Checking Wrong Pattern in Process Model Containing Invisible Task by Using Declarative Miner

Nadia P. Salsabila
Department of Mathematics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
mithasalsabila@gmail.com

Revi A. Palembiya
Department of Mathematics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
reviasprila@gmail.com

Kelly R. Sungkono
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
kelly@its.ac.id

Riyanarto Sarno
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
riyanarto@if.its.ac.id

***Abstract*—Event log records all events of performed business process on the system. Analysts used the event log to detect occurred anomalies, one of which is wrong pattern, in the process. However, there are conditions, i.e. skip condition, redo condition, and switch condition, which can be misinterpreted as wrong pattern. Uniquely, those conditions cannot be depicted in the reference model without utilizing additional tasks, namely invisible tasks. This research proposes rules which can check the wrong pattern in the process model containing those conditions. This research automatically formed declarative miner rules carrying invisible tasks based on a process model. The form of the used process model in this research is a graph model. Then, the rules are used to checking the wrong pattern. The experiment uses real data, i.e. port-container handling processes, and several simulation data. The analysis explains that declarative miners have 100% accuracy to check the wrong pattern in the process model that contains each invisible task including skip condition, redo condition, and switch condition.**

Keywords—declarative miner; event log; invisible task; wrong pattern

I. INTRODUCTION

Process mining is a field of solving business processes challenges based on the documentation of the process [1]. In business processes, one of the results by process mining is a process model [2]. Process mining aims consist of automatic discovery, conformity checking, and model improvement [3]. Automatic discovery, which is known as Process discovery, identifies business processes by considering the event log [4]. The event log is a file that records all the events on the system, such as the name, the executor, and the time of a task. Conformity checking, known as conformance checking, finds problems by analyzing the event log, set in a process model. This research focuses on finding the wrong pattern of processes. A wrong pattern is a problem that executes processes that do not follow relationships on a model process Standard Operational Procedure (SOP).

In business processes, the use of real activities not always depict all conditions of processes. Those indescribable conditions are skip, redo, and switch. Skip condition occurs when several activities in the middle of processes are permitted to abandon. Some tasks which enforce often are redo conditions. A situation is rated as a switch when a task is fulfilled after choice activities, and one of the choice activities has further activity options besides the task. These conditions are misconstrued as the wrong pattern if the reference model cannot form them. The attendance of invisible tasks avoids misconstrued.

This research proposes wrong pattern checking rules in the process model that contains invisible tasks with LTL automatically. Some papers have discussed checking anomaly patterns, but no one has handled anomaly patterns in the form of wrong patterns that contain invisible tasks [5]–[9]. Besides, research has made rules automatically, but only to handle non-free choice [10]. This research combines graph databases and declarative miners. Graph-database models a reference model, and the declarative miner builds the rules based on the reference model. The graph database applied in Neo4j. Neo4j was employed along with Cypher Query Language (CQL) [11]. The graph reference model is transformed into declarative miner rules [12] to check the wrong pattern in the process model that contains each invisible task. The offered declarative miner rules are tested by conducting two experiments for each type of indescribable condition. Tests in the field of accuracy ensure that declarative miner rules can be implemented.

Can be clarified, the contribution in this research are:

1. propose declarative miner rules to check for the wrong patterns in business processes that contain invisible tasks;

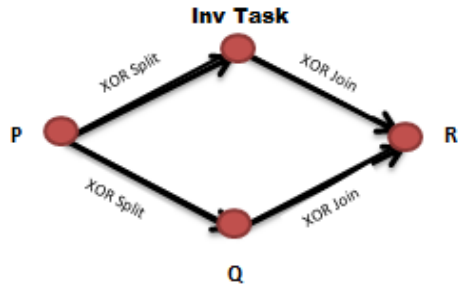
2. translate the process model in the graph model to the rules of declarative miners automatically.

