STRUCTURAL SIMILARITY ANALYSIS BETWEEN PROCESS VARIANTS

Noor Mazlina Mahmod 1, Nor Haslinda Ismail 2, Wong Yan Chiew 3

1, 3 Faculty of Electronics and Computer Engineering, Universiti Teknikal Malaysia Melaka, Locked Bag 1752, Pejabat Pos Durian Tunggal, 76109, Durian Tunggal, Melaka, Malaysia.
2 Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka, Locked Bag 1200, Hang Tuah Jaya, 75450, Ayer Keroh, Melaka, Malaysia.
{mazlina 1, lynda 2, ycwong 3}@utem.edu.my

ABSTRACT

The emergence in business process management has brought up the significance discussion in issue of managing variance in process instances. Variance in business process can lead to various changes and modifications of business requirements, strategies and functionalities. It is evident that providing methods to analyse the similarities between these process variants to indicate a preferred and likely successful work practice can be highly beneficial. This paper proposes structural similarity analysis as a means for researching the similarity between the structure elements and the execution construct of process variants. The main focus is to match process variants against a user-defined process query, subsequently analyse and measure the structural similarity degree between them. The research is quite challenging as there are various types of structural relationships exist between the process variants. However, the systematic method introduced in this paper can be applied successfully to solve the ambiguity issue in defining and measuring the structural similarity between the process variant.

Keywords: Structural Analysis, Similarity Degree, Process Variants, Process Query

1 INTRODUCTION

Variant of business process is resulted from the changes in business strategies, constraints and the emergence of unexpected events. It is also an important information resource representing preferred work practices.

The variant or instance adaptation is an ongoing subject in business process management. It can lead to the production of a very large number of process variants. These variants reflect the awareness of process constraints and requirements which provide valuable knowledge and insight of work practice and also produce valuable feedback in business process design and redesign.

Evidence from studies and researches in the same research field has shown that request for generic method to identify the similarity relationship between variants is very encouraging. Hence, we would like to propose an effective means to analyse, structure and compute the structural similarity degree between the process variants in honour of this paper.

Although a number of notions for similarity analysis between process variants already exist, however, most of the notions applied only provide a binary answer (i.e. either two processes are similar or not) which is not very helpful in real life applications. In real life, it is imperative to identify the constraints between the process variants and to distinguish which process models are exactly similar, slightly similar (i.e. to what degree?) or completely different. Therefore, the main intention of this paper is to provide an essential deliberation towards finding an efficient approach of structural similarity degree analysis and computation in order to assist the search and retrieval of preferred work practice which satisfy a certain criteria defined by user.

More about motivation and related works for the research are provided in the subsequent section. As in the third section, we will discuss on the groundwork or foundation of this paper as well as recall briefly about the instance adaptation framework in [7] and Process Variant Repository (PVR) architecture in [5] for process variant management. Later, in fourth section, we will describe the notion and approach for structural similarity degree analysis and computation of process variants against a given process query. Afterwards, in the fifth section, we will present the result of the structural similarity computation and validate our presented approach and computation formula. Finally in the final section, we evaluate and conclude the paper as well as describe its noteworthy expansions for future research.
2 MOTIVATION

The idea and concept of structural similarity analysis is generally difficult and problematic since we have to deal with a variety of structural relationships (i.e. sequential, fork, synchronization, decision, merge etc.) between the process variants. Furthermore, the existence notions about the structural similarity analysis mainly in generic approach provide inadequate results to precisely distinguish the type and the degree of structural similarity between the variants.

Motivated by the currently inadequate generic approach and the importance of structural aspect as a foundation for every process design model, we are highly interested to find an effective means to analyse the similarity of basic structures between process variants and introduce a useful approach to compute the similarity degree. We believed that our method introduced in this paper can assist process designers to discern the type and the degree of structural similarity between process variants as to improve their decision in producing a better business process design in the future.

There are several related works that contribute to the research. One of them is a notable work for business processes that has been addressed in process equivalence in [2] and another one is an approach about similarity analysis between process variants presented in [5]. The process equivalence described in [2] provides detail information about the execution sequences to conduct the similarity analysis. The approach presented in [5] explains on how to detect semantic business process variants using ontology approach.

