Process Mining, Discovery, and Integration using Distance Measures

Joonsoo Bae
Dept. of Industrial & Sys. Eng.
Chonbuk National Univ.South Korea
jsbae@chonbuk.ac.kr

James Caverlee
College of Computing
Georgia Institute of Technology
caverlee@cc.gatech.edu

Ling Liu
College of Computing
Georgia Institute of Technology
lingliu@cc.gatech.edu

William B. Rouse
Tennenbaum Institute
Georgia Institute of Technology
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 workflow designs. Concretely, we introduce a graph-based distance measure and a framework for utilizing this distance measure to mine the process repository and discover workflow designs that are similar to a given design pattern or to produce one integrated workflow design by merging two or more business workflows of similar designs. We derive the similarity measures by analyzing the workflow dependency graphs of the participating workflow processes. Such an analysis is conducted in two phases. We first convert each workflow dependency graph into a normalized process network 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 [16]. 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 (e.g., by adopting a web services framework for supporting customer service) to performing different work altogether (e.g., by outsourcing all non-essential functions and focusing on an enterprise's core strengths). 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 workflow 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 (process model for short in the rest of the paper) consists of business process definitions and the specification of workflows among the processes with respect to data flow, control flow, and operational views [17, 18]. We define a business process in terms of business activity patterns. An activity pattern consists of objects, messages, message exchange constraints, preconditions and postconditions [12, 19], 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, 7, 15]. 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. Activity patterns are constrained by a set of activity dependencies, which can be seen as cooperation contracts among activities that collaborate in accomplishing a complex service. The dependencies could capture complex interactions between activities.

We define a business workflow 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 workflow. We use control flow to capture the operational structure of the business workflow service, including the process execution ordering, the transactional semantics and dependencies of the workflow. A number of workflow specifications have garnered attention, including BPEL4WS (BEA, IBM, Microsoft), WSFL (IBM), XLANG (Microsoft), and XPDL (WfMC) [19, 20]. 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, and refer readers to our technical report for aggregation hierarchy based process similarity measures.

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 $n \geq 1$. Each node nd_i (*i=1, ..., n*) is described by a quadruple (NT, NN, TC, NS), where *NT* is the node type, *NN* denotes the node name, *TC* is the trigger condition of the node, *NS* is one of the two states of the node: fired or not fired
- $DE = \{e_1, e_2, ..., e_m\}$ is a set of edges, $m \geq 0$. Each edge is of the form $nd_i \rightarrow nd_j$. An edge $e_{ij} = nd_i \rightarrow nd_j$ is described by a quadruple $(EN, DP_{nd}, AV_{nd}, ES)$, where *EN* is the edge name, DP_{nd} is the departure node, AV_{nd} is the arrival node, *ES* is one of the two states of the edge: signaled and not signaled. ■

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.

3. Motivating Scenarios

Given the process reference model, we consider two motivating scenarios that benefit from the difference analysis methodology introduced in this paper.

3.1. Process Mining

Consider a scenario where a company has maintained a warehouse of existing processes used in various business locations. *Process mining* 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.

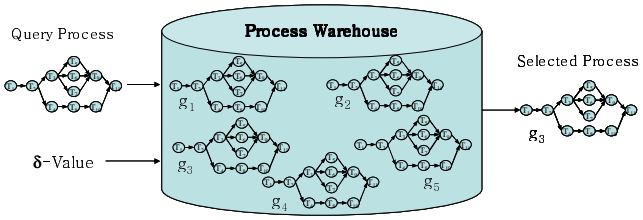


Fig. 1 Process mining example

In Fig. 1, 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 baseline process template in the process warehouse. Given a query process and a comparability threshold δ-value, the process mining will identify (g_3) 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.

