Mapping Graph-Based Process Model Into Discrete Event Simulation (DES)

Riza Dwi Andhika
Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
rizaandhika.19051@mhs.its.ac.id

Kelly Rossa Sungkono
Informatics
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
kelly@its.ac.id

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

***Abstract*— Many process discovery algorithms are emerging to form process models based on event logs automatically. There are several representations of the process model, one of which is the graph-based process model. The advantage of using a graph-based process model is that event logs, and process models can be represented in a graph model, so there is no conversion cost to converting event logs to process models. However, the weakness of the graph-based process model is that there are no features to simulate the model. To the contrary, Discrete Event Simulation (DES) is a type of process model that can be used for simulation; However, DES is created manually based on the knowledge of experts. This study proposes mapping a graph-based process model to the DES model. This research also makes an algorithm to determine the time span of the implementation of activities, which is used in the DES model. This study generates event logs based on DES obtained using the proposed mapping and builds a graph-based process model based on event logs. The evaluation verifies that the graph-based process model is the same as the existing one, which means that the proposed mapping is successful.**

Keywords*— *discrete event simulation, graph-based process model, process model mapping

I. INTRODUCTION

System auditors can leverage many process model types; some of them are Petri Nets, YAWL, BPMN, Discrete Event Simulation (DES), and Graph-based Process Model. Petri Nets (also known as Place or Transition Net) [1] is one process model which consists of interconnected states and transitions [2]. Furthermore, YAWL has a mature verification tool to detect structural errors [3], and its semantics are based on Petri Nets. The term business process refers to a group of distinguishable steps executed sequentially to achieve a certain aim. A business process defines the expected flow through several control-flow. The third perspective, the data required to complete each activity, is added by business process management systems (BPMSs). There have been several open sources and commercial BPMSs, each with a unique graphical notation. Another BPMS, known as YAWL, was produced by implementing this language. [4]. The current version of BPMN standard combines figurative and machine-understandable viewpoints to depict a process [5]. BPMN consists of a flow of events, activities, and sequences. The Object Management Group (OMG) maintains BPMN, an ISO/IEC 19510 international standard. Also, BPMN gives a way to represent processes graphically [6] formally. BPMN is another comprehensive strategy for enhancing an organization's workflow that focuses on process reengineering. Its objective is continuously to improve processes to maximize effectiveness and efficiency [7]. BPMN notations combine those viewpoints to produce a model. If process mining algorithms are concentrated on fusion model placement, then there are learned lessons, for instance, ordering by a particular group handled by a specific organization [8]. A process model obtained via a graph-based algorithm and a graph database is known as a "graph-based process model." The benefit of using a graph-based process depicting rules is that relationships are stored as well as actions in a graph database, allowing the algorithm to be built to create a low time complexity. [9, 10]. The advantage of using a Graph-based process model is that the event log and process model are on the same platform. Thus there is no conversion cost. Therefore, the Graph-based Process Model was chosen for this research paper. Despite its advantage, the Graph-based process model still cannot be simulated.

Discrete Event Simulation, a type of computer-based modeling methodology, offers a modest and adaptable way to accomplish things and is distinguished by its capacity to replicate the dynamic actions of complicated organizations and the cooperation of society, community, and circumstances [11]. Arguably, it is the most widely used OR simulation approach and OR technique in actual practice [12]. The ability to experiment with any component of a business system is a key benefit of simulation over other operational research (OR) methodologies [13]. However, there is no function of forming a Discrete Event Simulation model based on event logs, so it requires another application to convert the event log into a process model and requires expert knowledge to convert the results of the process model in the application into a DES model.

For those advantages and drawbacks of the Graph-based Process Model and Discrete Event Simulation have, this paper research proposed to create process model mapping from Graph-based Process Model to Discrete Event Simulation so that the explained drawbacks of both models can be eliminated. This paper creates defined rules to determine the execution time of activity that will be used in Discrete Event Simulation and the mapping notation from Graph-based Process Model to Discrete Event Simulation.

As the proposed model mapping, this research uses event log data from *Shopee* as the case study. The evaluation is done by creating Graph-based Process Model from event log data

2022 International Conference on Informatics Electrical and Electronics (ICIEE) | 978-1-6654-8622-4/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICIEE55596.2022.10009990

that is created based on proposed mapping. The proposed mapping can be concluded to true if the generated Graph-based Process Model has the same result as the Graph-based Process Model that is generated from *Shopee* event log data.

