Interface- and Usage-aware Service Discovery

Rimon Mikhaiel and Eleni Stroulia

Computing Science Department, University of Alberta, Edmonton AB, T6G 2H1, Canada
{rimon,stroulia}@cs.ualberta.ca

Abstract. To date, research on web-service discovery has followed the traditional component-discovery methodology and has examined signature matching, specification matching and information retrieval approaches, based on the interface description and documentation captured in WSDL. WSDL specifications, however, can be information poor, with standard data types, unintuitive identifiers for data, messages and operations and little natural-language documentation. The nature of the usage of the WSDL elements in the context of a BPEL composition can be an extremely useful source of information in the context of service discovery. In this paper, we discuss our method for service discovery based on interface and usage matching, exploiting the information captured in the WSDL and BPEL specifications. Our approach views both WSDL and BPEL as hierarchical structures and uses tree alignment to compare them in order to assess their similarity and to recognize the correspondences between their elements. We illustrate our method with two example scenarios.

Keywords: service discovery, WSDL matching, BPEL matching, process matching

1 Motivation and Background

Service discovery is an important task essential in developing service-oriented applications. In a typical service-discovery scenario, the service requester is looking for a service to complete a composite application and has specific expectations about the candidate service. In general, there are three types of desiderata for a service: it should (a) perform a certain task, i.e., maintain a shopping cart, (b) expose a particular interface, i.e., view, add-product and remove-product, and (c) behave in a certain manner, i.e., ignore any request for product removals if no product additions have been performed yet. Such expectations motivate and guide the developers' searches through web-services repositories, as they try to discover and select the service that best matches their needs.

To date, web-service discovery approaches have followed the "traditional" research methods of component discovery through either signature or specification matching. Signature matching assesses the similarity of the requested and provided interface, in terms of data types, messages and operations. Specification matching focuses more on the nature of the task delivered by the candidate service and its logical relation to the task required. Information-retrieval methods query the repository for services whose

identifiers and documentation are similar to the natural-language description of the developers' request.

This research views web services as traditional components, consisting of an accessible specification and a hidden implementation. As a result, the precision of the discovery process tends to be limited in that (a) often irrelevant services appear as plausible candidates and (b) even when relevant services are selected, it is not clear how exactly their data and operations match the requestor's specification. However, this stance is unnecessarily limiting: a WSDL specification of a complex web service may be associated with a BPEL specification declaratively describing the process by which the public WSDL operations are used. This specification provides two important types of information, particularly relevant to the task of service discovery: (a) the usage protocol of the provided operations and the (b) overall usage patterns of the data types. Essentially, this information enables us to consider the third type of the requester's expectations described above in the content of service discovery by assessing the similarity of two examined services based on how well the usage protocol of the candidate service operations aligns with the expected behavior of desired service in the context of its composition with other services in the service-oriented application.

In this paper, we discuss our method for service discovery using both WSDL and BPEL. Our method involves the following steps: (a) the relevant information from the WSDL specification of the services and the BPEL specification of its example usage is parsed and represented in a special-purpose tree representation; and (b) the tree representing the specifications of the desired service is compared against the tree specifications of the candidate services to select the most similar one and to precisely identify how the data and operations offered by the candidate map to the requestor's needs.

The rest of this paper is organized as follows. Section 2 reviews related research on service discovery. Section 3 discusses two case studies illustrating the insufficiency of WSDL for effective service discovery and the relevance of BPEL and usage-protocol information to the task. Section 4 presents our service-matching method based on tree alignment. Section 5 shows how the presented method resolves the issues raised in Section 3. Finally, Section 6 concludes with the essential ideas of our work.

2 Related Research

Interface matching examines the inter-operability between a published service and a requested one. This type of approaches are concerned with mapping the elements of a candidate published interface to the elements of the requested one. Usually, such a mapping is based on signature matching between the published operations and the requested ones. For example, Wang and Stroulia [11] proposed a family of WSDL matching methods that consider both the identifier and structural similarity of data types and methods. Payne et *al* [9] developed a DAML-S matching method, assuming a common ontology between the publisher and the requester, based on parameter matching using type subsumption and inheritance relationships. Syeda-Mahmood et *al* [10] proposed an interface matching approach that is based on name similarity.

