Indonesian Journal of Electrical Engineering and Computer Science
Vol. 18, No. 1, April 2020, pp. 124~134
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v18.i1.pp124-134

Gap analysis business process model by using structural similarity

Afrianda Cahyapratama[1], Kelly Rossa Sungkono[2], Riyanarto Sarno[3]
[1] Department of Information Technology Management, Institut Teknologi Sepuluh Nopember, Indonesia
[2,3]Department of Informatics, Institut Teknologi Sepuluh Nopember, Indonesia

Article Info

Article history:

Received Jul 18, 2019
Revised Oct 4, 2019
Accepted Oct 21, 2019

Keywords:

Business process
Dice coefficient similarity
Gap analysis
Graph database
Graph-matching algorithm

ABSTRACT

Gap analysis process model is a study that can help an institution to determine differences between business process models, such as a model of Standard Operating Procedure and a model of activities in an event log. Gap analysis is used for finding incomplete processes and can be obtained by using structural similarity. Structural similarity measures the similarity of activities and relationships depicting in the models. This research introduces a graph-matching algorithm as the structural similarity algorithm and compares it with dice coefficient algorithms. Graph-matching algorithm notices parallel relationships and invisible tasks, on the contrary dice coefficient algorithms only measure closeness between activities and relationships. The evaluation shows that the graph-matching algorithm produces 76.76 percent similarity between an SOP model and a process model generating from an event log; while, dice coefficient algorithms produces 70 percent similarity. The ability in detecting parallel relationships and invisible tasks causes the graph-matching algorithm produces a higher similarity value than dice coefficient algorithms.

Corresponding Author:

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

1. INTRODUCTION

Gap analysis of business processes is a tool for evaluating the performance of a company. Gap analysis is also one of the most important steps in the evaluation stage of performance suitability based on the Standard Operational Procedure (SOP) with actual events. Standard Operational Procedure (SOP) is a business process flows containing activities and tasks in achieving the goals of a company. Literally, the "gap" identifies a difference between one model and another model [1]. Gap analysis is often used in management for measuring service quality.

A business process is a series of performed activities and tasks to achieve the objectives of a company or institution [2]. The performance of a company can be reflected in the operational system and strategy used by a company. Gap analysis of business processes is very necessary to assess how much the gap value between actual performance and Standard Operational Procedure (SOP) has been designed by the company. Gap analysis [3] results are the basis for decision making related to readjustment of the company's operational standard procedures to meet good service standards.

An event log is a collection of events that contain information about the business processes that are running at the time [4]. The event log will be used by several process model building algorithms to generate business process models directly [5, 6]. Process modeling algorithm will run optimally if the event log that is processed is an event log that has complete information [7]. In the process modeling algorithm, there are two things that are of concern, namely non-free choice and invisible tasks. Non-free choice is the relationship

between activities in the chosen relationship with activities in other chosen relationships [5, 6]. Meanwhile, the use of invisible tasks on several process models to illustrate the existence of skip conditions or parallel overlapping relationships.

Graph-matching algorithm becomes one of the gap analysis techniques by measuring the similarities of a collection of business process models [8-10]. The ability of graph-matching algorithm is rules to create a process model containing parallel relationships, invisible tasks, or non-free choice directly based on the event log and matches the obtained process model and Standard Operational Procedure (SOP) model [11, 12] forming in a graph model. Standard Operational Procedure (SOP) model is the main model of business processes in a company which is the main reference for comparison of other business process models. Then, another gap analysis technique, dice coefficient algorithms identify process models that have similar structures or patterns from Standard Operational Procedure (SOP) model [13]. Dice coefficient algorithms only consider activities and relationships, without considering invisible tasks and the patterns describing parallel relationships.

This study aims to compare structural similarity algorithms, i.e. graph-matching algorithm and dice coefficient algorithms, to analyze the ability of other information containing non-free choice, parallel relationships, and invisible tasks influences the gap result of business process models. The used process is after-sales services, or can be called RMA (Return Material Automatically).

2. PROPOSED METHOD

A paper entitled "Software Measures for Business Processes" said that different business processes may have different levels of effectiveness. For example, based on the complexity of a business process model, the effectiveness can be measured based on formal descriptions of business processes [14]. The level of effectiveness is not only obtained from the complexity of business processes, but also can be measured by getting a gap value from the similarity of business process models based on Standar Operational Procedure (SOP) and an event log [15]. There are three used things to find gap values in business process models, which are structural similarity, behavior similarity, and semantic similarity. The structure similarity is viewed from the topology or forming structure, if represented in a graph, then the graph structure [16].

