Chapter 1
Web Services and Business Processes: A Round Trip

Mohammed AbuJarour and Ahmed Awad

Abstract Service-oriented Architecture (SOA) is considered as an implementation for business processes (BP). However, the relation between SOA and BPs is usually inspected in one direction only. In this chapter, we investigate the *bi-directional* relation between web services and business processes, and explore potential benefits therefrom. In particular, we introduce a novel approach to generate additional information about web services based on the configurations of business processes that consume these web services. This information is then used to enhance and smooth the modeling and configuration of future business processes. Through our approach, we can generate three types of information from consumers' business processes, namely annotations, context, and relations among web services. To evaluate our approach, we use the SAP reference model and we show the results in this chapter.

1.1 The Relation Between Business Processes and Web Services

Service-oriented Architecture (SOA) has been considered as an implementation platform for business processes (BP), nevertheless, each of them is typically investigated separately. Investigating both worlds (i.e., SOA and BP) together is expected to result in several benefits for both SOA and BP communities. For instance, getting a running instance of a business process model requires mapping its *service* tasks to (web) services. This mapping step requires sufficient information about the used web

M. AbuJarour (✉)
SAP AG, Potsdam, Germany
e-mail: mohammed.abujarour@sap.com

A. Awad
Faculty of Computers and Information,
Cairo University, Giza, Egypt
e-mail: a.gaafar@fci-cu.edu.eg

services that is understandable by process engineers or business people who create such mappings. Finding candidate web services to execute each service task is one of the key challenges in SOA, e.g., due to poor service descriptions [10]. We consider the configurations of BPs as a rich source of information about their consuming web services that enhance service discovery and future BP configurations.

Due to the increasing number of BPs, service consumers in several application domains maintain repositories of business process models (BPM) for their daily activities, e.g., “ship ordered item”. Each business process is composed of a set of manual (i.e., performed by employees) or service tasks (i.e., performed through web services). Building a new BPM incorporates three steps: (1) creating and labeling its tasks (2) determining the interaction between them, i.e., data and control flow (3) configuring the created model, i.e., *selecting* web services to perform service tasks. The built BPM is typically stored in the consumer’s repository for future requests.

Using the aforementioned scenario in practice involves several challenges, such as, service discovery, service selection, BP configuration. *Service discovery* is one of the main challenges in Service-oriented Computing (SOC), where a list of candidate web services are returned to service consumer as a result for their queries. Choosing a particular web service to invoke from this list is known as *service selection* [18], which has become a complex task due to several factors, e.g., lack of rich service descriptions. Therefore, additional information about web services—e.g., annotations, relations among web services, etc.—is expected to help meet this challenge [7]. *Business process configuration* represents the service selection step, where each service task is assigned a web service to execute it. Performing this task dynamically requires sufficient information about web services so that process engineers configure their BPMs accordingly. Technical information only is not sufficient, because it does not fit their backgrounds and knowledge [19].

Using web services within distributed business processes brings the challenges of service discovery and selection to business processes. In this work, we introduce a novel approach to bridge both worlds (SOA and BPM), where we use business process configurations to derive additional information about their implementing web services that reflect service consumers’ perspective (business view) to enrich their technical descriptions released by their providers. We are able to generate three types of information about web services, namely annotations, context, and relations among web services. Annotations for web services are generated from tasks’ labels and documentations, whereas context is derived from models’ titles and descriptions. We discover *additional* realistic and rich relations among web services in the form of linkage patterns using behavioral profiles [24] of their consuming business processes. Additionally, we derive a global representation of all relations among web services that are derived from multiple business processes. We use this global representation to predict relations among web services that are not used together in a single business process using the gained knowledge in this global representation.

The contributions of the work introduced in this chapter are:

1. Supporting smooth configuration of business processes by enabling context-aware service selection.

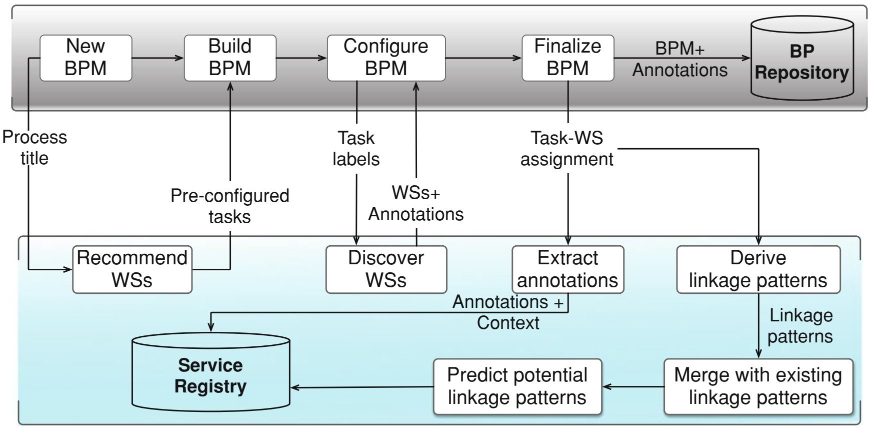


Fig. 1.1 An overview of our approach of integrating business processes and web services

2. Finding realistic, rich relations among web services as *linkage patterns.*
3. Disambiguating exclusive relations between web services using lexical ontologies, e.g., WordNet.
4. Merging behavioral profiles of BPs into a single global behavioral profile.
5. Revealing relations among web services that have not yet been used together.

The rest of this chapter is organized as follows: We give an overview of our approach in Sect. 1.2. Then, we introduce the fundamental concepts that are used throughout this chapter in Sect. 1.3. After that, we present our approach to generate annotations for web services from business process configurations in Sect. 1.4. In Sect. 1.5, we describe our approach to derive rich relations among web services in the form of linkage patterns. Deriving global behavioral profiles among web services is described in Sect. 1.6. Implementation details and experiments are introduced in Sect. 1.7. Related work is summarized in Sect. 1.8. We summarize this chapter in Sect. 1.9.

1.2 Overview of Our Approach

In this section we give an overview of our approach that represents a round trip between web services and business processes as shown in Fig. 1.1. The scenario starts when a business process designer creates a new BPM. At that point, the designer gives a descriptive name and summary for the new process, e.g., *establish a company in Germany*. Behind the scene, a request is sent to a service registry to find relevant web services that have been used in similar models, e.g., *establish a company in UK*. The returned recommended services are provided as *pre-configured* tasks that are made available to the designer to accelerate the process design.

The process designer might not use all web services recommended by the service registry in their new model. Therefore, they introduce new tasks to express the particular business needs in the process at hand. Each new task is given an identifying label that we pass to the service registry to find potential web services that can be candidate matches. For each new task in the model, a list of candidate web services is returned to the process designer during the configuration phase. Each web service in our collection is associated with a set of annotations that explain its functionality. These annotations are extracted and generated automatically from the websites of their providers [2], invocation analysis [3], and previous BP configurations. The new BPM is finalized when each service task is configured by assigning a web service to execute it. With the finalized BPM, two sources of information can be identified and generated, namely task-to-web service assignment and annotated BPM.

On the one hand, the *task-to-web service assignment* is passed from the modeling framework to the service registry, where annotations, context, and relations are generated therefrom. On the other hand, the tasks in the created BPM are automatically annotated with the annotations of the web services they are bound with, resulting in an *annotated BPM*. These annotations are crucial for BPM lookup, because not only task labels are used to index and find tasks, but enriched annotations are also used to achieve this goal [4] leading to better discovery of models from process repositories.

We use this assignment list also to generate behavioral profiles, based on which we derive rich relations among web services. Even more, we discover *hidden* relations among services that have not been used together in any business process configuration yet. The notion of behavioral profiles is developed by Weidlich et al. [24] to give a behavioral abstraction over business processes. We use this notion and extend it according to requirements for discovering relations among web services.

