Context-Based Service Recommendation for Assisting Business Process Design

Nguyen Ngoc Chan, Walid Gaaloul, and Samir Tata

Information Department
TELECOM SudParis
UMR 5157 CNRS Samovar, France

Abstract. The WS-BPEL provides a standard for business processes abstraction and execution, in which, the business processes abstraction is the key step for the completeness and success of business processes. The business processes abstraction includes the behavior and interactions between services which are sketched out by business processes designers. The current business process design is labor-intensive and time consuming, especially when it is required to be detailed to ensure the success of the business execution. In this paper, we propose an approach that helps the business process designers facilitate the design step by providing them a list of related services to the current designed model. We propose to capture the requested service's composition context specified through the process fragment surrounding it and recommend the services whose composition context in existing designed service compositions best match the given fragment context. Provided experimental evaluations in this paper show that our approach is efficient in realistic situations.

Keywords: business process modeling; web service composition; workflow pattern; context matching; recommender system.

1 Introduction

The current design of business process models is labor-intensive, especially when such models are required to be detailed to support the development of software systems [20]. It would be inefficient if every time a company engages in modeling and re-designing its process, it did so "from scratch" without consideration of how other companies perform similar processes. Indeed, to avoid the effort of creating process models from scratch, several consortia and vendors have defined so-called reference process models, for example SCOR [18] or SAP [7] models. These models capture proven practices and recurrent business operations in a given domain. They are designed in a generic manner and are intended to be individualized to fit the requirements of specific organizations or IT projects in order to enable systematic reuse of proven practices across process (re-)design projects. However, analysts take the reference models merely as a source of inspiration, but ultimately, they design their own model on the basis of the reference model, with little guidance as to which model elements need to be removed,

C. Huemer and T. Setzer (Eds.): EC-Web 2011, LNBIP 85, pp. 39–51, 2011.

added or modified to address a given requirement. Briefly, the current business process design still has shortcomings: (1) the reference models are human based and provided manually (this work is absolutely error-prone and time-consuming) and (2) they are always studied as a whole while sometimes only some parts of the model need to be considered.

In this paper, we present an original recommendation approach to help the business processes designers facilitate the design step of the business process abstraction. We propose to take into account the composition context specified through the business process fragment surrounding the requested service, and benefit from the modeling and usage of previous service compositions to build our recommendations. Concretely, we propose a process fragment model that computes similarities between services using the relations with their neighbors. The process fragment represents the composition context for a service described in terms of its relations with its neighbors. These relations are described through the control flow patterns. Then, we compute the similarity degrees between services by matching the respective process fragments. Indeed, the composition context informs us about the service behavior and thereafter can unveil its functionality. Therefore, our objective is two-fold: (1) takes into account the composition context, specified through the business process fragment surrounding the composed service, as an input in service discovery, and (2) benefits from the modeling and usage of existing composite services by extracting this implicit knowledge as process fragments to match with the composition context of the requested service. Furthermore, our approach can associate with the functionality-based service recommendation techniques to more precisely retrieve the expected services.

The paper is organized as follows: In section 2 we specify a graph based model to describe a service composition context. Section 3 elaborates the proposed matching algorithm. Section 4 illustrates our implementation and experimental results. Related work is presented in section 5 and we conclude our work in section 6.

2 Graph-Based Modeling of Service Composition Context

In this section, we present a graph-based service composition model whose control flow is modeled using workflow patterns. Firstly, we depict how we use workflow patterns to describe service interactions (see section 2.1). Secondly, we define the relations of each service with its neighbors using new defined layer and zone concepts (see section 2.2). Finally, we specify the composition context graph of each service (see section 2.3).

2.1 Graph-Based Service Composition Model

It is worthwhile to notice that the term composite service is usually used to denote composition of operations offered by different services [2]. Indeed, a composite service, defined using WS-BPEL for instance, is in fact a flow of services'

operations and not a flow of services. Thus, in our approach and in order to avoid such confusion, we supposed for simplicity purpose that a service has one operation so that its consumption coincides with its operation invocation.

A composite service implies several services and describes the order of their invocation, and the conditions under which these services are invoked. The control flow (or skeleton in the following) of a composite service specifies the partial ordering of component services (e.g., a service B is executed after the completion of a service A). We use (workflow-like) patterns to define a composite service skeleton. A workflow pattern [21] can be seen as an abstract description of a recurrent class of interactions. Applied to Web services, a pattern defines default dependencies (i.e. interactions) between services. For example, the Synchronize pattern [21] describes an abstract choreography by specifying services dependencies as following: a service is activated after the completion of several other services. We call *atomic pattern* a primitive control flow pattern that can be used in WS-BPEL such as : sequence, parallel split (AND-fork), synchronization (AND-join), multiple choice (OR-fork), an exclusive choice (XOR-fork), or a simple merge (OR-join). In the following we propose a graph-based model of service composition and we use Fig. 1 for all examples in our definitions.