Furthermore, we also rely on the graph reduction technique proposed in [4] which presents five effective reduction rules (i.e. terminal, sequential, closed, adjacent and overlapped) to verify structural correctness of process variant. For improvement in process structure reduction, we also adopt the selective reduce technique in [6] to modify the graph reduction technique in [4] by eliminating only the task nodes in process variants that are not contained in the node set of process query. This technique allows the process variants to be reduced into a non empty graph that is similar or almost similar with the process query.

We have applied both graph reduction techniques and algorithms in [4] and [6] in this research to identify which process variants are exactly similar or slightly similar when being compared to a user-defined process query. Besides, we also modify and enhance the flows counting algorithm in [6] to develop and improve our structural similarity degree computation formulas as presented in the fourth section.

3 GROUNDWORK

Process variants are very complex objects as they may vary significantly due to the complexity of data required to describe them even though they satisfy the same set of constraints. As consequence, it is not easy and very complicated to capture the type and compute the degree of structural similarity between them. To treat this problem, we apply Petri nets approach and its tool to model and analyse the structural aspect between the process variants.

Petri nets have a mathematical foundation to check the algorithms of process model at various abstractions and can be used as a design language for the specification of complex process structure. Please note that we only make use of the basic terminology of Petri nets and its subclass, Free Choice Petri nets (FCPN) to model the process variants presented in this paper with the purpose of reducing the semantic complexity in modelling the process models. More background material for Petri nets can be referred in [1, 3, 9, 10]. Please see [11] and [12] for detail information about FCPN.

In addition, we have implemented the instance adaptation framework in [7] to ensure that variants creation is organized and querying process is feasible. We also adopt the PVR architecture in [5] as shown in Figure 1 as a structured storage that consists all of the previous process designs to smooth up the search and retrieval activities of process variants. Further information about the reference architecture of PVR can be referred in [5].
4 STRUCTURAL SIMILARITY ANALYSIS

There are a number of notions and theories available that have been used to define the structural relationship between process variants such as in process equivalence in [1] and process subsumes and transform relations in [4] and [6]. However, the problem about the degree of similarity still remains because it is difficult to measure the degree of slightly similarity between the structural elements and relationships exist in the process variants.

To solve the problem, we will introduce an effective means through our conceptual approach of structural similarity analysis as summarized in Figure 2. Throughout the remaining subsections, we will present and discuss how this approach is used to analyse, measure and compute the structural similarity degree between the process variants in a systematic way.

4.1 Process Variants Modelling

An example of business process model illustrated in Figure 3 demonstrates the flow of activities in a vendor performance assessment process. Task like registering, assessing performance and selecting a vendor are predictable and repetitive. Somehow, there are a number of assessment tests can be performed based on preferred order during the assessing vendor performance procedure. The tests (i.e. T1, T2, ..., Tn) represent the tests that will be executed as assessment test for the vendor. Each process variant prescribed uniquely for each case that has to be coordinated and controlled.

![Figure 3. Example of Business Process for Vendor Performance Assessment](image)

We have select a set of test (i.e. T1, T2, ..., T6) to represent one of the variant for test assessment during assessing vendor performance process. Then, we restructure and remodelled the example of business process in Figure 3 using Petri nets and FCPN approach as presented in Figure 4. We apply fundamental terminology of Petri nets to define our new model in Figure 4 as follows:

- Each task \( t \in T \) is mapped onto a place \( E_t \) and a transition \( C_t \).
- Each synchronizer \( s \in S \) and each task object \( x \in X \) such that \( x \ Trig s \), a place with the name \( \sigma_{x,s} \) is created. Synchronization achieved by creating a transition \( H_s \) which has all these places as input places and has as output places the places corresponding to the task objects triggered by that synchronizer.
- Each fork \( d \in D \) is mapped to a place \( E_d \) and has for each of its split \( e \in X \) an arc to a unique transition \( G_{d,e} \) which has an outgoing arc to \( E_e \).
- Finally the initial marking of the net is a marking with one token in each of the places \( E_i \) with \( i \) an initial item.

