Process Mining, Discovery, and Integration using Distance Measures

Joonsoo Bae
Chonbuk National Univ.
South Korea
jsbae@chonbuk.ac.kr

Ling Liu
Georgia Institute of Technology, USA
lingliu@cc.gatech.edu

James Caverlee
Georgia Institute of Technology, USA
caverlee@cc.gatech.edu

William B. Rouse
Georgia Institute of Technology, USA
bill.rouse@isye.gatech.edu

Abstract

Business processes continue to play an important role in today's service-oriented enterprise computing systems. Mining, discovering, and integrating process-oriented services has attracted growing attention in the recent year. In this paper we present a quantitative approach to modeling and capturing the similarity and dissimilarity between different process designs. We derive the similarity measures by analyzing the process dependency graphs of the participating workflow processes. We first convert each process dependency graph into a normalized process matrix. Then we calculate the metric space distance between the normalized matrices. This distance measure can be used as a quantitative and qualitative tool in process mining, process merging, and process clustering, and ultimately it can reduce or minimize the costs involved in design, analysis, and evolution of workflow systems.

1. Introduction

With the increasing interest and wide deployment of web services, we see a growing demand for service-oriented architectures and technologies that support *enterprise transformation*. Effective enterprise transformation refers to strategic business agility in terms of how efficiently an enterprise can respond to its competitors and how timely an enterprise can anticipate new opportunities that may arise in the future. In the increasingly globalized economy, enterprises face complex challenges that can require rapid and possibly continual transformations. As a result, more and more enterprises are focused on the strategic management of fundamental changes with respect to markets, products, and services [14]. Such transformation typically has a direct impact on the business processes of an enterprise. Enterprise transformation may range from traditional business process improvement to wholesale changes to the processes supported by the enterprise – from performing current work in a new fashion to performing different work altogether. Each of these challenges may lead to a differing degree of enterprise transformation.

Fundamental to enabling the transformation of an enterprise is the development of novel tools and techniques for *transforming the business processes* of an enterprise. In this paper, we present a critical component to the problem of process transformation from a web services point-of-view. In particular, we present a novel process difference analysis method using distance measures between process definitions of two transactional web services. The process difference analysis focuses on process structure and process activity dependencies to identify distance measures between processes.

The proposed difference analysis method achieves three distinct goals. First, by analyzing the attributes of process models, we present a quantitative process similarity metric to determine the relative distance between process models. This facilitates not only the comparison of existing process models with each other, but also provides the flexibility to adapt to changes in existing business processes. Second, the proposed method is quick and flexible, which reduces the cost of both the analysis and design phases of web service processes. Third, the proposed method enables the flexible deployment of process mining, discovery, and integration – all key features that are necessary for effective transformation of an enterprise.

2. Web Service Process Reference Model

The web service process reference model consists of business process definitions and the specification of workflows among the processes with respect to data flow, control flow, and operational views [15, 16]. We define a business process in terms of business activity patterns. An activity pattern consists of objects, messages, message exchange constraints, preconditions and postconditions [17], and is designed to specify the service actions and execution dependencies of the business process. An activity pattern can be viewed as a web service process when it is executable as a web service. We consider two types of activity patterns –

elementary activity patterns and composite activity patterns [1, 6]. An elementary activity pattern is an atomic unit. A composite activity pattern consists of a one or more elementary activity patterns or other composite activity patterns. The dependencies could capture complex interactions between activities.

We define a business process as a collection of business activities connected by data flow and control flow, where each represents a business process. A process definition can be seen as a web service (or a collection of web services). We use data flow among processes to define the data dependencies among processes within a given business process. We use control flow to capture the operational structure of the business process service, including the process execution ordering, the transactional semantics and dependencies of the process. A number of workflow specifications have gathered attention, including BPEL4WS (BEA, IBM, Microsoft), WSFL (IBM), XLANG (Microsoft), and XPDL (WfMC) [17]. In our prototype development, we choose to use a variant of BPEL4WS.

Formally, each workflow service is specified in terms of process definitions. We can model each process definition using a directed graph, in which the nodes of the graph are activities. Depending on whether the edges indicate execution dependencies or data flow dependencies, we have a process aggregation hierarchy or a process dependency graph. The process aggregation hierarchy captures the hierarchical execution ordering of activities. The process *dependency graph* captures information about how activities share information and how data flows from one activity to another. Due to the space constraint, in this paper we focus our discussion only on the dependency graph. Concretely we present the process similarity measures based on the dependency graphs of the processes of interest.