Definition 1 (Direct link pattern). *A direct link pattern is a sequence of atomic patterns which connects two adjacent services. The direct link pattern is directed, and denoted by P.* $P_{\mathcal{C}}(s_i, s_j) = p_1 p_2 \dots p_k$ *indicates a direct link pattern with a sequence of k atomic patterns from* s_i *to* s_j *in the composition* $\mathcal{C}$.

For example, $P_{\mathcal{C}1}(s_1, s_2)$='Sequence', $P_{\mathcal{C}2}(s_3, s_4)$='OR-join''AND-join', $P_{\mathcal{C}2}(s_6, s_1)$ ='' (there is no direct link pattern from s_6 to s_1 in $\mathcal{C}2$).

Definition 2 (Composition graph). *A composition graph of a service composition* $\mathcal{C}$ *is a labeled directed graph* $G_{\mathcal{C}} = (V_{\mathcal{C}}, L_{\mathcal{C}}, A_{\mathcal{C}})$ *in which* $V_{\mathcal{C}} \neq \emptyset$ *is the set of vertices (services),* $L_{\mathcal{C}} \neq \emptyset$ *is the set of edge-labels (direct link patterns' names), and* $A_{\mathcal{C}} \subseteq V \times V \times L$ *is the set of directed edges (direct link patterns) in the composition* $\mathcal{C}$. *An edge* $a =< s_x, s_y, P_{\mathcal{C}}(s_x, s_y) > \in A_{\mathcal{C}}$ *is considered to be directed from* s_x *to* s_y *and labeled by* $P_{\mathcal{C}}(s_x, s_y)$. s_x *is called the tailed service,* s_y *is called the head service and* $P_{\mathcal{C}}(s_x, s_y)$ *is the direct link pattern from* s_x *to* s_y *in* $\mathcal{C}$.

For example, the composition $\mathcal{C}1$ can be presented by a graph $G_{\mathcal{C}1} = (V_{\mathcal{C}1}, L_{\mathcal{C}1}, A_{\mathcal{C}1})$, in which $V_{\mathcal{C}1}$={s_0, s_1, s_2, s_3, s_4}, $L_{\mathcal{C}}$={'AND-split', 'Sequence', 'AND-join'}, $A_{\mathcal{C}1}$={$< s_0, s_1$, 'AND-split'$>$, $< s_0, s_3$, 'AND-split'$>$, $< s_1, s_2$, 'Sequence'$>$, $< s_2, s_4$, 'AND-join'$>$, $< s_3, s_4$, 'AND-join'$>$}.

A path in a composition graph is named as a *pattern path.* A pattern path from s_i to s_j in a composition $\mathcal{C}$ is *indirected* and denoted by $\mathbb{P}_{\mathcal{C}}(s_i, s_j)$. For example, $\mathbb{P}_{\mathcal{C}1}(s_1, s_4) = P(s_1, s_2)P(s_2, s_4)$='Sequence''AND-join', $\mathbb{P}_{\mathcal{C}2}(s_3, s_1) = P(s_5, s_3)P(s_5, s_1)$='AND-split''OR-split''AND-split'. The *pattern path's length* is denoted by $Len(\mathbb{P})$ and the *shortest pattern path* is denoted by $SP_{\mathcal{C}}(s_i, s_j)$. For example, $SP_{\mathcal{C}1}(s_0, s_2) = P(s_0, s_1)P(s_1, s_2)$='AND-split''Sequence', $SP_{\mathcal{C}2}(s_7, s_6) = P(s_7, s_4)P(s_6, s_4)$='OR-join''AND-join''AND-join'.

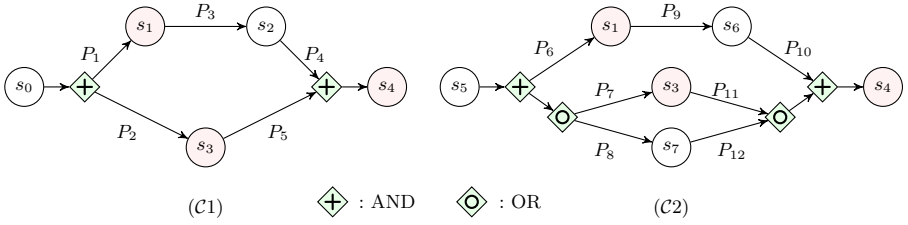


Fig. 1. Example: service compositions

2.2 Service Neighborhood

In this section, we propose some new definitions that are related to services' neighborhood and are used to define the composition context.

Definition 3 (k^{th}-layer neighbor). *A k^{th}-layer neighbor of a service s is a service connected to s via a k-length pattern path ($k \geq 0$). The set of k^{th}-layer neighbors of a service s in a composition $\mathcal{C}$ is denoted by $N^k_{\mathcal{C}}(s)$. $N^0_{\mathcal{C}}(s) = \{s\}$;*

For example, $N^1_{\mathcal{C}1}(s_2) = \{s_1, s_4\}$, $N^2_{\mathcal{C}1}(s_2) = \{s_0, s_3\}$.