2.1. Graph Database

A graph database is a database consisting of a number of graphs. Judging from the data structure, it will be a directed graph in a mathematical sense. SQL (Structured Query Language) is a set of special commands used to access data in a relational database. When compared to SQL, the database graph have a difference that is having an entity, while the SQL database has rows. The graph database is not an SQL database [17]. The purpose of using a graph database is to solve problems that cannot be resolved in a tabular database. To obtain data that has a relationship, users often have to write very complex SQL queries. The advantage of analysing with database graphs is much easier for users to see patterns of relationships between activities, which are very difficult to see when using a tabular database [18]. An example of a relational database is shown in Table 1, where Case, Activity, and Time are attributes.

Table 1. Event Log Example

Case	Activity	Time
C1	Confirm arrival of items	9/9/2018 07:00
C1	Prepare a place	9/9/2018 07:05
.	.	.
.	.	.
.	.	.
C2	Match items with a list	9/9/2018 07:25
C2	Sealing items	9/9/2018 07:40
C2	Input in database	9/9/2018 07:50

Data in graph database consists of nodes and edges. In Figure 1, an example of a graph is made from the data structure shown in Table 1 above and adds the NEXT relation between activities. A node is a point that contains all information about an object, while a node is a representation of the relationship between objects. In addition, graph databases also have more flexibility in treating data, unlike SQL databases. In a SQL database, if the data rows need to be inserted, then the table must be created first and if the elements of some data need to be expanded, it is necessary to edit the table. Meanwhile, NoSQL can create data without a table, so the data representation process becomes very flexible.

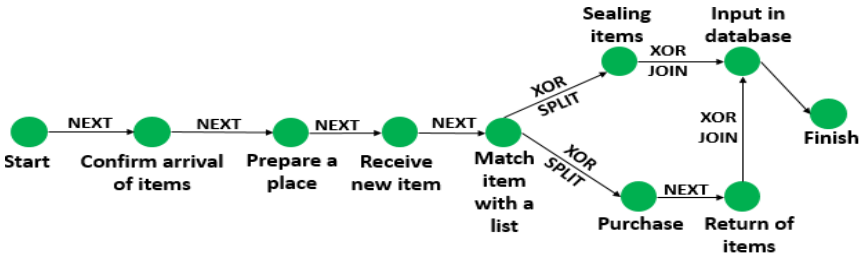


Figure 1. Example Structure of a Graph Database

2.2. Non-free Choice

The non-free choice process is a process that describes the linkages between activities in the choice process [19]. The non-free choice process involves an implicit relationship in the description of business process models. Examples of non-free choice processes in the business process model are described in Figure 2. The relationship between the "Check Stock RMA" activity and the "Fill the Device Request Form" activity and the relationship between "Check Sale Stock" activity and activity "Requesting the release of stock" is an example of an implicit relationship.

Figure 2 explains that the "Fill the Device Request Form" activity is carried out if the "Check Stock RMA" activity has been completed and the activity "Requesting the release of stock" is executed if the "Check Sale Stock" activity has been completed, even though the activity "Fill the Device Request Form" and "Requesting the release of stock" activity are included in the choice activity [20, 21].

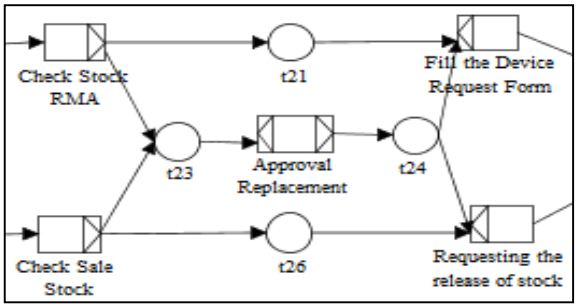


Figure 2. Non-Free Choice Processes

2.3. Graph-Matching Algorithm

In finding similarities in business process models, graph-matching algorithm are used as one of the approaches [8, 22-24]. A graph has nodes and edges. In mapping business processes into a graph, it is indicated by the node as activity and edges as a process or function that occurs between one activity and another activity. The similarity of the business process model is detected with the same process and function in each activity.

The similarity of the structure of a business process is the value of similarity between business process models by looking at the structure or shape of the pattern [25]. According to Remco Dijkman to calculate the similarity of the two processes, mapping must be found that induces maximum similarity. The structure is seen from each part of the existing model [16].

2.4. Dice Coefficient Similarity

The algorithm or calculation of dice is known by several other names, namely the Sørensen index or the dice coefficient. The other two names also refer to the representation of "similarity coefficients" and other variations of an index. The alternative spelling for Sørensen is Sorenson, Soerenson Index and Sorenson index, there are also other names of this algorithm including binary czekanowski (non-quantitative) index [13]. The equation of dice coefficient is shown in (1).

$$S(A,B) = 2x \frac{|A \cap B|}{|A| + |B|} \quad (1)$$

where:
A = The first process model
B = Second process model

3. RESEARCH METHOD

3.1. Data Collection

The data collected related to this research is a dataset from a distributor company. The dataset used in this study is the SOP and collection of event logs from the distributor company. Table 2 is an example of an RMA business process log event in a distributor company.