II. PREMILINARIES

A. Graph-based Process Model

Graph-based process model is represented like graph data structure (consists of nodes and relations) using graph database to store event log data and represent the activities and their relation.

Example of relations in graph-based process model for Sequence on Fig, 1, XOR on Fig, 2, AND on Fig, 3, and Non-Free Choice relation on Fig, 4.

B. Discrete Event Simulation

Discrete Event Simulation (DES), a type of computer-based modelling methodology, offers a modest and adaptable way to accomplish things. It is distinguished by its capacity to replicate the dynamic actions of complex organizations and the cooperation of society, community, and circumstances [11]. DES facilitates users to know the growth of the system and the description of the interaction of people and various histories of system entities in the operational realm [14]. The primary benefit of DES is the capacity to run experiments that cannot be performed on existing production systems. Creating a simulation model aids in learning new information that may improve the actual application. It can be challenging to determine whether an observation is the consequence of system interactions or chance because simulations involve unpredictability (random inputs) [15]. A software called *AnyLogic* can represent that process model. There are specific notations or blocks available to represent nodes and relations. As a side note, *AnyLogic* requires us to use source and sink blocks to denote the start and end point of a process model.

C. Event Log

An event log is a compilation of activities containing the id of the activity set, activity name, and the time when an activity is started. It also serves as a digital record of event executions in an information system. [16]. The case is a sequence of activities recorded in event log data. Those sequences can occur more than one. On the other hand, Trace is like a case without an overlapping sequence of activities. In other words, it is a variance of cases. Given that the event log serves as the dominant dossier for process mining approaches, the quality of this data significantly influences the final model. Low-quality event logs might produce complex results and unstructured, hard-to-understand models [17].

III. METHOD

Generally, the steps or flowchart of this research is shown in the Fig. 5. First, it needs to generate the Graph-based Process Model from the *Shoope* event log data. Next, generate the time span in the Graph-based Process Model which will be needed for mapping process. Then, map the process model from the Graph-based Process Model to Discrete Event Simulation.