1.3 Fundamentals: Business Process Knowledge

Researchers have proposed several approaches that investigate the behavioral relations among tasks within a process model. For instance, the α-algorithm [22], causal footprints [23] and behavioral profiles [24].[1] Although these approaches are developed for different purposes, they have a fundamental common feature; generating a set of behavioral relations among tasks in a process model. We use these behavioral relations in our approach to derive rich relations among web services. Causal footprints [23] and behavioral profiles [24] take as input a process model, represented as a WF-net [21]. Whereas the α-algorithm requires as input a set of process execution traces (i.e., log). We use the behavioral profiles approach as a starting point, because we have process models as input and because the behavioral profile approach is much more efficient than causal footprints. Nevertheless, our approach is independent of this selection and works with process behavior abstraction approaches that support the fundamental behavioral relations.

[1] There are other related approaches that share similar underlying concepts.

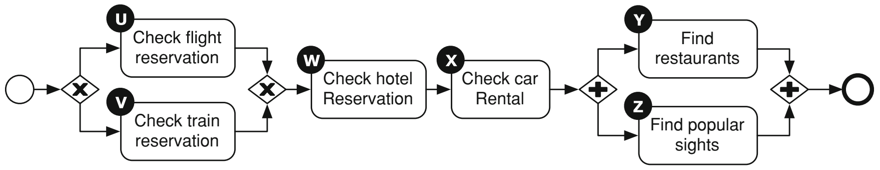


Fig. 1.2 A journey organizer business process modeled in BPMN 1.0

Behavioral profiles represent an abstract description of a business process and 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 Definition 1.1.

Definition 1.1 *(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 $bhp\colon N \times N \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#\}$ 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 always 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 Fig. 1.2 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 of discovering fine-grained linkage patterns among web services, 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 distance between a pair of tasks is calculated by counting the number of edges between them in the considered BPM. A preliminary definition of the extended behavioral profile is given in Definition 1.2. A comprehensive definition of the "Extended Behavioral Profile" is introduced in Definition 1.3.

Definition 1.2 *(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 $bhp'\colon N \times N \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#\} \times \mathbb{N}$ 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 Fig. 1.2 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, because their existence makes all tasks inside each BP concurrent.

Transforming BPMs into executable processes is achieved through a *configuration* step, where process engineers assign operations of web services to *service* tasks in the considered BPM. Configuring a BP can be expressed as a function that takes a task of a BP and assigns an operation to that task if it is a *service* task. The business process shown in Fig. 1.2 can be configured as follows: $conf(U) = \texttt{BookFlightTicket}$, $conf(V) = \texttt{BookTrainTicket}$, $conf(W) = \texttt{HotelReservation}$, $conf(X) = \texttt{CarRental}$, $conf(Y) = \texttt{FindRestaurants}$, and $conf(Z) = \texttt{FindSights}$. Where values to the right of the $conf$ function are operations of web services.

1.4 Annotating Web Services Using Business Process Knowledge

In this section, we describe our approach to generate annotations and derive contexts for web services based on the configurations of their consuming business processes. From each finalized (i.e., configured) BPM, we generate a *task-to-web service assignment* list, based on which we generate annotations for web services from the tasks of their consuming business processes. Task labels and documentation are extracted and their assigned web services are annotated with this extracted information. These labels and documentations are created by service consumers that represent the application level, i.e., business people. Additionally, the title of the created BPM is used to derive the context in which these web services are typically used. This information is then used to enable context-aware service selection for similar cases in the future.

Sharing information about business processes and web services used to execute them is not usually desired by service consumers, because this information might be considered one aspect of their competitive advantage. Nevertheless, our approach can be valuable in several scenarios and application domains, in particular, where high potentials for collaboration are expected and low potentials for competition among service consumers exist, such as government services, education and research, online modeling platforms, and quality-based service brokers.

Figure 1.3 shows a process model using BPMN for establishing a UK limited company. The first six activities of the process are services of the UK Companies

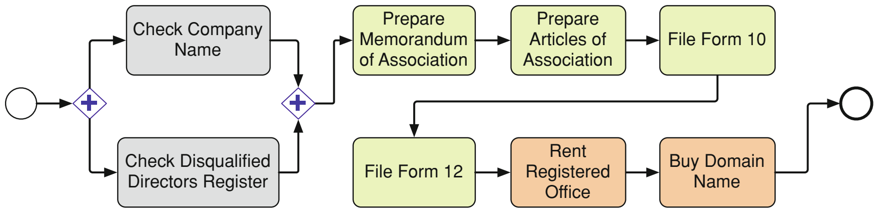

Check Company Name
Check Disqualified Directors Register
Prepare Memorandum of Association
Prepare Articles of Association
File Form 10
File Form 12
Rent Registered Office
Buy Domain Name

Fig. 1.3 Example process model of establishing a UK limited company

House. In the beginning, it has to be checked whether the desired company name is not already in use and the directors are not disqualified from leading a company. For the remaining steps, electronic forms are provided in the Portable Document Format (PDF). The last two activities in the model are web services offered by private service providers, e.g., "Buy domain name" can be executed using the `whois` web service (http://www.webservicex.net/whois.asmx).

Using our approach, a process engineer can configure their BPM to establish a company in UK using the existing annotations for web services and their operations used in such a model. These annotations are generated from the websites of their providers and through invocation analysis. Although they might be not rich enough, such annotation can be helpful in some cases. Generating additional annotations for such web services and operations from this BPM enrich their descriptions. In this particular use case, all used operations are associated with the context "establish a company in UK". Additionally, each operation is annotated with the label and documentation of each task that uses it. For instance, the `whois` web service is annotated with "buy domain name".

According to the German Federal Ministry of Economics and Technology, establishing a company in Germany incorporates 9 major steps.[2] For instance, check the company name, notarize the articles of association and foundation agreement, notify the Office of Business and Standards, register at the Trade Office (involves check manager's qualifications), etc. Some of these steps are similar to the ones involved in establishing a limited company in UK, such as "check the company name", "check qualified managers", "rent office", "buy domain name". For instance, `whois` web service (http://www.webservicex.net/whois.asmx), that is used to execute the "buy domain name" task in the case of UK company, can be suggested to execute its counterpart task in the German case.

Saving this model adds additional information to the service registry about the considered web services, such as the new labels and the current context. This additional information helps the service registry provide better results in future similar business processes, such as "Establishing a company in France".

1.5 Fine-Grained Linkage Patterns Among Web Services

Relations among web services are important to understand the functionalities of these web services and the interaction among them. We discover preliminary relations among web services from their `WSDL` files, and derive additional fine-grained ones in the form of linkage patterns using their consuming BPs (Fig. 1.1). Each of these linkage patterns has one of these types: *Predecessors*, *successors*, *similar*, *complementary*, and *related*. Moreover, we assign weights to such relations based on the usage of their web services 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

[2] http://www.existenzgruender.de/englisch/

the *predecessor* relation with a particular web service. In this section, we describe the types and weights of linkage patterns that we find based on business process knowledge.

1.5.1 Types of Linkage Patterns

Traditional approaches to discover relations among operations of web services usually give binary decisions whether two web services are related or not, without providing control flow dependency, 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, that are configured with OP_1 and OP_2, respectively. 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 *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 *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 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 within exclusive relations with OP_2 in the configurations of BPMs where both operations have been used (i.e., $bhp(A, B) = \#$) but there 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 related to another operation OP_2 if it appears *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"*.

1.5.2 Weights of Linkage Patterns

Existing approaches of discovering relations among web services give binary decisions on whether there is a relation between two web services or not. Such decisions are based on the co-occurrence of both services in service compositions, for instance. In our approach, we are able to discover fine-grained relations and assign a weight (between 0 and 1) to each relation to reflect its strength.