Table 2. Event Log of RMA Submission

Case_ID	Activity	Start_Timestamp	End_Timestamp
C1	Start	1/31/2018 8:00	1/31/2018 8:00
C1	Request RMA	1/31/2018 8:00	1/31/2018 8:04
C1	Check Device Status	1/31/2018 8:04	1/31/2018 8:06
C1	Status Info	1/31/2018 8:06	1/31/2018 8:10
C1	Create a service receipt	1/31/2018 8:10	1/31/2018 8:18
C1	Fill Form Registration RMA	1/31/2018 8:18	1/31/2018 8:23
C1	Submit the Device to the technician	1/31/2018 8:23	1/31/2018 8:30
.	.	.	.
.	.	.	.
.	.	.	.
C26	End	2/16/2018 16:42	2/16/2018 16:42

3.2. Modeling Event Logs into the Graph Form Using Neo4j

Business processes based on event logs must first be modeled into a graph database using Neo4j. The first time you have to form an event log file from csv to the link list format. Use the algorithm found in Table 3 to model this. This program will model each activity contained in the file on CSV to the node form on Neo4j. After that each node that has been formed in the same case will be connected to the NEXT relation. The same activity in different cases will not form a second time.

Table 3. Pseudocode for Process Modeling of Event Logs

No.	Code
Load RMA	
1	LOAD CSV with headers FROM "file:///data_log_done.csv"
2	AS line
3	Merge(:Activity{CaseId:line.Case_ID, Name:line.Activity, StartTime:line. Start_Timestamp, EndTime:line.End_Timestamp })
4	LOAD CSV with headers FROM "file:///data_log_done.csv"
5	AS line
6	Merge (:CaseActivity {Name:line.Activity })
Connect The Activity	
1	Match (c:Activity)
2	WITH COLLECT(c) AS Caselist
3	UNWIND RANGE(0,Size(Caselist) - 2) as idx
4	WITH Caselist[idx] AS s1, Caselist[idx+1] AS s2
5	match (b:CaseActivity),(a:CaseActivity)
6	WHERE s1.CaseId = s2.CaseId AND s1.Name = a.Name AND s2.Name =
7	b.Name
	MERGE (a)-[:NEXT {relation:"NEXT"}]->(b)

After getting the model in the form of a node and its relation, the next task is to add the Control Flow Pattern to the model process by using the algorithm in Table 4. This program will form a XORSplit relation if the number of outgoing relations in the form of NEXT at the initial node is more than one, and the number of incoming and outgoing relations is NEXT on the destination node in the form of 1. On the contrary, XORJoin relation will be formed if the outgoing relation in the form of NEXT at the initial node is one, and the incoming relation in the form of NEXT on the destination node is more than 1.

After getting Control Flow Pattern, the next step is checking whether there is a non-free choice in the business process model. This step can be run by using the algorithm found in Table 5.

Table 4. Pseudocode Forms a Control-Flow Pattern on A Business Process Model

No.	Code
XORSplit	
1	MATCH (n)-[r:NEXT]->(a)
2	WHERE size((n)-->()) > 1 and ((size((a)<--()) = 1) OR
3	(size((a)-->()) = 1))
	CREATE (n)-[:XORSPLIT {relation:"XOR Split"}]->(a)
4	delete r
XORJoin	
1	MATCH (n)-[r:NEXT]->(a)
2	WHERE size((n)-->()) >= 1 and (size((a)<--()) > 1)
3	CREATE (n)-[:XORJOIN {relation:"XOR Join"}]->(a)
4	delete r

Table 5. Pseudocode Displays Non-Free Choices on Business Process Models

No.	Code
Non-free Choice	
1	match (a)-[b:XORJOIN]->(s)
2	match (s)-[c:XORSPLIT]->(n)
3	match (a)-[d:XORSPLIT]->(n)
4	create (a)-[:NONFREECHOICE]->(n)
5	delete d

3.3. SOP Modeling into Graph Forms Using Neo4j

The next step is to map the business process model based on Standard Operational Procedure (SOP) in graphical form using the Neo4j database application graph. Mapping in the form of this graph is used in the process of matching graphs. Table 6 shows the pseudocode to form an activity or node and relations between activities based on Standard Operational Procedure (SOP). After getting the model in the form of a node and its relation, the next step is executing steps in Table 4 and Table 5.