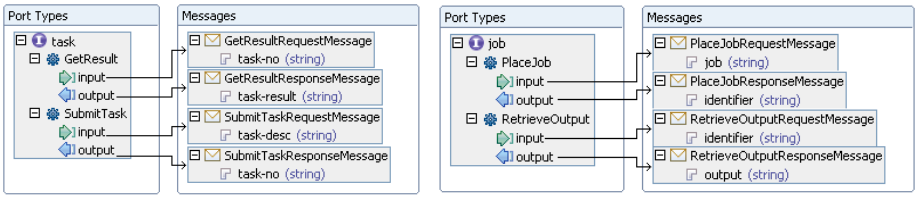
However, these approaches suffer from two drawbacks: first, interface matching does not guarantee a successful interaction because such an interface usually does not specify the usage conditions of the operations involved. Hence, an improper usage of the published operations would lead to an interaction failure. Second, interface matching may easily get confused when services are not distinctive; because it relies on documentation and on parameter lists, when the data types are simple and there is not much documentation there is not enough information on the basis of which to assess (dis)similarity.

Specification-matching approaches are based on matching the operation description of both the published service and the requested one. However, these approaches have a common drawback that they match one aspect of a process description: either control structure (like Petri nets [4], and π-calculus[7]), or message flow (like WSDL [2][3], and BPEL finite state machines [12]).

3 The Service-Discovery Problem in two Examples

In general, when looking for a service, a developer has in mind both the signatures of the operations desired and some behavioral scenarios, in which the candidate service is expected to participate. Discovery based on WSDL matching only is concerned only with the matching of the operations desired and provided. However, given a candidate service, there usually exist multiple likely mappings between the desired operations and the ones provided by the candidate service. Selecting one of these mappings is often impossible when neither the syntactic types nor the identifiers and documentation are distinctive. In this section, we discuss two such cases.

3.1 Simple data types



(a) A published WSDL description **(b) A requested WSDL description**

Fig. 1. A visual representation of two WSDL descriptions. *Operations* are shown on the left-hand side and *messages* are shown on the right-hand side. Each operation is connected to its associated *input* and *output* messages. Each message description includes its associated *parameter name* and *type*.

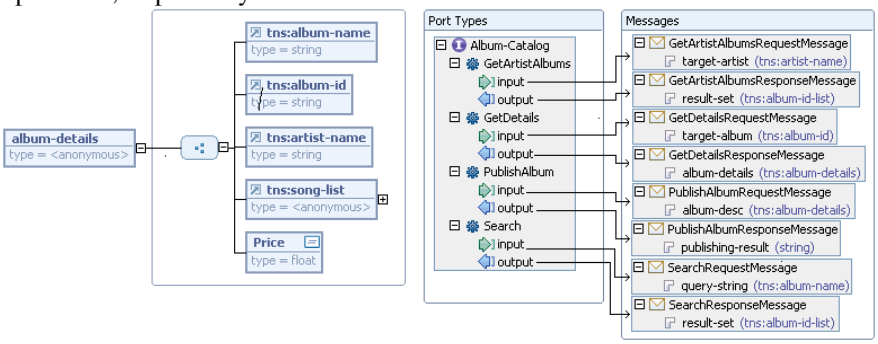
Consider, for example, the case when the provided service performs an asynchronous task: the consumer is expected to first submit the task and obtain a receipt and then return for the results associated to that receipt; if, at that time, the task is completed the results are returned to the consumer otherwise an exception is returned. Fig. 1

illustrates the WSDL specifications of a provided and requested service, both essentially involving such a task-submission service.

Assuming only the information shown in Fig. 1, there are two possible and equally likely mappings between the operations requested and provided. The published operation `GetResult` could be equally likely mapped to the requested `PlaceJob` or `RetrieveOutput` (with corresponding mappings for `SubmitTask`), since all the involved message parameters are strings with not particularly distinct names and there is no real reason to prefer mapping the `job` to the `task-desc` instead of the `task-no`, which would actually lead to the correct mapping of `GetResult` to `RetrieveOutput`.