We also adopt the Definition 1 (Process Model) and Definition 2 (Process Variant) from [8] to define the schema for a process model and process variant. Nevertheless, we still use explicit...
definition and symbol for coordinator type as defined earlier since we will use Petri nets approach in our examples of process model and process variants. We also make use of the query definition in [8] to describe our user-defined process query that will be structurally compared with the process variants.

4.2 Process Variants Structuring

The process variants potentially can be very large and in consequent they are getting harder to be managed properly. In view of this fact, we propose a pre-processing step through which variants will be restructured or filtered. To perform this, we apply the rules from graph reduction technique in [4] and selective reduce technique in [6] to reduce the variants into graphs consisting only the tasks present in the process query, but still preserve the original structure of the process variant.

We have produced the following algorithm to reduce and restructure the process variants until it become similar as the process query. This algorithm is a modification from both graph reduction [10] and selective reduce [8] algorithms which mapped into Petri nets approach.

Algorithm 1: Process Variants Structuring Algorithm

Input graph G
Output reduced graph G
procedure REDUCE(G)
lastsize ← size[G] + 1
while lastsize > size[G] do
    lastsize ← size[G]
    /* Terminal, sequential and adjacent reduction */
    for each task t ∈ T(G) do
        if din[t] + dout[t] ≤ 1 then
            delete t
        else if din[t] = 1 and dout[t] = 1 then
            toTask[task(top[InFlows[t]]]} ← top[OutTasks[t]]
            delete t
        else if din[t] = 1 and dout[t] > 1 and taskType[t] =
            taskType[top[InTasks[t]]] then
            for each flow f ∈ OutFlows[t] do
                fromTask[f] ← top[InTasks[f]]
                delete t
        else if dout[t] = 1 and din[t] > 1 and taskType[t] =
            taskType[top[OutTasks[t]]] then
            for each flow f ∈ InFlows[t] do
                toTask[f] ← top[OutTasks[f]]
                delete t
    /* Closed reduction */
    if lastsize ≤ size[G] then
        for each task t ∈ T(G) do
            if dout[t] > 1 then
                TaskSet ← t
                for each flow f ∈ OutFlows[t] do
                    if taskType[f] = taskType[toTask[f]]
                        then
                        if toTask[f] ∈ TaskSet then
                            TaskSet ← TaskSet ∪ { toTask[f] }
                        else
                            delete f
                /* Overlapped reduction */
        if lastsize ≤ size[G] then
            for each tasks t ∈ T(G) do
                if taskType[t] = FS and dout[t] = 1 and din[t] > 1
                    then
                        level1 ← top[OutTasks[t]]
                        t ← top[InTasks[t]]
                    if taskType[level1] = TASK and din[level1] > 1
                        and taskType[t] = TASK and dout[t] > 1 and din[t] = 1
                            then
                                level1 ← top[InTasks[t]]
                                if level1 = FS and dout[level1] > 1 then
                                    Level2 ← InTasks[t]
                                    Level3 ← OutTasks[t]
                                    if ∀ task ∈ Level2 ( taskType[task] = TASK and
                                        InTasks[task] = { level1 ] and OutTasks[task] = Level3 )
                                        then
                                            if ∀ task ∈ Level3 ( taskType[task] = FS
                                                and OutTasks[task] = { level1 ] and
                                                InTasks[task] = Level2 )
                                                then
                                                    fromTask[top[OutFlows[t]]] ← level1
                                                    delete all task ∈ Level2
                                                    delete all task ∈ Level3

Figure 5 provide an example of process query, Q and a process variant, V. The following Figure 6 demonstrates the process restructuring results of V, using the presented algorithm.
Structural Similarity Analysis Between Process Variants

Noor Mazlina Mahmod

ISSN 2085-1944

Figure 5. Example of Process Query, Q and Process Variant, V

Figure 6. Process Variant, V Restructuring

Variants may have different levels of similarity with the given query. In previous example, process variant, V is slightly similar to process query, Q. Several structure elements (i.e. sequential tasks, a fork and a synchronizer) in V model are reduced until V become similar as Q.