Definition 4 (k^{th}-area neighbor). *A k^{th}-area neighbor of a service s is a service connected to s via a l-length pattern path, where $0 \leq l \leq k$. The set of all k^{th}-area neighbors of s in a composition $\mathcal{C}$ creates a process fragment surrounding s and it is denoted by $\mathbb{N}^k_{\mathcal{C}}(s)$. $\mathbb{N}^k_{\mathcal{C}}(s) = \cup_{i=0}^{k} N^i_{\mathcal{C}}(s)$.*

For example, $\mathbb{N}^1_{\mathcal{C}1}(s_2) = \{s_2, s_1, s_4\}$; $\mathbb{N}^2_{\mathcal{C}1}(s_2) = \{s_2, s_1, s_4, s_0, s_3\}$.

Definition 5 (k^{th}-zone pattern). *A k^{th}-zone pattern of a service $s \in \mathcal{C}$ is a direct link pattern which connects a service in $N^{k-1}_{\mathcal{C}}(s)$ and a service in $N^k_{\mathcal{C}}(s)$. Set of all k^{th}-zone patterns of a service $s \in \mathcal{C}$ is denoted by $Z^k_{\mathcal{C}}(s)$. $Z^0_{\mathcal{C}}(s) = \emptyset$.*

For example, $Z^1_{\mathcal{C}1}(s_1) = \{< s_0, s_1,$ 'AND-split'$>, < s_1, s_2,$ 'Sequence'$>\}$, $Z^2_{\mathcal{C}2}(s_4)$ $= \{< s_1, s_6,$ 'Sequence'$>, < s_5, s_3,$ 'AND-split''OR-split'$>, < s_5, s_7,$ 'AND-split' 'OR-split'$>\}$

2.3 Service Composition Context Graph

We realize that the pattern paths between two services present their relation (closeness). The longer the pattern path is, the weaker their relation is. And if we capture the shortest pattern paths from a service to other, we can measure the best relation between them. On another hand, one edge in the composition graph can belong to more than one zone around a service, depending on the selected pattern paths to the that service. Therefore, we propose to build for each service a specific graph in which each edge belongs to its smallest zone. In other words, we propose to assign the smallest zone number for each direct link patterns computed by the shortest path's length to the associated service and represent the composition graph in another graph, so-call *composition context*

graph (definition 6). Concretely, the minimum zone value that is assigned to the pattern connecting s_i and s_j in the composition context graph of s will be $Min(Len(SP_{\mathcal{C}}(s_i, s)), Len(SP_{\mathcal{C}}(s_j, s))) + 1$ and the maximum zone value used to assign to all direct link patterns will be $n = Max(Len(SP_{\mathcal{C}}(s_x, s))) + 1 \; \forall s_x \in \mathcal{C}$.

Definition 6 (Composition context graph). *A composition context graph of a service $s \in \mathcal{C}$ is a labeled directed graph $G_{\mathcal{C}}(s) = (V_{\mathcal{C}}(s), Z_{\mathcal{C}}(s), L_{\mathcal{C}}(s), A_{\mathcal{C}}(s))$ that represents the composition graph $G_{\mathcal{C}} = (V_{\mathcal{C}}, L_{\mathcal{C}}, A_{\mathcal{C}})$ with the minimum k^{th}-zone patterns of s. $V_{\mathcal{C}}$ is the set of vertices (services), $Z_{\mathcal{C}}(s)$ is the minimum set of zones needed to represent the composition graph, $L_{\mathcal{C}}$ is the set of direct link patterns' names and $A_{\mathcal{C}}(s))$ is the set of direct link patterns labeled with their zone numbers. A composition context graph $G_{\mathcal{C}}(s)$ satisfies the followings:*

1. $V_{\mathcal{C}}(s) = V_{\mathcal{C}}$
2. $L_{\mathcal{C}}(s) = L_{\mathcal{C}}$
3. $Z_{\mathcal{C}}(s) = \{1, 2, \ldots, n\}$, *where:*
   $n = Max(Len(SP_{\mathcal{C}}(s_x, s))) + 1 \; \forall s_x \in \mathcal{C}$
4. $A_{\mathcal{C}}(s) = A_{\mathcal{C}} \times Z_{\mathcal{C}}(s)$, *where:*
   $a_s =< a_c, z_c(s) >=<< s_i, s_j, P(s_i, s_j) >, Min(Len(SP_{\mathcal{C}}(s_i, s)), Len(SP_{\mathcal{C}}(s_j, s))) + 1 >, \forall a_c =< s_i, s_j, P(s_i, s_j) > \in A_{\mathcal{C}}$