3.2 Divergent Application Domains

Consider now the case of a product catalog service, originally designed in the music domain (shown in Fig. 2), when a consumer is interested in a book catalog (shown in Fig. 3). Fig. 2(a) shows that the published service deals with an `album-details` ontology; while Fig. 3(a) shows that the consumer is interested in `book` ontology. Additionally, both Fig. 2(b) and Fig. 3(b) show the published and expected operations, respectively.

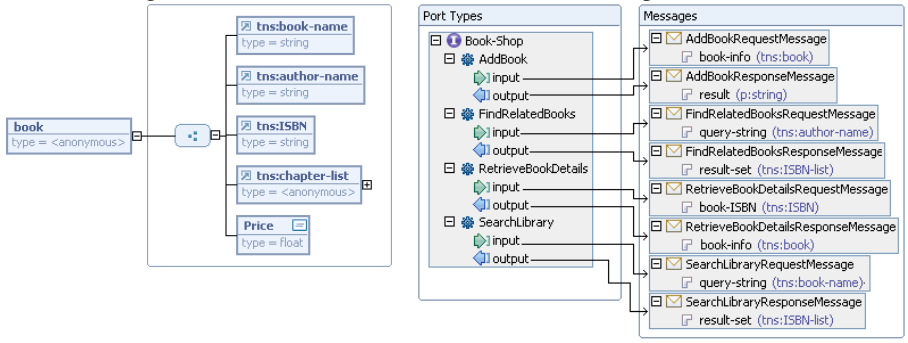


(a) The published album ontology **(b) The published WSDL description**

Fig. 2. A visual representation of the published WSDL description; to the left, is a visual representation of the XML *schema* definition; to the right, the operations and message descriptions are shown.

Given our current experience and understanding of on-line product catalogs, it is unlikely that a catalog service would be developed to include such domain-specific details. However, in general, domain-specific assumptions about the application ontology may "seep through" the service design. In such cases, although the published service deals with a different ontology than that of the service consumer, the consumer could still effectively use the service, if only the correct mapping between the divergent ontology elements were found. For example, in this scenario, both services register items, search the catalog, get item details, and find other items from the same producer/publisher. It would therefore be desirable to "discover" the published service in response to such a request.

The challenge then becomes to establish the proper mapping between the two divergent ontology elements and service operations. Mapping the `album-details` ontology of Fig. 2(a) against the `book` ontology of Fig. 3(a) would result in mapping the `Price` elements to each other (they have the same name and data type) and the `song-list` to the `chapter-list` (they are both `sequences`). However, there are no distinguishing features to guide the mapping between the remaining elements (all of them have the same data types and their names are inspired from different domain ontologies and are dissimilar) and hence there are six possible combinations.



(a) The requested book ontology **(b) The requested WSDL description**

Fig. 3. A visual representation of the consumer's expected WSDL description.

4 The Method

WSDL descriptions, including their associated XML Schemas underlying the service data types, do not contain sufficient information to resolve the ambiguity in mapping the requested and provided operations, messages, and data types. To inject more information in the service-discovery process, one might assume richer representations, such as those advocated by the semantic-web effort. Until the adoption of such representations becomes more prevalent, however, we can exploit the information on the operational semantics of the WSDL implied by its internal behavior represented in is associated BPEL.

In the web-services stack of standards, three aspects of a service are declaratively described in XML syntax: its data types (WSDL types and XML Schema), its functionalities (WSDL operations and messages) manipulating and transforming these data types, and, if it is part of more complex composite service, its behavior (BPEL flows) while delivering its functionalities.

The basic intuition underlying the method presented in this paper is that all these three aspects of a service description provide information that can be useful when considering a candidate service as a potential match for a requested one. More specifically, ambiguities in data mapping can be resolved by considering the roles of the data types in the operations provided by the service. Furthermore, ambiguities in operations' mapping can be resolved by examining the underlying behaviors accomplishing these operations.

Thus, our method is based on a three-level comparison of the corresponding service aspects, where a specialized tree-alignment approach adopted for each level. First, the BPEL behavioral specification of the provided service is compared against the behaviors expected of the requested service. Next, the operations of the desired and the candidate services are compared, using a cost function that is based on the results of BPEL matching of the first step. Finally, the data types of the two services are matched using another cost function that is based on operation matching of the second step.