3.2. Process Discovery & Integration

To illustrate the second scenario, we consider an enterprise transformation due to the merger of two companies. *Process discovery and integration* using process similarity can help the merged enterprise to identify the commonality between the existing business workflows of the two companies being merged and generate the new business workflow by eliminating unnecessary redundancies. Fig. 2 illustrates a situation when two companies have merged and attempt to implement a standardized process between the two. Since both companies have their own process warehouses before merging, there is a possibility that two different processes, g_A and g_B respectively, exist for the same purpose. One possible process transformation is to identify and integrate the two processes into a standard process (g_C). In this case, process similarity plays an important role in identifying similar processes and guiding the integration.

4. Process Difference Analysis

In this section, we present the process difference analysis method for evaluating the distance between two processes. We first describe the comparison matrices used for similarity analysis, including the concept of identical dependency graphs and δ-comparability. Both concepts are important for defining the distance measure between two dependency graphs. Then we define the concept of a process network matrix and introduce the concept of a normalized matrix. Finally, we define the dependency distance measure by measuring the difference between the normalized matrices.

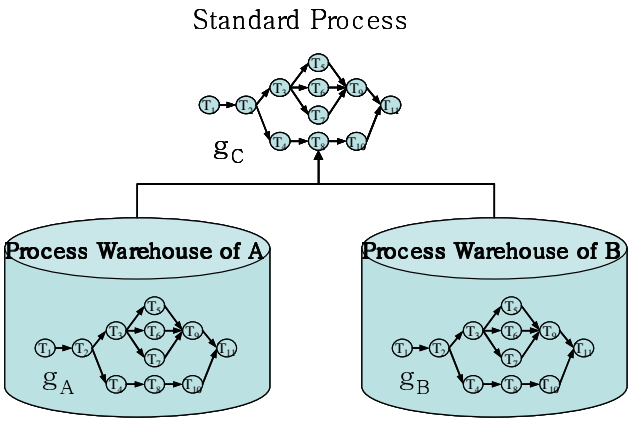


Fig. 2 Extracting a standard process from two similar ones

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 2 (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) $DE_1 =_{Set} DE_2$ ■

Given two workflow 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 3 (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. 3 with δ=0.5, g^0 and f^1 are not comparable because there is no common node in the two graphs, and also 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^1 are δ- comparable because all of the nodes in both graphs are common nodes. 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 δ = 0.5.

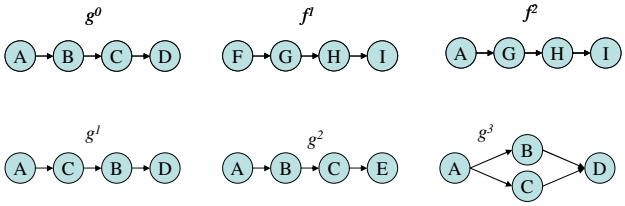


Fig. 3 Examples of δ-Comparability

From the examples in Fig. 3, it would be useful to know not only g^0 and g^1 and g^0 and g^2 are δ-comparable with δ = 0.5, but also which dependency graph is most similar to g^0. 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 network matrix. A process network 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*.

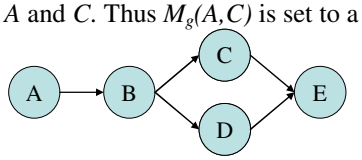
Definition 4 (Process Network Matrix, *M*)

Let $g = (DN, DE)$ be a dependency graph with $|DN| = n$ nodes. A process network 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 i^{th} row and the j^{th} column in *M*, $1 \leq i, j \leq n$. 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. 4 depicts the transformation of a process dependency graph of 5 activities shown in (a) into its process network matrix *M*, a 5×5 matrix. Each element of *M* is determined according to whether or not the corresponding two activities

(tasks) have precedence dependency. An edge between nodes *A* and *B* shows that activity A precedes activity B. Thus, $M_g(A,B)$ is set to a value of *1*. There is no direct edge between nodes *A* and *C*. Thus $M_g(A,C)$ is set to a value of 0.