Definition 1 (Dependency Graph, *DG*)
A dependency graph *DG* is defined by a binary tuple <*DN, DE*>, where

- $DN = \{nd_1, nd_2, ..., nd_n\}$ is a finite set of activity nodes where $n \geq 1$.
- $DE = \{e_1, e_2, ..., e_m\}$ is a set of edges, $m \geq 0$. Each edge is of the form $nd_i \rightarrow nd_j$. ■

Note that in the dependency graph formulation, self-edges are disallowed since edges are intended to denote data flow dependencies between different activities (nodes). Additionally, a dependency graph must be a connected graph. Unconnected nodes and isolated groups of nodes are disallowed in the graph, as isolated nodes or groups of nodes are considered a separate service process in our reference model.

As a real-life example of business process, there are many PIPs (Partner Interface Processes) as defined by RosettaNet[13]. PIPs define business processes between trading partners. PIPs fit into seven Clusters, or groups of core business processes, that represent the backbone of the trading network. Each Cluster is broken down into Segments and within each Segment are individual PIPs. RosettaNet standards provide the infrastructure for integrating business processes with trading partners across the globe, delivering essential value to industries and proven real-world business results. Fig. 1 shows a standard process of procurement order by buyer, which is in Segment 3A(Quote and Order Entry) of Cluster 3(Order Management). This example process has 13 activities and their dependencies.

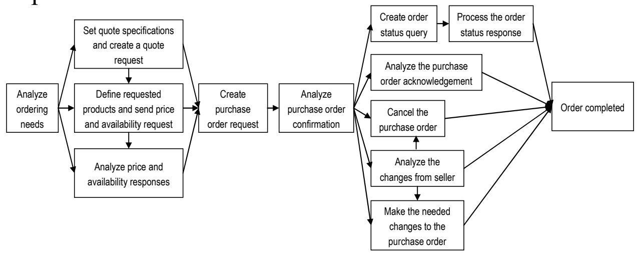

Analyze ordering needs
Set quote specifications and create a quote request
Define requested products and send price and availability request
Analyze price and availability responses
Create purchase order request
Analyze purchase order confirmation
Create order status query
Process the order status response
Analyze the purchase order acknowledgement
Cancel the purchase order
Analyze the changes from seller
Make the needed changes to the purchase order
Order completed

Fig. 1 **A real-life example of business process**

Given two processes and their respective dependency graphs, there are numerous ways these two graphs may differ. Typically, it makes more sense to compare only those graphs that have sufficient similarity in terms of their dependency graphs. Consider two extreme cases: one is when the two dependency graphs have the same set of nodes and the other is when there is no common node between two graphs. By assigning 1 for the first case and 0 for the latter case, we define a comparability measure that indicates the ratio of common nodes in two graphs. One way to measure the extent of comparability between two graphs is to use a user-controlled threshold, called δ-Comparability, which is set to be between 0 and 1. Because this value represents the ratio of common nodes over the union of all nodes in two graphs, the larger the value is, the greater degree of comparability between the two graphs. Note that δ value can not be 0 since δ = 0 means that there is no common node between two graphs, i.e., $DN_1 \cap DN_2 \neq \varnothing$.

Definition 2 (δ-Comparability of *DG*)
Let $DG_1 = (DN_1, DE_1)$ and $DG_2 = (DN_2, DE_2)$ be two dependency graphs, and δ be a user-defined control threshold. We say that DG_1 and DG_2 are *δ-*

comparable if the condition $\frac{|DN_1 \cap DN_2|}{|DN_1 \cup DN_2|} \geq \delta$ holds, where $0 < \delta \leq 1$ ■

If we apply the δ-Comparability to the example graphs shown in Fig. 2 with δ=0.5, g^0 and f^2 are not comparable because the number of common nodes is only one but the number of total nodes is 7, that is $\frac{|DN_1 \cap DN_2|}{|DN_1 \cup DN_2|} = \frac{1}{7} < 0.5$. On the other hand, g^0 and g^2 are δ-comparable because there are 3 common nodes and the total number of nodes is 5, thus the two graphs satisfy the δ-comparability condition $\frac{|DN_1 \cap DN_2|}{|DN_1 \cup DN_2|} = \frac{3}{5} \geq 0.5$ and $\delta = 0.5$.

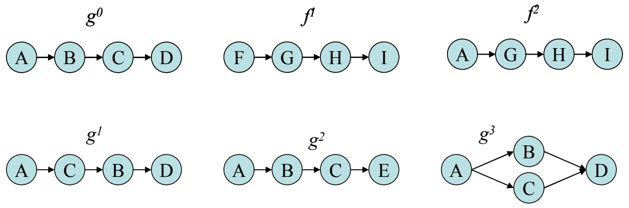


Fig. 2 **Examples of δ-Comparability**

3. Motivating Scenarios

Given the process reference model, we consider two motivating scenarios that benefit from the difference analysis methodology introduced in this paper. Consider a scenario where a company has maintained a warehouse of existing processes used in various business locations. *Process mining*[2, 3] of the process warehouse can help the enterprise to discover interesting associations or classifications among business processes running at different locations or branches of the company.

