# Workflow Common Fragments Extraction Based on WSDL Similarity and Graph Dependency

Riyanarto Sarno[1], Endang Wahyu Pamungkas[2], Dwi Sunaryono[3], Sarwosri[4]
Informatics Department, Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
riyanarto@if.its.ac.id[1], endang.wahyu.pamungkas10@mhs.if.its.ac.id[2], dwi@its-sby.edu[3], sri@its-sby.edu[4]

*Abstract*— **Business process management technology has been developing and growing. In the industry, business processes continue to change over time. Changes occur due to consumer demand, technological innovation, and policy changes. Therefore, existing business processes should be able to adapt to any changes. Currently, the design phase is always ignored as the focused point in the business process development. The development process has only focused on the implementation. Hence, in this paper, we propose a method to form a reconfigurable business process model. The method consist of a series of sequential process. The process started with calculating similarity value of business processes based on its structure. Then, we perform the clustering process to the models based on similarity values. From every cluster, we extract the common fragment using dependency graph calculation. The experimental result shows that the proposed method can produce common fragments or reconfigurable models which can be rearranged to its variations. The common fragment has high flexibility and granularity. Research on reconfigurable models of business processes are still limited, therefore the result of this research can be used to start other researches.**

***Keywords—business process management, reconfigurable model, structure similarity, common fragment, dependency graph.***

## I. Introduction

In the industry, business process will continue to change over time. Changes occur due to consumer demand, technological innovation, and policy changes. So, the existing business process must be able to adapt to any changes. Generally, Business Process Management Suite (BPMS) already provides facilities to change business process [1]. However, the existing reference model should be easy to modify. So that, the cost to make changes can be minimized. Currently, the design phase is always excluded as the focused point of business process development. The development process has always been focused on the implementation. So, we need a method to perform analysis and design in the development of business process. The objective to produce flexible business process from design and analysis phase. This flexible business process can lighten the work of both vendor and consumer in composing a new business process. According to this problem, we will try to observe the methodology to build flexible business process models. Flexible business process models can be represented by a reconfigurable model. We propose a method which utilize the formula of dependency graph. First, we need to perform clustering process on set of business process models based on its similarity. Clustering process is done using the method in [2]. However, we make a modification to improve the accuration of similarity calculation by using Web Service Description Language (WSDL) Similarity. Therefore, this method can only be implemented in business process environment that use web service.

The first part of this paper discuss about the introduction of this research. Then we discuss about several related studies that has been conducted about this research. In the next part, we explain about similarity calculation of business process models. After that, we explain the method to extract reconfigurable models from several similar models. In the next section, we conduct an experiment to prove that the method can give important result to the business process management field. In the last section we present the conclusion from this research as well as some further research plans.

## II. Related Work

This section discuss about several related studies to this research. There are a lot of studies that discuss about the calculation of similarity between business process models. There are two widely used approaches to calculate it, namely structural and behavioral similarity. Based on its name, structural similarity is done by comparing every element of business process models. In calculating structural similarity, there are two frequently used methods. They are graph edit distance [3] [4] [5] [6] and node and edge based similarity [7] [8] [9] [10]. In behavioral similarity, there are also two methods that are based on causal dependency [11] [12] [13] and based on trace and logs [14] [15]. Both of these methods have advantages and disadvantages based on its use. Then, to calculate the similarity value of string, syntactic and semantically method are the two common method. In syntactic, we can use well-known methods, namely Levenshtein distance and string edit distance. For semantic we can use method in [3]. In this method, string similarity is obtained by searching for the synonym of each compared string.

## III. THE PROPOSED METHOD

The proposed method includes three steps as described in the following subsections.