II. Related Work

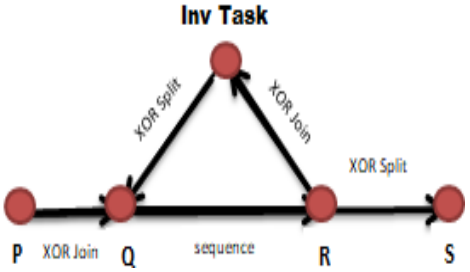
A. Invisible Task

Invisible tasks are activities that appear in the business process but are not stored in the event log. There are three types of invisible tasks [13]. Activities that have sequential relationships which are not carried out in separate processes can produce skip conditions. Fig. 1 (a) shows the process model [PQR] that has a sequential relationship. When the state of activity P goes directly to activity R, an invisible task appears. That condition is called a skip condition.

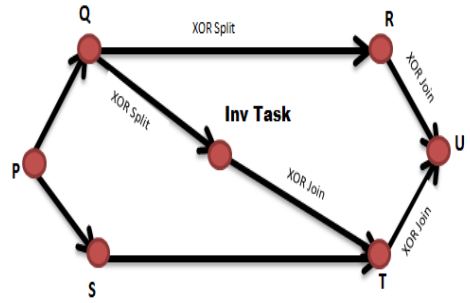
Two or more activities carried out repeatedly can produce redo conditions. Fig. 1 (b) shows the process model [PQRQRS] so that when activities Q and R are run frequently, an invisible task of redo condition appears. Activities that are not known for their original direction and the direction of their destination will produce a one-time state. Fig. 1 (c) shows the process model [PQRU, PQTU]. It looks that activities Q and R are in a switch condition so that an invisible task of switch condition appears.



(a) Skip



(b) Redo



(c) Switch

Fig. 1. Types of Invisible Tasks

B. Declarative Miner

Declarative miner is a rule that is formed from several Linear Temporal Logic (LTL). LTL is made by looking at the time [14]. Right and wrong constants, sets of atomic propositions, and temporal operators are the foundation on which LTL is built [10]. LTL is used to describe the problems that occur in the activity and ensure the system is safe [15]. Implication (→), conjunction (∧), disjunction (∨), and negation (¬) are logical temporal [3].

C. Research of Declarative Miner Rules

In paper [10], declarative algorithm rules are created automatically using LTL to detect wrong indirect relationships. However, this rule only covers process models that contain non-free choices. The non-free choice depends on activities' relation, where the chosen activity is carried out based on previous activities.

Meanwhile, in this research, the declarative miner rules made cover a broader model process. The indirect relationship is carried out by the invisible task, which becomes a solution for the activity to be carried out to not depend on the previous event. Besides, the existence of a pattern applied in this research enables problems in the process of the model to be quickly identified.

D. Accuracy

This research utilizes accuracy as the measurement of the results. Equation (1) shows the accuracy.

$$Acy = \frac{RP + RN}{RP + RN + WP + WN} x100\% \quad (1)$$

where:

RP : the right process that can be detected by the proposed rules

RN : the wrong-pattern process that can be detected by the proposed rules

WP : the right process that cannot be detected by the proposed rules

WN : the wrong-pattern process that cannot be detected by the proposed rules

III. Method

The event log in this research is automatically saved as a graph database using Neo4j and in the CSV file. The event log that is used is pre-processed so it does not have a process that contains anomalies. From the graph database in CSV file, new rules that are declarative miners automatically can be made for each type of invisible task based on the process model illustrated in the graph that contains event logs.

A. *Invisible Task: Skip Condition*

The event log of skip condition is automatically saved as a graph database using Neo4j and in the CSV file shown in table I. All activities in graph-database are described as a model with sequence relationships that are already known in Neo4j. Seen in Table I, there are activities {A, B, C}, and process models {A, B, C, A, C}. This process model contains the invisible task that is activity A which has a SPLIT relationship and a JOIN relationship between other activities.

Then, from the graph database in CSV file, a new rule of declarative miner created which is shown in Fig. 2. Table II shows the pseudocode containing the declarative miner results from the invisible task skip conditions.