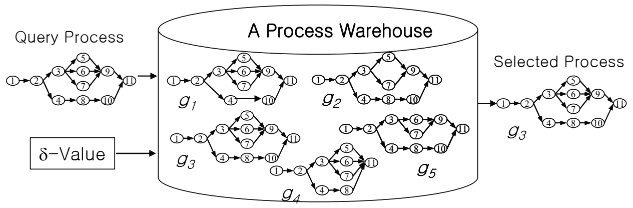


Fig. 3 **Process mining example**

In Fig. 3, we show a process warehouse that contains many types of processes (for example, g_1, g_2, g_3, g_4, g_5). A typical process mining scenario is the identification of the processes most similar to a query process template in the process warehouse. Given a query process and a comparability threshold δ-value, the process mining will identify (g_3) as the process that is most similar based on the comparability criterion. It is obvious that the concept of process similarity (or distance) is critical to the effectiveness of process mining.

4. Process Difference Analysis

In this section, we present the process difference analysis method for evaluating the distance between two processes. We first define the concept of a process matrix and introduce the concept of a normalized matrix. And then, we define the dependency distance measure by measuring the difference between the normalized matrices.

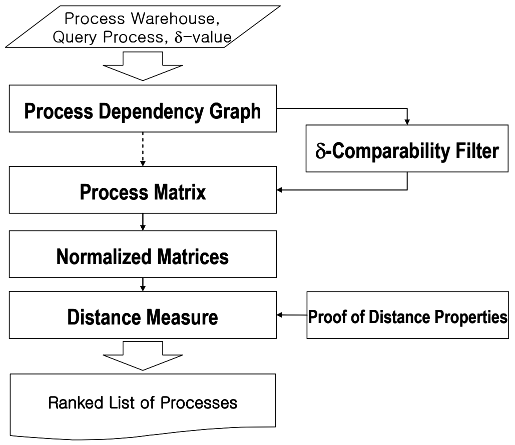


Fig. 4 **Flow chart of Difference Analysis**

In order to show the proposed procedure, we use two derived processes that are variations of procurement order process in Fig. 1. These two processes have 10 activities respectively but have different activities with each other. The first process (g_{11}) has A_6 but does not have A_8, and the second process (g_{22}) has A_8 but does not have A_6. These two graphs satisfy δ-Comparability as $\frac{|DN_1 \cap DN_2|}{|DN_1 \cup DN_2|} = \frac{9}{11} \geq 0.5$ and $\delta = 0.5$.

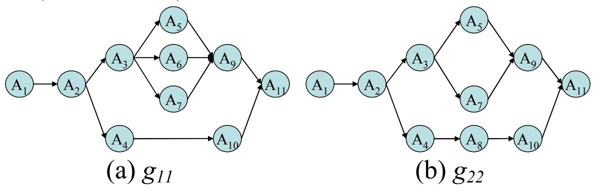


Fig. 5 **Two extended examples of Fig. 1**

4.1. Comparison Matrices

Two dependency graphs are said identical if the two graphs have the same set of nodes and the same set of edges. Formally we define identical dependency graphs as follows:

Definition 3 (Identical dependency graphs)

Let $DG_1 = (DN_1, DE_1)$ and $DG_2 = (DN_2, DE_2)$ be two dependency graphs. We say that DG_1 and DG_2 are identical if the two graphs have the same set of nodes and the same set of edges.

i) $DN_1 =_{Set} DN_2$ ii) $DE_1 =_{Set} DE_2$ ■

One way to compare and rank a set of similar process definitions is to transform each dependency graph into a numerical representation. This allows us to compare the dependency graphs using similarity distance in Euclidian distance metric space. This leads us to introduce the concept of a process matrix. A process matrix *M* is established in order to describe the precedence dependencies between two activities (tasks). The size of *M* is determined by the number of nodes in the dependency graph and each cell in the matrix denotes an element of *M.* The value of cell *M*(*i,j*) is set either to 1 or 0 depending on whether or not there is a precedence dependency between the two nodes *i* and *j*.

Definition 4 (Process matrix, *M*)

Let $g = (DN, DE)$ be a dependency graph with $|DN| = n$ nodes. A process matrix *M* of *g* is *n*-by-*n* matrix with *n* rows and *n* columns, and each row is named after the node name. Let $M_g(i,j)$ denote the value of the ith row and the jth column in *M,* $1 \le i, j \le n$. We define $M_g(i,j)$ as follows:

$$M_g(i,j) = \begin{cases} 1 & \text{if } \exists\, nd_i, nd_j \in DN \text{ such that } (nd_i, nd_j) \in DE \\ 0 & \text{else} \end{cases}$$

■

Fig. 6 depicts the transformation of a process dependency graph g_{11} shown in Fig. 5 (a) into its process matrix *M*, a 10×10 matrix. Each element of *M* is determined according to whether or not the corresponding two activities have precedence dependency. An edge between nodes A_1 and A_2 shows that activity A_1 precedes activity A_2. Thus, $M_g(A_1, A_2)$ is set to a value of 1. There is no direct edge between nodes A_1 and A_3. Thus $M_g(A_1, A_3)$ is set to a value of 0.