4.1 Web-service Specifications as Trees

For each of the three steps in the service aspects' comparison, we adopted the SPRC [7] tree-alignment algorithm, developed in the context of our work in RNA structure alignment. SPRC takes as input two *labeled ordered trees* and produces as output the minimum edit sequence that can transform one into the other. A labeled tree is a tree where each node has a label that if changed would result a different tree. Additionally, in ordered trees the relative order of sibling nodes and the parent-child relationships between nodes are significant, i.e. changes to the ancestor-descendant and sibling-order relationships results in a different tree.

Both BPEL and WSDL descriptions are expressed in terms of XML documents that can be represented as trees, where each XML element is represented by a tree node and an element's contained elements and attributes are represented as children nodes of that element's node.

4.2 Service Discovery as Tree-Edit Distance Minimization

Representing the BPEL and WSDL descriptions as ordered labeled trees, the problem of service discovery becomes to "align" the requested service with the candidates in order to select this candidate that has the minimum-cost edit sequence with the requested one, to which the SPRC algorithm can be applied.

The SPRC tree-alignment algorithm is based on the Zhang-Shasha tree-edit distance [13] algorithm, which calculates the minimum edit distance between two trees given a cost function for different edit operations like node change, deletion, and insertion. In SPRC, the Zhang-Shasha algorithm has been modified to use an affine –cost (i.e., context sensitive) policy and to report all the alignment solutions that are associated with the calculated tree edit distance.

As a post-processing step, SPRC applies a set of simplicity heuristics to reduce the cardinality of the set of reported edit sequences. The objective of SPRC's simplicity-based filtering is to discard the more unlikely solutions from the solution set produced by the SPRC tree-alignment phase. There are three simplicity heuristics.

(1) **Solution Minimality**: The first simplicity heuristic advises the algorithm to "*prefer minimal edit sequences*": when there is more than one different sequence with the same minimum cost, the one with *the least number of deletion and/or insertion operations* is preferable.

(2) **Vertical Simplicity**: The second simplicity heuristic advises the algorithm to "*prefer contiguous similar edit operations*". Intuitively, this rule says that the contiguous same edit operations could be considered as one single operation that involves long segments of XML nodes. When there are multiple different paths with the same minimum cost and the same number of editing operations, the one with *the least number of changes of operation types along a tree branch* is preferable.
(3) **Horizontal Simplicity**: In addition to maximizing the number of nodes along a tree branch to which the same edit operation is applied, SPRC also proposes that, to the extent possible, *sibling nodes should also suffer the same edit operations*.

4.3 A Usage-Aware Cost Function

As we have discussed earlier, the specific service-discovery issue we address in this work is to resolve the ambiguities –with respect to data and operations mappings– that frequently occur when a candidate service has been selected in response to a requestor's query using WSDL matching only. In many cases, because the data types are simple and the identifiers are not distinctive, there are multiple equally likely mappings between the requested service operations and the ones provided by the candidate. In such cases, we propose that the BPEL specifications of the usage protocols of the provided service operations and the behaviors expected of the requested service can be examined to guide the process towards a less ambiguous mapping.

In our three-level tree-alignment method, the similarity of the elements usage guides their mapping. More specifically, the correspondences identified between operations after aligning the BPEL specifications (i.e., the usage of the operations in the process flow) inform the WSDL alignment and, similarly, the correspondences identified between data during WSDL-message-operations alignment (i.e., the usage of data in the service functionalities) inform the WSDL-data-type alignment.

This is accomplished through the design of an appropriate cost function for each tree-alignment step. A cost function is used to evaluate the cost of various tree editing operations, e.g. change, delete, and insert. In our method, the results of a higher-level alignment step affect the cost function of the next level step. In other words, if the references of two elements are mapped to each other in the former step, then the cost of mapping the two elements in the later step is reduced, proportionally to the *degree of mapping* (DOM). The degree of mapping is defined as twice the number of mapped references of two elements divided by the total number of references of both elements.