TABLE I. GRAPH DATABASE: SKIP CONDITION

Act_Before	Relation	Act_After
A	XOR Split	B
A	XOR Split	Inv_Task
B	XOR Join	C
Inv_Task	XOR Join	C

TABLE II. PSEUDOCODE FOR CHECKING WRONG PATTERN IN INVISIBLE TASK: SKIP CONDITION

Pseudocode
Input : CSV file of graph database **Output:** Declarative miner rule
Process: For x as numberRowInTable: If Act_Before[x] = Inv_Task and relation(Act_Before[x], Act_After[x]) = XOR Join : For y as numberRowInTable: If Act_After[y] = Inv_Task and relation(Act_Before[y], Act_After[y]) = XOR Split and relation(Act_After[x], Act_Before[y]) = Sequence: A = Act_Before[y], B = Act_After of Act_Before[y] that is not Inv_Task, C = Act_After[x] do Rule Skip

Where :

Act_Before[x] : activity [x] as a beginning activity before relation
Act_After[x] : activity [x] as an end activity after relation
Act_Before[y] : activity [y] as a beginning activity before relation
Act_After[y] : activity [y] as an end activity after relation

(activity == A /_0(((activity == B /_0(activity == C)) \/ activity == C))) ;

Fig. 2. Declarative Miner of Invisible Task: Skip Condition

TABLE III. GRAPH DATABASE: REDO CONDITION

Act_Before	Relation	Act_After
A	XOR Join	B
B	Sequence	C
C	XOR Split	Inv_Task
Inv_Task	OR Join	B
C	XOR Split	D

TABLE IV. PSEUDOCODE FOR CHECKING WRONG PATTERN IN INVISIBLE TASK: REDO CONDITION

Pseudocode
Input : CSV file of graph database **Output:** Declarative miner rule
Process: For x as numberRowInTable: If Act_Before[x] = Inv_Task and relation(Act_Before[x], Act_After[x]) = XOR Join : For y as numberRowInTable: If Act_After[y] = Inv_Task and relation(Act_Before[y],Act_After[y]) = XOR Split and relation(Act_After[x], Act_Before[y]) = Sequence: A = Act_Before of Act_After[x] that is not Inv_Task B = Act_After[x] C = Act_Before[y] D = Act_After of Act_Before[y] that is not Inv_Task do Rule Redo

Where :

Act_Before[x] : activity [x] as a beginning activity before relation
Act_After[x] : activity [x] as an end activity after relation
Act_Before[y] : activity [y] as a beginning activity before relation
Act_After[y] : activity [y] as an end activity after relation

(activity == A /\<>((activity == B /_0((activity == C /_0(activity == D)))))) ;

Fig. 3. Declarative Miner of Invisible Task: Redo Condition

B. *Invisible Task: Redo Condition*

The event log of redo condition is automatically saved as a graph database using Neo4j and in the CSV file shown in table III. All activities in graph-database are described as a model with sequence relationships that are already known in Neo4j. Seen in Table III, there are activities {A, B, C, D}, and process models { A, B, C, D, A, B, C, Q, B, C, D}. This process model contains the invisible task that is activity C which has a SPLIT relationship and a JOIN relationship between other activities. Then, from the graph database in CSV file, a new rule of declarative miner created which is shown in Fig. 3. Table IV shows the pseudocode containing the declarative miner results from the invisible task redo conditions.

TABLE V. GRAPH DATABASE: SWITCH CONDITION

Act_Before	Relation	Act_After
A	XOR Split	B
B	XOR Split	C
B	XOR Split	Inv_Task
Inv_Task	XOR Join	E
C	XOR Join	F
E	XOR Join	F

TABLE VI. PSEUDOCODE FOR CHECKING WRONG PATTERN IN INVISIBLE TASK: SWITCH CONDITION

Pseudocode
Input : CSV file of graph database **Output:** Declarative miner rule
Process: For x as numberRowInTable: If Act_Before[x] = Inv_Task and relation(Act_Before[x], Act_After[x]) = XOR Join : For y as numberRowInTable: If Act_After[y] = Inv_Task and relation(Act_Before[y],Act_After[y]) = XOR Split and relation(Act_After[x], Act_Before[y]) = Sequence: A = Act_Before of Act_After[y] that is not Inv_Task B = Act_Before[y] C = Act_Before of Act_After[x] that is not Inv_Task E = Act_After[x] F = Act_After of Act_Before[x] that is not Inv_Task do Rule Switch