M_{11}	TO										
		A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_9	A_{10}	A_{11}
	A_1	0	1	0	0	0	0	0	0	0	0
	A_2	0	0	1	1	0	0	0	0	0	0
	A_3	0	0	0	0	1	1	1	0	0	0
F	A_4	0	0	0	0	0	0	0	0	1	0
R	A_5	0	0	0	0	0	0	0	1	0	0
O	A_6	0	0	0	0	0	0	0	1	0	0
M	A_7	0	0	0	0	0	0	0	1	0	0
	A_9	0	0	0	0	0	0	0	0	0	1
	A_{10}	0	0	0	0	0	0	0	0	0	1
	A_{11}	0	0	0	0	0	0	0	0	0	0

Fig. 6 **Process matrix of g_{11}**

In order to compare the two process dependency graphs g_{11} and g_{22}, we need to further normalize each process matrix that participates in the similarity computation. Each normalized process matrix includes the union of all sets of nodes, each from one participating process dependency graph. We formally introduce the concept of normalized process matrix in Definition 5 by extending the definition of a process matrix to include the entire union of nodes in the two graphs. The size of the normalized matrix is increased to the size of the union of the sets of nodes in both graphs. For those nodes that exist in a process matrix before normalization, the corresponding elements in the normalized matrix are the same as those in the process matrix. For those nodes added through the normalization, the corresponding elements in the normalized matrix are set to a value of 0. After normalization, both matrices have the same number of rows and columns, and share the same row and column names and sequences. The normalized matrices can then be used as an input to calculate distance.

Definition 5 (Normalized Matrix, *NM*)

Let $DG_1 = (DN_1, DE_1)$ and $DG_2 = (DN_2, DE_2)$ be two dependency graphs. Let NM_1 and NM_2 denote the normalized matrices for DG_1 and DG_2 respectively. We generate NM_1 and NM_2 from DG_1 and DG_2 as follows.

i) The number of rows and columns are computed by $m = |DN_1 \cup DN_2|$

ii) Let $DN_1 \bigcup DN_2 = \{A_1, A_2, ..., A_m\}$. Note that the row and column names of NM_1 and NM_2 are now normalized into the same node names $A_1, A_2, ..., A_m$ in the union of DN_1 and DN_2.

iii) Let $NM_1(i,j)$ denote the value of the ith row and the jth column in NM_1, and $NM_2(i,j)$ denote the value of the ith row and the jth column in NM_2

$$NM_1(i,j) = \begin{cases} 1 & \text{if } (A_i, A_j) \in DE_1 \\ 0 & \text{otherwise} \end{cases},$$

$$NM_2(i,j) = \begin{cases} 1 & \text{if } (A_i, A_j) \in DE_2 \\ 0 & \text{otherwise} \end{cases}$$ ■

Consider processes in Fig. 5 as an example. By constructing normalized matrices for g_{11} and g_{22}, denoted by NM_{11} and NM_{22} respectively, the size of NM_{11} of g_{11} is increased to 11 because NM_{11} should include node A_8, which was not originally included in g_{11}. All the elements of the newly added column for node A_8 are set to a value of 0 because there is no dependency between any node of g_{11} and node A_8. Similarly, node A_6 is added in NM_{22}. Now NM_{11} and NM_{22} have the same row names and column names: A_1

through A$_{11}$. We can use NM_{11} and NM_{22} to compare g_{11} and g_{22}.

NM_{11}	A$_1$	A$_2$	A$_3$	A$_4$	A$_5$	A$_6$	A$_7$	A$_8$	A$_9$	A$_{10}$	A$_{11}$
A$_1$	0	1	0	0	0	0	0	0	0	0	0
A$_2$	0	0	1	1	0	0	0	0	0	0	0
A$_3$	0	0	0	0	1	1	1	0	0	0	0
A$_4$	0	0	0	0	0	0	0	0	0	1	0
A$_5$	0	0	0	0	0	0	0	0	1	0	0
A$_6$	0	0	0	0	0	0	0	0	1	0	0
A$_7$	0	0	0	0	0	0	0	0	1	0	0
A$_8$	0	0	0	0	0	0	0	0	0	0	0
A$_9$	0	0	0	0	0	0	0	0	0	0	1
A$_{10}$	0	0	0	0	0	0	0	0	0	0	1
A$_{11}$	0	0	0	0	0	0	0	0	0	0	0