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 Eq. 1.1.

$$\omega(r) = \frac{len - d}{len} \tag{1.1}$$

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*. 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 [13] to find `synsets` for these terms and calculate the average distance, (syn_dist), among their *nearest common ancestors (NCA)* in WordNet [26]. The special value (-1) means that there is no similarity between both terms, i.e., they do not have a common ancestor in WordNet, 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. Linkage patterns are classified as *complementary* if the semantic similarity is low, i.e., syn_dist is higher than the predefined threshold. 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 recommendation. The score of each linkage pattern is the aggregation of weights of all instances of this pattern that are typically discovered from multiple BPMs. In practice, web services are used by different service consumers in multiple business processes with different arrangements. These differences result in incompatible relations among web services, i.e., ws_1 is a predecessor for ws_2 in one business process, but ws_1 is similar to ws_2 in another business process. To handle such situations, we merge all behavioral profiles of business process in a single global profile. Additionally, we use gained knowledge in this global profile to predict relations among web services that are not used together in the same business process, yet (Sect. 1.6.).

1.5.3 Example: Linkage Patterns of Purchase Order Processing

In this section, we apply our approach to a real-world purchase order processing scenario from the SAP Reference Model, whose BP is shown in Fig. 1.4. 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 previously. This extended behavioral profile is shown in Table 1.1.

From Table 1.1, 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 1.2. 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.

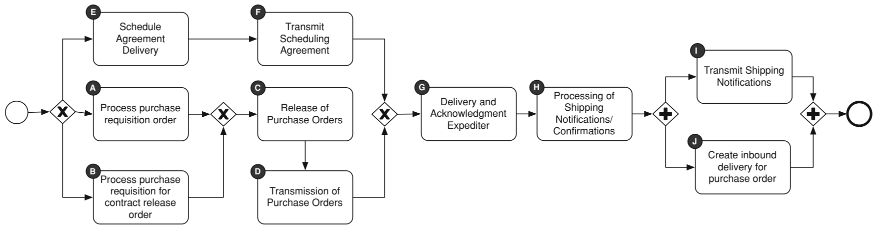

E
Schedule Agreement Delivery
F
Transmit Scheduling Agreement
A
Process purchase requisition order
C
Release of Purchase Orders
B
Process purchase requisition for contract release order
D
Transmission of Purchase Orders
G
Delivery and Acknowledgment Expediter
H
Processing of Shipping Notifications/ Confirmations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

Fig. 1.4 A business process for "purchase order processing" from *SAP* reference model represented in BPMN 1.0

Table 1.1 Extended behavioral profile for business process in Fig. 1.4

	A	B	C	D	E	F	G	H	I	J
A	(‖ , 0)	(#, 0)	(⇝, 2)	(⇝, 3)	(#, 0)	(#, 0)	(⇝, 5)	(⇝, 6)	(⇝, 8)	(⇝, 8)
B	(# , 0)	(‖, 0)	(⇝, 2)	(⇝, 3)	(#, 0)	(#, 0)	(⇝, 5)	(⇝, 6)	(⇝, 8)	(⇝, 8)
C	(⇜, 2)	(⇜, 2)	(‖, 0)	(⇝, 1)	(#, 0)	(#, 0)	(⇝, 3)	(⇝, 4)	(⇝, 6)	(⇝, 6)
D	(⇜, 3)	(⇜, 3)	(⇜, 1)	(‖, 0)	(#, 0)	(#, 0)	(⇝, 2)	(⇝, 3)	(⇝, 5)	(⇝, 5)
E	(# , 0)	(#, 0)	(#, 0)	(#, 0)	(‖, 0)	(⇝, 1)	(⇝, 3)	(⇝, 4)	(⇝, 6)	(⇝, 6)
F	(# , 0)	(#, 0)	(#, 0)	(#, 0)	(⇜, 1)	(‖, 0)	(⇝, 2)	(⇝, 3)	(⇝, 5)	(⇝, 5)
G	(⇜, 5)	(⇜, 5)	(⇜, 3)	(⇜, 2)	(⇜, 3)	(⇜, 2)	(‖, 0)	(⇝, 1)	(⇝, 3)	(⇝, 3)
H	(⇜, 5)	(⇜, 5)	(⇜, 3)	(⇜, 2)	(⇜, 4)	(⇜, 3)	(⇜, 1)	(‖, 0)	(⇝, 2)	(⇝, 2)
I	(⇜, 8)	(⇜, 8)	(⇜, 6)	(⇜, 5)	(⇜, 6)	(⇜, 5)	(⇜, 3)	(⇜, 2)	(‖, 0)	(‖, 0)
J	(⇜, 8)	(⇜, 8)	(⇜, 6)	(⇜, 5)	(⇜, 6)	(⇜, 5)	(⇜, 3)	(⇜, 2)	(‖, 0)	(‖, 0)

Table 1.2 Linkage patterns for business process in Fig. 1.4. *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	$(P, 0.125)$	$(P, 0.125)$	$(P, 0.375)$	$(P, 0.500)$	$(P, 0.375)$	$(P, 0.500)$	$(P, 0.750)$	$(P, 0.875)$	$(R, 1.000)$	–

1.6 Global Behavioral Profiles

Several approaches have been proposed to find relations among web services [5, 9, 12, 15], whose main goal is finding whether two web services are related or not, without further refinement of the suggested relations. Additionally, these approaches are not able to find *indirect* relations among web services, because they use knowledge about web services that are used together only. Relations between web services that are not used together remain missing.

Missing relations between web services do not necessarily indicate their independence. Several reasons can lead to such missing relations, such as lack of knowledge about web services and their functionalities, multiple web services with equivalent functionalities, and non-functional requirements (e.g., price, quality). We consider such missing relations as *hidden* ones and aim at revealing (part of) them. One approach of revealing such hidden relations is using knowledge concealed in the configurations of business processes that use these web services. Each configuration is considered an identifier for its tasks and its web services. Multiple tasks that have different labels are similar if they are bound with the same web service. Similarly, web services that have different names are considered similar if they are bound with tasks that share the same label.

In Sect. 1.5, we introduced an approach to discover rich relations among web services in the form of linkage patterns using business process knowledge that is contained in a single business process. As different consumers use web services in multiple business processes with different relations among them, multiple configurations over the same set of web services appear. These configurations are *local* to each individual process. In this section, we develop an approach to derive a *global* behavioral profile over the entire set of web services in a service registry and reveal *hidden* relations among web services within this global profile.

To validate our approach, we use a set of business processes from the SAP reference model [8]. These models represent possibilities to configure SAP R/3 ERP systems. Thus, it is analogous to business process configurations over a service landscape.

1.6.1 Extending Behavioral Profiles

Revealing hidden relations among web services requires a global behavioral profile, where all services in the considered registry are involved. A global profile is the result of *merging* all individual behavioral profiles of business processes. Merging two relations from two profiles results in *unknown* relations between web services that do not appear together in one business process. Moreover, this merging step might result in *contradicting* relations, e.g., merging $a \#_x b$ and $a \rightsquigarrow_y b$. Therefore, the four basic behavioral relations of the original behavioral profile in Definition 1.1 are not sufficient. We *extend* the four basic relations to capture such situations when merging individual profiles by introducing two additional relations: Unknown (?) and contradicts (※). These two relations do not appear on the level of individual raw profiles. They appear only when profiles are merged as we show in Sect. 1.6.2. We record the *distance* between tasks bound to web services in the process configuration similar to Definition 1.2. This distance is used in the derived linkage patterns among web services to rank services during service discovery. We obtain this distance by counting the edges on the shortest path between the nodes representing the tasks in the process graph of each BP. In this section, we present the formal notion of *extended* behavioral profiles. Additionally, we introduce a business process with its extended behavioral profile that is used as a running example in the rest of this section.

