Discovering Linkage Patterns among Web Services using Business Process Knowledge

Mohammed AbuJarour
Information Systems Group
Hasso-Plattner-Institut
University of Potsdam, Germany
mohammed.abujarour@hpi.uni-potsdam.de

Ahmed Awad
Business Process Technology Group
Hasso-Plattner-Institut
University of Potsdam, Germany
ahmed.awad@hpi.uni-potsdam.de

***Abstract*—Discovering relations among web services has been a fruitful research topic in services computing. Such relations are useful for service discovery, selection, composition, etc. The much information about web services is available, the better/more relations can be discovered. Based on the *technical* information provided in `WSDL` files, simple relations can be discovered. Finding more comprehensive relations requires additional information, such as semantic descriptions, information about service consumers, or service compositions. Rich semantic service descriptions are not so common in practice. Experiments have shown that similar users do not necessarily use the same web services. Using service compositions gives good results, however, researchers assume that they have access to such compositions, which is not the typical case in practice. Additionally, using service compositions is not sufficient to determine types and strengths of such relations. In this work, we propose a novel approach to discover relations among web services in the form of linkage patterns based on the configurations of business processes that use them. We specify types and weights of the discovered linkage patterns based on control flow patterns in business processes. We have implemented this approach using Oryx, a process modeling tool and repository, and Depot, a service registry, and validated it through real-world examples, as we show in this paper.**

***Keywords*-Service Discovery, Service Recommendation**

I. Introduction: Web Service Discovery

Service providers represent the main source of information about their offered web services in the form of service descriptions. Such descriptions are crucial in several tasks in services computing, such as service discovery, selection, ranking, substitution, recommendation, etc. Nevertheless, it has been observed that service providers usually release poor service description [1]. Typically, service descriptions are technical-oriented and their documentations appear as comments and notes made by service developers [2]. Additional factors that exacerbate this problem include the increasing number of available web services, the increasing complexity of existing web services, and the dynamic nature of business and IT landscapes [3].

One of the main approaches to alleviate the impacts of poor service descriptions on their usage is discovering relations among web services. These relations help service consumers find web services that match their business needs efficiently. The traditional techniques that have been used to achieve this task are input-output matching of web services using their technical service descriptions [4], semantic approaches [5], using compositions of web services [6], and using consumer-consumer similarity [7] (Cf. Sec. II).

It has been observed that the assumption of the existence of complete, correct, and rich service descriptions is not realistic in practice [8]. This fact limits the usability of such approaches to discover relations among web services using these descriptions. Additionally, studies have shown that similar service consumers do not usually use the same web services. This fact affects the discovered relations among web services using consumer-consumer similarity. Moreover, relations discovered from service compositions usually lack types or strengths.

In this work, we propose an approach to discover *additional* realistic and useful relations among web services in the form of linkage patterns using business processes (BP) that use them. We are able to identify five types of such linkage patterns: predecessors, successors, similar, complementary, and related web services. Additionally, we assign weights to these linkage patterns to reflect their strengths. Weights of predecessor and successor relations are based on the (shortest) distance between the tasks that use these web services in the corresponding BPs. Similarity between terms of tasks' labels and operations' names are used to give weights to the linkage pattern "similar " using WordNet. The other types of linkage patterns are assigned the weight 1. These weights are then used to rank lists of recommended web services in each of the five types, e.g., rank predecessor web services to a particular web service.

The main contributions of this work are:

1) Finding realistic linkage patterns among web services with fine-grained types and weights
2) Providing sources to rank recommended web services
3) Disambiguating exclusive relations between web services using lexical ontologies, e.g., WordNet.

The rest of this paper is organized as follows: Related work is summarized in Sec. II. Our approach is presented in Sec. III. A detailed real-world example is given in Sec. IV. Further implementation details and usage scenarios are given in Sec. V. We close this paper with a summary and future work in Sec. VI.

II. Related Work

In this section, we summarize common approaches to discover relations between web services. Then, we give a brief overview of the relationship between business processes and web services.

A. Finding Relations among Web Services

Finding relations among web services has been considered by several researchers in the community. Approaches that tackle this problem can be grouped roughly in four groups:

- **Input/output matching approach**: This approach matches inputs and outputs of operations of web services to find relations among them [4]. In this approach, the authors assume the existence of complete, rich, and correct service descriptions, e.g., `WSDL`. However, such assumption is not always realistic [8]. Additionally, experiments have shown that `WSDL` descriptions only are not sufficient [9]. This approach may lead to unrealistic/unusable relations, and misses relations between web services with *manual* tasks in between. The main goal of this approach is to investigate composability among web services [10], [11].
- **Semantic approach**: This approach applies Artificial Intelligence planning techniques to find valid compositions of web services [5], [12]. In this approach, they assume that web services are described formally using ontologies, such as `OWL-S`, `WSMO`, etc. In practice, semantic web services are not common [13]. Our approach does not necessarily require the existence of such formal descriptions. However, their existence could enable it to find additional relations.
- **Service compositions-based approach**: This approach is based on the idea that web services used in a service composition are related [6], [14]. In this approach, the authors assume that they have access to such compositions. This approach is not able to specify the type of relation between web services based on their usage in service compositions. It depends on the co-occurrence of web services in service compositions to decide that there is a relation among them. In our approach, we can determine types and weights of such relations using business process knowledge.
- **Consumer-consumer similarity approach**: This approach uses the idea that similar service consumers usually use the same web services [7]. However, this approach assumes the ability to track web services used by service consumers. This setting is not the typical one in practice. Moreover, it has been shown that similar service consumers do not necessarily use the same web services.

Additionally, researchers have shown that the assumption that service consumers can express their complex business needs using keywords for service discovery is not realistic [3]. Therefore, we consider such keywords as a starting point to help service consumers find their target web services. We track web services viewed by them and provide recommendations based on these web services. These recommendations include fine-grained types of relations among web services and are ranked according to their scores that reflect the strength of the relation. These recommendations enable service exploration, where services are inter-linked with relation similar to hyperlinks among webpages.

B. Web Services and Business Processes

Although Service-oriented Architecture (SOA) is seen as an implementation platform for business process models [15], only recently attention has been paid to bring those two worlds closer. On the one hand, business process models (BPM) have to be configured by assigning services to service tasks in order to be executable [16], [17]. On the other hand, service discovery can be enhanced with process configuration knowledge. We believe that an integrated view of both worlds is useful for both of them.

Buchwald et al. [17], propose an approach to bridge the gap between business process models and service compositions. The proposed approach introduces an intermediate layer between business process models (business view) and executable models, service compositions (technical view). The authors identify the need to store and maintain the relationship between business view tasks and technical view ones. To this point, the middle layer provides several types of transformation rules from the business to the technical view. However, this knowledge is kept in the middle layer and it is not the intention of that approach to reuse this knowledge to either enhance process modeling and/or service discovery.

The fact of having process repositories with hundreds to thousands of process models has attracted researchers to reuse-based process modeling. Smirnov et al. [18], use so-called behavioral profiles of business process model to extract association rules and action patterns among tasks. Based thereon, process modeling tools can suggest the user inserting certain tasks, if the user inserts other tasks within the model. Moreover, the approach can suggest a structuring relationship among the inserted tasks, e.g., tasks *A* and *B* should be exclusive to each other.

In this work, we make an explicit bi-directional link between business processes and web services. This link is used to discover fine-grained linkage patterns among web services used in BPs. The goal of this approach is to use these linkage patterns to enhance service discovery which reflects on the configuration of business process models.

III. Fine-grained Linkage Patterns

In our approach, we discover preliminary linkage patterns among web services from their WSDL files, and derive additional fine-grained ones using BPs that use them. Based on such business processes, we can discover more concrete relations such relations based on the usage of their web servicces in the corresponding BP. This weight is used to rank web services that have the same linkage pattern, e.g., rank web services that have the *predecessor* relation with a particular web service. Finding preliminary-relations among web services based on their WSDL descriptions is out of scope of this paper.

In this section, we describe the business process knowledge we use, and the types and weights of linkage patterns that we find based thereon.

A. Business Process Knowledge

Relevant information about a BP can be captured using the concept of behavioral profiles [19]. A behavioral profile represents an abstract description of a business process that identifies the behavioral relationship between any pair of its nodes. This relationship can be: (1) strict order $\rightsquigarrow$, (2) concurrent $\|$, (3) exclusive # or (4) inverse order $\leftsquigarrow$. The formal definition of behavioral profiles is introduced in Definiton 3.1.

Definition 3.1 (Behavioral Profile): Let N be the set of nodes within a business process model. The *behavioral profile* of a business process model is a function $bhp : N \times N \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#\}$ that assigns a behavioral property, strict order, inverse order, parallel, or exclusive, between each pair of nodes within the business process model.

If two tasks a, b appear in strict order, $bhp(a, b) = \rightsquigarrow$, then task a executes before task b. Similarly, if two tasks are concurrent then they can be executed in any order. Exclusiveness means that at most one of the two tasks can execute within a process instance. The behavioral profile of the BP shown in Figure 1 includes several behavioral properties, such as: $bhp(U, V) = \#$, $bhp(W, X) = \rightsquigarrow$, $bhp(X, W) = \leftsquigarrow$, $bhp(Y, Z) = \|$, $bhp(U, X) = \rightsquigarrow$, $bhp(Y, V) = \leftsquigarrow$, etc.