Table 6. Pseudocode of Standard Operational Procedure (SOP) Modeling into Neo4j Graph

No.	Create Aktivity (Node)	No.	Create Relation (Edge)
1	CREATE (A:RMA {id:1, description:'Start'}),	1	CREATE (A)-[:NEXT]->(B),
2	(B:RMA {id:2, description:'Request RMA'}),		(B)-[:NEXT]->(C),
3	(C:RMA {id:3, description:'Check Device Status'}),	2	(C)-[:NEXT]->(D),
4	(D:RMA {id:4, description:'Status Info'}),	3	(D)-[:NEXT]->(E),
5	(E:RMA {id:5, description:'Create a service receipt'}),	4	(E)-[:NEXT]->(F),
6	(F:RMA {id:6, description:'Fill Form Registration RMA'}),	5	(F)-[:NEXT]->(G),
7	(G:RMA {id:7, description:'Submit the Device to the technician'}),	6	(G)-[:NEXT]->(H),
8	(H:RMA {id:8, description:'Service'}),	7	(H)-[:NEXT]->(I),
9	(I:RMA {id:9, description:'Check Service Results via RMA Web'}),	8	(I)-[:NEXT]->(J),
10	(J:RMA {id:10, description:'Calculate the cost'}),	9	(J)-[:NEXT]->(K),
11	(K:RMA {id:11, description:'Make a Service Offer'}),	10	(K)-[:NEXT]->(L),
12	(L:RMA {id:12, description:'Running Billing SOP'}),	11	(L)-[:NEXT]->(M),
13	(M:RMA {id:13, description:'Make a DO RMA'}),	12	(M)-[:NEXT]->(Y),
14	(N:RMA {id:14, description:'Check Warranty'}),	13	(N)-[:NEXT]->(M),
15	(O:RMA {id:15, description:'Create Notice Damaged Device Form'}),	14	(I)-[:NEXT]->(N),
16	(P:RMA {id:16, description:'Customer Info'}),	15	(N)-[:NEXT]->(O),
17	(Q:RMA {id:17, description:'RMA to Vendor'}),	16	(O)-[:NEXT]->(P),
18	(R:RMA {id:18, description:'Replacement'}),	17	(P)-[:NEXT]->(M),
19	(S:RMA {id:19, description:'Check Stock RMA'}),	18	(P)-[:NEXT]->(Q),
20	(T:RMA {id:20, description:'Approval Replacement'}),	20	(Q)-[:NEXT]->(Y),
2	(U:RMA {id:21, description:'Fill the Device Request Form'}),	21	(N)-[:NEXT]->(R),
22	(V:RMA {id:22, description:'Make Job Costing'}),	22	(R)-[:NEXT]->(S),
23	(W:RMA {id:23, description:'Check Sale Stock '}),	23	(S)-[:NEXT]->(T),
24	(X:RMA {id:24, description:'Requesting the release of Stock'}),	24	(S)-[:NEXT]->(U),
25	(Y:RMA {id: 25, description:'End'})	25	(T)-[:NEXT]->(U),
		26	(U)-[:NEXT]->(V),
		27	(V)-[:NEXT]->(Y),
		28	(R)-[:NEXT]->(W),
		29	(W)-[:NEXT]->(T),
		30	(W)-[:NEXT]->(X),
		31	(T)-[:NEXT]->(X),
		32	(X)-[:NEXT]->(V)

3.4. Graph Matching

Referring to the calculation of graph-matching algorithm contained in the paper "Graph-Based Approach for Modeling and Matching Parallel Business Processes" [8], the first method is to use the functions that have been provided by Neo4j itself. That is by encrypting the process model and then the encryption results compared to using phonetic functions. Table 7 is an algorithm for performing the encryption process and Table 8 is the graph-matching algorithm.

Table 7. Pseudocode Encryption of Process Models

No.	Code
RMA SOP	
1	MATCH (n:RMA)-[r]->(m)
2	WITH collect(properties(n)) AS result
3	WITH apoc.util.md5(result) AS finalresult
4	RETURN finalresult
No.	Code
RMA EventLog	
1	MATCH (n:CaseActivity)-[r]->(m)
2	WITH collect(properties(n)) AS result
3	WITH apoc.util.md5(result) AS finalresult
4	RETURN finalresult

Table 8. Pseudocode Compares Graphs Using Phonetic Text Procedures

No.	Code
Phonetic function	
1	CALL apoc.text.phoneticDelta('String1', 'String2')

This program starts by modelling all the relationships found in the business process model and inserting it into a string "result". The string "result" will then be encrypted using md5 and saved to the string "finalresult". The "finalresult" results made from Business Process Model from Standard Operational Procedure (SOP) and Event Log will then be compared by carrying out phonetic functions Delta function on Neo4j. If the output is 4 then the result is very similar, whereas if you the output is 1 then the result is very far. The phonetic function itself has the disadvantage of only being able to compare identical process models.

This program starts by modeling all the relationships found in the business process model and inserting it into a string "result". The string "result" will then be encrypted using md5 and saved to the string "finalresult". The "finalresult" results made from Business Process Model from Standard Operational Procedure (SOP) and Event Log will then be by carrying out phonetic functions Delta function on Neo4j. If the output is 4 then the result is very similar, whereas if the output is 1 then the result is very far. However, the phonetic function itself has the disadvantage of only being able to compare truly identical process models.