1.6.1.1 Formal Model

The original definition of behavioral profiles is concerned with behavioral relations among tasks within a business process. However, in our approach, we are interested in discovering relations among web services used in such business processes. Therefore, we extend the notion of behavioral profiles and generalize the one introduced in Definition 1.2 to capture this requirement.

Definition 1.3 *(Extended Behavioral Profile)*[3] Let $\mathcal{W}$ be the set of web services within a service registry. The *extended behavioral profile* of web services in $\mathcal{W}$ is a function $xbhp : \mathcal{W} \times \mathcal{W} \rightarrow \mathcal{P}(\{\rightsquigarrow, \leftsquigarrow, \|, \#, ?, ※\} \times \mathbb{N})$ that assigns a set of pairs of a behavioral property (strict order, inverse order, parallel, exclusive, unknown, or contradicts) and a distance between each pair of web services within the service registry.

Comparing Definitions 1.2 and 1.3 of the extended behavioral profile with Definition 1.1, we notice that the behavioral relations are leveraged from the level of tasks within individual process models (configurations) to the level of web services within the service registry. Moreover, the extended profile records the distance between web services consumed within an individual profile. This distance is greater than zero if the behavioral relation is either $\rightsquigarrow$ or $\leftsquigarrow$ and zero otherwise. Finally, Definition 1.3 allows multiple behavioral properties to exist between two web services in the global behavioral profile where two additional behavioral relations (※&?) are introduced. An individual behavioral profile (Definition 1.1) of a process can be turned into an extended profile by adding all web services in the service registry to the services consumed by that process where their behavioral relations are set to unknown. For simplicity, we ignore these unknown relations for input behavioral profiles.

Definition 1.4 *(Projections over an Extended Behavioral Profile)* Let $\mathcal{W}$ be the set of web services within a service registry and let x be an extended behavioral profile. The function $rel_x : \mathcal{W} \times \mathcal{W} \rightarrow \{\rightsquigarrow, \leftsquigarrow, \|, \#, ?, ※\}$ projects the behavioral relation between two web services a and b in the registry with respect to profile x. Similarly

[3] This is a comprehensive definition for Definition 1.2

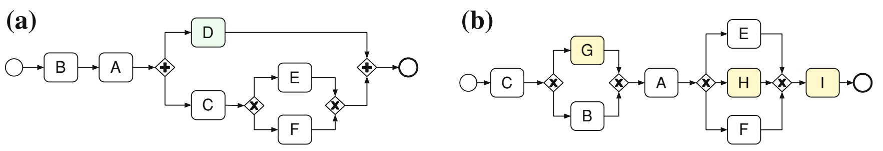


Fig. 1.5 Two anonymized business processes from SAP reference model

Table 1.3 The *extended* behavioral profile of BP_1 shown in Fig. 1.5a

	A	B	C	D	E	F
A	$(\|, 0)$	$(\leftsquigarrow, 1)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 4)$	$(\rightsquigarrow, 4)$
B	$(\rightsquigarrow, 1)$	$(\|, 0)$	$(\rightsquigarrow, 3)$	$(\rightsquigarrow, 3)$	$(\rightsquigarrow, 5)$	$(\rightsquigarrow, 5)$
C	$(\leftsquigarrow, 2)$	$(\leftsquigarrow, 3)$	$(\|, 0)$	$(\|, 0)$	$(\rightsquigarrow, 2)$	$(\rightsquigarrow, 2)$
D	$(\leftsquigarrow, 2)$	$(\leftsquigarrow, 3)$	$(\|, 0)$	$(\|, 0)$	$(\|, 0)$	$(\|, 0)$
E	$(\leftsquigarrow, 4)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 2)$	$(\|, 0)$	$(\|, 0)$	$(\#, 0)$
F	$(\leftsquigarrow, 4)$	$(\leftsquigarrow, 5)$	$(\leftsquigarrow, 2)$	$(\|, 0)$	$(\#, 0)$	$(\|, 0)$

$dist_x : \mathcal{W} \times \mathcal{W} \rightarrow \mathbb{N}$ projects the distance between the two services with profile x. For simplicity, we express $rel_x(a, b) = \{*\}$ as $a *_x b$ where $* \in \{\rightsquigarrow, \leftsquigarrow, \|, \#, ?, ※\}$.

1.6.1.2 Running Example of Global Profiles

In Fig. 1.5, we introduce two anonymized business processes from the SAP reference model that are used as a running example throughout this section. The common anonymized labels between both business processes indicate using the same service in their configuration. BP_1 has 6 tasks where only task D is not a common task with BP_2. On the other hand, BP_2 has 8 tasks among which 3 tasks are not common with BP_1, namely G, H, and I.

The extended behavioral profile of BP_1 is shown in Table 1.3, and that of BP_2 can be generated similarly, we omit it. It is worth mentioning that both BPs are configured such that each task is bound with a web service to execute it. According to Table 1.3, $xbhp_{BP_1}(E, F) = \{(\#, 0)\}$ and $xbhp_{BP_1}(A, D) = \{(\rightsquigarrow, 2)\}$. Relations that are not shown in this profile are implicitly *unknown*, e.g., $xbhp_{BP_1}(A, G) = \{(?, 0)\}$.

1.6.2 Deriving Global Behavioral Profiles

Knowledge about relations among web services is usually scattered in disparate profiles of business processes. Collecting this knowledge into a single profile is essential to reveal hidden relations among these web services. We call the result of this step a *global* behavioral profile. The global profile might include unknown or contradicting relations among some pairs of web services. We inspect the gained knowledge in the

Table 1.4 Merging the relations from two profiles x and y into an intermediate profile t

Profile	$a \rightsquigarrow_x b$	$a \leftsquigarrow_x b$	$a \parallel_x b$	$a \#_x b$	$a\, ?_x\, b$	$a \text{※}_x b$
$a \rightsquigarrow_y b$	$a \rightsquigarrow_t b$	$a \parallel_t b$ $a \text{※}_t b$	$a \parallel_t b$ $a \text{※}_t b$	$a \text{※}_t b$	$a \rightsquigarrow_t b$	$a \text{※}_t b$
$a \leftsquigarrow_y b$	$a \parallel_t b$ $a \text{※}_t b$	$a \leftsquigarrow_t b$	$a \parallel_t b$ $a \text{※}_t b$	$a \text{※}_t b$	$a \leftsquigarrow_t b$	$a \text{※}_t b$
$a \parallel_y b$	$a \parallel_t b$ $a \text{※}_t b$	$a \parallel_t b$ $a \text{※}_t b$	$a \parallel_t b$	$a \text{※}_t b$	$a \parallel_t b$	$a \text{※}_t b$
$a \#_y b$	$a \text{※}_t b$	$a \text{※}_t b$	$a \text{※}_t b$	$a \#_t b$	$a \#_t b$	$a \text{※}_t b$
$a ?_y b$	$a \rightsquigarrow_t b$	$a \leftsquigarrow_t b$	$a \parallel_t b$	$a \#_t b$	$a ?_t b$	$a \text{※}_t b$
$a \text{※}_y b$	$a \text{※}_t b$	$a \text{※}_t b$	$a \text{※}_t b$	$a \text{※}_t b$	$a \text{※}_t b$	$a \text{※}_t b$

global profile to *predict* possible resolutions for its unknown relations. Both steps, i.e., deriving a global profile and predicting unknown relations are of incremental nature. That is, at the point that a new process configuration is available, this new profile is merged with the global profile, to obtain a new global profile, and the prediction of unknown relations is performed again.