The definition of behavioral profiles is not sufficient to achieve our goal, in particular, assigning weights to the discovered linkage patterns. Therefore, we extend it by incorporating the shortest distance between each pair of tasks in addition to their behavioral property. The formal definition of the extended behavioral profile is given in Definition 3.2.

Definition 3.2 (Extended Behavioral Profile): Let N be a set of nodes within a business process model. The *extended behavioral profile* of a business process model is a function $bhp' : N \times N \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#\} \times \mathbb{N}$ that assigns a behavioral property, strict order, inverse order, parallel, or exclusive, and a distance, between each pair of nodes within the business process model.

For instance, the extended behavioral profile of the BP in Figure 1 includes several pairs, such as: $bhp'(U, V) = (\#, 0)$, $bhp'(W, X) = (\rightsquigarrow, 1)$, $bhp'(X, W) = (\leftsquigarrow, 1)$, $bhp'(Y, Z) = (\|, 0)$, $bhp'(U, X) = (\rightsquigarrow, 3)$, $bhp'(Y, V) = (\leftsquigarrow, 5)$, etc. To derive useful behavioral properties between tasks of a BP, we remove cyclic edges from such BPs because their existence makes all connected tasks concurrent.

Tasks in a business process can be either manual or service tasks. Manual tasks are performed by employees, whereas, service tasks are executed through web services. Transforming BPMs into executable processes is done through a *configuration* step. In this step, business engineers assign operations of web services to *service* tasks in the considered BPM. This assignment involves service discovery and selection. The formal definition of BP configuration is given in Definition 3.3.

Definition 3.3 (BP Configuration): A service registry usually contains a collection of web services. Each web service, WS_i, has one or more operations, OP_j. Formally, $WS_i = \{OP_1, \cdots, OP_n\}$; $n \in \mathbb{N}$. A service registry SR is the collection of available operations, $SR = \bigcup_{i=1\ldots m} WS_i$. The *configuration* of a business process model – containing service tasks T – is a function $conf_{BPM}$[1] $: T \rightarrow SR$ that assigns an operation for each *service* task in that business process model.

The BP shown in Figure 1 can be used as a journey organizer, where its tasks are configured as follows: $conf(U)$ = `BookFlightTicket`, $conf(V)$ = `BookTrainTicket`, $conf(W)$ = `HotelReservation`, $conf(X)$ = `CarRental`, $conf(Y)$ = `FindRestaurants`, $conf(Z)$ = `FindSights`. Where values to the right of the $conf$ function are operations of web services.

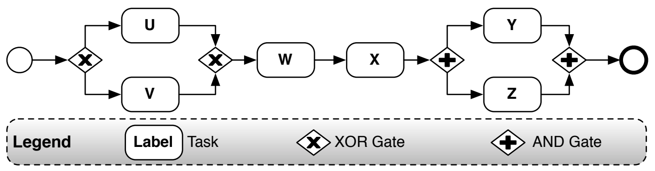

Figure 1. A simple business process modeled in BPMN 1.0

Saving a configured BPM ships the extended behavioral profile and its configuration to the service registry. This shipped information is then used by the service registry to discover linkage patterns among operations of web services used by that BP. Each linkage pattern has a *type* and *weight*, as we show in the sequel.

[1]When the BPM is clear from the context, we drop the subscript

B. Types of Linkage Patterns

Traditional approaches to discover relations among operations of web services use the heuristic that operations used in service compositions or by similar service consumers are related. However, they do not provide execution semantics, e.g., parallel, sequence, etc. Such relations are not sufficient given the increasing number and complexity of web services and business processes. In our approach – based on extended behavioral profiles –, we are able to identify five types of relations among operations of web services based on their usage in BPMs. Consider two tasks, A and B, whose configurations are: $conf(A) = OP_1$ and $conf(B) = OP_2$. Based on their behavioral properties, the following five types of linkage patterns can be identified:

1) **Predecessor**: An operation OP_1 is a predecessor of another operation OP_2 if it appears typically *before* OP_2 in the configurations of BPMs where both operations have been used, i.e., $bhp(A, B) = \rightsquigarrow$.
2) **Successor**: An operation OP_1 is a successor of another operation OP_2 if it appears typically *after* OP_2 in the configurations of BPMs where both operations have been used, i.e., $bhp(A, B) = \leftsquigarrow$.
3) **Similar**: An operation OP_1 is similar to another operation OP_2 if it appears typically within exclusive relations with OP_2 in the configurations of BPMs where both operations have been used (i.e., $bhp(A, B) = \#$) and there is a high semantic similarity between the terms used to label both tasks and their executing operations, e.g., *"rent a bike"* and *"buy a bike"*.
4) **Complementary**: An operation OP_1 is complementary to another operation OP_2 if it appears typically within exclusive relations with OP_2 in the configurations of BPMs where both operations have been used (i.e., $bhp(A, B) = \#$) but their is *no high* semantic similarity between the terms used to label both tasks and their executing operations, e.g., *accept* and *reject*.
5) **Related**: An operation OP_1 is a related to another operation OP_2 if it appears typically *concurrently* to OP_2 in the configurations of BPMs where both operations have been used, i.e., $bhp(A, B) = \|$. For instance, *"validate address"* and *"validate email address"*.