The second method is to use a brute force algorithm in Java. To do graph-matching algorithm in this way, the first thing to do is export the data link list on Neo4j to the CSV format with the algorithm in Table 9. This program displays all the origin nodes, relations, and destination nodes, then exports them to CSV.

Table 9. Make CSV Files from Business Process Models

No.	Code
Create CSV	
1	START n=node(*) MATCH (n)-[r]->(m) RETURN n,r,m;

After the data is formed in CSV format, then run the Brute Force algorithm according to the algorithm below. There are 4 stages in the Brute Force Pattern Matching Algorithm. The first stage is initialization, in this stage importing all libraries needed and also initializing variables in the form of integer, double, string, and also initializing the location of the file to be read. The file is read in the form of export to CSV from the business process model that has been formed. Table 10 shows the pattern matching algorithm in the first stage.

The second stage is to enter the CSV file that is formed from the Business Process Model based on Standard Operational Procedure (SOP). At this stage the loop will be repeated continuously, each loop of the file will be read per line, then it will be separated every time there is a "," (comma) sign. The first separation result will be entered into the nodefrom1 string, the second separation is entered into relation1, and the third separation will be entered into the nodeto1 string. The loop stops when the line that is read from the file is null. Table 11 shows the pattern matching algorithm in the second stage.

The third stage is to enter the Comma-sepeated values (CSV) that is formed based on the event log. At this stage the loop will be repeated continuously, each loop of the file will be read per line, then it will be separated every time there is a "," (comma) sign. The first separation result will be entered into the nodefrom2 string, the second separation is entered into relation2, and the third separation will be entered into the nodeto2 string. The loop stops when the line that is read from the file is null. Table 12 shows the pattern matching algorithm in the third stage.

The fourth stage is comparing the results of the string that has been made in the second stage with the results of the string in the third stage. At this stage double loop is performed. The first loop to read the index string from the Standard Operational Procedure (SOP) results, the second loop to read the index string from the event log results. If between "nodefrom1" and "nodefrom2", "relation1" and "relation2", "nodeto1" and "nodeto2" are all the same, then the value of "match" will increase by one. After the looping ends the "match"

value will be shared with the largest number of rows and multiplied by 100%. The results will then be stored in the "percentage" variable and will be displayed. Table 13 shows the pattern matching algorithm in the fourth stage.

Table 10. Pattern Matching First Stage

```
1  import java.io.*;
2  import java.util.ArrayList;
3  public class Bufferread {
4          public static void main(String[] args) throws IOException
5          {
6                  String csvFile1 = "src/RMA_SOP.csv";
7                  String csvFile2 = "src/RMA_EventLog.csv";
8   String line1 = "";
9   String line2 = "";
10  String cvsSplitBy = ",";
11  int a=0;
12  int divide=0;
13  ArrayList<String> nodefrom1 = new ArrayList<String>();
14  ArrayList<String> nodeto1 = new ArrayList<String>();
15  ArrayList<String> relation1 = new ArrayList<String>();
16  ArrayList<String> nodefrom2 = new ArrayList<String>();
17  ArrayList<String> nodeto2 = new ArrayList<String>();
18  ArrayList<String> relation2 = new ArrayList<String>();
```

Table 11. Pattern Matching Second Stage

```
1  BufferedReader br = new BufferedReader(new FileReader(csvFile1));
2   while ((line1 = br.readLine()) != null) {
3          String[] node1 = line1.split(cvsSplitBy);
4   nodefrom1.add(node1[0]);
5   relation1.add(node1[1]);
6   nodeto1.add(node1[2]);
7   a++;
8   }
```

Table 12. Pattern Matching Third Stage

```
1  BufferedReader br2 = new BufferedReader(new FileReader(csvFile2));
2   while ((line2 = br2.readLine()) != null) {
3          String[] node2 = line2.split(cvsSplitBy);
4          nodefrom2.add(node2[0]);
5   relation2.add(node2[1]);
6   nodeto2.add(node2[2]);
7   a++;
8   }
```

Table 13. Pattern Matching Fourth Stage

```
1  a=nodefrom1.size();
2  int match = 0;
3  for (int i=0;i<a;i++){
4          for (int j=0;j<a;j++)
5          { if (nodefrom1.get(i).equals(nodefrom2.get(j)) && nodeto1.get(i).equals(nodeto2.get(j)))
                  { match += 2;
6                     if (relation1.get(i).equals(relation2.get(j)))
7           { match++; }
8      }
9                   if (nodefrom1.size() < nodefrom2.size())
10                  { divide = nodefrom1.size(); }
11                  else
12                  { divide = nodefrom2.size(); }
13           }
14         double percentage = match/(divide*3.0) * 100;
15         System.out.print(percentage+"\n");}
16
```

4. RESULT AND DISCUSSION

4.1. Business Process Modeling into the Graph Form

In the implementation of the establishment of the business process model of the event log. The method that has been done can be seen in Table 3, the first is entering the event log to Neo4j so that it is in the form of a link list. After forming a link list (node and relation), the next thing is to form a control flow pattern of the process model by using the algorithm contained in Table 4. In the next step is to check whether there is a non-free choice in the business process model using an algorithm that can be seen in Table 5. The results from business process modeling based on event log into graph form using Neo4j can be seen as shown in Figure 3. The process model uses abbreviations of the activities describing in Table 6.