For example, the WSDL operation matching step's cost function is defined as

$$\gamma_{operation}(op) = \begin{cases} \gamma_{change}(1 - Dom_{BPEL}(x,y)) & op = (x,y) \\ \gamma_{delete} & op = (x,-) \\ \gamma_{insert} & op = (-,y) \end{cases}$$

where $Dom_{BPEL}(x,y)$ is the degree of mapping ratio that is calculated as twice the number of references where element (operation or message) x in $BPEL_1$ is mapped to references of element (operation or message) y in $BPEL_2$ divided by the total number

of references of both elements. Similarly, the data-type matching step's cost function is defined as:

$$\gamma_{onto\log y}(op) = \begin{cases} \gamma_{change}(1 - Dom_{operation}(x,y)) & op = (x,y) \\ \gamma_{delete} & op = (x,-) \\ \gamma_{insert} & op = (-,y) \end{cases}$$

Incorporating the DOM ratio of a successive matching step into the current matching step would incorporate the usage similarity of references (of the successive step) with the matching the real components of the current step.

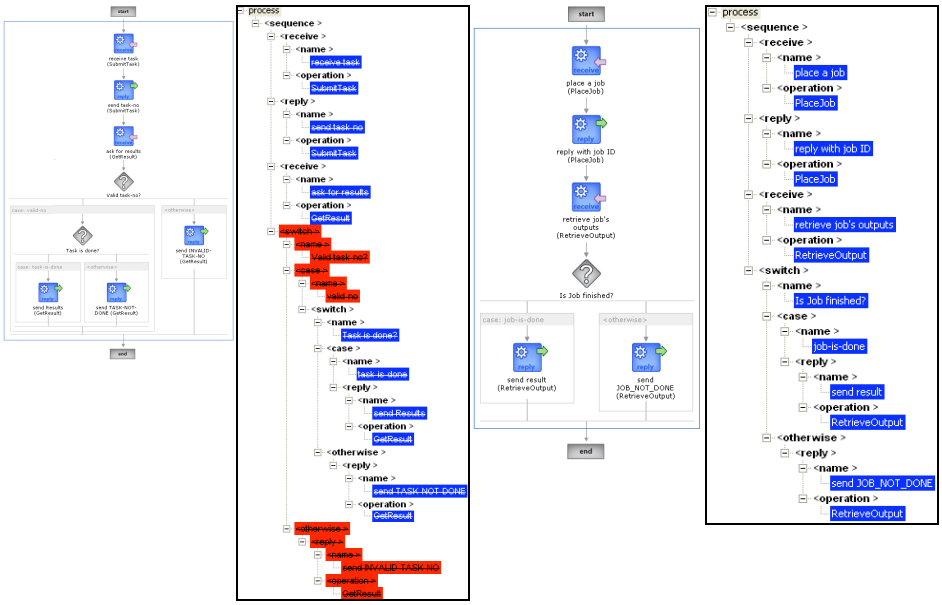
5 The Examples Revisited

In this section, we revisit the two examples in Section 3. Fig. 4 shows the BPEL descriptions of the candidate published service and the requested service from the example discussed in Section 3.1. Fig. 4(a) illustrates the workflow of a published service. In this figure, the service provider implicitly describes how that service is supposed to be used. Fig. 4(c) shows the expected scenario from the consumer point of view. The diagrams (a) and (c) provide a visual representation of the BPEL specification while the diagrams (b) and (d) represent the two specifications as trees. In the diagrams (a) and (c), each *action* is annotated with its name and, between parentheses, the name of the *operation* to which it belongs. There are two types of message actions: (1) a *receive* action annotated with a pink left arrow, and (2) a *reply* action annotated with a green right arrow. In the diagrams (b) and (d), *deleted* nodes are highlighted with a red background, while *changed* (or replaced) nodes are highlighted with a blue background.

The result of aligning the BPEL descriptions of Fig. 4(a) and Fig. 4(c) is shown in Fig. 4(b) and Fig. 4(d): the receive action named "receive task" associated with the operation named `SubmitTask` is mapped to the receive action named "place a job" associated with the operation named `PlaceJob`. Similarly, all references to the operation `SubmitTask` are mapped to references to the operation `PlaceJob`, and vice versa. Fig. 4(b) and Fig. 4(d) show that all references to the operation `GetResult` are mapped to references to the operation `RetrieveOutput` and vice versa. Hence, we can conclude the following $Dom_{BPEL}(x,y)$ function:

X\y	PlaceJob	RetrieveOutput
GetResult	0%	100%
SubmitTask	100%	0%

Therefore, the cost function for the subsequent alignment step is advised to reduce the mapping cost for both (`SubmitTask, PlaceJob`) and (`GetResult, RetrieveOutput`) to zero. Thus, there is no longer any ambiguity for the operations and message matching process.