For example, a composition context graph $G_{\mathcal{C}1}(s_2) = (V_{\mathcal{C}1}(s_2), Z_{\mathcal{C}1}(s_2), L_{\mathcal{C}1}(s_2), A_{\mathcal{C}1}(s_2))$ of the service s_2 can be inferred from the composition graph $G_{\mathcal{C}1}$ (in Fig. 1), in which: $V_{\mathcal{C}1}(s_2) = \{s_0, s_1, s_2, s_3, s_4\}$, $L_{\mathcal{C}} = \{$'AND-split', 'Sequence', 'AND-join'$\}$, $Z_{\mathcal{C}1}(s_2) = \{1,2,3\}$, $A_{\mathcal{C}}(s) = \{<< s_0, s_1,$ 'AND-split'$>, 2 >, << s_0, s_3,$ 'AND-split'$>, 3 >, << s_1, s_2,$ 'Sequence'$>, 1 >, << s_2, s_4,$ 'AND-join'$>, 1 >, << s_3, s_4,$ 'AND-join'$>, 2 >\}$.

In graphical view, the composition context graphs $G_{\mathcal{C}1}(s_2)$ and $G_{\mathcal{C}2(s_6)}$ of the service s_2 and s_6 can be shown as in Fig. 2. We note that a composition context graph of a service is related to the composition where this service is used, so this composition context graph can differ from one composition to another. For example, the composition context graph of s_3 in $\mathcal{C}1$ is different to the composition context graph of s_3 in $\mathcal{C}2$, ie. $G_{\mathcal{C}1}(s_3) \neq G_{\mathcal{C}2}(s_3)$.

3 Service Recommendation Based on Composition Context Matching

To illustrate each step in the computation, we use an example to compute the pattern matching of two services s_2 and s_6 which respectively belong to the composition $\mathcal{C}1$ and $\mathcal{C}2$ (Fig. 1). The services s_1, s_3, s_4 exist in both compositions. The direct link patterns are: $P_1 = P_2 =$ 'AND-fork', $P_3 =$ 'Sequence', $P_4 = P_5 =$ 'AND-join', $P_6 =$ 'AND-fork', $P_7 = P_8 =$ 'AND-fork''OR-fork', $P_9 =$ 'Sequence', $P_{10} =$ 'AND-join', $P_{11} = P_{12} =$ 'OR-join''AND-join'. The distributions of neighbors of s_2 and s_6 in layers are easily inferred from the compositions and redrawn in Fig. 2. We elaborate step by step how we compute the similarity in the following.

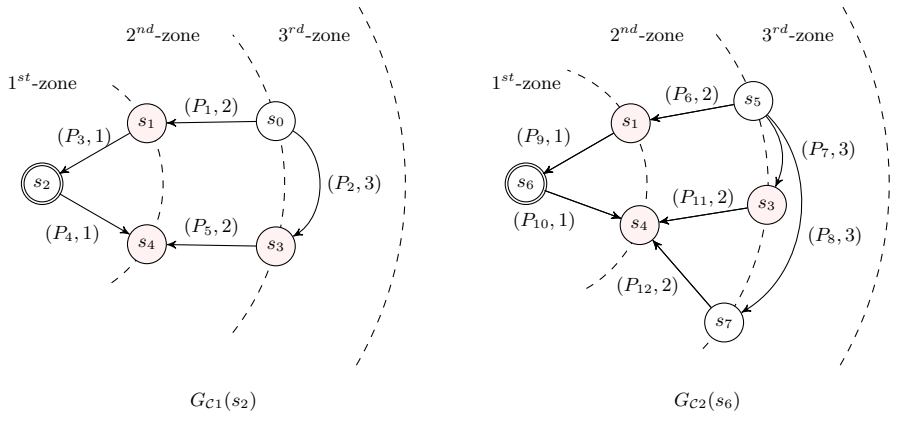


Fig. 2. Composition context graphs inferred from the compositions graph

3.1 Direct Link Pattern Matching

In our approach, the direct link pattern matching is considered as the fundamental weight of the composition context matching. Since each direct link pattern between two adjacent services is a sequence of atomic patterns which can easily mapped to a sequence of characters, we propose to use the Levenshtein distance [13] to compute the matching between two direct link patterns. We consider each atomic pattern as a character, then *a direct link pattern is presented by a sequence of characters* (or string) and the similarity between two direct link patterns can be easily computed.

Concretely, given two direct link patterns $P(s_i, s_j) = p_1 p_2 \dots p_n$ and $P'(s_{i'}, s_{j'}) = p'_1 p'_2 \dots p'_m$, the pattern matching between them is computed by the equation (1).

$$M_p(P, P') = 1 - \frac{LevenshteinDistance(P, P')}{Max(n, m)} \tag{1}$$

The equation (1) also covers two special cases:

① If $P(s_i, s_j) = P(s_{i'}, s_{j'})$,ie.$(m = n) \wedge (p_t = p'_t, \forall t \in [1, n])$,then $M_p(P, P') = 1$
② If $P(s_i, s_j) \subset P(s_{i'}, s_{j'})$, ie. $\exists k < (m - n) : p_t = p'_{(k+t)}, t = \overline{1..n}$, then $M_p(P, P') = \frac{n}{m}$