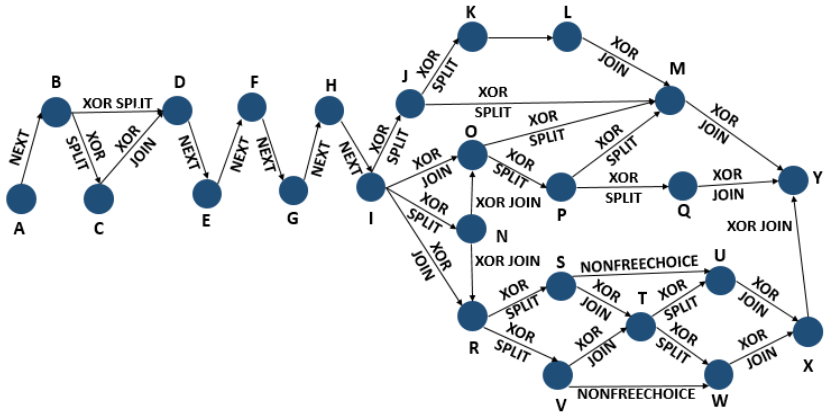


Figure 3. The event log modeling results into a Neo4j graph

The next relation is the establishment of a business process model of the Standard Operational Procedure (SOP). The method that has been done can be seen in Table 6, the first is to make the node and its relation. Next is to form a control flow pattern of the process model using the algorithm contained in Table 4. In the next step is to check whether there is a non-free choice in the business process model using the algorithm contained in Table 5. Results from business process modeling based on SOPs into graph shapes using Neo4j can be seen as Figure 4. The process model uses abbreviations of the activities describing in Table 6.

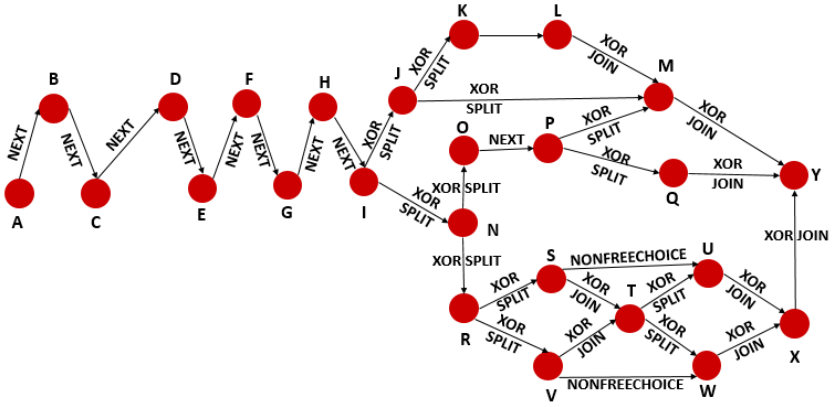


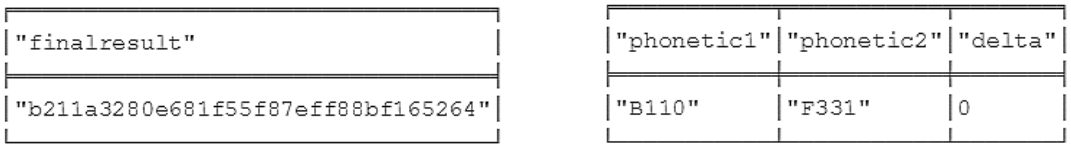
Figure 4. Standard operational procedure (SOP) modeling results into a Neo4j graph

4.2. Graph Matching

In this stage, the next step of the process business process is formed from the Standard Operational Procedure (SOP) and event log. The next thing to do is to compare the two graphs. To compare it, there are 2 methods, namely using the function on Neo4j and also using the brute force algorithm in Java. The first method is to encrypt a business process model. Figure 5(a) shows an example of the result of the encryption

using the algorithm found in Table 7. Final result is a string that stores the encryption results from a business process model. The results of the final result are displayed in the next line of Figure 4.

After performing the encryption process, a Neo4j function is called phonetic is used to compare the two encryption results from the business process model. Algorithm in Table 8 executes this process. The output will be in the form of numbers 1 to 4. The number 1 indicates different, and in contrast the number 4 indicates very similar. Figure 5(b) shows the results of phonetic functions.