(a)

M	TO					
		A	B	C	D	E
FROM	A	0	1	0	0	0
	B	0	0	1	1	0
	C	0	0	0	0	1
	D	0	0	0	0	1
	E	0	0	0	0	0

(b)

Fig. 4 An example of process network matrix

Recall that when two workflow dependency graphs are δ-comparable, they may not have the same set of nodes. Recall Fig. 3, g^0 and g^2 are 0.5-comparable but node D in g^0 does not exist in g^2 and node E in g^2 does not exist in g^0. In order to compare the two workflow dependency graphs g^0 and g^2, we need to further normalize each process network matrix that participates in the similarity computation such that each normalized process network matrix includes the union of all sets of nodes, each from one participating process dependency graph. We formally introduce the concept of normalized process network matrix in Definition 4 by extending the definition of a process network 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 network matrix before normalization, the corresponding elements in the normalized matrix are the same as those in the process network 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 $DN_1 \cup DN_2 = \{A_1, A_2, ..., A_m\}$. 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.

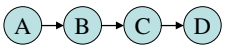
iii) Let $NM_1(i, j)$ denote the value of the *i*th row and the *j*th column in NM_1, and $NM_2(i, j)$ denote the value of the *i*th row and the *j*th column in NM_2

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

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

Consider processes in Fig. 3 as an example. By constructing normalized process network matrices for g^0 and g^2, denoted by NM_1 and NM_2 respectively, the size of NM_1 of g^0 is increased to 5 because NM_1 should include node E, which was not originally included in g^0. All the elements of the newly added column for node E are set to a value of 0 because there is no dependency between any node of g^0 and node E. Similarly, node D is added in NM_2. Now NM_1 and NM_2 have the same row names and column names: A, B, C, D, and E. We can use NM_1 and NM_2 to compare g^0 and g^2.

g^0

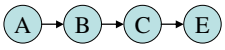


(a)

NM_1	TO					
		A	B	C	D	E
FROM	A	0	1	0	0	0
	B	0	0	1	0	0
	C	0	0	0	1	0
	D	0	0	0	0	0
	E	0	0	0	0	0

(b)

g^2



(c)

NM_2	TO					
		A	B	C	D	E
FROM	A	0	1	0	0	0
	B	0	0	1	0	0
	C	0	0	0	0	1
	D	0	0	0	0	0
	E	0	0	0	0	0

(d)

Fig. 5 An example of comparison matrices

The algorithm for construction of normalized process network 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 $\{A_1, A_2, ...A_m\}$ 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. The pseudo code of this algorithm is given in Algorithm 1.

Algorithm 1 (Normalization of Comparable Matrices)

Input: $DG_1 = (DN_1, DE_1)$; $DG_2 = (DN_2, DE_2)$;

Output: $NM_1(i, j)_{m\times m}$; $NM_2(i, j)_{m\times m}$

Description:
Compute m, Construct NM_1, NM_2 for each DG_1, DG_2

Begin
if δ-comparable (DG_1, DG_2) = false then exit;
Compute union of sets DG_1 and DG_2 ;
Set $m = \left| DN_1 \cup DN_2 \right|$;
Label nodes in $\{DN_1 \cup DN_2\}$ as $\{A_1, A_2, ...A_m\}$;
/* Construct NM_1, NM_2 for each DG_1, DG_2 */
Initialize $NM_1(i, j)_{m\times m} = 0$;
Initialize $NM_2(i, j)_{m\times m} = 0$;
for i = 1 step 1 until m
for j = 1 step 1 until m {
if $(A_i, A_j) \in NE_1$ then set $NM_1(i, j) = 1$;
if $(A_i, A_j) \in NE_2$ then set $NM_2(i, j) = 1$;
}
return $NM_1(i, j)_{m\times m}$; $NM_2(i, j)_{m\times m}$;
End ■

4.2 Distance-based Process Similarity Measures

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

Consider the example processes g^0 and g^2 in Fig. 5. One way of computing the distance between g^0 and g^2 by matrix subtraction is to simply perform subtraction element by element. By subtracting NM_2 from NM_1, only the element (C, D) and (C, E) have values 1 and -1 respectively and the rest of the elements are 0. This means that (C, D) and (C, E) are the two dependencies unmatched between the two dependency graphs g^0 and g^2.