### A. *Business Proses Similarity using WSDL Similarity*

There are several methods can be used to calculate the similarity value between business process model. There are three kinds of similarity between business process model amongst structural, behavioral, and contextual. In this study, we calculate similarity value among business process using structural similarity. Structural similarity is the similarity value that focused on comparing shape and structure. We adopt Jaccard Structural Similarity equation (1) to calculate structural similarity.

$$Sim = \frac{Total\ of\ similar\ element}{total\ of\ similar\ element + total\ of\ different\ element} \quad (1)$$

We use Petrinet as main notation of business process model. Petrinet has three main structure consist of transition, place, and edge. For transition and place, similarity was obtained by comparing their identifier. Source and sink node is compared to get the similarity value of edges. Afterwards, we calculate the distance of string identifier using the similarity value. But, in this study we propose a new method to calculate similarity of transition element. In every transition, we look deeper to the web services that have been called by the activity. Web service elements is in the WSDL extension file. Therefore, we proposed a method to compare every element in the WSDL file. There are several elements defined in the web service as follows:

- *Message*
- *Port*
- *Service*
- *Operation*

We compare string elements of web service definition to obtain similarity score between WSDL file. In addition to the four elements in Table I, we should also compare input and output messages. We use semantic method to compare string of every element. Furthermore, we need to calculate the average similarity score of each element to earn transition similarity. We also need to discuss about method to earn similarity value on message element. Message elements consist of input message and output message. Both input and output message contain variable name and variable type.

TABLE I. ELEMENT OF WSDL FILE

| | |
|---|---|
| **Message** | Input Message Name |
| | Output Message Name |
| | Variable Name and Variable Type |
| **Port** | Port Name |
| **Operation** | Operation Name |
| **Service** | Service Name |

There is special treatment in calculating string similarity of variable type. Complex type is a variable type that contains more than one atomic variable. It has same structure like array data structure. The example of similarity calculation between two complex types can be seen in the next section

Fig. 1 gives example of two business process fragments that contains two places, one transition, and two edges. In this case, we want to calculate the similarity value of transition. If we just compare their identifier, them their similarity score is 1. However, the result will be different if we look deeper to WSDL file has been called in this activity. Fig.2 is the example of WSDL on input and output message part

From Fig. 2, we know that two transitions with same

$$Sim = \frac{1 * sum\ of\ words\ +\ 0{,}75 * sum\ of\ synonym}{maximum\ words\ in\ each\ sentences} \quad (2)$$

identifier can call different web service. The first transition calls gettingActualItemBalance and the other one calls gettingPlanItemBalance. Subsequently, we need to compare the name of message and the variable. The name of message includes input and output message. We use WordNet to compare two strings semantically. But, we need to stem and tokenize those strings first before compare it. Semantic similarity formula to find similarity value between two strings can be seen on (2).

We got similarity score 0.75 for input message, 0.85 for output message, and 0.66 for variable. So, the similarity value for message element is 0.73. Here is the list of elements that should be compared. The final result of similarity value is average of similarity of all elements.

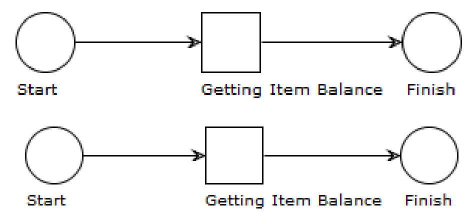
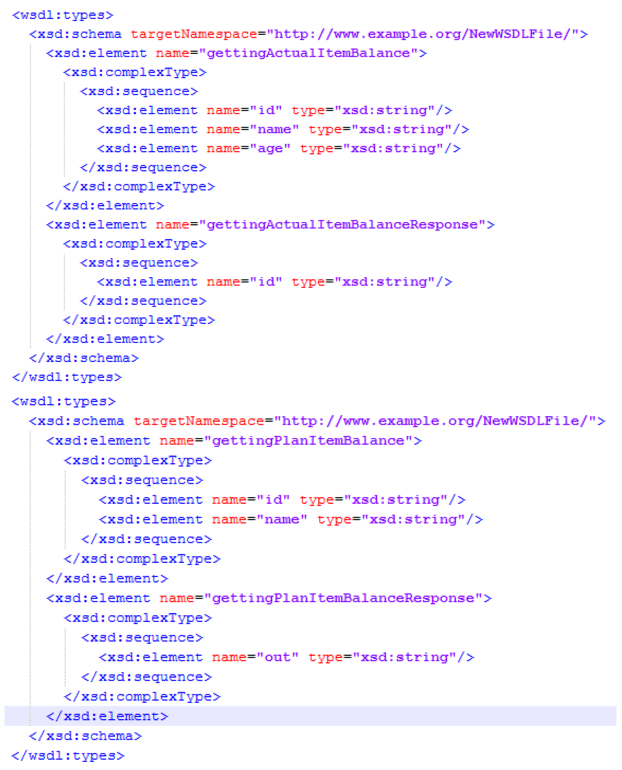