C. Weights of Linkage Patterns

Existing approaches of discovering relations among web services determine whether there is a relation between two web services or not. Such decisions depend on the co-occurrence of both services in service compositions, for instance. In our approach, we can determine fine-grained relations and give a weight (between 0 and 1) to each relation to reflect its strength. These weights are used to rank web services in each type of relations, e.g., predecessors, similar, etc. during service recommendation.

The first type of information that we use to calculate the weight of a relation between two web services is the distance between their consuming tasks in the corresponding BP. This information is provided in the extended behavioral profile of the BP. The distance between any two tasks in a BP is greater than 0 if their behavioral property is either strict order or inverse order. Therefore, the distance is used to assign a weight to predecessor and successor linkage patterns only. The weight, ω, of a linkage pattern, r, between two operations where the distance between their consuming tasks is, d, and the maximum distance between any pair of tasks in their BP is len, is given by Equation 1.

$$\omega(r) = \frac{len - d}{len} \tag{1}$$

Exclusive tasks can be similar (doing the same functionality) or complementary to each other (doing different functionalities). For instance, "Get weather by city" and "Get weather by post code" are *similar* tasks. Whereas, "Send acceptance" and "Send rejection" are *complementary* tasks. To determine the linkage pattern between two web services whose consuming tasks are exclusive to each other, we investigate the semantics of terms appearing in their names and their consuming tasks. We use WordNet [20] to find `synsets` for these terms and calculate the average distance, (syn_dist), among their *nearest common ancestors (NCA)* in WordNet [21]. The special value (-1) means that there is no similarity between both terms, e.g., acceptance and rejection. If the average distance, (syn_dist), is between 0 and a predefined threshold, then the linkage pattern between both services is *similar* and its weight is calculated using the same equation above, where len is replaced by our threshold value, and d is replaced by syn_dist. For instance, $syn\_dist(``bookFlightTicket", ``BookTrainTicket") = 16$. Based on our experiments, we set the value of the maximum WordNet distance threshold to 20. Given this value, the aforementioned web services are *similar* and the weight of their linkage pattern is 0.2. It is worth mentioning that the linkage patterns *complementary* and *related* are assigned the weight 1.

Whenever a new BPM is created by a service consumer, we discover all possible linkage patterns from that BPM and store them in the database of the service registry. Frequencies and weights of linkage patterns are used to derive scores for these patterns to rank recommended web services within each type of recommendations. The score of each linkage pattern is the aggregation of weights of all instances of this pattern that are typically discovered from multiple BPMs. Currently, we ignore inconsistent linkage patterns, i.e., different types of linkage patterns for the same web services from different BPMs. As a result, a web service could appear in more than one type of recommendation for the same web service.

IV. Example: Purchase Order Processing

In this section, we apply our approach to a real-world purchase order processing scenario from the SAP Reference Model. The BP behind this scenario is shown in Figure 2. When this process is configured, we assume that a single operation of a web service is assigned to each task in this model. For instance, operation *A* is assigned to task "process purchase requisition order". Following the traditional approaches of discovering relations among web services, we get the result that there is a relation between *A* and *B*. No further information about the type and strength of this relation is provided. In our approach, we get the extended behavioral profile that encapsulates business process knowledge for these operations as shown in the previous section. This extended behavioral profile is shown in Table I.

From Table I, we notice that operations *A* and *E* are exclusive and also are the operations *A* and *B*. We refine this relation further as either *similar* or *complementary*. To achieve this refinement, we analyze the semantics of the terms in the labels of their corresponding tasks. Based on our experiments, we set our threshold maximum WordNet distance to 20 to control the similarity search in WordNet.We repeat this step for each pair of operations that are exclusive to each other. With result obtained, we establish the linkage patterns among the operations as shown in Table II.

Based on the semantic analysis, the *exclusive* relation between operations *A*, *E* – obtained from the profile – is refined to a complementary linkage pattern. On the other hand, the exclusive relation between *A*, *B* is identified as similar because of the high similarity between terms appearing in the labels of their counterpart tasks.

Using these linkage patterns, users who search for a particular service, e.g, *A*, get useful lists of recommendations. These recommendations represent inter-links among web services that help service consumers explore web service comfortably.