(a) (b)

Figure 5. Result in Neo4j, (a) Result of SOP encryption, (b) Result of Phonetic function

The similarity obtained is 0 because this phonetic function can only accept graphs that have identical identities. If the delta value produces a number 0 then the two graphs are declared to have no similarity. Therefore to be able to get precise results then proceed with the second method using the brute force algorithm in java. First, the business process model must be changed first into the form of CSV format. The data taken are "origin node", "type of relation", and "destination node". Table 14 below is a cut of the results of the business process model changes from SOP to CSV. After forming the data node and its relation in CSV format. The data will then be used as input to run the Brute Force Pattern Matching Algorithm. The end result of the Pattern Matching algorithm is a percentage of the compatibility of the two process models. The percentage of graph matching is 76.76%.

Table 14. Results of the Business Process Model Changes into CSV Format

n.description	type(r)	m.description
Start	NEXT	Request RMA
Request RMA	NEXT	Check Device Status
.	.	.
.	.	.
.	.	.
Requesting the release of Stock	XORJOIN	Make Job Costing

4.3. Dice Coefficient Similairty

In using dice coefficient similarity, objects that are compared will be grouped first based on nodes and edges to be able to measure the similarities of business processes that are used as objects. Based on the results of grouping of 2 business process models that have been compared to produce 112 domains, and in the comparative business process model obtained 98 domains. Based on the two business process models, the intersected values are 73, so by using (1), the obtained similarity value is shown in (2).

$$S(A,B) = 2x \frac{|A \cap B|}{|A|+|B|}$$
$$S(A,B) = 2x \frac{73}{111+97} \quad (2)$$
$$S(A,B) = 0.70 = 70\%$$

5. CONCLUSION

This paper compares the graph-matching algorithm and dice coefficient similarity algorithm to analyze the ability of obtaining other information containing non-free choice, parallel relationships, and invisible tasks influences the gap result of business process models. Graph-matching algorithm is chosen because this algorithm can form a process model containing that information. In the graph matching phase, the two business process models are compared by taking all existing nodes and relations from a model process and comparing them with nodes and relations from other model processes.

The evaluation conducted in this study showed that the measurement of gap by using graph-matching algorithm on Neo4j in a similarity level is 76.76%. Then, the gap measurement using dice coefficient similarity produces a percentage of 70%. These values prove that graph-matching algorithm is superior because the percentage value generated from graph matching is higher than that from dice coefficient. The high value obtaining by graph matching algorithm shows that the ability of obtaining parallel relationships, invisible tasks, and non-free choice influences the gap value of business process models. In this study, matching patterns based on graph matching algorithm can only be used with event logs generated from CSV files. Future work that can be developed from this study is to do a gap analysis process model that has non-free choice conditions and raises the invisible task condition, so that the gap analysis process to find out the similarity of the process model can be obtained in full.

ACKNOWLEDGEMENTS

Authors give a deep thanks to God Almighty, Institut Teknologi Sepuluh Nopember, Directorate of Research and Community Service, Directorate General of Research and Development Strengthening Ministry of Research, Technology and Higher Education Republic of Indonesia for supporting this research.