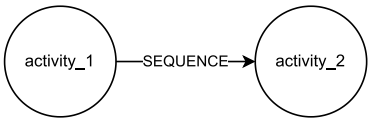


Fig. 1. Sequence relation

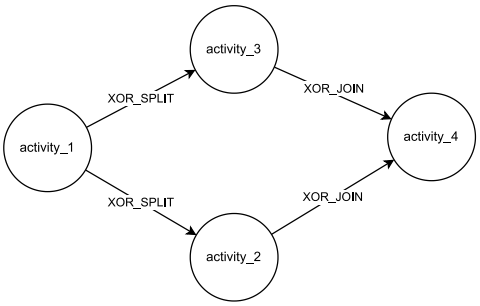


Fig. 2. XOR relation

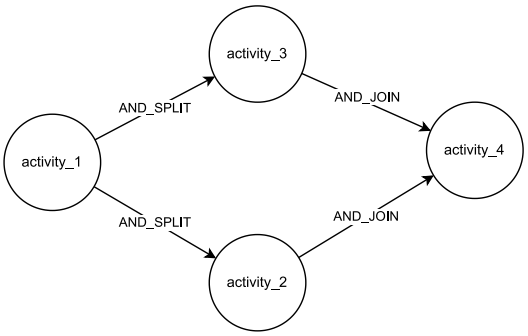


Fig. 3. AND relation

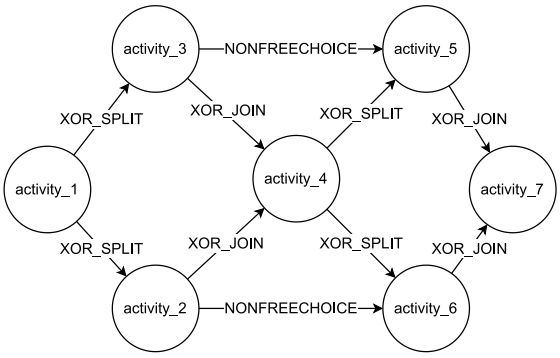


Fig. 4. Non-Free Choice relation

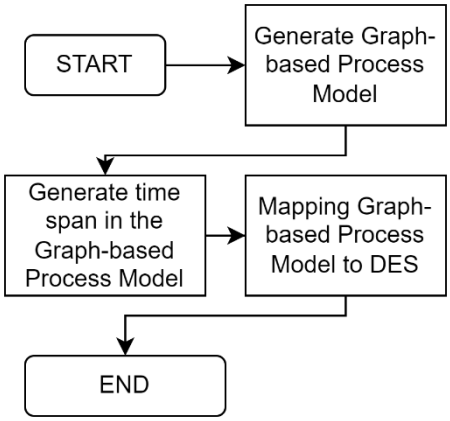


Fig. 5. Research general flowchart

A. Generate Graph-based Process Model

In order to create the graph-based process model from event log, the data needs to be loaded first in the graph database and then create unique node based on the activity name attribute in the event log. Later, it will be referred as case activity. Then, the sequence relation between case activity needs to be constructed along with the average, minimum, and maximum time of the case activity node. To do that, the process will be to track two case activity node, iterate over activity record, and calculate the duration of current activity by subtracting the timestamp of next activity with current activity where the case id of both activity is the same and the activity name of current and next activity match with both tracked case activity and finally the sequence relation can be made between both tracked case activity.

Now to construct the XOR relation, first, it needs to create XOR SPLIT relation by finding two case activity that have relation where the destination node has minimum one outgoing relation, minimum one incoming relation, and the source node has minimum two outgoing relations. Then the XOR SPLIT relation can be created. For the XOR JOIN, it needs to find two case activity that have relation where the destination node has minimum two incoming relations, and the source node has minimum one outgoing relation. Then, the XOR JOIN relation can be created.

Next, to create AND relation, first, it needs to create AND SPLIT relation by finding case activity source node where has two outgoing relations to two different destination nodes where each destination cannot have sequence relation directing to the source node and each of them has the same number of outgoing relations with the source node outgoing relation. Then, the AND SPLIT relation can be created. For the AND JOIN relation, it needs to find case activity node where source nodes have outgoing relation to the same destination node, the number of incoming relations to destination node should be the same as each source node outgoing relations, and incoming relations to destination node cannot be AND SPLIT relation. Then, the AND JOIN relation can be created.

Additionally, Invisible Task relation needs to be created. To do that, find source node that has relation y to destination node a and relation x to node b. If relation y is XOR SPLIT and r is XOR JOIN, then create invisible task node and create XOR SPLIT relation from source node to invisible task node and XOR JOIN from invisible task node to node a.

To create Non-Free Choice, find node a that has outgoing XOR JOIN relation to node m, node m has outgoing XOR SPLIT relation to node n. If the source node name is not equal as destination node name, and then track two additional nodes called x and y. If node a name is not equal as node n, node x name is equal as node a, node y name is equal as node n, node x and y have equal case id, the number of incoming relations to node m is more than one, the number of outgoing and incoming relation to node m is equal, and id of node n is greater than node a. Then, create Non-Free Choice relation.The complete cypher query for this Graph-based process model generation is shown in the Fig. 6, Fig. 7, Fig. 8, Fig. 9, and Fig. 10.

B. Generate Time Span in the Graph-based Process Model

The process to generate time span needs the duration data of activity node. To do that, first, track two case activity nodes, iterate over activity record, and calculate the duration of current activity by subtracting the timestamp of next activity with current activity where the case id of both activity is the same and the activity name of current and next activity match with both tracked case activity. The process is done together when creating Sequence relation as shown in the Fig. 6. After every activity records duration have been calculated, then to calculate the average time of a case activity, it can be done by aggregating the average duration of the case activity as shown in the Fig. 7. Using the same method, the minimum and maximum time of case activity can be calculated.