Table I
Extended Behavioral profile for business process in Figure 2

	A	B	C	D	E	F	G	H	I	J
A	$(\parallel, 0)$	$(\#, 0)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 3)$	$(\#, 0)$	$(\#, 0)$	$(\rightsquigarrow, 5)$	$(\rightsquigarrow, 6)$	$(\rightsquigarrow, 8)$	$(\rightsquigarrow, 8)$
B	$(\#, 0)$	$(\parallel, 0)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 3)$	$(\#, 0)$	$(\#, 0)$	$(\rightsquigarrow, 5)$	$(\rightsquigarrow, 6)$	$(\rightsquigarrow, 8)$	$(\rightsquigarrow, 8)$
C	$(\leftsquigarrow, 2)$	$(\leftsquigarrow, 2)$	$(\parallel, 0)$	$(\rightsquigarrow, 1)$	$(\#, 0)$	$(\#, 0)$	$(\rightsquigarrow, 3)$	$(\rightsquigarrow, 4)$	$(\rightsquigarrow, 6)$	$(\rightsquigarrow, 6)$
D	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 1)$	$(\parallel, 0)$	$(\#, 0)$	$(\#, 0)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 3)$	$(\rightsquigarrow, 5)$	$(\rightsquigarrow, 5)$
E	$(\#, 0)$	$(\#, 0)$	$(\#, 0)$	$(\#, 0)$	$(\parallel, 0)$	$(\rightsquigarrow, 1)$	$(\rightsquigarrow, 3)$	$(\rightsquigarrow, 4)$	$(\rightsquigarrow, 6)$	$(\rightsquigarrow, 6)$
F	$(\#, 0)$	$(\#, 0)$	$(\#, 0)$	$(\#, 0)$	$(\leftsquigarrow, 1)$	$(\parallel, 0)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 3)$	$(\rightsquigarrow, 5)$	$(\rightsquigarrow, 5)$
G	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 2)$	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 2)$	$(\parallel, 0)$	$(\rightsquigarrow, 1)$	$(\rightsquigarrow, 3)$	$(\rightsquigarrow, 3)$
H	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 2)$	$(\leftsquigarrow, 4)$	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 1)$	$(\parallel, 0)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 2)$
I	$(\leftsquigarrow, 8)$	$(\leftsquigarrow, 8)$	$(\leftsquigarrow, 6)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 6)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 2)$	$(\parallel, 0)$	$(\parallel, 0)$
J	$(\leftsquigarrow, 8)$	$(\leftsquigarrow, 8)$	$(\leftsquigarrow, 6)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 6)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 3)$	$(\leftsquigarrow, 2)$	$(\parallel, 0)$	$(\parallel, 0)$

V. Implementation and Usage Scenarios

We have implemented a prototype that realizes our approach to discover additional linkage patterns among web services using business process knowledge. In this section, we give details about the implementation of this prototype that integrates *Oryx*, which is a business process modeling platform and repository [22], and *Depot*, which is a public web service registry [23]. Additionally, we show how our approach can be used to enable service exploration through service recommendations.

A. Implementation Details

The integration between both projects is depicted in Figure 3. During the configuration phase of a business process model, a web service is assigned to each service task in that model. This assignment is done using Depot to find relevant web services. The selection made for each task affects the ranking of web services in the result list for next tasks. Saving the finalized, configured BPM ships the extended behavioral profile and the configurations of that BPM to Depot. Based on this information, Depot derives linkage patterns between web services used in that BPM. These linkage patterns are then used to provide recommendations to service consumers during service discovery, either in Oryx or Depot.

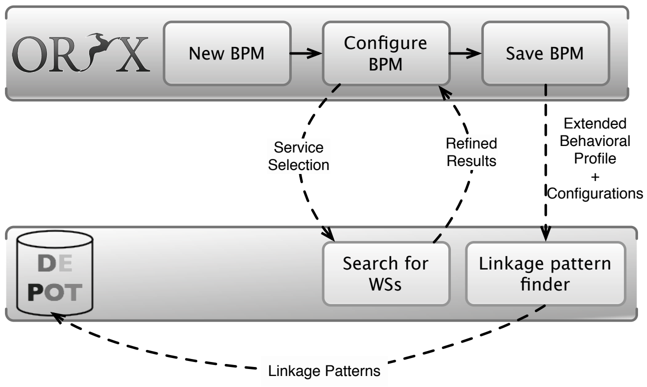

Figure 3. Context: An integrated view of Business Process Modeling (*Oryx*) and Web Services (*Depot*)

It is worth mentioning that our approach does not suffer a cold start because we discover preliminary linkage patterns among web services from their *annotated* `WSDL` descriptions. These annotations for web services are generated automatically from the websites of their providers [24] and through automatic invocation analysis [25]. These additional annotations for web services can be used as a semantics source. This additional source of semantics is extremely helpful to determine fine-grained linkage patterns among web services that are typically used with exclusiveness in BPs. In particular, for operations with technical or semantically-poor names.