REFERENCES

[1] Bordley RF. "Integrating Gap Analysis and Utility Theory in Service Research". *J Serv Res*. 2001;3(4): 300-309. doi:/10.1177/109467050134003.
[2] Anugrah IG, Sarno R. "*Business Process model similarity analysis using hybrid PLSA and WDAG methods*". *2016* Int Conf Inf Commun Technol Syst. 2016:231-236. doi:10.1109/ICTS.2016.7910304.
[3] Kuss E, Stuckenschmidt H. "Automatic Classification to Matching Patterns for Process Model Matching Evaluation". :1-14.
[4] Anugrah IG, Sarno R, Anggraini RNE. "*Decomposition using Refined Process Structure Tree (RPST) and control flow complexity metrics*". Proc 2015 Int Conf Inf Commun Technol Syst ICTS 2015. 2016:203-208. doi:10.1109/ICTS.2015.7379899.
[5] Sungkono KR, Sarno R. "*HMM for Discovering Intentional Process Model From Event Logs by Considering Sequence of Activities*". 4th Int Conf Electr Eng Comput Sci Informatics. 2017:1-6.
[6] Sungkono KR, Sarno R, Ariyani NF. "*Refining Business Process Ontology Model With Invisible Prime Tasks Using SWRL Rules*". 11th Int Conf Inf Commun Technol Syst. 2017:215-220.
[7] Saldivar J, Vairetti C, Rodríguez C, Daniel F, Casati F, Alarcón R. "Analysis and improvement of business process models using spreadsheets". *Inf Syst*. 2016;57:1-19. doi:10.1016/j.is.2015.10.012.
[8] Sarno R, Sungkono KR, Septiarakhman R. "Graph-Based Approach for Modeling and Matching Parallel Business Processes". *Int Inf Inst*. 2018;21(5):1603-1614.
[9] Dijkman R, Rosa M La, Reijers H a. "Managing large collections of business process models-Current techniques and challenges". *Comput Ind*. 2012;63(2):91-97. doi:10.1016/j.compind.2011.12.003.
[10] Cardoso J, Mendling J, Neumann G, Reijers HA. "A Discourse on Complexity of Process Models". :1-12.
[11] Weidlich M, Dijkman R, Mendling J. "The ICoP framework: Identification of correspondences between process models". *Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics)*. 2010;6051 LNCS:483-498. doi:10.1007/978-3-642-13094-6_37.
[12] Wen L, Wang J, Van Der Aalst WMP, Huang B, Sun J. "Mining process models with prime invisible tasks". *Data Knowl Eng*. 2010;69(10):999-1021. doi:10.1016/j.datak.2010.06.001.
[13] Jimenez S, Gonzalez FA, Gelbukh A. "Mathematical properties of soft cardinality: Enhancing Jaccard, Dice and cosine similarity measures with element-wise distance". *Inf Sci (Ny)*. 2016;367-368:373-389. doi:10.1016/j.ins.2016.06.012.
[14] Antonini A, Ferreira AM, Morasca S. "Software Measures for Business Processes".
[15] Becker M, Laue R. （29）2011Analysing Differences Between Business Process Similarity Measures.pdf.
[16] Dijkman R, Dumas M, García-Bañuelos L. "*Graph matching algorithms for business process model similarity search*". 7th Int Conf Bus Process Manag. 2009;Business P:48-63. doi:10.1007/978-3-642-03848-8_5.
[17] Bhatia A. "Graph Databases- An Overview". 2014;5(1):657-660.
[18] Pande IN, Dharmawan W, Sarno R. "Book Recommendation Using Neo4j Graph Database in BibTeX Book Metadata". 2017:47-52. doi:10.1109/ICSITech.2017.8257084.
[19] Wen L, Van Der Aalst WMP, Wang J, Sun J. "Mining Process Models with Non-Free-Choice Constructs". :1-32. http://wwwis.win.tue.nl/~wvdaalst/publications/p394.pdf.
[20] Sarno R, Sungkono KR. "*Coupled Hidden Markov Model for Process Discovery of Non-Free Choice and Invisible Prime Tasks*". Procedia Comput Sci Direct. 2018;124:134-141. doi:10.1016/j.procs.2017.12.139.
[21] Sungkono KR, Sarno R. "Constructing Cintrol-Flow Patterns Containing Invisible Task and Non-Free Choice Based On Declarative Model". *Int J Innov Comput Inf Control*. 2018;14(4):1285-1299. doi:10.24507/ijicic.14.04.1285.
[22] Klinkmüller C, Weber I. "Analyzing control flow information to improve the effectiveness of process model matching techniques". *Decis Support Syst*. 2017;100:6-14. doi:10.1016/j.dss.2017.06.002.
[23] Becker M. "Business Process Management Workshops". 2012;100(August 2011). doi:10.1007/978-3-642-28115-0.

[24] Dijkman R, Dumas M, Garcia-Banuelos L, Kaarik R. "*Aligning Business Process Models*". Proc 13th IEEE Int Conf Enterp Distrib Object Comput. 2009:40-48. doi:10.1109/EDOC.2009.11.
[25] Vanderfeesten I, Reijers HA, Mendling J, Aalst WMP Van Der, Cardoso J. "On a Quest for Good Process Models : The Cross-Connectivity Metric".

BIOGRAPHIES OF AUTHORS

Afrianda Cahyapratama is a student at the Department of Information Technology Management, Institut Teknologi Sepuluh Nopember, Indonesia. He received his Bachelor of Computer degree from the Informatics Engineering Department, Faculty of Computer Science, Brawijaya University in 2016. His research interest is Information Management and has an interest in Process Management, Software Analyst and Data Analyst. Email : afriandacahyapratama@gmail.com

Kelly Rossa Sungkono is a lecturer in Informatics Department, Institut Teknologi Sepuluh Nopember (ITS). Kelly received B.Sc and M.Sc from ITS. Her topics of researches are process mining and business process management. One of her papers is *Constructing Control-Flow Patterns Containing Invisible Task and Non-Free Choice Based on Declarative Model*. Email : kelly@its.ac.id

Riyanarto Sarno was born in Surabaya on August 1959. Riyanarto received M.Sc and Ph.D. in Computer Science from the University of Brunswick Canada in 1988 and 1992. He is a lecturer of Institut Teknologi Sepuluh Nopember (ITS) Surabaya. His research focuses on the internet of things, business process management, process-aware information, enterprise computing and smart grids. Email : riyanarto@if.its.ac.id