonstrate that the precision value of the model formed by the Graph-merging method (1.0) was higher than that of Graph Semantic Similarity (0.44) and those of the Derguech and Yohanes methods, which both have a value of 0.53. Although the one-way ANOVA declares the absence of statistical significance in precision value ($F(3,8) = 1.289$, $p = 0.34$), the effect size ($\eta^2 = 0.326$) indicates a significant practical impact. A more than 80% improvement in precision compared to the benchmark algorithms proves that the Graph-merging method improves the quality of the combined process model.

The Graph-merging method has not been able to combine process models with different activity names with the same meaning and accommodate looping relationships, so that in the future, it can be developed by combining semantic similarity and looping relationships.

6. CRediT authorship contribution statement

K. R. Sungkono: Conceptualization, Methodology, Validation, Formal analysis, Resources, Data Curation, Writing – Original Draft. **R. Sarno:** Resources, Writing – Review & Editing, Supervision. **I. G. A. C. P. Dewi:** Software, Data Curation, Writing – Original Draft. **M. S. Hitam:** Writing – Review & Editing, Supervision.

7. Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

8. Acknowledgment

The authors gratefully acknowledge support from the Institut Teknologi Sepuluh Nopember for this work, under project scheme of the Publication Writing and IPR Incentive Program (PPHKI).

9. Data availability

The data used to support the findings of this study are available from the corresponding author upon request

10. Declaration of generative AI and AI-assisted technologies in the writing process

The authors utilized generative AI to improve writing clarity of this paper. Authors reviewed and edited AI-assisted content. They also bear full responsibility for the final publication.

References

[1] C. Castiglione, “Automated generation of digital models for manufacturing systems: The event-centric process mining approach,” *Computers & Industrial Engineering*, vol. 197, p. 110596, 2024, doi: https://doi.org/10.1016/j.cie.2024.110596.

[2] T. R. de Boer, R. J. Arntzen, R. Bekker, B. M. Buurman, H. C. Willems, and R. D. van der Mei, “Process Mining on National Health Care Data for the Discovery of Patient Journeys of Older Adults,” *Journal of the American Medical Directors Association*, vol. 26, no. 1, p. 105333, 2025, doi: https://doi.org/10.1016/j.jamda.2024.105333.

[3] F. Corradini, A. Marcelletti, A. Morichetta, B. Re, and L. Verducci, “A Mapping Methodology Enabling Object-Centric Process Mining on Blockchain Applications,” *Blockchain: Research and Applications*, p. 100392, 2025, doi: https://doi.org/10.1016/j.bcra.2025.100392.

[4] K. R. Sungkono, R. Sarno, F. D. Amadea, and A. F. Irbah, “Graph-Based Process Model Discovery for Mining Invisible Non-Prime Tasks in Hybrid Choice-Parallel Relationships,” *International Journal of Intelligent Engineering & Systems*, vol. 14, no. 3, pp. 425–434, 2021, doi: 10.22266/ijies2021.0630.35.

[5] K. R. Sungkono, R. Sarno, B. S. Onggo, and M. F. Haykal, "Enhancing model quality and scalability for mining business processes with invisible tasks in non-free choice," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 9, p. 101741, 2023, doi: 10.1016/j.jksuci.2023.101741.

[6] I. Waspada, R. Sarno, E. S. Astuti, H. N. Prasetyo, and R. Budiraharjo, "Extending Graph-Based Process Discovery for Hierarchical Block Detection and Incomplete Concurrent Relationship Handling," *IEEE Access*, vol. 11, no. , pp. 123382–123391, 2023, doi: 10.1109/ACCESS.2023.3324742.

[7] M. Boubakir and A. Chaoui, "An Approach and A Tool for Merging A Set of Models in Pairwise Way," *Malaysian Journal of Computer Science*, vol. 34, no. 1, pp. 13–33, Jan. 2021, doi: 10.22452/mjcs.vol34no1.2.

[8] A. Bottrighi, M. Guazzone, G. Leonardi, S. Montani, M. Striani, and P. Terenziani, "Integrating ISA and Part-of Domain Knowledge into Process Model Discovery," *Future Internet*, vol. 14, no. 12, 2022, doi: 10.3390/fi14120357.

[9] M. Sharbaf, B. Zamani, and G. Sunyé, "Automatic resolution of model merging conflicts using quality-based reinforcement learning," *Journal of Computer Languages*, vol. 71, no. December2021, p. 101123, 2022, doi: 10.1016/j.cola.2022.101123.

[10] M. Sharbaf, B. Zamani, and G. Sunyé, *Conflict management techniques for model merging: a systematic mapping review*, vol. 22, no. 3. Springer Berlin Heidelberg, 2023, pp. 1031–1079. doi: 10.1007/s10270-022-01050-9.

[11] K. R. Sungkono, R. Sarno, M. C. Salsabila, and C. P. Dewi, "A Graph-based Method for Merging Business Process Models by Considering Semantic Similarity," *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 2, pp. 166–175, 2023, doi: 10.22266/ijies2023.0430.14.

[12] B. Di Martino, L. Colucci Cante, A. Esposito, and M. Graziano, "A tool for the semantic annotation, validation and optimization of business process models," *Software: Practice and Experience*, vol. 53, no. 5, pp. 1174–1195, May 2023, doi: 10.1002/spe.3184.