(a) Visual representation **(b) Tree representation** **(c) Visual representation** **(d) Tree representation**

Provider's published BPEL **Consumer's expected BPEL**

Fig. 4. Visual and tree representations of the BPEL specifications of the two services in the simple-data-types example.

Applying the tree-alignment algorithm for the BPEL trees in Fig. 4(a) and Fig. 4(c) results in ten possible edit sequences. For example, in Fig. 4(a), there are two switch actions: an outer one named "valid task no?", and an inner one named "Task is done?"; while in Fig. 4(c), there is only one switch action named "Is job finished". Hence, intuitively, there should be at least two possible alignments: (1) to map the outer switch in $BPEL_1$ to the switch in $BPEL_2$ while deleting the inner one, and (2) to map the inner switch in $BPEL_1$ to the switch in $BPEL_2$ while deleting the outer one. Fig. 5 shows a subset of the produced solutions of the tree-alignment algorithm. Due to space limitations, only the tree representing $BPEL_1$ is shown. This figure illustrates that there are different combinations of mapping the elements between the two given trees.

It is interesting to note that applying the simplicity heuristics vertically and then horizontally reduced the size of the solution set from 10 to 3 and then 2 solutions. For example, counting the vertical refraction points for the four different solutions in Fig. 5(a), (b), (c), and (d) would result in 2, 2, 5, and 6 points, respectively. Hence, solution in Fig. 5 (c), and (d) are discarded because they don't have the least number of refraction points. The remaining solutions are the best in that they map complete branches across the two trees. Furthermore, the number of horizontal refraction points for the maintained solutions (namely, in Fig. 5(a) and (b)) are 2 and 4 points, respectively. Hence, solution Fig. 5(b) is discarded as not having the least number of horizontal refraction point, which is also justified as it tries to map dispersed siblings

to adjacent ones. Hence, solution in Fig. 5(a) is relatively the best in mapping whole sub-trees to whole sub-trees.