Figure 1. Comparing two transition



```
<wsdl:types>
  <xsd:schema targetNamespace="http://www.example.org/NewWSDLFile/">
    <xsd:element name="gettingActualItemBalance">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="id" type="xsd:string"/>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="age" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="gettingActualItemBalanceResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="id" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</wsdl:types>
<wsdl:types>
  <xsd:schema targetNamespace="http://www.example.org/NewWSDLFile/">
    <xsd:element name="gettingPlanItemBalance">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="id" type="xsd:string"/>
          <xsd:element name="name" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="gettingPlanItemBalanceResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="out" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</wsdl:types>
```

Figure 6. Comparing WSDL files

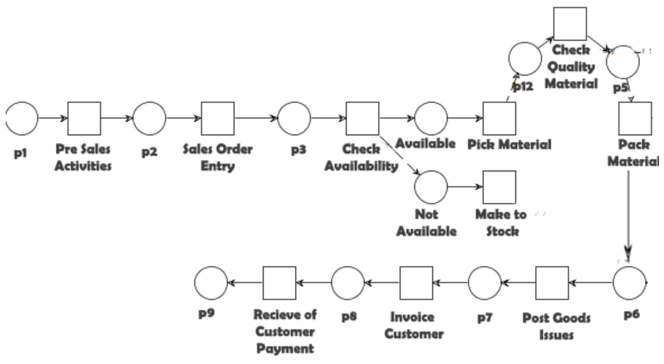


Figure 2. Business process variation 1

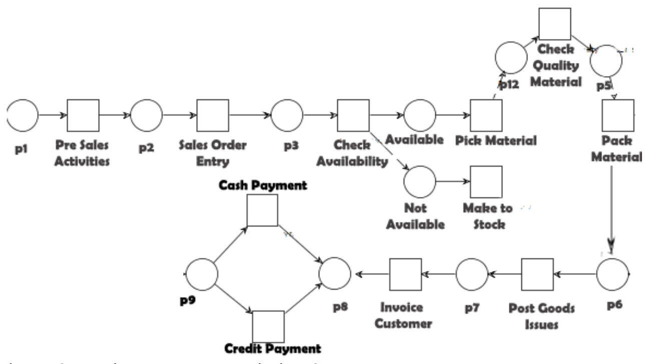


Figure 3. Business process variation 2

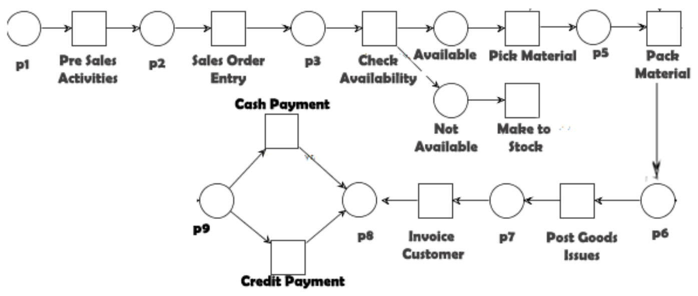


Figure 5. Business process variation 3

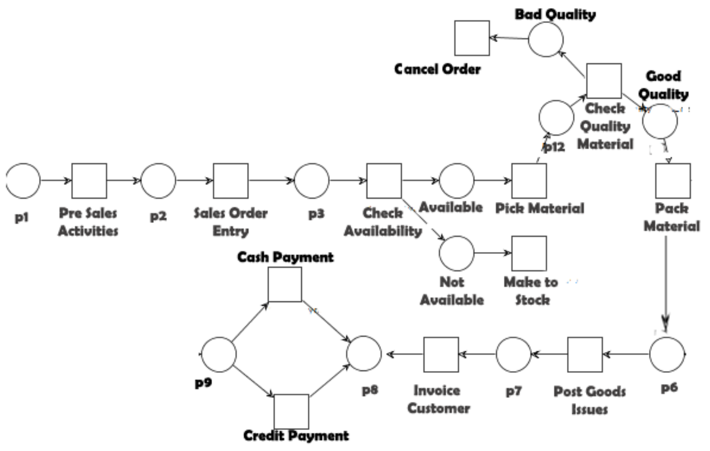


Figure 4. Business process variation 4

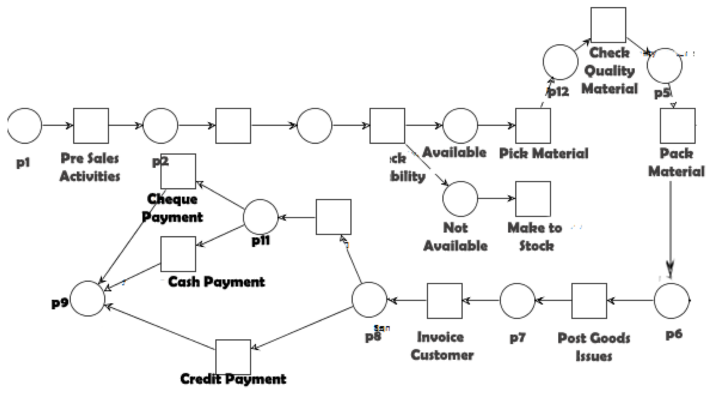


Figure 5. Business process variation 5

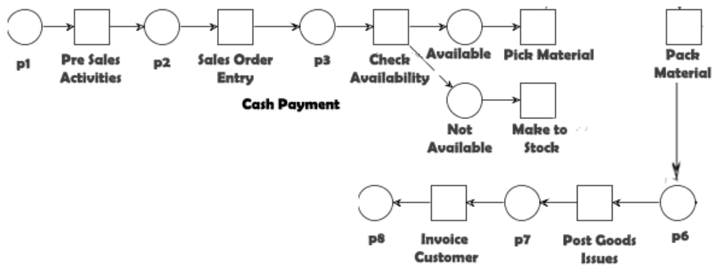


Figure 8. Main Business Process Fragment

### B. *Dependency Graph*

Dependency graph is a metric contains the occurrence frequency of relation between two events in the event log. The value is between -1 and 1. Heuristic Miner [17] is the first algorithm to introduce dependency graph. The first step to mining process using this algorithm is forming the dependency graph. The result from this process is a frequency based metric that explain how strong a relation between two events in a completed process. The formula to calculate dependency metric between events is on (3).

$$a \Rightarrow_W b = \left( \frac{|a >_W b| - |b >_W a|}{|a >_W b| + |b >_W a| + 1} \right) \qquad (3)$$