$$NM_1 - NM_2 = \begin{bmatrix} & A & B & C & D & E \\ A & 0 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 1 & -1 \\ D & 0 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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^0 and g^2 in nodes C, D, E. 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_1 and NM_2 is to use the sum of the squares of elements in $NM_1 - NM_2$ as shown below, which is $(1)^2 + (-1)^2 = 2$ because element (C, D) and (C, E) have non-zero values 1 and -1.

$$(NM_1 - NM_2)(NM_1 - NM_2)^T$$
$$= \begin{bmatrix} & A & B & C & D & E \\ A & 0 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 1 & -1 \\ D & 0 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} & A & B & C & D & E \\ A & 0 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 1 & 0 & 0 \\ E & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$
$$= \begin{bmatrix} & A & B & C & D & E \\ A & 0 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 1^2 + (-1)^2 & 0 & 0 \\ D & 0 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Interestingly, we can calculate the sum of the squares of elements in a matrix by the notion of trace in linear algebra. The best way to calculate the sum of the squares of elements in a matrix is using the concept of inner products.

Before we introduce the concept of trace and inner products in our context, we first give a general description of the matrix multiplication. A multiplication of a general matrix A and its transpose matrix A^T is defined as follows. It indicates that the sum of diagonal elements equals the exact sum of the squares of each element in matrix A.

$$A \times A^T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \dots & & & & \dots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \times \begin{bmatrix} a_{11} & a_{21} & a_{31} & \dots & a_{n1} \\ a_{12} & a_{22} & a_{32} & \dots & a_{n2} \\ a_{13} & a_{23} & a_{33} & \dots & a_{n3} \\ \dots & & & & \dots \\ a_{1n} & a_{2n} & a_{3n} & \dots & a_{nn} \end{bmatrix}$$
$$= \begin{bmatrix} \sum_{j=1}^{n} a_{1j}^2 & \dots & \dots & \dots & \dots \\ \dots & \sum_{j=1}^{n} a_{2j}^2 & \dots & \dots & \dots \\ \dots & \dots & \sum_{j=1}^{n} a_{3j}^2 & \dots & \dots \\ \dots & & & & \dots \\ \dots & \dots & \dots & \dots & \sum_{j=1}^{n} a_{nj}^2 \end{bmatrix}$$

According to [6], the sum of diagonal elements in a matrix is defined as the trace of the matrix.

Definition 6 (Trace of Matrix, *tr*)
The trace of an m-by-m square matrix *A* is defined as the sum of the elements on the main diagonal (the diagonal from the upper left to the lower right) of *A*, i.e.,

$$tr(A) = A_{1,1} + A_{2,2} + ... + A_{m,m}$$

where A_{ij} represents the (i, j)th element of *A* ■

Now we can define the inner product of a matrix using the definition of trace.

Definition 7 (Inner Product of Matrix)
For an m-by-n matrix *A* with complex (or real) entries, we have

$$< A, A^T >= tr(A \times A^T) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^{\ 2} \geq 0,$$

with the trace of A equals to 0 only if *A*=0. ■
This inner product yields a real number (a scalar) that is the product of two vectors on the space of all complex (or real) m-by-n matrices.

Definition 8 (Dependency Difference Metric, *d*)
Let $DG_1 = (DN_1, DE_1)$ and $DG_2 = (DN_2, DE_2)$ be two dependency graphs. Let *NM1* and *NM2* be the normalized matrix of *DG1* and *DG2* respectively. We define the symmetric difference metric on graphs *DG1* and *DG2* by the trace of the difference matrix of *NM1* and *NM2* as follows:

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

where *tr*[·] denotes the trace of a matrix, i.e., the sum of the diagonal elements. ■

This distance function counts the number of edge discrepancies between *DG1* and *DG2*. In fact, the metric *d* is the Hamming metric used in information theory [8]. Now, we want to show that the dependency difference metric *d* satisfies the distance measure properties. First, we want to show that the inner product of a matrix and its transpose matrix satisfies the distance measure properties. Let G(n) represent the set of all graphs on *n* distinct vertices which have undistinguished (unweighted) edges and no loops. (Loops are edges that connect a vertex to itself). Even though our discussion excludes the loop situation for simplicity, the method we describe can be extended generally. Let R denote the real numbers. Recall that a function *d*: G(n)×G(n)→R metrizes G(n) if and only if for all networks $g_1, g_2, g_3 \in$ G(n) the following conditions hold:

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) $d(g_1, g_2) \leq d(g_1, g_3) + d(g_3, g_2)$.