Since a service in a composition has either the incoming direct link patterns from its precedent services or the outgoing direct link patterns to its following services, *we take into account the direct link pattern's directions* in our computation. Concretely, to compute the direct link pattern matching between s_i and s_j, we match incoming direct link patterns of s_i to incoming direct link patterns of s_j and outgoing patterns of s_i to outgoing direct link patterns of s_j then we sum these matching results to get the final matching value. *The matching between*

two direct link patterns that have inverse directions is equal to 0, which means if $P(s_i, s_j)$ and $P(s_{i'}, s_{j'})$ are two direct link patterns from s_i to s_j and $s_{i'}$ to $s_{j'}$ respectively, then $M_p(P(s_i, s_j), P(s_{j'}, s_{i'})) = M_p(P(s_j, s_i), P(s_{i'}, s_{j'})) = 0$.

In our example, when the direct link patterns are mapped to sequences of characters, we have $M_p(P_3, P_9) = M_p$('Sequence','Sequence')=1; $M_p(P_4, P_{10}) = M_p$('AND-join','AND-join')=1; $M_p(P_5, P_{11}) = M_p$('AND-join','OR-join''AND-join')=0.5, and so on.

3.2 Composition Context Matching

The k^{th}-zone neighbors of a service create a fragment composition around it. This fragment contains the composition context of the associated service within k layers. In our approach, we propose to compute the similarity between two services based on the matching of their composition context. Concretely, to compute the similarity between two services s_i and s_j, we *match all direct link patterns that belong to the same zone and are ended by either* s_i *or* s_j *or the same services*. By this way, our approach *captures latently the service matching* of two compositions, *focuses only on the related services* and *avoids the time-consuming problem* of redundant matching computations. In our illustrated example, we will match (P_3 and P_9), (P_4 and P_{10}), (P_5 and P_{11}) as they have the same ending services, not (P_1 and P_6) or other pairs.

In formula, suppose that $a =<< s_x, s_y, P_{\mathcal{C}m}(s_x, s_y) >, z >$ is the edge connecting s_x and s_y by the direct link pattern $P_{\mathcal{C}m}(s_x, s_y)$ belongs to zone z in the composition context graph $G_{\mathcal{C}m}(s_i)$, $a \in V_{\mathcal{C}m}(s_i)$. Similarly, $a' =<< s_{x'}, s_{y'}, P_{\mathcal{C}n}(s_{x'}, s_{y'}) >, z' > \in V_{\mathcal{C}n}(s_j)$. The composition context matching of s_i and s_j within k^{th}-area with the direction consideration is given by Equation (2).

$$\mathbb{M}^k_{\mathcal{C}a,\mathcal{C}b}(s_i, s_j) = \frac{\sum_{a \in V_{\mathcal{C}m}(s_i)} \sum_{a' \in V_{\mathcal{C}n}(s_j)} M^*(a, a')}{|Z^k_{\mathcal{C}m}(s_i)|} \tag{2}$$

in which, $M^*(a, a') = M_p(P_{\mathcal{C}m}(s_x, s_y), P_{\mathcal{C}n}(s_{x'}, s_{y'}))$ in cases:

① $(z = z' = 1) \wedge ((s_x = s_i \wedge s_{x'} = s_j \wedge s_y = s_{y'}) \vee (s_x = s_{x'} \wedge s_y = s_i \wedge s_{y'} = s_j))$
② $(1 < z = z' \leq k) \wedge (s_x = s_{x'}) \wedge (s_y = s_{y'}))$

and $M^*(a, a') = 0$ in other cases.

We can easily check that, $\mathbb{M}^k_{\mathcal{C}a,\mathcal{C}b}(s_i, s_j)$ is different from $\mathbb{M}^k_{\mathcal{C}b,\mathcal{C}a}(s_j, s_i)$, and if $Z^k_{\mathcal{C}m}(s_i) \subseteq Z^k_{\mathcal{C}m}(s_j)$, $\mathbb{M}^k_{\mathcal{C}a,\mathcal{C}b}(s_i, s_j)$ will be equal to 1, which means if all patterns from s_i to its k^{th}-layer neighbors are patterns from s_j to its k^{th}-layer neighbors, s_j will be absolutely able to replace s_i.

The k^{th}-area neighbors of a service s create a process fragment surrounding s, which is presented by a sub composition graph. Therefore, the matching problem becomes the graph matching problem which was proved to be a NP-complexity problem [1]. However, in our case, we know the root points of the graph comparison, which are s_i and s_j, and we match only the same pairs of services in both composition graphs, thus *we avoid the NP-complexity problem* of the original

graph matching. In another hand, with the composition context graph definition and the direct link patterns presentation in zones, we can compute the composition context matching of any pair of services in compositions. Moreover, direct link patterns locates in the closest zones to the associated service. Therefore, our algorithm run smoothly and returns very high quality results.