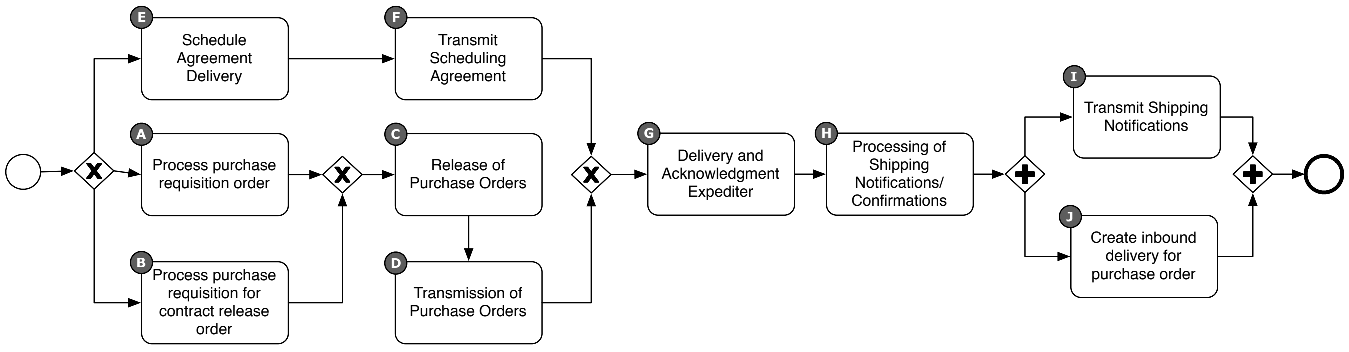
E
Schedule Agreement Delivery
F
Transmit Scheduling Agreement
A
Process purchase requisition order
C
Release of Purchase Orders
B
Process purchase requisition for contract release order
D
Transmission of Purchase Orders
G
Delivery and Acknowledgment Expediter
H
Processing of Shipping Notifications/ Confirmations
I
Transmit Shipping Notifications
J
Create inbound delivery for purchase order

Figure 2. A Business Process for "Purchase Order Processing" from *SAP* Reference Model represented in BPMN 1.0

Table II
Linkage Patterns for business process in Figure 2. *P*: Predecessor, *S*: Successor, *M*: Similar, *C*: Complementary, *R*: Related

	A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
A	–	$(M, 0.300)$	$(S, 0.875)$	$(S, 0.750)$	$(C, 1.000)$	$(C, 1.000)$	$(S, 0.500)$	$(S, 0.375)$	$(S, 0.125)$	$(S, 0.125)$
B	$(M, 0.300)$	–	$(S, 0.875)$	$(S, 0.750)$	$(C, 1.000)$	$(C, 1.000)$	$(S, 0.500)$	$(S, 0.375)$	$(S, 0.125)$	$(S, 0.125)$
C	$(P, 0.875)$	$(P, 0.875)$	–	$(S, 1.000)$	$(C, 1.000)$	$(C, 1.000)$	$(S, 0.750)$	$(S, 0.625)$	$(S, 0.375)$	$(S, 0.375)$
D	$(P, 0.750)$	$(P, 0.750)$	$(P, 1.000)$	–	$(C, 1.000)$	$(C, 1.000)$	$(S, 0.875)$	$(S, 0.750)$	$(S, 0.500)$	$(S, 0.500)$
E	$(C, 1.000)$	$(C, 1.000)$	$(C, 1.000)$	$(C, 1.000)$	–	$(S, 1.000)$	$(S, 0.750)$	$(S, 0.625)$	$(S, 0.375)$	$(S, 0.375)$
F	$(C, 1.000)$	$(C, 1.000)$	$(C, 1.000)$	$(C, 1.000)$	$(P, 1.000)$	–	$(S, 0.875)$	$(S, 0.750)$	$(S, 0.500)$	$(S, 0.500)$
G	$(P, 0.500)$	$(P, 0.500)$	$(P, 0.750)$	$(P, 0.875)$	$(P, 0.750)$	$(P, 0.875)$	–	$(S, 1.000)$	$(S, 0.750)$	$(S, 0.750)$
H	$(P, 0.500)$	$(P, 0.500)$	$(P, 0.750)$	$(P, 0.875)$	$(P, 0.625)$	$(P, 0.750)$	$(P, 1.000)$	–	$(S, 0.875)$	$(S, 0.875)$
I	$(P, 0.125)$	$(P, 0.125)$	$(P, 0.375)$	$(P, 0.500)$	$(P, 0.375)$	$(P, 0.500)$	$(P, 0.750)$	$(P, 0.875)$	–	$(R, 1.000)$
J	$(P, 0.125)$	$(P, 0.125)$	$(P, 0.375)$	$(P, 0.500)$	$(P, 0.375)$	$(P, 0.500)$	$(P, 0.750)$	$(P, 0.875)$	$(R, 1.000)$	–

B. Exploration and Recommendation of Web Services

One of our goals in this work is to enhance service discovery. In particular, with poor service descriptions. Service exploration is a common approach to enhance service discovery [25]. For instance, exploring web services based on provider, category, etc. In our approach, we use service recommendation to enable service exploration, where recommendations are provided based on the discovered linkage patterns among web services.