C. Mapping Graph-based Process Model to DES

In order to map from graph-based process model to DES, the writer used the TABLE I to map between Graph-based process model and DES entity. As a side note, in DES, there are two additional mandatory entities that need to be presented (not mandatory in Graph-based Process Model) which are source and sink that act as a starting point and endpoint.

```
// Load all data from event log
LOAD CSV with headers FROM "file:///shoope.csv" AS ln
MERGE (:ActNode{IdCase: ln.IdCase,
Name: ln.Name,
Time: ln.Time})

// Create case activity nodes
LOAD CSV with headers FROM "file:///shoope.csv" AS ln
MERGE (:CaseActNode {Name: ln.Name })

// Create sequence relation between case activity nodes
MATCH (c:ActNode)
WITH COLLECT(c) AS Caselist
UNWIND RANGE(0, Size(Caselist) - 2) as index
WITH Caselist[index] AS s1, Caselist[index + 1] AS s2
MATCH (b:CaseActNode),(a: CaseActNode)
WHERE s1.IdCase = s2.IdCase AND s2.Name = b.Name AND s1.Name = a.Name
SET s2.Duration = duration.between(s1.Time, s2.Time).seconds
MERGE (a)-[r:SEQUENCE]->(b)
```

Fig. 6. Cypher query to load data from csv file, create activity node, case activity node, and Sequence relation

```
// Set minimum time for each case activity
MATCH (c:ActNode)
WITH COLLECT(c) AS Caselist
UNWIND RANGE(0, Size(Caselist)) as index
WITH Caselist[index] AS s1
MATCH (a:CaseActNode)
WHERE s1.Name = a.Name
WITH a, MIN(tofloat(s1.Duration)) as timemin
SET a.MinTime = timemin

// Set maximum time for eacg case activity MATCH (c:ActNode)
WITH COLLECT(c) AS Caselist
UNWIND RANGE(0, Size(Caselist)) as index
WITH Caselist[index] AS s1
MATCH (a:CaseActNode)
WHERE s1.Name = a.Name
WITH a, MAX(tofloat(s1.Duration)) as timemax
SET a.MaxTime = timemax

// Set average time for each case activity
MATCH (c:ActNode)
WITH COLLECT(c) AS Caselist
UNWIND RANGE(0, Size(Caselist)) as index
WITH Caselist[index] AS s1
MATCH (a:CaseActNode)
WHERE s1.Name = a.Name
WITH a, AVG(tofloat(s1.Duration)) as timeavg
SET a.AvgTime = timeavg
```

Fig. 7. Cypher query to generate time span

```
// Create XOR relation between case activity nodes
MATCH (src)-[rel]->(dest)
WHERE SIZE((src)-->())>1 AND SIZE((dest)<--())=1 AND ( SIZE((dest)--
>())=1 OR SIZE((dest)-->())>1 )
CREATE (src)-[:XOR_SPLIT]->(dest)
DELETE rel

MATCH (src)-[rel]->(dest)
WHERE ( size((src)-->())=1 OR size((src)-->())>1 ) AND size((dest)<-
-())>1
CREATE (src)-[:XOR_JOIN]->(dest)
DELETE rel

// Create AND relation between case activity nodes
MATCH (dest1)<-[rel1]-(src)-[rel2]->(dest2)
WHERE size((src)-->())>1
AND size((src2)-->())=size((src)-->()) AND size((dest1)--
>())=size((src)-->())
AND not (dest1)-[:SEQUENCE]->(src) AND not (dest2)-[:SEQUENCE]-
>(src)
MERGE (dest1)<-[:AND_SPLIT]-(src)-[:AND_SPLIT]->(dest2)
DELETE rel1, rel2

MATCH (dest1)-[rel1]->(src)<-[rel2]-(dest2)
WHERE size((src)<--())>1
AND size((dest2)-->())=size((src)<--()) AND size((dest1)--
>())=size((src)<--())
AND not ()-[:AND_SPLIT]->(src)
MERGE (dest1)-[:AND_JOIN]->(src)<-[:AND_JOIN]-(dest2)
DELETE rel1, rel2

// Create Invisible task relation
MATCH (dest2)<-[rel1]-(src)-[rel2]->(dest 1)
WHERE ( TYPE(rel2)='XOR_SPLIT' AND TYPE(rel1)='XOR_JOIN')
CREATE (i:Inv_Task_Node {Name: "Inv_Task_Node"})
WITH src, i, dest2, rel1, rel2
CALL apoc.create.relationship(src, type(rel2), {}, i) YIELD rel as
relation1
CALL apoc.create.relationship(i, type(rel1), {}, dest2) YIELD rel as
relation2
DELETE rel1

// Combine 2 invisible task
match (a:Inv_Task_Node)<-[r]-(b)-[l]->(c:Inv_Task_Node)-[s]->(d)
where id(a)<id(c)
merge (a)-[:XOR_JOIN]->(d)
delete l,s,c
```