The computation given by equation (2) can generate recommendations regardless the zones that a pattern path belongs to. However, in reality, the behavior of a service is strongly reflected by the direct link patterns to its closet neighbors while the interactions among other neighbors in the higher layers do not heavily affect its behavior. Therefore, the impact of k^{th} zones needs to be examined and we propose to assign a weight (w_k) for each k^{th}-zone, so called zone-weight and integrate this parameter into our computation. Since the zone-weight has to have greater values in smaller k zones, we propose to assign the zone-weight a value computed by a polynomial function which is given by equation (3).

$$w_z = \frac{k+1-z}{k} \tag{3}$$

where z is the zone number ($1 \leq z \leq k$), k is the number of considered zones around the service. All direct link patterns connect either to or from the associated service has the greatest weight ($w_1 = 1$) and the direct link patterns connect to/from services in the furthest zone has the smallest weight ($w_k = 1/k$).

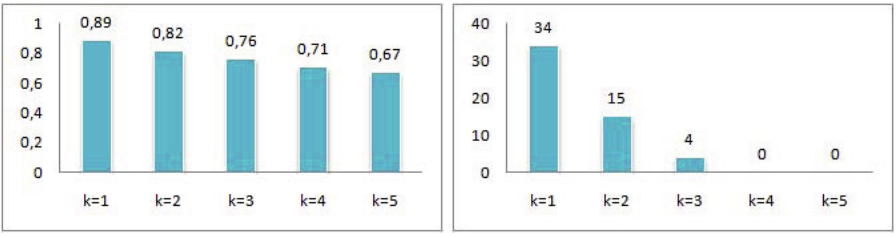
With the zone's weight consideration, the pattern matching computation on two services $s_i \in V_{\mathcal{C}a}$ and $s_j \in V_{\mathcal{C}b}$ is the combination of composition context matching (the equation(2)) and the zone-weight impact (the equation(3)), which is given by the equation (4).

$$\mathcal{M}^k_{\mathcal{C}a,\mathcal{C}b}(s_i, s_j) = \frac{2}{k+1} \times \sum_{z=1}^{k} \frac{\sum_{a.z=a'.z'=z} \frac{k+1-z}{k} \times M^*(a, a')}{|Z^z_{\mathcal{C}m}(s_i)| - |Z^{z-1}_{\mathcal{C}m}(s_i)|} \tag{4}$$

where $|Z^z_{\mathcal{C}m}(s_i)| - |Z^{z-1}_{\mathcal{C}m}(s_i)|$ is the number of direct link patterns in the zone z^{th} of $G_{\mathcal{C}m}(s_i)$ (see Definition 5). In case $|Z^z_{\mathcal{C}m}(s_i)| - |Z^{z-1}_{\mathcal{C}m}(s_i)| = 0$, which means there is no direct link pattern in the zone z^{th} of $G_{\mathcal{C}m}(s_i)$, $M^*(a, a')$ is also equal to 0, we consider the fraction $\frac{0}{0}$ as 0.

Return to our example with the zone-weight consideration, in case $k = 1$, $t = 0$, only the nearest neighbors are taken into account, the zone-weight does not affect the results, therefore, the matching values are the same to the case that we do not take into account the zone-weight, which means $\mathcal{M}^1_{\mathcal{C}1,\mathcal{C}2}(s_2, s_6)$=$\mathbb{M}^1_{\mathcal{C}1,\mathcal{C}2}(s_2, s_6)$ and $\mathcal{M}^1_{\mathcal{C}2,\mathcal{C}1}(s_6, s_2)$=$\mathbb{M}^1_{\mathcal{C}2,\mathcal{C}1}(s_6, s_2)$. In case $k = 2$, we have: $\mathcal{M}^2_{\mathcal{C}1,\mathcal{C}2}(s_2, s_6) = (1/3) \times (2 \times (M_p(P_{i3}, P_{j3}) + M_p(P_{i4}, P_{j4})/2) + (M_p(P_{i5}, P_{j5})/2)) = 0.75$ and $\mathcal{M}^2_{\mathcal{C}2,\mathcal{C}1}(s_6, s_2) = (1/3) \times (2 \times (M_p(P_{j3}, P_{i3}) + M_p(P_{j4}, P_{i4})/2) + (M_p(P_{j5}, P_{i5})/3)) \approx 0.72$.

The matching values among services present their similarity and they are used to make recommendation. For a selected service, we pick up top-n services which have the highest matching values to recommend it. In our experiments, we recommend the top-5 services for each one.



(a) Average matching values **(b)** Services with matching value=1

Fig. 3. Statistics of matching values with different k^{th}-zone

4 Implementation and Experiments

We tried to get experiments from real business process abstracts or web service compositions but finding proper datasets for our approach is really a big challenge. We searched from the Internet in parallel with asking other researchers on our domain but unfortunately we did not found any good dataset or the datasets are under non-disclosure agreements. Finally, we decided to collect manually the scattered business processes and web service compositions (focus on car rental, car selling and travel booking business context) on previous contributions, re-engineer them to be used by our application and added some business processed designed by ourselves. On synthesis of the collected data, we got a database with 46 web services and 27 compositions. The largest composition consists of 14 services and the smallest composition consists of 4 services (in average, 6.8 services per composition). Our application[1] is implemented as a Java applet in order to allow public users adding web services, creating compositions and get recommendations based on our proposed model.