The typical service discovery scenario starts when a service consumer specifies a particular (business) need in keywords, for instance. Candidate web services that match these keywords are ranked according to their degrees of match. After that, the service consumer examines these candidate web services further. Detailed information about each chosen web service is provided, such as provider, operations, etc. In our approach, this information is augmented with recommended web services. These recommendations are grouped in five groups as shown in Figure 4.

Predecessor and successor web services are ranked according to their weights based on distances between their consuming task in BPs that use them. Similar web services are ranked according to the similarity between terms used in their names and labels of tasks that consumer them. Complementary and related web services are ranked according to their frequencies.

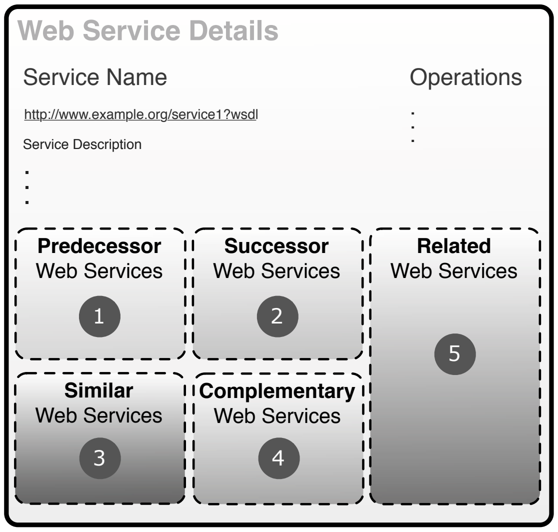

Figure 4. Five types of recommendation of web services in Depot

Based on this approach, service consumers can explore huge collections of web services that can be used together to build useful and realistic business needs (BPs). This approach can be compared to browsing webpages that are linked together using hyperlinks.

Configuring BPs becomes easier using our approach. Service selection made in a task affects ranking web services for the next task. This ranking help process engineers build and configure their BPs quickly and easily.

VI. Summary and Future Work

Service discovery has been among the top challenges in services computing. Several factors exacerbate this challenge, such as the lack of rich service descriptions, the increasing number and complexity of web services, etc. To enhance service discovery, several approaches have been proposed to investigate relation among web services. Typically, four approaches are used to find such relations, namely, input-output matching, semantic web services, service compositions, and consumer-consumer similarity approaches. Despite their efficiency (under certain assumptions), these approaches cannot specify fine-grained types or strengths of the found relations.

In our approach, we utilize the business process configuration to discover fine-grained linkage patterns among web services used in such processes. The required business process knowledge is captured using the notion of extended behavioral profiles. Based on these profiles, we can determine five types of linkage patterns among web services, namely, predecessor, successor, similar, complementary, and related. Additionally, each linkage pattern is assigned a weight that reflects its strength. These weights are used to rank service recommendations that enables service exploration.

In this work, we ignore contradicting and inconsistent behavioral properties among web services. Investigating such inconsistencies is part of our future work.

References

[1] D. Kuropka, P. Tröger, S. Staab, and M. Weske, *Semantic Service Provisioning*. Germany: Springer, 2008.

[2] M. Sabou, C. Wroe, C. Goble, and G. Mishne, "Learning Domain Ontologies for Web Service Descriptions: An Experiment in Bioinformatics," in *Proceedings of the 14th international conference on World Wide Web*, 2005. [Online]. Available: http://portal.acm.org/ft_gateway.cfm?id=1060776

[3] Sun Microsystems, "Effective SOA Deployment using an SOA Registry Repository, A Practical Guide," 2005, white paper.

[4] X. Dong, A. Halevy, J. Madhavan, E. Nemes, and J. Zhang, "Similarity search for web services," in *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 372–383. [Online]. Available: http://portal.acm.org/citation.cfm?id=1316689.1316723

[5] L. Lin and I. B. Arpinar, "Discovery of semantic relations between web services," in *Proceedings of the IEEE International Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 357–364.

