Enabling Business Experts to Discover Web Services for Business Process Automation

Sebastian Stein, Katja Barchewitz, and Marwane El Kharbili

IDS Scheer AG
Altenkesseler Str. 17
66115 Saarbrücken, Germany
(sebastian.stein|katja.barchewitz|marwane.elkharbili)
@ids-scheer.com,
WWW home page: http://www.ids-scheer.com/soa/

Abstract. Using Web services for business process automation is an accepted approach in context of service-oriented architectures (SOA). Business process models are created by business experts usually not having an IT background and who are therefore not able to use the technical descriptions available for web services. In this paper, we show how we extended the market leading business process management suite ARIS to enable business experts to discover, assess, and select Web services for business process automation. We developed a structural and a semantic matching algorithm as well as a graphical user interface for Web service assessment. We use a schema to classify Web service discovery literature and we relate our work to it. Our completely integrated discovery tool helps bridging the gap between business and IT, because business experts can now discover Web services needed for business process automation on their own.

1 Introduction

A company is driven by its business processes and their interfaces to the outside world. Those business processes are documented using business process notations like Event-driven Process Chains (EPC) [1]. Such process models are used on different abstraction levels, for example to specify how a business activity should run in general but also how a certain sub-process should be implemented using information systems. It is the grand vision of business process management (BPM) [2] to directly derive the process implementation from the business process models created by business experts.

In past years, the idea of service-oriented architectures (SOA) [3] became a popular approach for integrating information systems to support business process automation. SOA itself is just an architectural style, but not a specific technology. There seems to be a preliminary consensus in enterprise computing that Web service technology [4, see e. g.] is the preferred SOA implementation solution. Steps in a business process are automated by Web services and the business process models are afterwards transformed into process execution languages like the Business Process Execution Language (BPEL) [5].

Using Web services for business process automation has a major drawback: Business processes are described by business experts, whereas Web services are described on a technical level. The technical Web service description is not usable for business experts and therefore they are not able to select a web service to automate a certain step in a business process. To overcome this problem, we developed a tool and method allowing business experts to discover Web services for process automation. The discovery tool is completely integrated in the world market leading[1] software for business process analysis and management *ARIS*[2] and the belonging ARIS methodology [8] for business process management.

This article is structured as follows. In the next section we provide a detailed literature review of Web service discovery algorithms. Based on existing literature, we develop a classification for Web service discovery approaches. In section 3 we present our solution. First, we discuss in sub-section 3.1 the general structure of our solution and how our solution can be classified according to the classification schema developed. In sub-sections 3.2 and 3.3 we describe how Web services and data structures are represented in ARIS. This information is important in order to understand the description of the two matching algorithms. Our structural matching algorithm for Web service discovery is explained in sub-section 3.4. Our semantic matching algorithm for Web service discovery is explained in sub-section 3.5. In sub-section 3.6 we present the graphical user interface we developed to allow business experts to assess the matching results and to make an informed decision of the Web service to be used. In section 4 we give an example to better illustrate our solution. Finally, we present our conclusions at the end of the article.

2 Literature Review and Theoretical Foundation

We investigated service discovery literature. Even though we were not able to identify any specific publication dealing with Web service discovery for business process automation, we found many publications related to web service discovery in various domains. For example, many publications are targeting Web service discovery in context of Grid computing [9]. Here, services are bound during execution often based on quality of service (QoS) parameters. A related domain is agent systems [10, see e. g.] trying to identify a communication partner with a set of defined capabilities. Other publications deal with identifying Web services in context of software engineering. The idea is to construct complex (software) systems by combining basic Web services. Public market places are created so that service providers can advertise their offerings and service consumers can evaluate and bind them. Today, public standards for such service registries are available like UDDI [11] and ebXML Registry Information Model [12], even though we cannot confirm a quick adoption of those standards in industry.

The idea of offering well encapsulated functionality to an anonymous market is not new. During the early 1990s the idea of component-oriented software

[1] ... according to Gartner [6] and Forrester [7] market research reports...
[2] http://www.aris.com/

engineering [13, see e. g.] became popular and here again it was the idea to reuse existing software components to construct more complex applications. However, there was no agreed standard for describing and binding the components and so their application was always limited to users with the same technology platform.