We experiment on asserting *the impact of k^{th}-zones on the composition context matching*. For each service, we computed its similarity values with others and selected the most related service which had the highest similarity value. We run the proposed model with different k^{th}-zones. The results showed that the average matching values computed by our algorithm are very high and decrease when k increases (Fig. 3a). When k increases, services and patterns in the further zones are taken into account and in most cases, the matching among these patterns in comparison to the number of patterns in the further zones is lower than the matching in the nearer zones, therefore, it decreases the final matching values.

The k^{th}-zone also affects the number of services which are retrieved with the highest matching values. When k increases, the number of services which are retrieved with high similarity values decreases and the number of services which are retrieved with the lower similarity values increases. Fig. 3b show the distribution of matching values with different k^{th}-zones. With $k = 1$, 34 services were recommended with the similarity equal to 1 (Fig. 3b). When k increases, the unmatched patterns in the further zones around the associated services reduce

[1] `http://www-inf.int-evry.fr/SIMBAD/tools/CMSR/`

the similarity values and the number of services which were retrieved by the highest similarity values also decreased. However, since the nearest neighbors to a service have the greatest weight, most of the results are stable when k increases (Fig. 4).

In practice, the behavior of a service is reflected by its connections to the nearest neighbors (1^{st}-zones). The relations to its higher k^{th}-zone ($k > 1$) neighbors have less impact to its behavior. In our approach, we target to recommend services for business process design, not for an agnostic use. Therefore, widening the k^{th}-zones allows reducing the searching space by taking into account richer composition contexts and get the closer results to the required services. The more zones to a services are considered, the better candidates are retrieved.

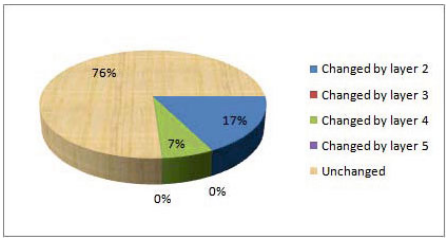


Fig. 4. Statistics of retrieved services

5 Related Work

In recent years, there are many efforts on helping the business process designers to faster and more accurately create new business process models using available reference models. They proposed either to rank the existing business process models in the repository for searching [8,22], or to measure the similarity between them [10,19,14] to create new process models. Some of them [8,22] encountered the NP-complexity graph matching problem and they have to find a compromise between computation complexity and quality of results. Different from them, our approach focused partially on the business process, described as a service composition, to take into account just the composition context to retrieve the most related services. Therefore, we do not face the NP-complexity problem (as we explained in section 3.2). In another hand, we focus on matching a partial context instead of matching the whole business process.

Another contribution that aims at refining business process reference by merging existing process models was recently presented by M.L. Rosa et. al.[16]. However, different from our work, they did not do further to help process designers and users with recommendations. D. Schleicher et. al. [17] also presented an approach based on so-called compliance templates to develop and manage compliant business processes involving different stakeholders. Their work targeted at refining the business process in layers using compliance constraints. These constrains can be composition constraints although the authors did not mention how they can be inferred. In our paper, we propose to implicitly extract them.

Thomas Gschwind et. al. [11] also applied workflow patterns for business process design. They aimed at helping business users understand the context and apply patterns during the editing process. In our work, we help the business users better design a business process by automatically providing them the most related services instead of patterns.

In the web service discovery domain, many solutions for faster reaching to the desired services were also proposed. Most of them based on the traditional service descriptions (WSDL documents) and targeted at finding the similarity between requests from users (query, profile, historic data, etc) and data stored in service repositories. They generated recommendations based on the text processing, including query analysis and web service description similarity. They can be classified in the categories: clustering [9], rating based [15], words analysis [3] and vector space model [12]. Since these approaches are text-based, they can encounter the synonym and polysenym problems (one word can have many meanings and one meaning can be described by different words). In another hand, since they captured only the explicit knowledge described in WSDL files, they lack the implicit knowledge which can be inferred from past usage data.

Our previous contributions [5,4,6] on proposing a web service recommender system based on user's behavior can overcome the shortcomings of the text-based analysis systems. We solved the problem from the user's side and we can provide good recommendations which are close to user's behaviors. However, in our previous work, we did not take into account the relations among web services in compositions. We fulfilled this shortcoming in this work.

Last but not least, it is worth to notice that our approach can associate with the functionality-based service recommendation techniques to generate more precise recommendations since the service connections to its neighbors do not infer fully its functionality. Our approach can be applied as preprocessing step either in the design phase to limit the search space or later in the execution phase to filter the selected recommended services.