Fig. 8. Cypher query to create XOR, AND, and Invisible Task relation

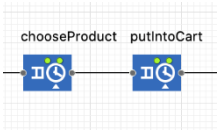
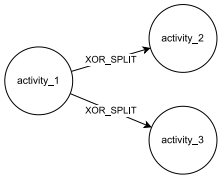
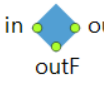
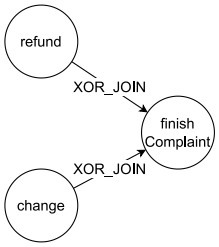
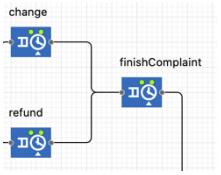
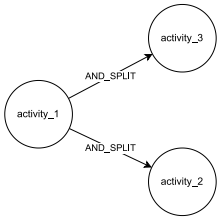
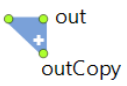
```
// Create Invisible Non-Prime Task relation
MATCH (a)-[b]->(c)-[d]->(e)
WHERE type(d)='XOR_JOIN' AND type(b)='AND_SPLIT'
DELETE b
MERGE (a)-[:XOR_SPLIT]->(c)

MATCH (a)-[c]->(b)-[d]->(a)
MATCH (b)<-[g]-(e)-[f]->(a)
MATCH (e)-[j]->(k)
MATCH (m:ActNode),(n:ActNode)
WHERE a.Name=m.Name AND b.Name=n.Name AND id(b)>id(a) AND
TYPE(j)='XOR_SPLIT'
CALL{
                    WITH m, n
                    CALL apoc.meta.data() yield property
                    WHERE m.IdCase<>n.IdCase AND property = 'IdCase'
RETURN COUNT(property) as Count2
}
CALL {
                    WITH m,n
                    CALL apoc.meta.data() yield property
                    WHERE m.idCase=n.IdCase AND property = 'IdCase'
RETURN COUNT(property) as Count1
}
WITH SUM(Count2)%SUM(Count1) as mod, a, b, c, d, e, f, g, j
WHERE mod=0
CREATE (i:Inv_Task_Node {Name: "Inv_Task_Node"})
WITH e,i,c, f, g, d, j, a, b
CALL apoc.create.relationship(e, type(j), {}, i) YIELD rel as
relation1
DELETE c, f, g, d
MERGE (b)<-[:AND_SPLIT]-(i)-[:AND_SPLIT]->(a)


MATCH (d)<-[c]-(a:Inv_Task_Node)-[b]->(e)
MATCH (d)<-[g]-(f:Inv_Task_Node)-[h]->(e)
MATCH (j)-[x]->(a)
MATCH (j)-[y]->(f)
WHERE id(a)<id(f)
delete g,h,y,f

match (b)<-[r:XOR_JOIN]-(a:Inv_Task_Node)-[l:XOR_JOIN]->(c)
match (f)-[s]->(d:Inv_Task_Node)-[:AND_SPLIT]->(e)
delete r,l,s
merge (a)-[:XOR_JOIN]->(d)<-[:XOR_JOIN]-(f)
```