Given a set of behavioral profiles, we want to derive a global profile that contains pairwise relations between all web services. We achieve this by merging all individual profiles iteratively in a pairwise manner. The result of each merging iteration is an intermediate profile that is merged with another profile. This step is repeated until all individual profiles are incorporated. Merging individual profiles might result in unknown or contradicting relations among web services. Unknown relations appear between web services that are not used together in the same business process, whereas contradicting relations appear due to *conflicting* relations in source profiles. For instance, the relations ($a \#_x b$) (i.e., a and b are exclusive in profile x) and ($a \rightsquigarrow_y b$) (i.e., a precedes b in profile y) might imply that one of these relations is wrong, i.e., used incorrectly by a process engineer. Exclusiveness usually means that web services do similar or complementary jobs [1]. Currently, we propagate such conflicts to the resulting intermediate profile by adding two relations ($a \text{※}_z b$) and ($b \text{※}_z a$) that represent a contradiction to the resulting intermediate profile z.

We merge two relations between web services a and b that appear in both input profiles x and y into the global profile t according the rules that are summarized in Table 1.4. These rules can be grouped as follows:

1. Merging ($a *_x b$) with ($a *_y b$) gives ($a *_t b$), where $*$ is the same type of relation.
2. Merging ($a \leftsquigarrow_x b$) with ($a \rightsquigarrow_y b$) gives ($a \parallel_t b$) and ($a \text{※}_t b$).
3. Merging ($a *_x b$) with ($a \bullet_y b$) gives ($a \parallel_t b$) and ($a \text{※}_t b$), where $* \in \{\rightsquigarrow, \leftsquigarrow\}$ and $\bullet = \parallel$.
4. Merging ($a *_x b$) with ($a \#_y b$) gives ($a \#_t b$) if $* = \#$, and $a \text{※}_t b$ otherwise.
5. Merging ($a\, ?_x\, b$) with ($a *_y b$) gives ($a *_t b$), where $*$ is a basic relation.
6. Merging ($a \text{※}_x b$) with ($a *_y b$) gives $a \text{※}_t b$.

Some merging rules are non-deterministic, i.e., produce multiple alternatives (Table 1.4). For instance, merging ($a \rightsquigarrow_x b$) and ($a \leftsquigarrow_y b$) gives two options: ($a \parallel_t b$) and ($a \text{※}_t b$). Parallelism means that there is no dependency between a and b, i.e., they can be used in any order. On the other hand, a dependency between a and b means that either profile x or y is incorrect, where it includes a data anomaly, e.g., missing data [20]. In this case, we conclude that there is a contradiction ($a \divideontimes_t b$). To resolve such uncertainties, a human intervention is needed, which is out of scope of this work.

An important property of these rules is *associativity*, where the order of merging behavioral profiles of business processes does not affect the global behavioral profile. Consider three profiles of three business processes, where two tasks appear in all three profiles. We can identify the following cases:

1. Three *similar* relations in all three profiles: According to the first rule, the result will always be the same relation in the global profile.
2. Three *different* ordering relations: According to the second rule, at least one of the merging steps results in a parallel relation. This resulting parallel relation occurs either as an intermediate (first merging two different ordering relations) or a final one (first merging two similar ordering relations). As an intermediate relation is further merged with the remaining order relation. This last merging step results in a parallel relation in the global profile according to the third rule. This shows that such merging steps always result in a parallel relation in the global profile.
3. Parallel and ordering: Merging three relations that include parallel and ordering relations results in a parallel relation in the global profile according to the third rule.
4. Exclusive and others: Merging three relations that include exclusive and ordering or parallel relations results in a contradiction relation in the global profile according to the fourth rule.
5. Unknown and others: Unknown relations do not affect the result of such merging steps according to fifth rule.
6. Contradicts and others: Contradicts relations always result in a contraction relation in the global profile according to the sixth rule.

The second component in the extended behavioral profile (besides the relation's type) is *distance* between services. This distance between two services in an intermediate profile is calculated as the shortest distance in the corresponding profiles unless one of both distances is zero, i.e., # or ||. In that case, we use the non-zero distance from both input relations.

1.6.2.1 Example Revisited

Assuming that the two processes from Fig. 1.5 are the only individual profiles in our knowledge base. By applying our merging rules shown in Table 1.4, we get the global profile shown in Table 1.5. This global profile has 9 web services that represent the union of all services in its source profiles, i.e., BP_1 and BP_2. For instance, merging relations ($\leftsquigarrow$, 1) and ($\leftsquigarrow$, 2) between web services A and B from BP_1 and BP_2, receptively, gives the relation ($\leftsquigarrow$, 1) in the global profile. The distance of the relation in the global profile is the minimum distance from input relations. Some merging

Table 1.5 Merging profiles of BP_1 and BP_2 (Fig. 1.5a,b) in one global profile

	A	B	C	D	E	F	G	H	I
A	(‖, 0)	(⇜, 1)	(‖, 0) (※, 0)	(⇝, 2)	(⇝, 2)	(⇝, 2)	(⇜, 2)	(⇝, 2)	(⇝, 8)
B	(⇝, 1)	(‖, 0)	(‖, 0) (※, 0)	(⇝, 3)	(⇝, 4)	(⇝, 4)	(# , 0)	(⇝, 4)	(⇝, 10)
C	(‖, 0) (※, 0)	(‖, 0) (※, 0)	(‖, 0)	(‖, 0)	(⇝, 2)	(⇝, 2)	(⇝, 2)	(⇝, 8)	(⇝, 14)
D	(⇜, 2)	(⇜, 3)	(‖, 0)	(‖, 0)	(‖, 0)	(‖, 0)	**(?, 0)**	**(?, 0)**	**(?, 0)**
E	(⇜, 2)	(⇜, 4)	(⇜, 2)	(‖, 0)	(‖, 0)	(# , 0)	(⇜, 4)	(# , 0)	(⇝, 2)
F	(⇜, 2)	(⇜, 4)	(⇜, 2)	(‖, 0)	(# , 0)	(‖, 0)	(⇜, 4)	(# , 0)	(⇝, 2)
G	(⇝, 2)	(# , 0)	(⇜, 2)	**(?, 0)**	(⇝, 4)	(⇝, 4)	(‖, 0)	(⇝, 4)	(⇝, 10)
H	(⇜, 2)	(⇜, 4)	(⇜, 8)	**(?, 0)**	(# , 0)	(# , 0)	(⇜, 4)	(‖, 0)	(⇝, 2)
I	(⇜, 8)	(⇜, 10)	(⇜, 14)	**(?, 0)**	(⇜, 2)	(⇜, 2)	(⇜, 10)	(⇜, 2)	(‖, 0)

rules produce multiple alternatives. For instance, A and C has the relation $(\rightsquigarrow, 2)$ and $(\leftsquigarrow, 4)$ in BP_1 and BP_2, respectively. Merging both relations gives two alternatives in the global profile between A and C: $(\parallel, 0)$ and (※, 0). The remaining relations can be derived in the same way. Merging extended behavioral profiles of BPs that do *not* have the same exact set of web services results in unknown relations between web services that do not appear in the same BP. For instance, relations between D from one side and G, H, and I on the other side in the global profile. In the sequel, we aim at using the knowledge gained from merging both profiles to reveal such unknown relations.

1.6.3 Predicting Unknown Relations ($a\,?\,b$)

Merging two profiles that do not have the same set of web services results in a global profile with unknown relations among web services that do not appear in both source profiles. In this section, we describe our approach to reveal such *unknown* relations by predicting potential resolutions for them.