6 Conclusion and Future Work

In this paper, we propose an original approach to capture the composition context to generate process design recommendation. Since this composition context presents the requested service's behavior and they can implicitly infer the service's functionality, our approach performed well in recommending related services. Our approach retrieves not only the services for an incomplete abstract process but also possibly the replaceable services to a selected one. It can be very useful in helping the composition designers and managers find suitable services to replace a vulnerable one to enhance the availability of the service compositions. In our future work, we intend to investigate the co-existence of patterns in compositions, as well as the number of time that a web service is used in oder to refine our matching algorithm. We also aim at extending our approach to infer existing service composition from log execution.

Acknowledgement. The work presented in this paper is supported by the ANR French funding under the PAIRSE project.

References

1. Abdulkader, A.M.: Parallel Algorithms for Labelled Graph Matching. PhD thesis, Colorado School of Mines, USA (1998)
2. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures and Applications. Springer, Berlin (2003)
3. Blake, M.B., Nowlan, M.F.: A web service recommender system using enhanced syntactical matching. In: ICWS, pp. 575–582 (2007)
4. Ngoc Chan, N., Gaaloul, W., Tata, S.: Collaborative filtering technique for web service recommendation based on user-operation combination. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 222–239. Springer, Heidelberg (2010)
5. Chan, N.N., Gaaloul, W., Tata, S.: Web services recommendation based on user's behavior. In: ICEBE, pp. 214–221 (November 2010)
6. Chan, N.N., Gaaloul, W., Tata, S.: A web service recommender system using vector space model and latent semantic indexing. In: AINA, pp. 602–609 (2011)
7. Curran, T., Keller, G., Ladd, A.: SAP R/3 business blueprint: understanding the business process reference model. Prentice-Hall, Inc., NJ (1998)
8. Dijkman, R., Dumas, M., García-Bañuelos, L.: Graph matching algorithms for business process model similarity search. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 48–63. Springer, Heidelberg (2009)
9. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: VLDB, pp. 372–383 (2004)
10. Ehrig, M., Koschmider, A., Oberweis, A.: Measuring similarity between semantic business process models. In: APCCM 2007, pp. 71–80 (2007)
11. Gschwind, T., Koehler, J., Wong, J.: Applying patterns during business process modeling. In: Dumas, M., Reichert, M., Shan, M.-C. (eds.) BPM 2008. LNCS, vol. 5240, pp. 4–19. Springer, Heidelberg (2008)
12. Kokash, N., Birukou, A., D'Andrea, V.: Web service discovery based on past user experience. In: Abramowicz, W. (ed.) BIS 2007. LNCS, vol. 4439, pp. 95–107. Springer, Heidelberg (2007)
13. Levenshtein, V.: Binary Codes Capable of Correcting Deletions, Insertions and Reversals. Soviet Physics Doklady 10, 707 (1966)
14. Li, C., Reichert, M., Wombacher, A.: On measuring process model similarity based on high-level change operations. In: Li, Q., Spaccapietra, S., Yu, E., Olivé, A. (eds.) ER 2008. LNCS, vol. 5231, pp. 248–264. Springer, Heidelberg (2008)
15. Manikrao, U.S., Prabhakar, T.V.: Dynamic selection of web services with recommendation system. In: NWESP 2005, p. 117. IEEE Computer Society, Washington (2005)
16. La Rosa, M., Dumas, M., Uba, R., Dijkman, R.: Merging business process models. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 96–113. Springer, Heidelberg (2010)
17. Schleicher, D., Anstett, T., Leymann, F., Schumm, D.: Compliant business process design using refinement layers. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 114–131. Springer, Heidelberg (2010)
18. Stephens, S.: Supply chain operations reference model version 5.0: A new tool to improve supply chain efficiency and achieve best practice. Information Systems Frontiers 3, 471–476 (2001)

19. van der Aalst, W.M.P., de Medeiros, A.K.A., Weijters, A.J.M.M.T.: Process equivalence: Comparing two process models based on observed behavior. In: Dustdar, S., Fiadeiro, J.L., Sheth, A.P. (eds.) BPM 2006. LNCS, vol. 4102, pp. 129–144. Springer, Heidelberg (2006)
20. van der Aalst, W.M.P., Dumas, M., Gottschalk, F., ter Hofstede, A.H.M., Rosa, M.L., Mendling, J.: Preserving correctness during business process model configuration. Formal Asp. Comput. 22(3-4), 459–482 (2010)
21. Van Der Aalst, W.M.P., Ter Hofstede, A.H.M., Kiepuszewski, B., Barros, A.P.: Workflow patterns. Distrib. Parallel Databases 14(1), 5–51 (2003)
22. Yan, Z., Dijkman, R., Grefen, P.: Fast business process similarity search with feature-based similarity estimation. In: Meersman, R., Dillon, T.S., Herrero, P. (eds.) OTM 2010. LNCS, vol. 6426, pp. 60–77. Springer, Heidelberg (2010)