The function *d* is called a metric, and (G(n), *d*) is called a metric space. One can define many possible metrics on the set of graphs with *m* vertices. In a particular application, the metric should reflect a sense of distance that honors the context of the data.

Lemma 1 (Inequality of Absolute Values)
For two real numbers x and y, $|x| + |y| \geq |x + y|$. ■

Theorem 1
$d(DG_1, DG_2)$ **satisfies the Distance 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 *DG1* and *DG2* are identical, because the matrix A becomes 0.

ii) $d(DG_1, DG_2) = d(DG_2, DG_1)$

iii) $d(DG_1, DG_2) \leq d(DG_1, DG_3) + d(DG_3, DG_2)$

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 ii) holds.

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

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 $d(DG_1, DG_2) = \sum_{i,j} |NM_1(i, j) - NM_2(i, j)|$. Similarly,

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

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 task (activity) precedence, task commonality, flow structure, etc. Task 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 tasks and can be extended to represent the sequence disparities between all tasks. In Fig. 6, the distance of two processes g^0 and g^1, denoted by $d(g^0, g^1)$, illustrates the difference of task precedence. Task 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 Figure 6. 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.

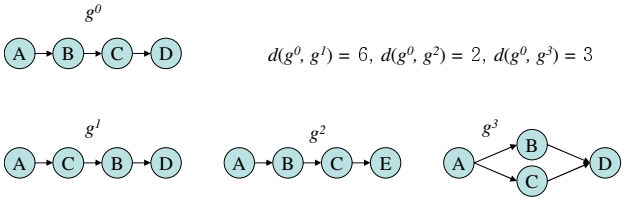


Fig. 6 Examples of dependency distance measurements

In Fig. 6, 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. Due to the space restriction, we will provide a discussion on the dependency distance measure for processes that are executed in parallel in our technical report.

5. 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, 5, 10, 18]. 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 objective of this research is to develop a distance based similarity measure to discover, mine and integration of existing workflow definitions by analysis of workflow dependency graphs. Process discovery and process mining are useful for new enterprises or businesses to create their own workflow processes based on successful experience of others. Process integration is critical for supporting a successful merger of two business units or enterprises.

In addition, the wide spread use of process-centric systems has made it possible to accumulate process definitions and to accelerate the analysis and comprehension of process definitions.

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 [9, 11, 21, 22]. 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 [9, 22]. The GED implemented a set of editing operations, for example, the deletion, insertion, and substitution of nodes and edges, and defined the similarity of two graphs in terms of the shortest (or least cost) sequence of editing operations that transforms one graph into the other. The MCS measures the distance between graphs by measuring the missing structural information expressed as the difference between the minimal common super-graph and maximal common sub-graph. Such an approach can naturally deal with several types of noises and distortions, such as the addition or deletion of nodes in both graphs, and has the advantage of not requiring the use of any cost function, thereby avoiding the major drawback of edit-distance-based approaches. It is also worth mentioning that Bunke [9] 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 [9, 11, 14].

Finally, web services are standard means to provide remote access to many Web applications. One of the important problems in Web services research is developing efficient methods for automatically discovering and invoking remote Web services. Furthermore, composing multiple Web services, rather than accessing a single service, is essential for many mission critical applications and provides more benefits to users. 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 [21]. 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. However, this approach may not work for complex web services where each service is a

multi-step workflow with an overlapping set of task (activity) nodes. In this paper we present a quantitative approach to modeling and capturing the similarity and dissimilarity between different workflow designs. We derive the similarity measures by analyzing the workflow dependency graphs of the participating workflow processes in two phases. First, we convert each workflow dependency graph into a normalized process network matrix. Then we calculate the metric space distance between the normalized matrices. We show how this distance measure can be used as a quantitative and qualitative tool in process mining, process merging and process clustering. We believe that our approach can ultimately reduce or minimize the costs involved in design, analysis, and evolution of workflow systems.