In our approach, we classify the type of structural similarity based on the similarity and dissimilarity of structural elements between the process variants. If all the tasks and structure construct are exactly similar as the process query, then, it is classified as exact similar but if only some tasks or structures are similar, then it is classified as slightly similar. Detail classification for slightly similarity is required to precisely identify the type and the degree of similarity between the structure aspect of process variants and a given process query. Figure 7 provides an overview of the structural similarity classification in a hierarchical order.

We concern more on slightly similar class for structural similarity than the exact similar class because the slightly similar class represents the process variants that have partial similarity with process query and the variants are classified into several specific classes based on the type of structural similarity criteria they have with a defined process query. On the other hand, the exact similar class just represents the process variants that have exactly the same task and structure construct as process query. In this class the variants is completely as same as the defined process query.

Figure 7. Overview of Structural Similarity Classification

In our approach, we have classified four specific classes to identify the different type of structural similarity for slightly similar category. The first class known as Exact Task and Same Construct (ETSC) grouped the process variants which have the entire task similar with the tasks in process query and similar structure construct after gone through the restructuring procedure as shown in Algorithm 1. Another class called Exact Task and Different Construct (ETDC) is almost similar with ETSC class but the structural construct of the process variants are different from the process query. Two other classes are known as Slightly Similar Task and Same Construct (STSC) and Slightly Similar Task and Different Construct (STDC) are a bit different from ETSC and ETDC as the tasks in process variants are not completely as same as the process query after restructuring no matter either the structural construct is similar or not.

4.3 Structural Similarity Computation

In this section, we propose a computation or formula scheme based on selected basic structural elements (i.e. the extra and missing task, fork and synchronizer) to demonstrate a simple preliminary result for structural similarity analysis. Figure 8 describes the flow chart of structural similarity computation applied in our approach.
With the intention to improve the computation formula, we make use of the justification rationale of dissimilarity weight assigned to every type of structural elements [8] which is not similar with the structure elements in process query. As a starting point, we have come out with a simple but effective computation formula that can be implemented for slightly partial structural similarity. The formulas are as follows:

**Formula 1: Structural Similarity Computation for Different Structure Elements**

1) **matchFlow:**
   
   for each task \( t \in T[P] \), \( taskType[t] \in \{\text{task, coordinator}\} \) do
   
   if \( InFlows[t] \in F[Q] \) then
     count ← count + 1
   end if
   
   if \( OutFlows[t] \in F[Q] \) then
     count ← count + 1
   end if
   
   matchFlow = \( 100\% \times (count / F[P]) \)

2) **matchTask:**
   
   if \( T[P] \cap T[Q] > 0 \)
     matchTask = \( \#(\text{matchFlow}) / \#T[P] \) * 100%
   end if

3) **extraTask:**
   
   if \( taskType[t] = \text{task} \)
     extraTask = \( \#(T[P]-T[Q]) * 0.5 / |T[P]| \)
   end if

4) **extraFork:**
   
   if \( taskType[t] = \text{coordinator and coordinatorType[t] = fork} \)
     extraFork = \( \#(T[P]-T[Q]) * 0.8 / |T[P]| \)
   end if

5) **extraSync:**
   
   if \( taskType[t] = \text{coordinator and coordinatorType[t] = synchronizer} \)
     extraSync = \( \#(T[P]-T[Q]) * 1.0 / |T[P]| \)

6) **missingTask:**
   
   for \( T[Q] - T[P] \) do
     if \( taskType[t] = \text{task} \)
       missingTask = \( \#(T[Q]-T[P]) * 1.5 / |T[Q]| \)
     end if
   end if

7) **missingFork:**
   
   if \( taskType[t] = \text{coordinator and coordinatorType[t] = fork} \)
     missingFork = \( \#(T[Q]-T[P]) * 1.8 / |T[Q]| \)
   end if