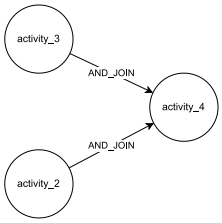
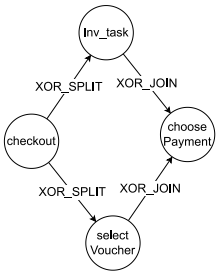
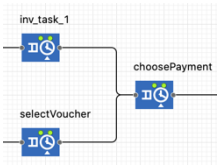
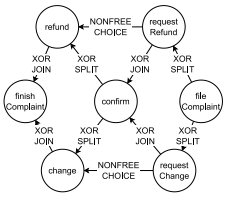
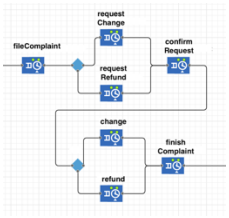
Fig. 9. Cypher query to create Invisible Non-Prime Task relation

```
// Create Non-Free Choice relation
MATCH (src)-[rel1:XOR_JOIN]->()
MATCH ()-[rel2:XOR_SPLIT]->(dest)
MATCH (x:ActNode),(y:ActNode)
WHERE src.Name<>dest.Name AND x.Name=src.Name AND y.Name=dest.Name
AND x.IdCase=y.IdCase AND x.Time<l.Time
MERGE (src)-[:NON_FREE_CHOICE]->(dest)

MATCH (src1)-[rel1:AND_JOIN]->(mid:Inv_Task_Node)-[rel2]->(dest1)
MATCH (src2)-[rel3]->(mid:Inv_Task_Node)-[rel2]->(dest1)
MATCH ()-[rel4:XOR_SPLIT]->(dest2)
MATCH (x:ActNode),(y:ActNode)
WHERE src1.Name<>dest2.Name AND dest1.Name<>dest2.Name AND
x.Name=src1.Name AND y.Name=dest2.Name AND x.IdCase=y.IdCase AND
x.Time<y.Time
MERGE (mid)-[:NON_FREE_CHOICE]->(dest2)

MATCH k = (dest1)<-[l:NON_FREE_CHOICE]-(src)-[r:NON_FREE_CHOICE]-
>(dest2)
where dest1.Name <> dest2.Name
delete l,r
```

Fig. 10. Cypher query to create Non-Free Choice relation

TABLE I. TABLE ENTITY MAPPING

Entity Name	Graph-based Process Model	Discrete Event Simulation
START	-	
ACTIVITY		
SEQUENCE		
XOR_SPLIT		
XOR_JOIN		
AND_SPLIT		

AND_JOIN		
INVISIBLE_ TASK		
NON_FREE_ CHOICE		
END	-	

IV. Experiment

A. Material

The process model used in this research is about the transaction process from *Shopee* company, where a person selects a product, processes the payment, and later, the person will get points from that transaction. In this process model, five different relations occur which are Sequence, XOR, AND, Non-Free Choice, and Invisible Task.

The event log data contains 25 cases and 23 traces. Moreover, there are 17 different case activity which are *chooseProducts, putIntoCart, checkout, selectVoucher, choosePayment, receivePackage, writeReview, giveRating, getShopeePoints, fileComplaint, requestRefund, confirmRequest, refund, finishComplaint, uploadReceipt, requestChange,* and *change*.

B. Result

The result of the Graph-based Process Model created from the Shopee event log is shown in Fig. 11. Moreover, as a case activity can have many actual activities along with its execution time, the Graph-based Process Model creation process also calculates the maximum, minimum, and average time result of every case activity as shown in TABLE II. For example, in this experiment, the *chooseProduct* case activity has 25 actual activities, therefore the aggregation of maximum, minimum, and average execution times of all those actual activities is 86,00 seconds, 2,00 seconds, and 7,25 seconds respectively. The mapping result from the Graph-based Process Model to Discrete Event Simulation is shown in Fig. 12. It shows how all the case activities and relations are adapted in the Discrete Event Simulation based on the TABLE I mapping. After simulating the Discrete Event Simulation, it generated another event log. The generated event log is then used to test the mapping whether is correct or not by generating again the graph-based process model. The result of a new graph-based process model is shown in Fig. 13.

As the result shows, Fig. 11 and Fig. 13 have the same result in the Graph-based Process Model, where both have the same case activities and the relationship between each of the case activities. Therefore, the mapping process from the Fraph-based process model to discrete event simulation is correct.