Where :

Act_Before[x] : activity [x] as a beginning activity before relation
Act_After[x] : activity [x] as an end activity after relation
Act_Before[y] : activity [y] as a beginning activity before relation
Act_After[y] : activity [y] as an end activity after relation

(activity == A /_0((activity == B /_0(((activity == C /_0(activity == F)) \/ (activity == E /_0(activity == F))))))) ;

Fig. 4. Declarative Miner of Invisible Task: Switch Condition

TABLE VII. EXAMPLE OF EXPERIMENT EVENT LOG: SKIP CONDITION

Case_ID	Activity	Date	Time
PP1	A	3/8/2016	10:32
PP2	B	3/8/2016	10:47
PP2	C	3/8/2016	10:51

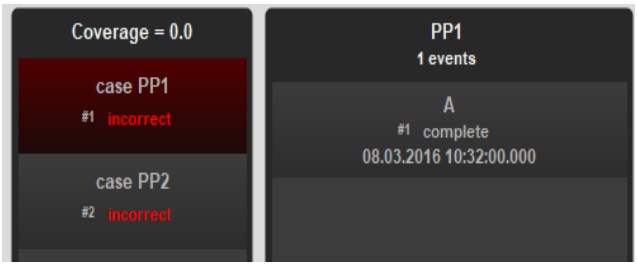


Fig. 5. Result of Wrong Pattern Checking in Processes with Invisible Task: Skip Condition

C. *Invisible Task: Switch Condition*

The event log of switch condition is automatically saved as a graph database using Neo4j and in the CSV file shown in Table V. All activities in graph-database are described as a model with sequence relationships that are already known in Neo4j. Seen in Table VII there are activities {A, B, C, E, F} and process models {A, B, C, F, A, B, E, F }. This process model contains the invisible task that is activity C which has a SPLIT relationship and a JOIN relationship between other activities. Then, from the graph database in CSV file, a new rule of declarative miner created which is shown in Fig. 4. Table VI shows the pseudocode containing the declarative miner results from the invisible task switch conditions.

IV. RESULT

This evaluation uses simulation event logs that contain three types of conditions of invisible tasks and a real event log from the Port Container Handling case. The rules are created based on a process model of a pre-processed event log, so the process model describes the correct processes (no anomalies). This research creates two evaluations of each condition, except for the Port Container Handling case.

A. *Invisible Task: Skip Condition*

For verifying the proposed declarative miner rules of skip condition, this research creates two simulation log. The first contains 20 cases that depict 2 traces. The example of cases that illustrate the traces are presented in Table VII. Case PP1 has activity {A}, and case PP2 has activities {B, C}. Both the simulation log traces are the wrong pattern, so the proposed rules should identify all cases as incorrect. The example of output by the proposed rules is shown in Fig. 5. This research also creates a simulation log with 20 cases that are the right processes. The accuracy of the results based on all experiments is shown in Table XI.

B. *Invisible Task Redo Condition*

A simulation log for examining redo condition consists of 20 cases and 2 traces. Table VIII shows all traces by the example cases of the simulation event log. Based on Table VIII, traces of the log area {A, B, D} and {A, B, C, B, C, D}. Case PP1 is an example of a trace with the wrong pattern, and Case PP2 is an example of a right trace. The output by the proposed rules is shown in Fig. 6. This research also applies the rules on another simulation log, where the accuracy result of the experiment is shown in Table XI.