6. 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).

7. 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. van der Aalst, W.M.P., van Dongen, B.F., Herbst, J., Maruster, L., Schimm, G. & Weijters. A.J.M.M. (2003). Workflow Mining: A Survey of Issues and Approaches, Data and Knowledge Engineering. 47(2), 237-267, 2003
3. van der Aalst, W.M.P., Weijters, A.J.M.M., & Maruster, L. (2004). Workflow Mining: Discovering Process Models from Event Logs. IEEE Transactions on Knowledge and Data Engineering. 16(9), pp. 1128-1142, 2004
4. van der Aalst, W.M.P. and Weijters, A.J.M.M. (2004), “Process Mining: A Research Agenda,” Computers in Industry, 53(3), pp. 231-244, 2004
5. 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.
6. Howard Anton and Chris Rorres, Elementary Linear Algebra: Applications, John Wiley&Sons, 1994.
7. Bae, J., Bae, H., Kang, S., Kim, Y.: Automatic control of workflow process using ECA rules. IEEE Trans. on Knowledge and Data Engineering, vol.16, no.8 (2004) 1010-1023.
8. Banks, D., Carley, K.: Metric inference for social networks. Journal of classification. vol.11 (1994) 121-149.
9. Bunke, H., Shearer, K.: A Graph Distance Metric based on the Maximal Common Subgraph. Pattern Recognition Letters, vol.19, issues 3-4, (1998) 255-259.
10. 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.
11. Hammouda, K.M., Kamel, M.S.: Efficient Phrase-Based Document Indexing for Web Document Clustering. IEEE Transactions on Knowledge and Data Engineering, vol.16, no.10 (2004) 1279-1296.
12. Hollingsworth, D., “Workflow management coalition specification,” The workflow reference model, WfMC, 1995.
13. Leymann, F., Roller, D.: Production workflow: concepts and techniques. Prentice Hall PRT, New Jersey (2000).
14. Lian, W., Cheung, W.W., Mamoulis, N., Yiu, S.: An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. IEEE Transactions on Knowledge and Data Engineering, vol.16, no.1 (2004) 82-96.
15. Ling Liu and Calton Pu. "ActivityFlow: Towards Incremental Specification and Flexible Coordination of Workflow Activities", In: The 16th International Conference on Conceptual Modeling (ER'97), Los Angeles, California, USA (3 - 6 November 1997).
16. Rouse, W. B., “A Theory of Enterprise Transformation. Systems Engineering,” vol. 8, no. 4, 2005.
17. Rush, R., Wallace, W.A., “Elicitation of knowledge from multiple experts using network inference,” IEEE Transactions on Knowledge and Data Engineering, vol. 9, no. 5 (1997) 688-698.
18. Guido Schimm, “Mining exact models of concurrent workflows,” Computers in Industry, 53, pp 265-281, 2004.
19. WfMC, Workflow Management Coalition Terminology and Glossary 3.0 (WFMC-TC-1011). Technical report, Workflow Management Coalition, Brussels, 1999.
20. WfMC, Workflow Management Coalition Workflow Standard Process Definition Interface -- XML Process Definition Language, Document Number WFMC-TC-1025 Version 1.13, September 7, 2005
21. Andreas Wombacher, Peter Fankhauser, Bendick Mahleko, an Erich Neuhold, “Matchmaking for Business Processes Based on Choreographies,” in International Journal of Web Services, Vol. 1, No. 4, ISSN: 1545-7362 Idea Group Publishing, pp. 14-32, 2004
22. Zhang, K., Shasha, D.: Simple Fast Algorithms for the Editing Distance between Trees and Related Problems. SIAM Journal of Computing, vol.18, no.6 (1989) 1245-1262.