We predict potential resolutions for the unknown relation between web services a and b in the global profile g with the help of a common service between them, e.g., c. Having more than one common service is resolved by intersecting all predicted relations from each common service according to Alg. 1. Our goal is to resolve the relation $(a\,?_g\,b)$ into $(a \leftsquigarrow_g b)$, $(a \rightsquigarrow_g b)$, $(a \parallel_g b)$, or $(a \#_g b)$ by investigating the relations between a and c on the one hand and between b and c on the other hand. We select a common service c such that we can derive useful information from its relations with the considered services. For instance, selecting c such that (a ※$_g$ c) is not of value. Therefore, the common service c has to be in one of the basic four relations with both a and b. Furthermore, the predicted relation has to be consistent

Table 1.6 Resolving the unknown relation $a\ ?_g\ b$ via a common service c

Relation	$a \rightsquigarrow c$	$a \leftsquigarrow c$	$a \parallel c$	$a \# c$
$b \rightsquigarrow c$	$a \rightsquigarrow b$ $a \leftsquigarrow b$ $a \parallel b$ $a \# b$	$a \leftsquigarrow b$	$a \parallel b$ $a \leftsquigarrow b$	$a \# b$ $a \leftsquigarrow b$
$b \leftsquigarrow c$	$a \rightsquigarrow b$	$a \leftsquigarrow b$ $a \rightsquigarrow b$ $a \parallel b$ $a \# b$	$a \parallel b$ $a \rightsquigarrow b$	$a \# b$ $a \rightsquigarrow b$
$b \parallel c$	$a \parallel b$ $a \rightsquigarrow b$	$a \parallel b$ $a \leftsquigarrow b$	$a \parallel b$ $a \rightsquigarrow b$ $a \leftsquigarrow b$	$a \parallel b$ $a \# b$
$b \# c$	$a \# b$ $a \rightsquigarrow b$	$a \# b$ $a \leftsquigarrow b$	$a \parallel b$ $a \# b$	$a \# b$ $a \rightsquigarrow b$ $a \leftsquigarrow b$

with existing relations in the global profile. Finding a useful resolution for unknown relations depends on the used knowledge, therefore it is not always possible to predict such a resolution. In such cases, the unknown relation between a and b ($a\ ?_g\ b$) in the global profile g remains and a human expert is informed about the situation to find a resolution manually if necessary.

We predict potential resolutions for each unknown relation between web services a and b in the global profile g—i.e., ($a\ ?_g\ b$)—using a common service, c, according to the set of rules that is summarized in Table 1.6. For instance, resolving ($a *_g c$) and ($b *_g c$) gives ($a \rightsquigarrow_g b$), ($a \leftsquigarrow_g b$), ($a \parallel_g b$), and ($a \#_g b$), where $* = \rightsquigarrow$ or $* = \leftsquigarrow$. Each of these predicted relations still preserves the existing relations ($a \rightsquigarrow c$) and ($b \rightsquigarrow c$) or ($a \leftsquigarrow c$) and ($b \leftsquigarrow c$). Resolving ($a \leftsquigarrow_g c$) and ($b \rightsquigarrow_g c$) gives ($a \leftsquigarrow_g b$). Any other relation, e.g., ($a \rightsquigarrow b$), does not preserve the existing relation between a, b on the one hand and c on the other hand. For instance, ($a \rightsquigarrow b$) means that b executes *before* a, that contradicts ($a \leftsquigarrow c$). Similarly, we cannot deduce that ($a \# b$) as it contradicts ($b \rightsquigarrow c$), since that implies either ($c \leftsquigarrow b$) or ($c \# b$), which is not the case.

Distances of the predicted $\rightsquigarrow$ and $\leftsquigarrow$ relations in the global profile are calculated according to the functions shown in Table 1.7. Distances are used to rank relevant web services during service discovery [1]. Additionally, we use them to prune possible resolutions. For some cases, the new distance is the absolute value of the difference of two distance. As an example, consider the case where we have ($a \rightsquigarrow c$) and ($b \rightsquigarrow c$). According to Table 1.6, all four basic relations are valid resolutions. For the predicted ($a \parallel b$) and ($a \# b$) we set distance to zero. However, for the two remaining cases, i.e. ($a \rightsquigarrow b$) and ($a \leftsquigarrow b$), the distance is the absolute value of the difference in input distances. When we have no information to calculate the distance, we set it to an artificial value infinity, e.g., the case of ($a \parallel c$) and ($b \parallel c$). For the

Table 1.7 Distances of the predicted relation $a\,?\,b$ via a common service c

Relation	$a \rightsquigarrow c$	$a \leftsquigarrow c$	$a \parallel c$	$a \# c$
$b \rightsquigarrow c$	$\lvert diff()\rvert$	$sum()$	$dist(b, c)$	$dist(b, c)$
$b \leftsquigarrow c$	$sum()$	$\lvert diff()\rvert$	$dist(b, c)$	$dist(b, c)$
$b \parallel c$	$dist(a, c)$	$dist(a, c)$	∞	N/A
$b \# c$	$dist(a, c)$	$dist(a, c)$	N/A	∞

cases where there is no order in the predicted relation between a and b, we express this using N/A in the table.

According to our rules of resolution shown in Table 1.6, possible resolutions to an unknown relation $a\,?\,b$ can include both $a \rightsquigarrow b$ and $a \leftsquigarrow b$. We use the distance information to prune one or both of these resolutions according to the following rules. Consider two relations $(a *_x b)$ and $(b \bullet_y c)$ with distances d_x and d_y, respectively, where $*$ and $\bullet$ are either $\rightsquigarrow$ or $\leftsquigarrow$, and Δd is defined as $d_x - d_y$, we identify three cases:

1. $\Delta d = 0$: The unknown relation $(a\,?\,b)$ cannot be predicted to $(a \rightsquigarrow b)$ or $(a \leftsquigarrow b)$.
2. $\Delta d > 0$: The unknown relation $(a\,?\,b)$ can be predicted to $(a \rightsquigarrow b)$, but not to $(a \leftsquigarrow b)$.
3. $\Delta d < 0$: The unknown relation $(a\,?\,b)$ can be predicted to $(a \leftsquigarrow b)$, but not to $(a \rightsquigarrow b)$.

Table 1.6 shows possible resolutions of $a\,?\,b$ using one common service c. However, a and b might have a set of common services, which includes services that have useful behavioral relations ($\rightsquigarrow$, $\leftsquigarrow$, $\parallel$, or #) with both a and b. In Algorithm 1, we show the steps we follow to achieve this resolution. We use each element in this set to predict the unknown relation between a, b according to the rules in Table 1.6 (Line 7). After that, we do an intersection among all possible resolutions deduced from each element in that set (Line 11). The resulting relations from this intersection are then used as potential resolutions to that unknown relation between a and b. If this intersection gives an empty set (e.g., due to contradictions), we are unable to predict resolutions for $a\,?\,b$ (Lines 12–13). These steps are repeated for all unknown relations in the global profile until no further resolutions are found.

1.6.3.1 Example Revisited

In Table 1.5, we show the global profile that we get by merging the extended global profiles of BP_1 (Fig. 1.5a) and BP_2 (Fig. 1.5b). That global profile has three unknown relations between service D on the one hand and services G, H, and I on the other hand, because these services are not used in the same BP. However, BP_1 and BP_2 have other common web services, e.g., A, B and C. We use such common services to predict resolutions for (part of) these three unknown relations.