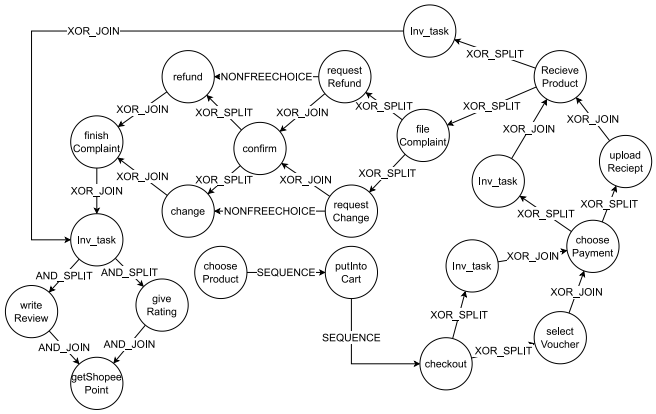


Fig. 11. Graph-based Process Model created using *Shopee* event log data

TABLE II. Table Time Span Result

Activity Name	Max Time (seconds)	Min Time (seconds)	Average Time (seconds)
chooseProduct	86,00	2,00	7,25
putIntoCart	17,00	2,00	3,63
checkout	127,00	3,00	11,80
selectVoucher	52,00	2,00	8,55
choosePayment	14,00	3,00	5,79
recievePackage	69,00	3,00	9,96
writeReview	6,00	2,00	2,50
giveRating	9,00	1,00	2,79
getShopeePoint	7,50	1,50	2,65
fileComplaint	66,00	4,00	12,75
requestRefund	9,00	4,00	4,63
confirmRequest	5,00	4,00	3,5
refund	63,00	2,00	9,75
finishComplaint	6,00	2,00	3,19
uploadReceipt	50,00	2,00	6,42
requestChange	6,00	3,00	4,25
change	7,00	2,00	3,13

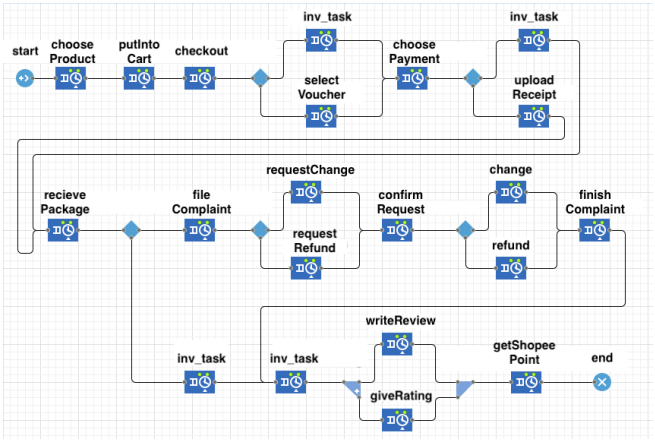


Fig. 12. Discrete Event Simulation mapping result

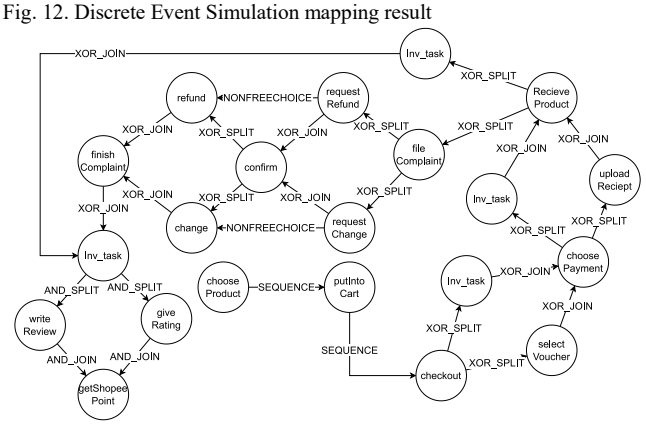


Fig. 13. Graph-based Process Model generated from Discrete Event Simulation event log data

V. Conclusion

The mapping steps to map Graph-based Process Model to DES can be summarized by get the event log data. Next, generate the Graph-based Process Model by using the cypher query. Then, map all the entities from Graph-based Process Model to DES.

The event log data used in this research, is from *Shopee* event log data where the attributes consist of case id, activity, timestamp, and originator. As the research result, both Graph-based Process Model created from *Shopee* event log and the DES simulation event log show the same result. Therefore, the mapping process in this research is correct.

For future work, the mapping process from graph-based process model to discrete event simulation can be done automatically instead of doing it manually to retrieve the result faster using the *AnyLogic* API.