It can be said that Web service technology resolves this major interoperability issue by defining the WSDL [14] and SOAP [15] standards. The Web Service Description Language (WSDL) is used to define the interface of a Web service as well as where the Web service can be reached in terms of a unique resource identifier. SOAP defines a standard protocol to access the remote resource.

Web service discovery aims at identifying a Web service able to fulfil the requirements defined by the Web service request. We were able to identify three basic approaches to Web service discovery:

1. *Structural* discovery approaches use syntactical information available like the interface description and the definition of the data messages exchanged between the communication partners. This kind of matching is very technical, as it requires the service requester to specify structural requirements like a certain operation signature or data type. A typical example of such a discovery approach is given by Ramasamy [16]. Ramasamy compares operation names and operation parameters to the service request to discover Web services.
2. *Lexical* discovery approaches use natural language descriptions. For example, Web service operation names usually contain some terms describing their functionality. Also, WSDL and other standards allow embedding natural language descriptions. The lexical algorithms remove stop words from those descriptions, find synonyms using lexical databases like WordNet [17] and compute similarity coefficients. For example, Zhuang et al. [18] present an algorithm to compute the similarity of two web services. Their approach uses the information given in the WSDL files and does not require any additional annotations. They do manual pre-processing of the WSDL files to remove abbreviations, but it should be possible to use lexical databases like WordNet to automate this task in the future.
3. *Semantic* descriptions often based on ontologies are another major approach for Web service discovery. They use formal methods to describe web service capabilities and properties so that machine reasoning can be used to identify possible candidates for a service request. There are competing formalisms for describing this semantic information like the Web Service Modeling Ontology (WSMO) [19] or OWL-S [20]. A standard called WSDL-S [21] was proposed which provides some extensions for WSDL so that semantic descriptions in any formalism can be referenced from a WSDL file and so semantic annotation of existing Web services becomes possible. An early example for semantic matching is Paolucci et al. [22]. They use DAML-S to describe the capabilities of a web service as well as the service request. In a more recent example Kritikos and Plexousakis [23] describe quality of service (QoS) parameters using OWL-S allowing matching on non-functional Web service properties.

Most discovery algorithms combine different approaches to achieve a better result. For example, Wang and Stroulia [24] combine structural and lexical analysis. Kokash et al. [25, p. 526] have identified several strategies how to combine the results of different discovery approaches.

- The *mixed* strategy uses different discovery approaches and matching algorithms in parallel and unites the returned result sets into one final result set. Normally, duplicates are removed from the final result set.
- The *cascading* strategy applies different discovery approaches and matching algorithms in sequence. A matching is only performed on the result set returned by the previous algorithm. This helps to reduce the amount of processing needed and it can increase the overall result quality. This can be seen as a stepwise refinement.
- The *switching* strategy selects between different discovery strategies and matching algorithms based on predefined criteria. For example, if the results returned by an algorithm are not satisfactory, another algorithm is used. The cascading and switching strategy can be combined to create more complex strategies.

Our experience and literature investigation show that there is another important characteristic to correctly classify Web service discovery approaches and matching algorithms. One has to distinguish between discovery during design time and run-time. The former is normally initiated by a user designing a web service composition for example to create a custom software application or to automate a business process. This is also sometimes referred to as early binding. The latter is used during execution of a service composition. In this case, the composition only contains a requirements definition for a service call but it does not specify which specific Web service to use. At run-time, discovery is done to find all Web services matching the requirements specification and the best fitting Web service is used. This is sometimes referred to as late binding.

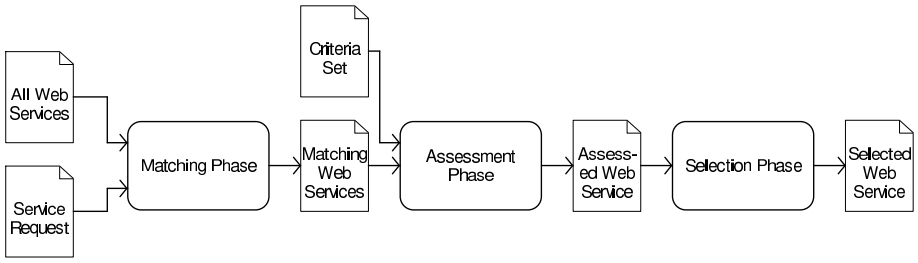

Fig. 1. Major Phases of Web Service Discovery

According to Kokash et al. [25, p. 522] Web service discovery consists of three major phases, also illustrated in figure 1:

1. During the *matching* phase matching algorithms belonging to the different discovery approaches are applied. The results are combined according to the chosen strategy. The result set might just consist of all Web services matching the request or they might be ranked according to their fitness.
2. During the *assessment* phase the matching results are further refined by a set of criteria. Where matching is normally done automatically, the assessment is often done manually, especially if Web service discovery is done during design time.
3. In the final *selection* phase a Web service is chosen and used in the composition as intended. This might also mean to adapt either the web service or the consuming process or application.

In this section, we provided an overview of current Web service discovery approaches and how to classify them. In the following section we show how our work relates to and extends existing approaches for Web service discovery.

3 Our Approach to Web Service Discovery

3.1 Overview

In theory it might be possible to discover a Web service during run-time to automate a certain step in a business process. Still, we have not seen something like that nor did our customers asked for it. They carefully design their processes and select Web services during design time. Therefore, our approach focuses on Web service discovery during design time.

We do not use any algorithms for lexical matching of Web services, even though the user can refine the matching results during the assessment phase using ordinary string search. In contrast, we make heavy use of structural matching to identify Web services able to handle the data objects modelled in the business process. We compare the business objects given in the business process to the message types used by the Web service in the message exchange. In that sense our matching algorithm is very similar to what Ramasamy [16] describes. We also do a lightweight variant of semantic matching. The users of our tool are able to create a taxonomy and use this taxonomy to classify the functionality of Web services. Even though we are not using ontologies or reasoning algorithms, it is still a way of capturing semantics.

We use the mixed strategy to unite the results of structural and semantic matching. We consider a Web service to fulfil the service request, if it is either discovered by structural or semantic matching or by both approaches. We remove any duplicates from the final result set before it is presented to the user for assessment.

We have structured the Web service discovery tool according to the three phases of service discovery. The user initiates Web service discovery by selecting the business process step to be automated. During the first phase, we analyse the context of the selected business process step and derive the service request. All Web services available in our tool are matched against the service request.

Afterwards, the result set is presented to the user for assessment. Finally, the user selects the Web service to use and the Web service is automatically added to the business process.

The following sub-sections describe our solution in detail. We do not describe the user roles involved using this solution to make the description not too complicated. An example is given in section 4. This example provides a walk-through also describing the involved user roles.

3.2 Web Service Representation in ARIS SOA Architect

The ARIS Platform is a set of integrated products to manage all aspects of an enterprise model. Besides defining and documenting a business strategy and business processes, one important aspect of an enterprise is the supporting IT infrastructure. Today, many companies are migrating their IT infrastructure to service-oriented architectures. A common piece in such an architecture are web services. Therefore, the specific SOA related ARIS product called ARIS SOA Architect allows importing Web services, if they are described using the Web Service Description Language (WSDL) version 1.1. Instead of just dumping the file in the underlying database, we extract the content and represent it using the Unified Modelling Language (UML). For example, WSDL porttypes are mapped to UML interfaces and the belonging operations to UML operations. The WSDL import functionality of ARIS SOA Architect also allows importing embedded or referenced XML schema definitions. Those definitions are mapped to UML as well.

Using UML models to visualise the information contained in a WSDL file is a proven approach for technical oriented users, but it is insufficient for business users. Therefore, we also create an object representing the web service from a business perspective. This object has no technical information like operations, interfaces or technical message types, because a business user should not have to deal with this kind of information in order to use a web service. Instead, the Web service is described from a business perspective by adding tags to it. The tag concept is described in detail in section 3.5. Other information includes the hardware the Web service is running on, the application system the Web service belongs to, and the person responsible for the Web service. The user can also evaluate, who or which process is currently using the Web service and the company locations the Web service is available for.

3.3 Information Architecture and Business Objects in ARIS SOA Architect

As described in the previous sub-section, technical data structures like XML schema definitions are mapped to UML models. However, for a business user it is not useful to deal with such a detailed data model. For example, there might be different technical message types or database schemas to represent customer data, but from a business perspective there is only one customer data

object. Such data objects are often called business objects or logical data objects. As in case of technical data modelling, business objects are also further refined into more concrete parts. For example, the business object customer can be decomposed into the name, address, payment history, and an interest profile. The models describing all relevant business objects for the whole enterprise are called information architecture. An internationally operating company should only have one information architecture, but there are usually several implementation of this architecture.