Algorithm 1: Predicting unknown relations in the global profile

```
Require: $g$ the global profile
Ensure: $g'$ the global profile with some unknown relations revealed
 1: $pred \leftarrow \emptyset$
 2: for all $a?b \in g$ do
 3:   $CT \leftarrow getCommonTasks(a, b)$
 4:   for all $c \in CT$ do
 5:     $ac \leftarrow rel_g(a, c)$
 6:     $bc \leftarrow rel_g(b, c)$
 7:     $tmp \leftarrow predictRelaton(ac, bc)$        {According to Tables 1.6 and 1.7}
 8:     if $pred = \emptyset$ then
 9:       $pred \leftarrow tmp$
10:     else
11:       $pred \leftarrow intersect(pred, temp)$
12:       if $pred = \emptyset$ then
13:         break
14:       end if
15:     end if
16:   end for
17:   $g \leftarrow merge(g, pred)$        {According to Table 1.4}
18: end for
19: $g' \leftarrow g$
20: return $g'$
```

Table 1.8 Possible relations between services *D* & *G* via common services {A, B, C, E, F}

Common task	A	B	C	E	F
Relation with D	$D \leftsquigarrow A$	$D \leftsquigarrow B$	$D \parallel C$	$D \parallel E$	$D \parallel F$
Relation with G	$G \rightsquigarrow A$	G # B	$G \leftsquigarrow C$	$G \rightsquigarrow E$	$G \rightsquigarrow F$
Deduced relation	$D \leftsquigarrow G$	$D \leftsquigarrow G$	$D \rightsquigarrow G$	$D \leftsquigarrow G$	$D \leftsquigarrow G$
		D # G	$D \parallel G$	$D \parallel G$	$D \parallel G$

To predict $(D\ ?\ G)$, we select the set of common tasks among them. In this example, this set is {A, B, C, E, F}. Because $(D \leftsquigarrow A)$ and $(G \rightsquigarrow A)$, we deduce that $(D \leftsquigarrow G)$ according to the transitivity rule. Similarly, we deduce all potential relations between D and G using their common services as shown in Table 1.8. The intersection of these alternatives is ϕ, i.e., there is no common relation among potential relations. Therefore, the relation between D and G in the global profile remains unknown.

We follow the same steps to predict the relation $(D\ ?\ H)$. The set of common tasks is the same. Intersecting all potential relations between D and H gives the new relation $(D \parallel H)$. Again, the same set of common tasks is used to reveal the $(D\ ?\ I)$. In this case, the intersection of all potential relations between these tasks gives two alternatives: $(D \parallel I)$ and $(D \rightsquigarrow I)$. The distance in the new strict order relation is the minimum distance between I and the common tasks. In this case, the distance is 2. The global profile after revealing potential hidden relations is shown in Table 1.9.

Table 1.9 Revealing hidden relations in the global profile of BP_1 and BP_2

	A	B	C	D	E	F	G	H	I
A	(‖, 0)	(⇜, 1)	(‖, 0) (※, 0)	(⇝, 2)	(⇝, 2)	(⇝, 2)	(⇜, 2)	(⇝, 2)	(⇝, 8)
B	(⇝, 1)	(‖, 0)	(‖, 0) (※, 0)	(⇝, 3)	(⇝, 4)	(⇝, 4)	(# , 0)	(⇝, 4)	(⇝, 10)
C	(‖, 0) (※, 0)	(‖, 0) (※, 0)	(‖, 0)	(‖, 0)	(⇝, 2)	(⇝, 2)	(⇝, 2)	(⇝, 8)	(⇝, 14)
D	(⇜, 2)	(⇜, 3)	(‖, 0)	(‖, 0)	(‖, 0)	(‖, 0)	**(?, 0)**	(‖, 0)	(‖, 0) (⇝, 2)
E	(⇜, 2)	(⇜, 4)	(⇜, 2)	(‖, 0)	(‖, 0)	(# , 0)	(⇜, 4)	(# , 0)	(⇝, 2)
F	(⇜, 2)	(⇜, 4)	(⇜, 2)	(‖, 0)	(# , 0)	(‖, 0)	(⇜, 4)	(# , 0)	(⇝, 2)
G	(⇝, 2)	(# , 0)	(⇜, 2)	**(?, 0)**	(⇝, 4)	(⇝, 4)	(‖, 0)	(⇝, 4)	(⇝, 10)
H	(⇜, 2)	(⇜, 4)	(⇜, 8)	(‖, 0)	(# , 0)	(# , 0)	(⇜, 4)	(‖, 0)	(⇝, 2)
I	(⇜, 8)	(⇜, 10)	(⇜, 14)	(‖, 0) (⇜, 2)	(⇜, 2)	(⇜, 2)	(⇜, 10)	(⇜, 2)	(‖, 0)

1.7 Implementation and Evaluation

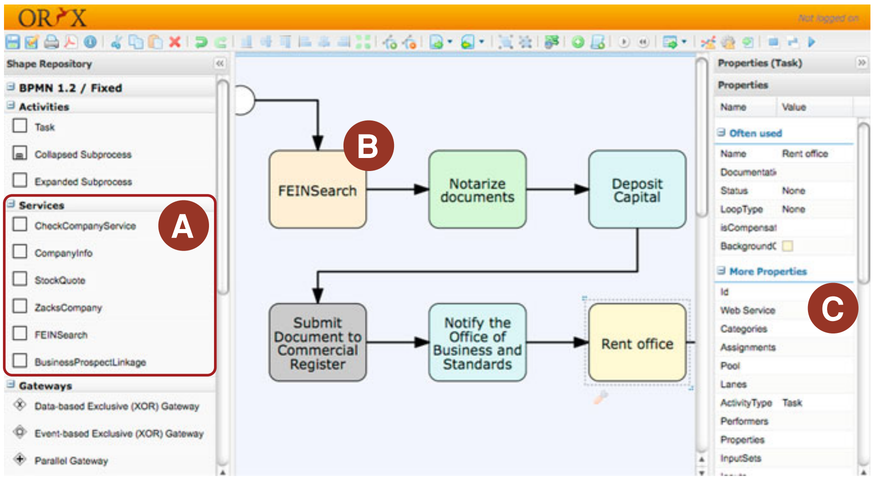
In this section, we describe the implementation of our prototype to validate our proposed approach, in addition to a set of experiments using a subset of the BPs from the SAP reference model.

1.7.1 Implementation: Integrating Depot and Oryx

We have developed a prototype that implements our approach to enrich service descriptions using business process configurations. In this section, we give details about the implementation of this prototype that integrates *Oryx*—a business process modeling platform and repository—and *Depot*—a web service registry.[4] The front-end of our prototype is Oryx, whereas Depot represents the backend. Figure 1.6 shows a screenshot of using our prototype to design a business process for establishing a company in Germany.

To create a new model for this process in Oryx, a proper title, such as "Establishing a company in Germany" is given by the process designer. The area labeled with *A* in Fig. 1.6 shows a list of web services discovered in Depot that have been used is similar contexts and are relevant to this process. For instance, the `whois` web services used in the UK example can be shown in this example despite the fact that there is no high similarity between terms appearing in "establish a company in Germany" and "whois". Each of these web services is already configured and can be simply dragged-

[4] The authors conducted this work during their stay at Hasso-Plattner-Institut at University of Potsdam, Germany



Name	Value
Name	Rent office
Documentati	
Status	None
LoopType	None
isCompensat	
BackgroundC	
Id	
Web Service	
Categories	
Assignments	
Pool	
Lanes	
ActivityType	Task
Performers	
Properties	
InputSets	

Fig. 1.6 A screenshot of our prototype used to model the process of establishing a company in Germany. *A* Suggested web services, *B* A pre-configured task from *A*, *C* Task properties

and-dropped to the design area. Indeed, the task labeled with *B* "FEINSearch" is an example of such pre-configured tasks. The web service assigned to this task gives company's details by its name or address. If the provided name does not exist, this hints that this name can be used as a name for the new company to be established.

1.7.2 Experiments

In this section, we show a set of experiments to evaluate our approach of predicting potential relations among web services using business process knowledge. We use a set of business processes from the SAP reference model [8], because these models represent possibilities to configure SAP R/3 ERP systems. Thus, it is analogous to business process configurations over a service landscape. We use 18 BPs with related missions from the SAP reference model. In particular, they are concerned with purchase order/requisition processing. These processes include 146 tasks in total. On average, each BP has about 8 tasks. Among the 146 tasks, 81 tasks are distinct, i.e., bound (configured) with distinct web services. We performed this configurations manually and verified the results manually as well. We analyzed the labels of the tasks and decided which labels (tasks) that can be bound to the same web service. Additionally, we had to manually restructure the models to have a single start and a single end node so that the behavioral profile calculation algorithm can be applied to them. Moreover, we excluded loops to obtain useful behavioral relations among tasks. A loop yields relations among all nodes within that loop concurrent.