TABLE VIII. EXPERIMENT EVENT LOG: REDO CONDITION

Case_ID	Activity	Date	Time
PP1	A	3/8/2016	10:32
PP1	B	3/8/2016	11:33
PP1	D	3/8/2016	13:28
PP2	A	3/8/2016	16:56
PP2	B	3/8/2016	17:32
PP2	C	3/8/2016	18:15
PP2	B	3/8/2016	20:11
PP2	C	3/8/2016	20:42
PP2	D	3/8/2016	20:50

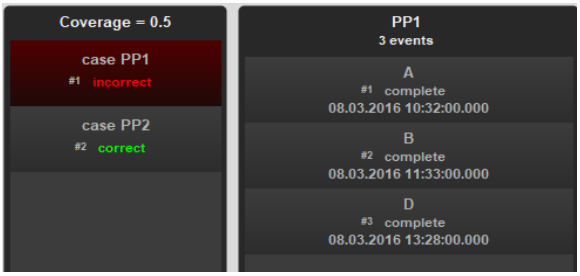


Fig. 6. Result of Wrong Pattern Checking in Processes with Invisible Task: Redo Condition

C. *Invisible Task Switch Condition*

TABLE IX. EXPERIMENT EVENT LOG: SWITCH CONDITION

Case_ID	Activity	Date	Time
PP1	A	3/8/2016	10:32
PP1	C	3/8/2016	10:38
PP1	F	3/8/2016	10:44
PP2	A	3/8/2016	10:47
PP2	B	3/8/2016	10:50
PP2	E	3/8/2016	10:53
PP2	F	3/8/2016	10:56

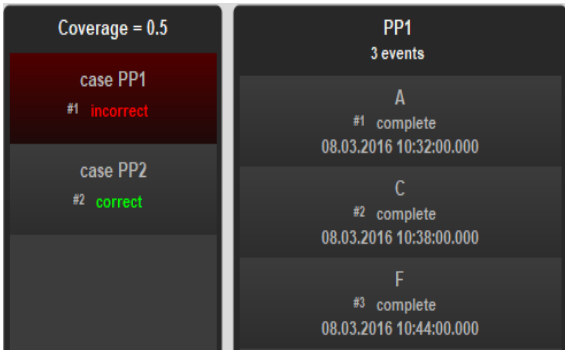


Fig. 7. Result of Wrong Pattern Checking in Processes with Invisible Task: Switch Condition

This research creates two simulation log that contains the wrong pattern in processes containing invisible task switch condition. Table IX shows the example of cases that occur on the first simulation log containing 20 cases and 2 traces. Case PP1 is an example of a wrong-pattern process, and Case PP2 is an example of the right process. The example of output by the proposed rules is shown in Fig. 7. The accuracy of results based on all simulation log is described in Table XI.

D. *Port Container Handling*

This experiment was carried out using a Port container handling process. Graph database of port container handling is applied in the table contains event logs that match the pattern. The event log must provide case_id as a process instance, activities name, date of process instance, and timestamp. Table X describes the part of an event log used in this experiment, where PP1 until PP3 are the right processes, and the others are wrong-pattern processes. Event logs in table X are imported into ProM to be converted into an XES file. An XES file will be run with a declarative miner of Port Container Handling that has been created. All experiments indicate the accuracy of the declarative miner for each condition of the invisible task so that the accuracy can be obtained, shown in Table XI.

TABLE X. PORT CONTAINER HANDLING EVENT LOG

Case_ ID	Activity	Date	Time
PP1	Specify_Container_Type	3/8/2016	10:05
PP1	Specify_Dry	3/8/2016	10:35
PP1	Make_Decisions_Before_Transport_ The_Container	3/8/2016	10:38
PP1	Transport_The_Container_Truck	3/8/2016	10:41
PP1	The_Truck_Left	3/8/2016	10:44
PP2	Specify_Container_Type	3/8/2016	11:05
PP2	Specify_Reefer	3/8/2016	11:35
PP2	Make_Decisions_Before_Transport_ The_Container	3/8/2016	11:38
PP2	Pull_Out_Reefer_Cable	3/8/2016	11:41
PP2	Transport_The_Container_Truck	3/8/2016	11:44
PP2	The_Truck_Left	3/8/2016	11:45
PP3	Specify_Container_Type	3/8/2016	12:05
PP3	Specify_Uncontainer	3/8/2016	12:20
PP3	Make_Decisions_Before_Transport_ The_Container	3/8/2016	12:35 12:38
PP3	Set_Up_Tools	3/8/2016	12:41
PP3	Transport_The_Container_Truck	3/8/2016	12:44
PP3	The_Truck_Left	3/8/2016	12:45
PP4	Transport_The_Container_Truck	3/8/2016	19:23
PP4	The_Truck_Left	3/8/2016	19:40
PP5	Specify_Container_Type	3/9/2016	08:00
PP5	Transport_The_Container_Truck	3/9/2016	08:20
PP5	The_Truck_Left	3/9/2016	08:40
PP6	Specify_Container_Type	3/9/2016	09:40
PP6	Specify_Dry	3/9/2016	10:03
PP6	Set_Up_Tools	3/9/2016	10:16
PP6	Transport_The_Container_Truck	3/9/2016	10:50
PP6	The_Truck_Left	3/9/2016	12:00