$a$ and $b$ are 2 events that are related to each other. $a \rightarrow w^b$ is the dependency value where event $a$ is followed by event $b$. $a > w^b$ is the frequency of event $a$ followed by event $b$. Otherwise, $b > w^a$ is the frequency of event $b$ followed by event $a$.

In process mining, dependency graph is used to find the strength of relation between 2 events. Noise and the real relation can also be found by using this metric. Noise is defined by comparing between dependency value and threshold value. The problem is finding the exact threshold to find noise. However, dependency graph can not detect branches in the model process. So that, Heuristic Miner algorithm add some extra calculations.

### C. *Common Fragment Extraction*

Common fragment or usually called the main business process is a part of process that contains similar part to all business process variations [1]. Common fragment is represented by a reconfigurable model. There are 2 purposes of extracting common fragment. First of all, it is used to make the process of storing business process more efficient. Business process management suite does not have to store all of variations that developed from same model. Secondly, it is facilitate the process of composing business process variation from the main business process. Therefore, a model designer does not have to make the model from scratch. It also can increase the reusability of a model.

The extraction process of reconfigurable model started by clustering process on business processes based on its similarity. The aim is to form a cluster consist of similar business processes. The same method in [2] is used to perform clustering process with a modification in the similarity calculation process. That research explains 2 methods to calculate similarity, structural and behavioral. But, this research only uses structural method and adds WSDL similarity calculation for each event. Groups of similar model that obtained from the previous process is then being calculated to produce dependency table. Relation of activities that has the highest dependency value in the dependency table is forming reconfigurable model. The highest dependency value can be calculated mathematically using formula (4).