(a) NM_{11}

NM_{22}	A$_1$	A$_2$	A$_3$	A$_4$	A$_5$	A$_6$	A$_7$	A$_8$	A$_9$	A$_{10}$	A$_{11}$
A$_1$	0	1	0	0	0	0	0	0	0	0	0
A$_2$	0	0	1	1	0	0	0	0	0	0	0
A$_3$	0	0	0	0	1	0	1	0	0	0	0
A$_4$	0	0	0	0	0	0	0	1	0	0	0
A$_5$	0	0	0	0	0	0	0	0	1	0	0
A$_6$	0	0	0	0	0	0	0	0	0	0	0
A$_7$	0	0	0	0	0	0	0	0	1	0	0
A$_8$	0	0	0	0	0	0	0	0	0	1	0
A$_9$	0	0	0	0	0	0	0	0	0	0	1
A$_{10}$	0	0	0	0	0	0	0	0	0	0	1
A$_{11}$	0	0	0	0	0	0	0	0	0	0	0

(b) NM_{22}

Fig. 7 **An example of comparison matrices**

The algorithm for construction of normalized process matrices consists of three steps. First, we must determine whether or not DG_1 and DG_2 are δ-comparable for the given δ value. Second, we compute the size of the normalized NM by $m = |DN_1 \cup DN_2|$ and label nodes in $\{DN_1 \cup DN_2\}$ as $\{A_1, A_2, \ldots A_m\}$ using a uniform naming scheme. Third, we create the matrix data structures for DG_1 and DG_2: $NM_1(i, j)$ and $NM_2(i, j)$, where i, j = 1, 2, ..., m, and assign a value of 1 or 0 to each element in the two normalized matrices.

4.2 Distance-based Process Similarity Measures

With the concept of a normalized matrix, we now transform the problem of comparing two processes into the problem of computing the distance-based similarity of the two normalized process matrices. One obvious idea is to compute the distance of two normalized matrices using matrix subtraction.

Consider the example processes g_{11} and g_{22} in Fig. 7. One way of computing the distance between g_{11} and g_{22} by matrix subtraction is to simply perform subtraction element by element. By subtracting NM_{22} from NM_{11}, we can see only five elements have values 1 and -1 respectively and the rest of the elements are 0. This means that five elements are unmatched between the two dependency graphs g_{11} and g_{22}.

NM_{11}-NM_{22} =

	A$_1$	A$_2$	A$_3$	A$_4$	A$_5$	A$_6$	A$_7$	A$_8$	A$_9$	A$_{10}$	A$_{11}$
A$_1$	0	0	0	0	0	0	0	0	0	0	0
A$_2$	0	0	0	0	0	0	0	0	0	0	0
A$_3$	0	0	0	0	0	1	0	0	0	0	0
A$_4$	0	0	0	0	0	0	0	-1	0	1	0
A$_5$	0	0	0	0	0	0	0	0	0	0	0
A$_6$	0	0	0	0	0	0	0	0	1	0	0
A$_7$	0	0	0	0	0	0	0	0	0	0	0
A$_8$	0	0	0	0	0	0	0	0	0	-1	0
A$_9$	0	0	0	0	0	0	0	0	0	0	0
A$_{10}$	0	0	0	0	0	0	0	0	0	0	0
A$_{11}$	0	0	0	0	0	0	0	0	0	0	0

A drawback of this approach is that both 1 and -1 values in the resulting matrix represent the fact that there are some discrepancies between two graphs g_{11} and g_{22} in five elements. But it does not tell the degree of such discrepancies in terms of concrete distance measure. Thus we need an efficient way to represent the total number of non-zero values in the resulting matrix.

One obvious way to capture the degree of the difference between NM_{11} and NM_{22} is to use the sum of the squares of elements in NM_1–NM_2 as shown below, which is $(1)^2 + (-1)^2 + (1)^2 + (1)^2 + (-1)^2 = 5$ because only five elements have non-zero values 1 and -1.

$(NM_{11} - NM_{22})(NM_{11} - NM_{22})^T =$

	A$_1$	A$_2$	A$_3$	A$_4$	A$_5$	A$_6$	A$_7$	A$_8$	A$_9$	A$_{10}$	A$_{11}$
A$_1$	0	0	0	0	0	0	0	0	0	0	0
A$_2$	0	0	0	0	0	0	0	0	0	0	0
A$_3$	0	0	1	0	0	0	0	0	0	0	0
A$_4$	0	0	0	2	0	0	0	-1	0	0	0
A$_5$	0	0	0	0	0	0	0	0	0	0	0
A$_6$	0	0	0	0	0	1	0	0	0	0	0
A$_7$	0	0	0	0	0	0	0	0	0	0	0
A$_8$	0	0	0	-1	0	0	0	1	0	0	0
A$_9$	0	0	0	0	0	0	0	0	0	0	0
A$_{10}$	0	0	0	0	0	0	0	0	0	0	0
A$_{11}$	0	0	0	0	0	0	0	0	0	0	0

Interestingly, we can calculate the sum of the squares of elements in a matrix by the notion of trace in linear algebra. According to [5], the sum of diagonal elements in a matrix is defined as the trace of the matrix. The best way to calculate the sum of the squares of elements in a matrix is using the concept of inner products, which is defined by the trace concept.