TABLE XI. ACCURACY OF RESULTS

Dataset	Description Experiments	Accuracy
Simulation Event Log – Invisible Skip (Experiment 1)	This experiment contains 0 right processes and 20 wrong-pattern processes.	$\frac{(0+20)}{(0+20+0+0)} x100\% = 100\%$
Simulation Event Log – Invisible Skip (Experiment 2)	This experiment contains 20 right processes and 0 wrong-pattern processes.	$\frac{(20+0)}{(20+0+0+0)} x100\% = 100\%$
Simulation Event Log – Invisible Redo (Experiment 1)	This experiment contains 10 right processes and 10 wrong-pattern processes	$\frac{(10+10)}{(10+10+0+0)} x100\% = 100\%$
Simulation Event Log – Invisible Redo (Experiment 2)	This experiment contains 15 right processes and 5 wrong-pattern processes	$\frac{(15+5)}{(15+5+0+0)} x100\% = 100\%$
Simulation Event Log – Invisible Switc (Experiment 1)	This experiment contains 10 right processes and 10 wrong-pattern processes	$\frac{(10+10)}{(10+10+0+0)} x100\% = 100\%$
Simulation Event Log – Invisible Switc (Experiment 2)	This experiment contains 15 right processes and 5 wrong-pattern processes	$\frac{(15+5)}{(15+5+0+0)} x100\% = 100\%$
Port Container Handling Processes	This experiment contains 27 right processes and 6 wrong-pattern processes.	$\frac{(27+6)}{(27+6+0+0)} x100\% = 100\%$

Based on Table XI, all evaluations obtain 100% because the process model as the reference of rules is obtained based on the right processes (no anomalies), and the rules can describe all relationships of that process model. The accuracy will be reduced if the event log used as a rule reference contains wrong-pattern processes.

V. CONCLUSION

Patterns can accelerate the process of designing a solution and reduce modeling time. However, the pattern used might be wrong. A wrong pattern in-process model that contains invisible tasks can occur in skip condition, redo condition, or switch condition.

Therefore, this research proposes rules which can check the wrong pattern. Moreover, this research also translates the process model in the graph model to the rules of declarative miners automatically. Firstly, the Event log is saved as a graph database using Neo4j and in the CSV file. Then, the declarative miners' rules can be made automatically and used to checking the wrong pattern in the process model containing invisible tasks.

From the experimental results, we can conclude that the declarative miners have 100% accuracy. It is because the process model as the reference of rules is obtained based on the right processes (no anomalies), and the rules can describe all relationships of that process model. The accuracy will be reduced if the event log that is used as a rule reference contains wrong-pattern processes.

ACKNOWLEDGMENT

This research was funded by the Indonesian Ministry of Education and Culture, and the Indonesian Ministry of Research and Technology/National Agency for Research and Innovation, under Penelitian Dana Departemen managed by Institut Teknologi Sepuluh Nopember (ITS) research grant (Contract No. 1664/PKS/ITS/2020).

REFERENCES

[1] T. Becker and W. Intoyoad, "Context Aware Process Mining in Logistics," *Procedia CIRP*, vol. 63, pp. 557–562, 2017, doi: 10.1016/j.procir.2017.03.149.