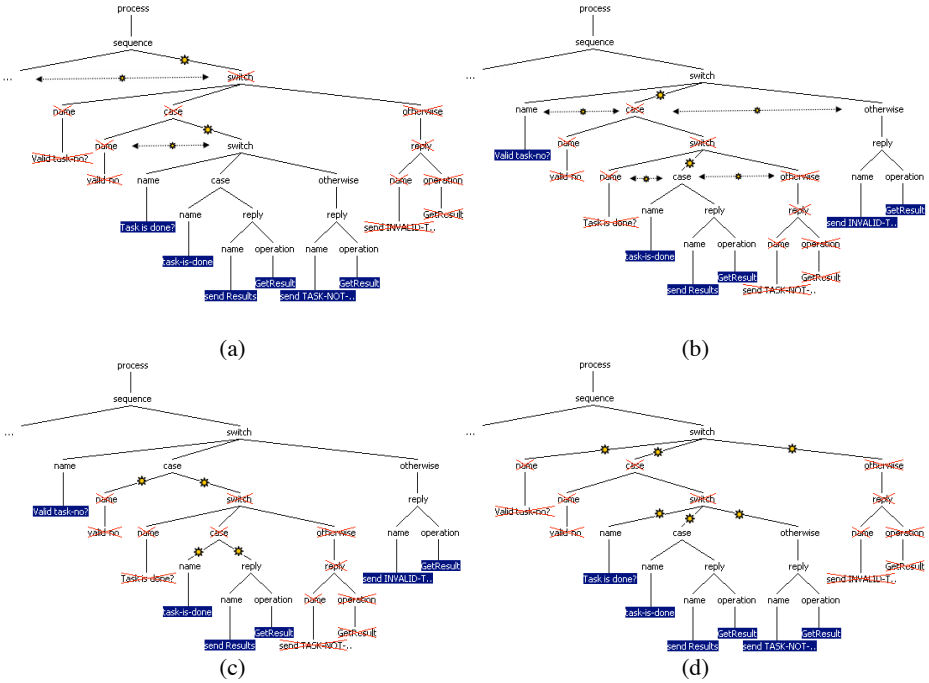
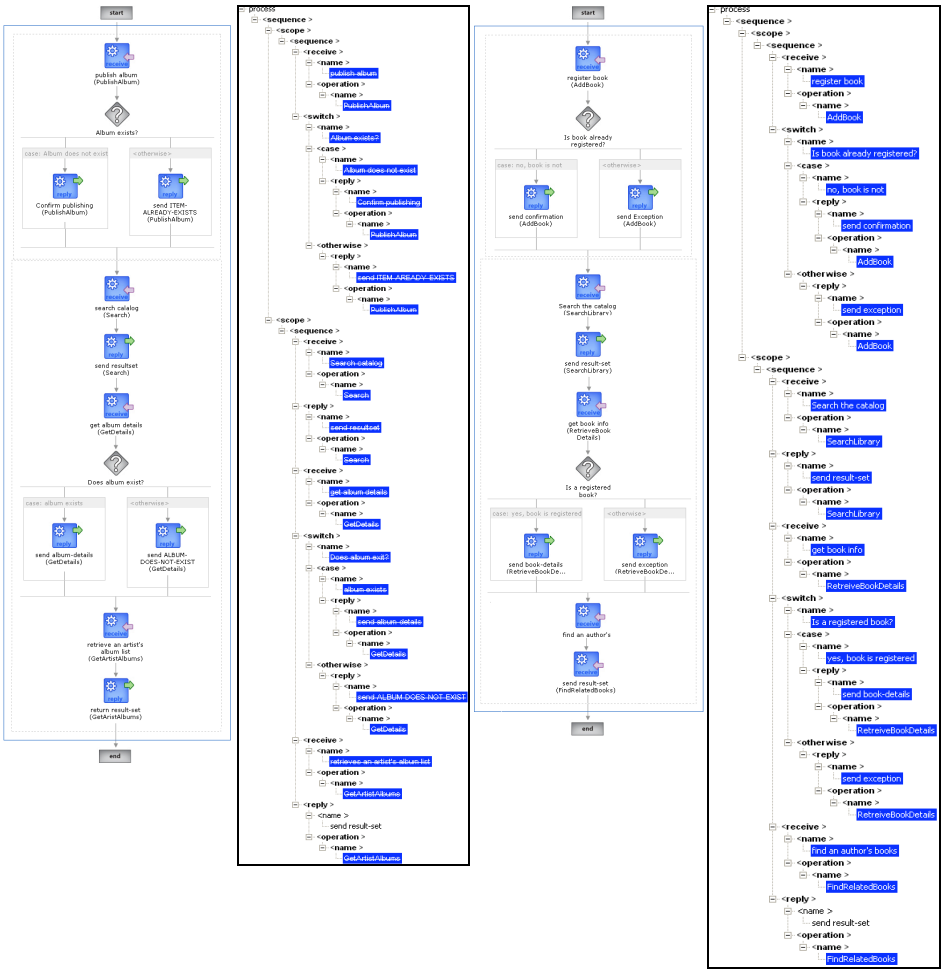


Fig. 5. A subset of the solution set of example 1. Each of these figures shows a visual representation of the changes to $BPEL_I$ implied by the alignment process. In each of these figures, a *red cross* means a deletion, a *blue highlight* means a change. Additionally, a *yellow star* refers to a vertical refraction point, while a *dotted arrow with a yellow star in the middle* refers to a horizontal refraction point.

The ambiguity issues in the example of Section 3.2 are also resolved through matching the usage of both the operations and the data types. For example, matching the BPEL description in Fig. 6(a) against the one in Fig. 6(b) reports that references to operations `PublishAlbum`, `Search`, `GetDetails`, and `GetRelatedAlbums` match the references to operations `AddBook`, `SearchLibrary`, `GetBookInfo`, and `FindAuthorBooks`, correspondingly. Using the resulting degree of mapping (DOM), these operations are mapped to each other, and, as a result, their signatures including data-type references are also mapped. Thereby, the earlier ambiguity in the mapping of the data-type elements is also resolved. For example, the published `album-name`, `artist-name`, and `album-id` match the requested `book-name`, `author-name`, and matches `ISBN`, respectively.