Definition 6 (Dependency Difference Metric, *d*)
Let $DG_1 = (DN_1, DE_1)$ and $DG_2 = (DN_2, DE_2)$ be two dependency graphs. Let NM_1 and NM_2 be the normalized matrix of DG_1 and DG_2 respectively. We define the symmetric difference metric on graphs DG_1 and DG_2 by the trace of the difference matrix of NM_1 and NM_2 as follows:

$$d(DG_1, DG_2) = tr[(NM_1 - NM_2) \times (NM_1 - NM_2)^T]$$

where $tr[\cdot]$ denotes the trace of a matrix, i.e., the sum of the diagonal elements. ■

This distance function counts the number of edge discrepancies between DG_1 and DG_2. Now, we want to show that the dependency difference metric d satisfies the distance measure properties. The function d is called a metric if and only if for all graphs g_1, g_2, g_3, the following conditions hold [7]:

i) $d(g_1, g_2) = 0$ iff g_1 and g_2 are identical
ii) $d(g_1, g_2) = d(g_2, g_1)$
iii) $d(g_1, g_2) \le d(g_1, g_3) + d(g_3, g_2)$.

Theorem 1. *d*(*DG*₁,*DG*₂) satisfies Distance Measure Properties.
Proof:
Concretely, we want to prove that if $A = NM_1 - NM_2$ and $d(DG_1, DG_2) = < A, A^T >= tr(A \times A^T) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{\;2}$, then this distance $d(DG_1, DG_2)$ satisfies the three distance measure properties:

i) $d(DG_1, DG_2) = 0$ iff DG_1 and DG_2 are identical, because the matrix A becomes 0.
ii) $d(DG_1, DG_2) = d(DG_2, DG_1)$ by the d definition.
iii) $d(DG_1, DG_2) \le d(DG_1, DG_3) + d(DG_3, DG_2)$

For any two nodes i, j, let

$$NM_k(i,j) = \begin{cases} 1 & \text{if } (A_i, A_j) \in DE_1 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } k = 1, 2, 3$$

Then we can show the property iii) holds.

$$\begin{aligned} d(DG_1, DG_2) &= tr[(NM_1 - NM_2) \times (NM_1 - NM_2)^T] \\ &= \sum\nolimits_{i,j} \{NM_1(i,j) - NM_2(i,j)\}^2 \\ &= d(DG_2, DG_1) \end{aligned}$$

Now we show that the property iii) holds as well, because $NM_1(i,j) - NM_2(i,j)$ is either 0 or ±1, thus we have $d(DG_1, DG_2) = \sum_{i,j} |NM_1(i,j) - NM_2(i,j)|$.

$$\begin{aligned} &d(DG_1, DG_3) + d(DG_3, DG_2) \\ &= \sum\nolimits_{i,j} |NM_1(i,j) - NM_3(i,j)| + \sum\nolimits_{i,j} |NM_3(i,j) - NM_2(i,j)| \\ &= \sum\nolimits_{i,j} \{|NM_1(i,j) - NM_3(i,j)| + |NM_3(i,j) - NM_2(i,j)|\} \\ &\ge \sum\nolimits_{i,j} |NM_1(i,j) - NM_3(i,j) + NM_3(i,j) - NM_2(i,j)| \\ &= \sum\nolimits_{i,j} |NM_1(i,j) - NM_2(i,j)| \\ &= d(DG_1, DG_2) \end{aligned}$$

So the new process distance measure is, in fact, a distance metric. ■

Since the dependency distance metric $d(g^1, g^2)$ counts the number of asymmetric arcs, it can reflect the difference of some characteristics between two processes, such as activity precedence, activity commonality, flow structure, etc. Activity precedence describes how the activities are linked and sequenced in terms of execution ordering. The dependency distance metric denotes the disparity of sequence between two activities and can be extended to represent the sequence disparities between all activities. In Fig. 8, the distance of two processes g^0 and g^1, denoted by $d(g^0, g^1)$, illustrates the difference of activity precedence. Activity commonality means how many activities are shared between two process models. This counts the different activities or new activities of two processes, as illustrated by processes g^0 and g^2 in Fig. 8. In addition, flow structure denotes the difference between serial and parallel flows. Two processes g^0 and g^3 show the difference measurement of flow structures, serial and parallel flows.