[13] M. La Rosa, M. Dumas, R. Uba, and R. Dijkman, "Business Process Model Merging: An Approach to Business Process Consolidation," *ACM Transactions on Software Engineering and Methodology*, vol. 22, no. 2, Mar. 2013, doi: 10.1145/2430545.2430547.

[14] N. L. Ha and T. M. Prinz, "Partitioning behavioral retrieval: An efficient computational approach with transitive rules," *IEEE Access*, vol. 9, pp. 112043–112056, 2021, doi: 10.1109/ACCESS.2021.3102634.

[15] M. de Leoni, W. M. van der Aalst, and M. Dees, "Nine years later: Reflecting on our article: A general process mining framework for correlating, predicting, and clustering dynamic behavior based on event logs," *Information Systems*, vol. 137, p. 102644, 2026, doi: https://doi.org/10.1016/j.is.2025.102644.

[16] K. S. Kim, D. L. Pham, Y. I. Park, and K. P. Kim, "Experimental verification and validation of the SICN-oriented process mining algorithm and system," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 10, pp. 9793–9813, 2022, doi: 10.1016/j.jksuci.2021.12.013.

[17] C. Zhou, D. Zhang, D. Chen, and C. Liu, "A Change-Sensitive Complexity Measurement for Business Process Models Based on Control Structure," *Systems*, vol. 11, no. 250, 2023, doi: 10.3390/systems11050250.

[18] Y. Setiawan, K. R. Sungkono, and R. Sarno, "A New Similarity Method based on Weighted Graph Models for Matching Parallel Business Process Models," *International Journal of Intelligent Engineering & Systems*, vol. 13, no. 5, pp. 267–276, 2020, doi: 10.22266/ijies2020.1031.24.

[19] J. Guo and H. Fang, "Behavior Differentiation of Process Variants With Invisible Tasks," *IEEE Access*, vol. 11, pp. 64815–64830, 2023, doi: 10.1109/ACCESS.2023.3289876.

[20] W. Derguech, S. Bhiri, and E. Curry, "Designing business capability-aware configurable process models," *Information Systems*, vol. 72, pp. 77–94, Dec. 2017, doi: 10.1016/j.is.2017.10.001.

[21] M. A. Zemni, A. Mammar, and N. B. Hadj-Alouane, "An automated approach for merging business process fragments," *Computers in Industry*, vol. 82, pp. 104–118, Oct. 2016, doi: 10.1016/j.compind.2016.05.002.

[22] G. Hübscher *et al.*, "Graph-based managing and mining of processes and data in the domain of intellectual property," *Information Systems*, vol. 106, p. 101844, 2022, doi: https://doi.org/10.1016/j.is.2021.101844.

[23] F. Drewes and Y. Stade, "On the power of local graph expansion grammars with and without additional restrictions," *Theoretical Computer Science*, vol. 1015, p. 114763, Nov. 2024, doi: 10.1016/j.tcs.2024.114763.

[24] M. Kobeissi, N. Assy, W. Gaaloul, B. Defude, B. Benatallah, and B. Haidar, "Natural language querying of process execution data," *Information Systems*, vol. 116, p. 102227, 2023, doi: 10.1016/j.is.2023.102227.

[25] H. Fang, W. Liu, W. Wang, and S. Zhang, "Discovery of process variants based on trace context tree," *Connection Science*, vol. 35, no. 1, p. 2190499, 2023, doi: 10.1080/09540091.2023.2194578.

[26] A. Kalenkova, A. Polyvyanyy, and M. L. R. Rosa, "Structural and Behavioral Biases in Process Comparison Using Models and Logs," in *Conceptual Modeling*, Cham: Springer International Publishing, 2021, pp. 62–73. doi: 10.1007/978-3-030-89022-3_6.

[27] A. Khannat, H. Sbai, and L. Kjiri, "Extended Inductive Miner to Discover Semantic Annotated Process Model: A Variability-Based Approach.," *Journal of Information Science Theory & Practice (JIStaP)*, vol. 12, no. 4, 2024.

[28] M. T. Garcia, M. M. Nunes, M. Fantinato, S. M. Peres, and L. H. Thom, "BPMN-Sim: A multilevel structural similarity technique for BPMN process models," *Information Systems*, vol. 116, p. 102211, 2023, doi: https://doi.org/10.1016/j.is.2023.102211.

[29] J. Zhou and G. Reniers, "Developing a Petri-net approach for emergency response modeling and time analysis of process accidents considering the execution characteristics of emergency tasks," *Journal of Loss Prevention in the Process Industries*, vol. 98, p. 105753, 2025, doi: https://doi.org/10.1016/j.jlp.2025.105753.

[30] A. Augusto, J. Mendling, M. Vidgof, and B. Wurm, "The connection between process complexity of event sequences and models discovered by process mining," *Information Sciences*, vol. 598, pp. 196–215, 2022, doi: 10.1016/j.ins.2022.03.072.

[31] L. N. Safitri, R. Sarno, and K. R. Sungkono, "LTL similarity and classification using fuzzy rules for evaluating environment sustainability business process indicator," *2019 International Conference on Information and Communications Technology, ICOIACT 2019*, pp. 348–353, 2019, doi: 10.1109/ICOIACT46704.2019.8938430.