Acknowledgment

This research was funded by Lembaga Pengelola Dana Pendidikan (LPDP) under Riset Inovatif-Produktif (RISPRO) Invitation Program, the Indonesian Ministry of Education and Culture under Penelitian Terapan Unggulan Perguruan Tinggi (PTUPT) Program, and Institut Teknologi Sepuluh Nopember (ITS) under Penelitian Dana Departemen and project scheme of the Publication Writing and IPR Incentive Program (PPHKI).

References

[1] A. Pinna, R. Tonelli, M. Orrú, and M. Marchesi, “A petri nets model for blockchain analysis,” The Computer Journal, vol. 61, no. 9, pp. 1374–1388, Jan. 2018.

[2] K. B.B. Mo’Minov, “Modelling asynchronous parallel process with Petri Net,” International Journal of Engineering and Advanced Technology, vol. 8, no. 5S3, pp. 1–2, Jul. 2019.

[3] M. Kherbouche, K. Bouafia, and B. Molnar, “Transformation of UML State Machine to Yawl,” 2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS), pp. 215–220, Dec. 2019.

[4] M. Adams, A. V. Hense, and A. H. M. ter Hofstede, “Yawl: An open source business process management system from science for science,” SoftwareX, vol. 12, pp. 1–2, Jul. 2020.

[5] M. Ramos-Merino, L. M. Álvarez-Sabucedo, J. M. Santos-Gago, and F. de Arriba-Pérez, “A pattern based method for simplifying a BPMN process model,” Applied Sciences, vol. 9, no. 11, p. 2322, Jun. 2019.

[6] M. Häußler, S. Esser, and A. Borrmann, “Code compliance checking of railway designs by integrating BIM, BPMN and DMN,” Automation in Construction, vol. 121, pp. 4–5, Jan. 2021.

[7] P. Wiśniewski, K. Kluza, and A. Ligęza, “An approach to Participatory Business Process Modeling: BPMN model generation using constraint programming and graph composition,” Applied Sciences, vol. 8, no. 9, p. 1428, Aug. 2018.

[8] A. Kalenkova, A. Burattin, M. de Leoni, W. van der Aalst, and A. Sperduti, “Discovering high-level BPMN process models from Event Data,” Business Process Management Journal, vol. 25, no. 5, pp. 995–1019, Oct. 2018. = M. Adams, A. V. Hense, and A. H. M. ter Hofstede, “Yawl: An open source business process management system from science for science,” SoftwareX, vol. 12, pp. 1–2, Jul. 2020.

[9] 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.

[10] I. Waspada, R. Sarno, and K. Sungkono, “An improved method of parallel model detection for graph-based process model discovery,” *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 2, pp. 127–139, Apr. 2020.

[11] X. Zhang, “Application of discrete event simulation in Health Care: A Systematic Review,” BMC Health Services Research, vol. 18, no. 1, pp. 1–11, Sep. 2018.

[12] A. Greasley and J. S. Edwards, “Enhancing discrete-event simulation with Big Data Analytics: A Review,” Journal of the Operational Research Society, vol. 72, no. 2, pp. 247–267, Nov. 2019.

[13] K. Agalianos, S. T. Ponis, E. Aretoulaki, G. Plakas, and O. Efthymiou, “Discrete event simulation and Digital Twins: Review and challenges for Logistics,” Procedia Manufacturing, vol. 51, pp. 1636–1641, Jun. 2020.

[14] J. S. Morgan, S. Howick, and V. Belton, “A toolkit of designs for mixing discrete event simulation and system dynamics,” European Journal of Operational Research, vol. 257, no. 3, pp. 907–918, Aug. 2017.

[15] A. Kampa, G. Go≈Çda, and I. Paprocka, "Discrete event simulation method as a tool for improvement of manufacturing systems," Computers, vol. 6, no. 1, p. 10, Feb. 2017

[16] R. Andrews, C. G. J. van Dun, M. T. Wynn, W. Kratsch, M. K. E. Röglinger, and A. H. M. ter Hofstede, "Quality-informed semi-automated event log generation for process mining," Decision Support Systems, vol. 132, p. 113265, Feb. 2020.

[17] H. M. Marin-Castro and E. Tello-Leal, “Event log preprocessing for Process Mining: A Review,” Applied Sciences, vol. 11, no. 22, p. 10556, Nov. 2021.