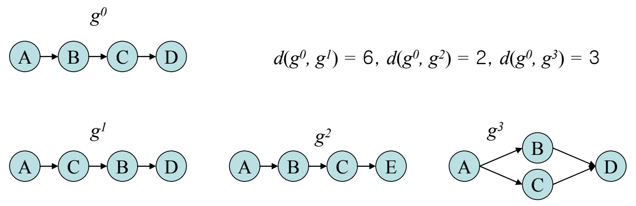


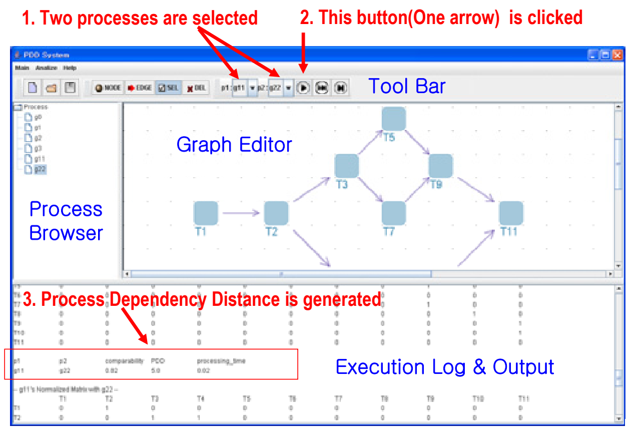
Fig. 8 **Examples of dependency distance**

In Fig. 8, if we follow the previous procedure to calculate the dependency distance, all of the graphs are transformed to process network matrices and normalized process matrices. Then the distance of dependency between g^0 and g^1 is 6, the distance of g^0 and g^2 is 2, and the distance of g^0 and g^3 is 3. This means that g^0 and g^2 are the most similar, which is intuitively correct because the first three activities are in the same sequence but only the last activity is different. g^0 and g^1 are mostly different because the sequence of the activities in g^1 is quite different from g^0. In this dependency distance measure, the parallel execution in g^3 is not considered important and only the precedence relationships and common activities are considered important.

If we look into more extended examples in Fig. 5 again, each graph is transformed into process matrix, and then normalized matrix. These two normalized matrices are subtracted and squared. Finally we can get the proposed dependency distance 5 by obtaining the trace of it.

5. Prototypes and Evaluations

The presented concepts of this paper were implemented to analyze the similarity of processes in process warehouse. This system, called “BPSAT(Business Process Similarity Analysis Tool)”, is developed by using Java language. This prototype system has three windows: process browser, graph editor, and execution log output window. We can select some processes in the left process browser, and the selected process is shown and modified in the right graph editor. All the execution log and analysis outputs are displayed in the bottom window. There are also necessary buttons in tool bar. The basic manipulation such as creating and editing of process graph can be done in this prototype system, and the functionality of similarity analysis methods proposed in this paper can be done in this system. Also other new similarity criteria can be added in this system. The current version of this system can be downloaded at http://ebiz.chonbuk.ac.kr/~jsbae/bpsat.


1. Two processes are selected
2. This button(One arrow) is clicked
Tool Bar
Graph Editor
Process Browser
3. Process Dependency Distance is generated
Execution Log & Output
T1
T2
T3
T5
T7
T9
T11

```
p1      p2      comparability   PDD     processing_time
g11     g22     0.82            5.0     0.02
```

Fig. 9 **Prototype system of BPSAT**

After we check the candidate processes to be compared, we select two processes to be compared, g_{11} and g_{22}. Then we can get the proposed process dependency distance is generated and shown in the output window.

6. Related Work

Although business process management systems have been deployed in many industrial engineering fields, research on analysis, mining and integration of business processes are still in its infancy. One of the representative existing studies on process improvement is workflow mining, which investigates the traces and results of workflow execution, and determines significant information in order to improve the existing workflow processes [2, 3, 4, 9, 16]. However, most of the existing workflow mining research does not provide a quantitative measure to compare and capture the similarity of different workflow designs.

The graph theory in a traditional algorithm textbook is a useful means to analyze the process definitions. Graphs, or representative data structures, are used as an accepted effective tool to represent the problem in various fields, which include pattern matching and machine recognition, such as pattern recognition, web and XML document analysis, and schema integration [8, 10, 18, 19]. For example, research on similarities in graph structures can be divided into three categories. The first category of traditional similarity is based on graph and sub-graph isomorphism, which has several weaknesses and distortions in the input data and the models. In order to overcome these weaknesses, other graph similarity analysis techniques, such as the graph edit distance (GED) metric and maximal common sub-graph (MCS) have been introduced [8, 19]. It is also worth mentioning that Bunke [8] has shown that with generic graphs, under certain assumptions concerning the edit-costs, determining the maximum common sub-graph is equivalent to computing the graph edit-distance. This MCS is a basic concept of workflow similarity that measures the common activities and transitions of workflow processes. In this paper we utilize the graph theory results to derive the metric space distance metric for measuring process similarity and difference.

Our research on workflow similarity measure is mainly inspired by the research results on document similarity analysis and graph similarity measures. A large number of document similarity measures are presented in existing literature for building document management systems, knowledge management systems, as well as search engines [8, 10, 12].