[6] S. Basu, F. Casati, and F. Daniel, "Toward web service dependency discovery for soa management," in *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 422–429. [Online]. Available: http://portal.acm.org/citation.cfm?id=1443230.1444300

[7] W. Rong, K. Liu, and L. Liang, "Personalized web service ranking via user group combining association rule," in *Proceedings of the 2009 IEEE International Conference on Web Services*, ser. ICWS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 445–452. [Online]. Available: http://dx.doi.org/10.1109/ICWS.2009.113

[8] D. Fensel, U. Keller, H. Lausen, A. Polleres, and I. Toma, "WWW or what is wrong with web service discovery?" in *Proceedings of the W3C Workshop on Frameworks for Semantics in Web Services*, 2005. [Online]. Available: http://members.deri.at/~uwek/publications/WWW_or_What_is_Wrong_with_Web_service_Discovery.pdf

[9] S. Sharma and S. Batra, "Applying association rules for web services categorization," *International Journal of Computer and Electrical Engineering*, vol. 2, no. 3, pp. 465–468, 2010.

[10] A. Segev, "Circular context-based semantic matching to identify web service composition," in *Proceedings of the 2008 international workshop on Context enabled source and service selection, integration and adaptation: organized with the 17th International World Wide Web Conference (WWW 2008)*, ser. CSSSIA '08. New York, NY, USA: ACM, 2008, pp. 7:1–7:5.

[11] A. M. Omer and A. Schill, "Web service composition using input/output dependency matrix," in *Proceedings of the 3rd workshop on Agent-oriented software engineering challenges for ubiquitous and pervasive computing*, ser. AUPC 09. New York, NY, USA: ACM, 2009, pp. 21–26. [Online]. Available: http://doi.acm.org/10.1145/1568181.1568189

[12] F. Lecue and A. Leger, "Semantic web service composition based on a closed world assumption," in *Proceedings of the European Conference on Web Services*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 233–242. [Online]. Available: http://portal.acm.org/citation.cfm?id=1191829.1192660

[13] A. Bose, "Effective Web Service Discovery using a combination of a Semantic Model and a Data mining technique," Master's thesis, Queensland University of Technology, Queensland, Australia, 2008.

[14] M. Winkler, T. Springer, E. D. Trigos, and A. Schill, "Analysing dependencies in service compositions," in *Proceedings of the 2009 international conference on Service-oriented computing*, ser. ICSOC/ServiceWave'09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 123–133. [Online]. Available: http://portal.acm.org/citation.cfm?id=1926618.1926633

[15] M. Weske, *Business Process Management*. Springer, 2007.

[16] S. Buchwald, J. Tiedeken, and M. Reichert, "Anforderungen an ein metamodell für soa-repositories," in *ZEUS*, ser. CEUR Workshop Proceedings, C. Gierds and J. Sürmeli, Eds., vol. 563. CEUR-WS.org, 2010, pp. 17–24.

[17] S. Buchwald, T. Bauer, and M. Reichert, *Bridging the Gap Between Business Process Models and Service Composition Specifications*, 2011, ch. Int'l Handbook on Service Life Cycle Tools and Technologies: Methods, Trends and Advances.

[18] S. Smirnov, M. Weidlich, J. Mendling, and M. Weske, "Action patterns in business process models," in *ICSOC/ServiceWave*, ser. Lecture Notes in Computer Science, L. Baresi, C.-H. Chi, and J. Suzuki, Eds., vol. 5900, 2009, pp. 115–129.

[19] M. Weidlich, A. Polyvyanyy, J. Mendling, and M. Weske, "Efficient computation of causal behavioural profiles using structural decomposition," in *Petri Nets*, ser. Lecture Notes in Computer Science, J. Lilius and W. Penczek, Eds., vol. 6128. Springer, 2010, pp. 63–83.

[20] G. A. Miller, "Wordnet: a lexical database for english," *Commun. ACM*, vol. 38, pp. 39–41, November 1995. [Online]. Available: http://doi.acm.org/10.1145/219717.219748

[21] "The yago-naga project: Harvesting, searching, and ranking knowledge from the web," http://www.mpi-inf.mpg.de/yago-naga/.

[22] G. Decker, H. Overdick, and M. Weske, "Oryx - An Open Modeling Platform for the BPM Community," in *BPM*, ser. LNCS, vol. 5240. Springer, 2008, pp. 382–385.

[23] M. AbuJarour and F. Naumann, "Information integration in Service-oriented Compuitng," in *Ph.D. Symposium at the European Conference on Web Services*, Ayia Napa, Cyprus, 2010.

[24] M. AbuJarour, F. Naumann, and M. Craculeac, "Collecting, Annotating, and Classifying Public Web Services ," in *OTM 2010 Conferences*. Crete, Greece: Springer, 2010.

[25] M. AbuJarour and F. Naumann, "Dynamic Tags For Dynamic Data Web Services," in *Workshop on Enhanced Web Service Technologies*. Ayia Napa, Cyprus: ACM, 2010.