TABLE II. DEPEDENCY TABLE PART I

| **Activities** | **Pre Sales Activity** | **Sales Order Entry** | **Check Availabili ty** | **Pick Material** | **Check Quality Material** | **Make to Stock** |
|---|---|---|---|---|---|---|
| Pre Sales Activity | | 0.83 | | | | |
| Sales Order Entry | | | 0.83 | | | |
| Check Availability | | | | 0.83 | | 0.83 |
| Pick Material | | | | | 0.80 | |
| Check Quality Material | | | | | | |
| Make to Stock | | | | | | |
| Pack Material | | | | | | |
| Post Goods Issues | | | | | | |
| Invoice Customer | | | | | | |
| Receive Customer Payment | | | | | | |
| Cash Payment | | | | | | |
| Credit Payment | | | | | | |
| Cancel Order | | | | | | |
| Direct Payment | | | | | | |
| Cheque Payment | | | | | | |

TABLE III. DEPEDENCY TABLE PART II

| **Activities** | **Pack Material** | **Post Goods Issues** | **Invoice Custome r** | **Receive Custome r Payment** | **Cash Payment** |
|---|---|---|---|---|---|
| Pre Sales Activity | | | | | |
| Sales Order Entry | | | | | |
| Check Availability | | | | | |
| Pick Material | | | | | |
| Check Quality Material | 0.80 | | | | |
| Make to Stock | | | | | |
| Pack Material | | 0.83 | | | |
| Post Goods Issues | | | 0.83 | | |
| Invoice Customer | | | | 0.50 | 0.75 |
| Receive Customer Payment | | | | | |
| Cash Payment | | | | | |
| Credit Payment | | | | | |
| Cancel Order | | | | | |
| Direct Payment | | | | | |
| Cheque Payment | | | | | |

$$\frac{total\ of\ dataset\ model}{total\ of\ dataset\ model + 1} \quad (4)$$

TABLE IV. DEPEDENCY TABLE PART III

| Activities | Credit Payment | Cancel Order | Direct Payment | Cheque Payment | Cash Payment |
|---|---|---|---|---|---|
| Pre Sales Activity | | | | | |
| Sales Order Entry | | | | | |
| Check Availability | | | | | |
| Pick Material | | | | | |
| Check Quality Material | | 0.50 | | | |
| Make to Stock | | | | | |
| Pack Material | | | | | |
| Post Goods Issues | | | | | |
| Invoice Customer | 0.75 | | 0.50 | | |
| Receive Customer Payment | | | | | |
| Cash Payment | | | | | |
| Credit Payment | | | | | |
| Cancel Order | | | | | |
| Direct Payment | | | | 0.50 | 0.50 |
| Cheque Payment | | | | | |

Fig. 3 until Fig. 7 are the example of main business process extraction. There are 5 business process model that can be categorized into one group. The highest dependency value for this group is 5/6 or 0.83. Hence, activity relation with the dependency value of 0.83 can be classified as main business process. The series of A-B, B-C, C-D, C-F, G-H, H-I is the main business process for this group as can be seen at Fig. 8. The calculation can be seen in the Table II until Table IV.

## IV. EXPERIMENT

In this experiment, we use 20 business process models as dataset. The models is taken from a running ERP business process flow. It was made using Petri Net notation with .pnml file extension. ERP application that used as case study has the WSDL file. We calculate similarity value of all models using structural similarity and WSDL similarity as described before. We obtain WSDL file from the implemented ERP application. The calculation process produce similarity metric of all business process models. The calculation results is used as the base to do clustering that adopt graph partition approach. The clustering result shows that the 20 models form 7 cluster and each cluster contains similar business process. After that, we take the common fragment from each cluster using dependency graph calculation. Formed common fragment contains shared activity from all the models in one cluster.

## V. CONCLUSION

In this paper we presented an improvement to do similarity calculation between business processes. We perform this improvement to cover the possibility of mistake when an activity has same name but different execution process. The key of our improvement is by using WSDL file to get the information of process that executed by an activity. So that, we can lessen the possibility of false positive. In addition, we also propose new method to extract common fragment in business process models repository. Common fragments are taken from every formed cluster based on similarity calculation. This process adopt the dependency graph method that is a part of process mining method. Common fragment is a main model that can be used as a base to form business process variation. The research about common fragment extraction is very new. So that, we hope our propose method in this research can be used to produce another use of common fragment in business process management field.