Finally, in order to support web service composition, an infrastructure for searching and matchmaking of business processes is needed. One example is using annotated deterministic finite state automata (aDFA) to model the business processes [18]. If a business process is specified as aDFA, the match between two aDFAs is determined by the intersection of their languages. When there is non-empty intersection, the two business processes are matched.

7. Conclusion and Future work

We have presented a difference analysis methodology using distance measures between process definitions of web services. The proposed difference analysis method achieves three distinct goals. First, by analyzing the attributes of process models, we can present a quantitative process similarity metric to determine the relative distance between process models. This facilitates not only the comparison of existing process models with each other, but also provides the flexibility to adapt to changes in processes. Second, the proposed method is fast and flexible, which reduces the cost of both the analysis and design phases of complex web service processes. Third, the proposed method enables the flexible deployment of process mining, discovery, and integration – all desirable functionality that are critical for fully supporting the effective transformation of an enterprise.

Our research on process mining, discovering and integration through similarity analysis continues along several directions. First, we are interested in distance measures that can compare workflow designs with complex block structure and various execution constraints. Second, we are interested in developing a prototype system that provides efficient implementation of various similarity analysis methods, including the dependency distance metric presented in this paper. Furthermore we are interested in applying the method developed to concrete case studies of existing enterprise transformations and to evaluate and improve the similarity measures proposed in this paper.

Acknowledgements: The first author was supported by the Korea Research Foundation Grant (KRF-2004-003-D00477).

8. References

1. W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, “Workflow Patterns,” *Distributed and Parallel Databases*, 14(3), pages 5-51, July 2003.
2. W.M.P. van der Aalst, B.F.van Dongen, J. Herbst, L. Maruster, G. Schimm and A.J.M.M. Weijters, “Workflow Mining: A Survey of Issues and Approaches,” *Data and Knowledge Engineering*, 47(2), 237-267, 2003
3. W.M.P. van der Aalst, A.J.M.M. Weijters and L. Maruster, ”Workflow Mining: Discovering Process Models from Event Logs,” *IEEE Transactions on Knowledge and Data Engineering*, 16(9), pp. 1128-1142, 2004
4. R. Agrawal, D. Gunopulos, and F. Leymann, “Mining Process Models from Work-flow Logs”, 6th International Conference on Extending Database Technology, pp. 469-483, 1998.
5. Howard Anton and Chris Rorres, *Elementary Linear Algebra: Applications*, John Wiley&Sons, 1994.
6. J. Bae, H. Bae, S. Kang, Y. Kim, “Automatic control of workflow process using ECA rules,” *IEEE Trans. on Knowledge and Data Engineering*, vol.16, no.8, pp. 1010-1023, 2004.
7. D. Banks and K. Carley, “Metric inference for social networks,” *Journal of classification*, vol.11, pp. 121-149, 1994.
8. H. Bunke, K. Shearer, “A Graph Distance Metric based on the Maximal Common Subgraph,” *Pattern Recognition Letters*, vol.19, issues 3-4, pp. 255-259, 1998.
9. J.E. Cook and A.L. Wolf, “Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model,” *ACM Transactions on Software Engineering and Methodology*, 8(2), pp. 147-176, 1999.
10. K.M. Hammouda and M.S. Kamel, “Efficient Phrase-Based Document Indexing for Web Document Clustering,” *IEEE Transactions on Knowledge and Data Engineering*, vol.16, no.10, pp. 1279-1296, 2004.
11. F. Leymann and D. Roller, *Production workflow: concepts and techniques*, Prentice Hall PRT, New Jersey, 2000.
12. W. Lian, W.W. Cheung, N. Mamoulis, S. Yiu, “An Efficient and Scalable Algorithm for Clustering XML Documents by Structure,” *IEEE Transactions on Knowledge and Data Engineering*, vol.16, no.1, pp. 82-96, 2004.
13. RosettaNet, http://www.rosettanet.org, RosettaNet Standard (RosettaNet Partner Interface Processes)
14. W. B. Rouse, “A Theory of Enterprise Transformation,” *Systems Engineering*, vol. 8, no. 4, 2005.
15. R. Rush and W.A. Wallace, “Elicitation of knowledge from multiple experts using network inference,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 5, pp. 688-698, 1997.
16. Guido Schimm, “Mining exact models of concurrent workflows,” *Computers in Industry*, 53, pp 265-281, 2004.
17. WfMC, Workflow Management Coalition Workflow Standard Process Definition Interface -- XML Process Definition Language, Document Number WFMC-TC-1025 Version 1.13, September 7, 2005
18. Andreas Wombacher, Peter Fankhauser, Bendick Mahleko, and Erich Neuhold, “Matchmaking for Business Processes Based on Choreographies,” *International Journal of Web Services*, vol.1, no.4, pp. 14-32, 2004
19. K. Zhang and D. Shasha, “Simple Fast Algorithms for the Editing Distance between Trees and Related Problems,” *SIAM Journal of Computing*, vol.18, no.6, pp. 1245-1262, 1989