[2] I. Zakarija, F. Škopljanac-Mačina, and B. Blašković, "Automated simulation and verification of process models discovered by process mining," *Automatika*, vol. 61, no. 2, pp. 312–324, Apr. 2020, doi: 10.1080/00051144.2020.1734716.

[3] F. M. Maggi, R. P. J. C. Bose, and W. M. P. van der Aalst, "Efficient Discovery of Understandable Declarative Process Models from Event Logs," *Advanced Information Systems Engineering. CAiSE 2012*, vol. 7328, pp. 270–285, 2012, doi: 10.1007/978-3-642-31095-9_18.

[4] R. Sarno and K. R. Sungkono, "Coupled Hidden Markov Model for Process Discovery of Non-Free Choice and Invisible Prime Tasks," *Procedia Computer Science*, vol. 124, pp. 134–141, 2017, doi: 10.1016/j.procs.2017.12.139.

[5] S. Pauwels and T. Calders, "An anomaly detection technique for business processes based on extended dynamic bayesian networks," *The 34th ACM/SIGAPP Symposium*, pp. 494–501, Apr. 2019, doi: 10.1145/3297280.3297326.

[6] T. Nolle, A. Seeliger, and M. Mühlhäuser, "BINet: Multivariate Business Process Anomaly Detection Using Deep Learning," *International Conference on Business Process Management*, pp. 271–287, 2018, doi: 10.1007/978-3-319-98648-7_16.

[7] S. J. van Zelst, M. Fani Sani, A. Ostovar, R. Conforti, and M. La Rosa, "Filtering Spurious Events from Event Streams of Business Processes," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10816, pp. 35–52, 2018, doi: 10.1007/978-3-319-91563-0_3.

[8] R. Sarno, F. Sinaga, and K. R. Sungkono, "Anomaly detection in business processes using process mining and fuzzy association rule learning," *Journal of Big Data*, vol. 7, no. 5, pp. 1–19, Dec. 2020, doi: 10.1186/s40537-019-0277-1.

[9] H. Darmawan, R. Sarno, A. S. Ahmadiyah, K. R. Sungkono, and C. S. Wahyuni, "Anomaly Detection based on Control-flow Pattern of Parallel Business Processes," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 16, no. 6, pp. 2809–2816, Dec. 2018, doi: 10.12928/telkomnika.v16i6.10568.

[10] D. B. Prakoso, K. R. Sungkono, and R. Sarno, "Declarative Algorithm for Checking Wrong Indirect Relationships of Process Model Containing Non-Free Choice," *2019 12th International Conference on Information & Communication Technology and System (ICTS)*, pp. 37–42, Jul. 2019, doi: 10.1109/ICTS.2019.8850931.

[11] N. Francis *et al.*, "Cypher: An Evolving Query Language for Property Graphs," *Proceedings of the 2018 International Conference on Management of Data - SIGMOD '18*, pp. 1433–1445, 2018, doi: 10.1145/3183713.3190657.

[12] D. B. Prakoso, K. R. Sungkono, and R. Sarno, "Declarative algorithm for checking wrong indirect relationships of process model containing non-free choice," *Proceedings of 2019 International Conference on Information and Communication Technology and Systems, ICTS 2019*, pp. 37–42, 2019, doi: 10.1109/ICTS.2019.8850931.

[13] R. Sarno, K. Sungkono, R. Johanes, and D. Sunaryono, "Graph-Based Algorithms for Discovering a Process Model Containing Invisible Tasks," *International Journal of Intelligent Engineering and Systems*, vol. 12, no. 2, pp. 85–94, Apr. 2019, doi: 10.22266/ijies2019.0430.09.

[14] N. Kamide and R. Yano, "Logics and translations for hierarchical model checking," *Procedia Computer Science*, vol. 112, pp. 31–40, 2017, doi: 10.1016/j.procs.2017.08.014.

[15] H. S. Bank, S. D'souza, and A. Rasam, "Temporal Logic (TL)-Based Autonomy for Smart Manufacturing Systems," *Procedia Manufacturing*, vol. 26, pp. 1221–1229, 2018, doi: 10.1016/j.promfg.2018.07.159.