## ACKNOWLEDGEMENT

We would like to thank Institut Teknologi Sepuluh Nopember and DIKTI - Directorate General of Higher Education - Ministry of Education and Culture of Indonesia for supporting the research.

## REFERENCES

[1] M. L. Rosa, M. Dumas, R. Uba and R. Dijkman, "Business Process Model Merging: An Approach to Business Process Consolidation," *ACM Transactions on software Engineering and Methodology (TOSEM),* vol. 22, no. 2, 2013.

[2] R. Sarno, H. Ginardi, E.W. Pamungkas and D. Sunaryono, "Clustering of ERP Business Process Fragments," in the Proceeding of IEEE *International Conference on Ccomputer, Control, Informatics and Its Applications (IC3INA),* pp. 319-324, 2013.

[3] Dijkman, R. Dumas, M. v. Dongen, B. Kaarik and R. Mendling, "Similarity of Business Process Models: Metrics and Evaluation," *Information System,* vol. 36, pp. 498-516, 2011.

[4] Grigori, D. Corrales, J. C. Bouzeghoub and M. Gater, "A Ranking BPEL Process for Service Discovery," *IEEE Transactions on services computing,* vol. 3, pp. 178-192, 2010.

[5] Kunze and M. Weske, "Metrics Trees for Efficient Similarity Search in Process Model Repositories," in *Proceedings of the 1st International Workshop Process in the Large*, pp. 535-546, 2010.

[6] J. Bae, J. Caverlee, L. Liu and H. Yan, "Process Mining by Measuring Process Block Similarity," *In Proceedings of the business process management workshop,* vol. 4103, pp. 141-152, 2006.

[7] Z. Yan and Dijkman, "Fast Business Process Similarity Search with Feature-based Similarity Estimation," in *On the move to meaningful internet systems - confederated international conferences proceedings*, pp. 60-77, 2010.

[8] M. Minor, A. Tartakovski and R. Bergmann, "Representation and Structured-based Similarity Assessment for Agile Workflows," *&th International conference on Case-Based Reasoning,* pp. 224-238, 2007.

[9] K. Huang, Z. Zhou and Y. Han, "An Algorithm for Calculating Process Similarity to Cluster Open Source Process Designs," *Grid and Cooperative Computing 2004 Workshops,* pp. 107-114, 2004.

[10] R. Sarno, P.L.I. Sari and D. Sunaryono., "Decision Mining for Multi Choice Workflow Patterns." in the Proceeding of IEEE *International Conference on Ccomputer, Control, Informatics and Its Applications (IC3INA),* 2013.

[11] J. Bae, L. Liu and J. Caverlee, "Process Mining, Discovery, and Integration using Distance Measures," in *Proceedings of the IEEE International Conference on web Services*, pp. 479-488, 2006.

[12] J. J. Y, J. Bae and L. Liu, "Hierarchical Clustering of Business Process Models," *International Journal of Innovative Computing Information and Control,* pp. 1-11, 2009.

[13] H. Zha, J. Wang and L. Wen, "A Workflow Net Similarity Measure Based on Transition Adjacency Relations," *Computers in Industry,* vol. 61, pp. 463-471, 2010.

[14] K. Gerke, J. Kadorso and A. Claus, "Measuring the Compliance of Processes with Reference Models," in *in Proceedings of the Confederated International Conferences on the Move to Meaningful Internet System*, pp. 76-93, 2009.

[15] J. Wang, T. He and L. Wen, "A Behavioral Similarity Measure between Labeled Petri nets based on Principal Transition Sequences," in *Proceedings of the Confederated International Conferences On the Move to Meaningful Internet Systems*, pp. 394-401, 2010.

[16] Milani and Fedrik, "Modeliling Families of Business Process Variants: A Decomposition Driven Method," *arXiv preprint arXIv,* pp. 1311-1322, 2013.

[17] Weijters, A. J. M. M., Wil MP van Der Aalst, and AK Alves De Medeiros. "Process Mining with the Heuristics Miner Algorithm," *Technische Universiteit Eindhoven, Tech. Rep. WP* 166 (2006): 1-34.