(a) Visual representation **(b) Tree representation** **Provider's published BPEL** **(c) Visual representation** **(d) Tree representation** **Consumer's expected BPEL**

Fig. 6. Visual and tree representations of the BPEL specifications of the two services in the divergent-application-domains example.

6 Conclusions

In this paper, we discussed our approach to resolving ambiguities in the mapping of discovered service elements to those of the service requested by the consumer, by examining the usage of these elements in the context of BPEL-specified behavioral specifications of the service in action. The basic intuition underlying our work is that there are three types of information (in the web-services stack of specification

standards) relevant to deciding whether or not a particular discovered service should be adopted in the context of a new service-oriented application under composition: (a) its data types and their identifiers (indicating the application-domain ontology), (b) the syntactic structure of its interface (indicating the number and type of functionalities provided by the service), and (c) the behavioral usage protocol of the service functionalities (indicating the role that these functionalities can play in the context of an overall application). Our service discovery method is based on tree-alignment of the three corresponding specifications, with results from each step feeding into the alignment of the next step below. In this paper, we have illustrated our method with two illustrative examples; a more extensive and systematic experiment against a full-fledged service repository is under way.

Acknowledgements

This research was supported by ICORE, NSERC and the Alberta IBM CAS.

References

[1] Andrews, T., Curbera, F., Dholakia, H., Goland, Y., Klein, J., Leymann, F., Liu, K., Roller, D., Smith, D., Thatte, S., Trickovic, I., and Weerawarana, S. “Business Process Execution Language for Web Services.” version 1.1, 2003, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/

[2] Blanchet, W., Elio, R., Stroulia, E. “Conversation Errors in Web Service Coordination: Run-time Detection and Repair”. Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2005.

[3] Blanchet, W., Elio, R., Stroulia, E. “Supporting Adaptive Web-Service Orchestration with an Agent Conversation Framework”. Proceedings of the third IEEE International Conference on Web Services (ICWS), 2005.

[4] Brockmans, S., Ehrig, M., Koschmider, A., Oberweis, A., and Studer, R., “Semantic Alignment of Business Processes”. In 8th International Conference on Enterprise Information Systems, 2006.

[5] Cho, J. McGregor, and L. Krause. "A protocol-based approach to specifying interoperability between objects". In Proceedings of the 26th Technology of Object-Oriented Languages and Systems (TOOLS'26), 1998.

[6] Christensen, E., et al. “The web services description language WSDL.” http://www.w3.org/TR/wsdl

[7] Mikhaiel, R., Lin, G., and Stroulia, E., “Simplicity in RNA Secondary Structure Alignment: Towards biologically plausible alignments” Submitted to BIBE 2006.

[8] Milner, R., Parrow, J., and Walker, D., “A Calculus of Mobile Processes, Part I+II”. Journal of Information and Computation, September 1992, 1--87.

[9] Payne, T.R., Paolucci, M., and Sycara, K., “Advertising and Matching DAML-S Service Descriptions”. Semantic Web Working Symposium (SWWS), 2001.

[10] Syeda-Mahmood, T.F., Shah, G., Akkiraju, R., Ivan, A.A., and Goodwin, R., “Searching Service Repositories by Combining Semantic and Ontological Matching”. ICWS, 2005, 13--20.

[11] Stroulia, E., and Wang, Y., “Structural and Semantic Matching for Assessing Web-Service Similarity”, International Journal of Cooperative Information Systems, 14(4):407-437, June 2005.
[12] Wombacher, A., Fankhauser, P., and Neuhold, E. “Transforming BPEL into Annotated Deterministic Finite State Automata for Service Discovery.” ICWS, 2004, 316--323.
[13] Zhang, K., Stgatman, R., and Shasha, D. “Simple fast algorithm for the editing distance between trees and related problems.” SIAM Journal on Computing, 18(6), 1989, 1245--1262.