8) **missingSync:**
   
   if \( taskType[t] = \text{coordinator and coordinatorType[t] = synchronizer} \)
     missingSync = \( \#(T[Q]-T[P]) * 2.0 / |T[Q]| \)
   end if

It will be more professional if we could rank the structural similarity degree based on the structural similarity class described earlier. The result should be in a logic manner (i.e. the highest rank should be the exact similar, followed by ETSC, ETDC, STSC and finally STDC). The ranking formula for each structural similarity classification is as follows:

**Formula 2: Structural Similarity Computation for Different Structure Elements**

For Total Match = matchFlow

For Partial Match:

**ETSC** = \( (matchFlow + matchTask) - ((extraTask + extraFork + extraSync) * (#(T[P]-T[Q]) / (T[P] *100%)) \)

**ETDC** = \( (matchFlow + matchTask) - ((extraTask + extraFork + extraSync) * (#(T[P]-T[Q]) / (T[P] *100%)) - ((missingFork + missingSync) * (#(T[Q]-T[P]) / (T[Q] *100%)) \)

**STSC and STDC** = \( (matchFlow + matchTask) - ((extraTask + extraFork + extraSync) * (#(T[Q]-T[P]) / (T[Q] *100%)) - ((missingTask + missingFork + missingSync) * (#(T[Q]-T[P]) / (T[Q] *100%)) \)

5 RESULT

This section demonstrates the preliminary result using the overall techniques and computation formula from our conceptual approach of structural similarity analysis.

The result for exact similar should be 100% since every task and structure construct between the process variants from this class are exactly similar to process query. Meanwhile for slightly similar case, we will apply the simple computation formula.
introduced in previous sub section 4.3 to compute the different type of structural similarity degree between the variants. To demonstrate this, we provide a process query, Q1 as a user-defined process query. The Figure 9 shows the structure model of Q1.

Figure 9. Example of Process Query 1, Q1

We also present five process variants as shown in Figure 10 (i.e. V1, V2 and V3) and in Figure 11 (i.e. V4 and V5). These variants will be structurally compared with Q1 using our effective approach. Please note that all examples are focus on slightly similarity computation only because the rank computation for exact similar is always 100%.

Figure 10. Example of Process Variants, V1, V2 and V3

Table 1 presents a set of preliminary result of structural similarity degree computation and ranking between V1, V2, ..., V5 against Q1 using our approach for structural similarity analysis and computation formula.

<table>
<thead>
<tr>
<th>Process Variant</th>
<th>Classification of Structural Similarity</th>
<th>Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>ETSC</td>
<td>83.59%</td>
</tr>
<tr>
<td>V2</td>
<td>ETDC</td>
<td>44.28%</td>
</tr>
<tr>
<td>V3</td>
<td>STDC</td>
<td>22.04%</td>
</tr>
<tr>
<td>V4</td>
<td>STSC</td>
<td>28.86%</td>
</tr>
<tr>
<td>V5</td>
<td>STDC</td>
<td>13.47%</td>
</tr>
</tbody>
</table>

Based on the rank result in Table 1, process variant V1 carries the highest structural similarity rank which is 83.59% followed by V2, V4, V3 and lastly V5 which carries 13.47%. The reason that drive the above ranking result can be observed from the reduction algorithm and computation formula demonstrated in the previous section 4. Through observation, the overall result is sensible according to their type of structural similarity classification.

6 EVALUATION AND CONCLUSION

Throughout this paper, we have presented an effective approach to analyse and measure the structural similarity between process variants in a well-structured manner. We have proposed an improvement for process reduction algorithm and computation formula to produce a reasonable
ranking percentage for the structural similarity degree of process variants.

The notions and approach presented in this paper in particular has potential for further assisting the similarity analysis in the future as it could be an alternative reference method. Moreover, the results from the proposed method can improve the process design and redesign in the future ensuring organization wide consistency. Furthermore, we intend to refine and enhance the presented approach and computation formula with the intention to make it more reliable and intuitive to be applied in a larger framework with diverse dimension of process variants.

REFERENCES