Table 1.10 Types and ratios of relations in raw profiles, derived global profile, and resolved profile

Type	Raw processes (%)	Global profile (%)	Resolved global profile (%)
Strict Order $\rightsquigarrow$	33.25	3.87	14.48
Inverse Order $\leftsquigarrow$	33.25	3.87	14.48
Parallel $\parallel$	9.7	1.60	34.03
Exclusive #	23.8	3.42	17.97
Conflict ※	0	0.15	0.12
Unknown ?	0	87.10	18.91

The baseline approach is predicting relations among tasks of BPs *without* using their configurations information, i.e., only identical labels of tasks in different BPs are considered similar. Following this approach, we are able to predict resolutions for 54.8 % of all unknown relations in the generated global behavioral profile. The ratio of resolved relations using labels of tasks depends considerably only on the degree of similarity and cohesion among labels. Using the configurations of these BPs where semantically similar tasks are bound to a single web services, we are able to predict resolutions for around 72 % of all unknown relations among tasks used in our experiments.

We are able to reveal different types of relations among web services. In Table 1.10, we show the ratio of each type of relations with respect to the total number of relations in source profiles, their derived global profile, and after revealing part of the hidden relations in that global profile. Note that percentages in this table are local to each column. The majority of relations in the revealed global profile are parallel (34 %). Additional knowledge about such tasks and their bound web services can be used to resolve such relations in more concrete ones. This further resolution is part of our future work. Conflicting relations appear due to inaccurate configurations of BPs or due to lack of sufficient knowledge about tasks and web services. Unknown relations are still in the global profile even after applying our resolutions approach. Either the used knowledge is not sufficient to reveal such relations or there are no such useful relations. For instance, a music web service and a web service for Gene analysis.

1.8 Related Work

Bringing SOA and BPMs has been an active research topic. For instance Buchwald et al. propose an approach to bridge the gap between business process models and service compositions [6]. 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. use so-called behavioral profiles of business process models to extract association rules and action patterns among tasks [17]. Based thereon, process modeling tools can suggest to the user the insertion of 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 our 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 during the configuration of business process models.

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 approaches*: These approaches match inputs and outputs of operations of web services to find relations among them [9]. The main goal of these approaches is to investigate composability among web services [14, 16].
- *Semantic approaches*: These approaches apply Artificial Intelligence planning techniques to find valid compositions of web services [11, 12]. They are based on the assumption that web services are described formally using ontologies, such as `OWL-S`, `WSMO`, etc.
- *Service compositions-based approaches*: These approaches are based on the idea that web services used in a service composition are related [5, 25]. Compared to our approach, these approaches are unable to reveal hidden relations among web services that were never used in the same process model.
- *Consumer-consumer similarity approaches*: These approaches use the idea that similar service consumers usually use the same web services [15].

1.9 Summary

In this chapter, we introduced a novel approach to enrich poor service descriptions with information extracted from the configurations of BPs that consume them. We use business process configurations to discover fine-grained relations among web services used in such processes in the form of linkage patterns. 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.

Additionally, we introduced an approach to reveal hidden relations among web services by exploiting process configurations over these services. Typically, several process configurations exist. Therefore, we merge these individual profiles into a single global profile. After that, unknown relations within the global profiles were input to our prediction approach to reveal possible behavioral relations that might exist among them. To reveal these relations, we use common services between the two services with an unknown relation. We applied our approach to a subset of the SAP reference models and our experiments show that we could reveal about 72 % of the unknown relations in the global profile.

References

1. AbuJarour, M., Awad, A.: Discovering linkage patterns among web services using business process knowledge. In: Proceeding of the 8th International Conference on Services Computing, SCC (2011)
2. AbuJarour, M., Naumann, F., Craculeac, M.: Collecting, annotating, and classifying public web services. In: Proceedings of the 2010 International Conference on On-the-Move to Meaningful Internet Systems, OTM (2010)
3. AbuJarour, M., Oergel, S.: Automatic sampling of web services. In: Proceeding of the 9th International Conference on Web Services, ICWS (2011)
4. Awad, A.: BPMN-Q: A language to query business processes. In: Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures, EMISA (2007)
5. Basu, S., Casati, F., Daniel, F.: Toward web service dependency discovery for SOA management. In: Proceedings of the 5th International Conference on Services Computing, SCC (2008)
6. Buchwald, S., Bauer, T., Reichert, M.: Bridging the Gap Between Business Process Models and Service Composition Specifications, Chap. Methods, Trends and Advances, Int'l Handbook on Service Life Cycle Tools and Technologies (2011)
7. Buchwald, S., Tiedeken, J., Reichert, M.: Anforderungen an ein Metamodell für SOA-Repositories. In: Proceedings of the 2nd Central-European Workshop on Services and their Composition (Services und ihre Komposition), ZEUS (2010)
8. Curran, T.A., Keller, G., Ladd, A.: SAP R/3 Business Blueprint: Understanding the Business Process Reference Model, 1st edn. Prentice Hall (1997)
9. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB, (2004)
10. Fensel, D., Keller, U., Lausen, H., Polleres, A., Toma, I.: WWW or what is wrong with web service discovery? In: Proceedings of the W3C Workshop on Frameworks for Semantics in Web Services (2005)
11. Lecue, F., Leger, A.: Semantic web service composition based on a closed world assumption. In: Proceedings of the 2006 European Conference on Web Services (2006)
12. Lin, L., Arpinar, I.B.: Discovery of semantic relations between web services. In: Proceedings of the 2006 International Conference on Web Services, ICWS (2006)
13. Miller, G.A.: WordNet: a lexical database for english. Commun. ACM **38**, 38–41 (1995)
14. Omer, A.M., Schill, A.: 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, AUPC 09 (2009)
15. Rong, W., Liu, K., Liang, L.: Personalized web service ranking via user group combining association rule. In: Proceedings of the 2009 International Conference on Web Services, ICWS (2009)
16. Segev, A.: 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, CSSSIA (2008)

17. Smirnov, S., Weidlich, M., Mendling, J., Weske, M.: Action patterns in business process models. In: Proceedings of the 7th International Conference on Service-Oriented Computing, ICSOC/ServiceWave (2009)
18. Sreenath, R.M., Singh, M.P.: Agent-based service selection. J. Web Sem. **1**(3), 0–0 (2004)
19. Stein, S., Barchewitz, K., El Kharbili, M.: Enabling business experts to discover web services for business process automation. In: Proceedings of the 2nd Workshop on Emerging Web Services Technology, WEWST (2007)
20. Sun, S.X., Zhao, J.L., Nunamaker, J.F., Liu Sheng, O.R.: Formulating the data-flow perspective for business process management. Info. Sys. Research **17**(4), 374–391 (2006)
21. van der Aalst, W.M.P., van Hee, K.M.: Workflow management: models, methods, and systems. MIT Press, London (2002)
22. van der Aalst, W.M.P., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
23. van Dongen, B.F., Mendling, J., van der Aalst, W.M.P.: Structural patterns for soundness of business process models. In: EDOC, pp. 116–128. IEEE Computer Society (2006)
24. Weidlich, M., Polyvyanyy, A., Mendling, J., Weske, M.: Efficient computation of causal behavioural profiles using structural decomposition. In: Proceedings of the 31st International Conference on Applications and Theory of Petri Nets, Petri Nets (2010)
25. Winkler, M., Springer, T., Trigos, E.D., Schill, A.: Analysing dependencies in service compositions. In: Proceedings of the 2009 International Conference on Service-Oriented Computing, ICSOC/ServiceWave (2009)
26. Weikum G. et al.: The YAGO-NAGA project: harvesting, searching, and ranking knowledge from the web. http://www.mpi-inf.mpg.de/yago-naga/