Message types defined by Web services are an implementation of business objects, too. A business user is using business objects to specify the data flow in a business process. If the technical data objects defined by the web services are mapped to the business objects used by the business user, it is possible to discover Web services for business process automation. The underlying algorithm is explained in detail in the following sub-section.

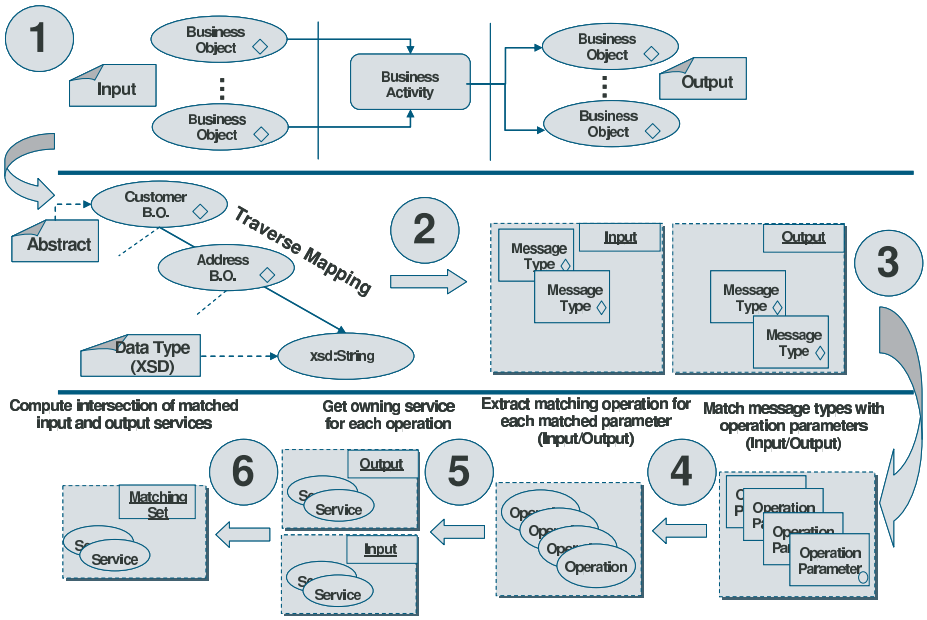

Fig. 2. Structural Web Service Matching Algorithm

3.4 Structural Matching Algorithm

Web services use message types to define their input and output. On the other hand, a business process uses business objects to describe the data flow. Both concepts are not equivalent, because they are on completely different abstraction levels as explained in the previous section. Instead of using message types in

business processes to model the data flow, one should create a mapping between business objects and message types. This mapping can be used to discover Web services by navigating from the business objects over the message types to the belonging Web services.

Our structural matching algorithm works as follows: We first extract all business objects modelled as input and output of the business process step as illustrated in step 1 in figure 2. Afterwards, we have two sets, one containing all business objects required as input and the other one containing all business objects required as output. For each of those business object sets we navigate through the mapping to identify all message types. Implementing this navigation is not trivial, because the modelling capabilities of the ARIS suite allows as many abstraction levels between business object and message type as the user wants including cyclic dependencies. Optimisation techniques must be used to implement a high performing solution. For example, the business object customer used in a business process is further decomposed into an address. This address can be represented using different message types. The algorithm has to identify all message types mapped to the business object. This is illustrated between steps 1 and 2 in figure 2. After this step, we have two sets of message types, one for message types required as input and one for message types required as output. In step 3 we check to which operation parameters those message types belong and if the parameters have the same direction as the message types (input or output). Extracting this information is possible, because we map the complete content of the WSDL file and related XSD files to UML models as described in section 3.2 and 3.3. If the message type is an operation parameter with the correct direction, we extract the belonging operation as shown in step 4. Afterwards, the operation's owning Web service is extracted in step 5. At the end we have two sets of Web services: one set supporting all input business objects and the other set containing all web services supporting the output business objects. In the final step 6 both result sets are intersected. The final result set of the structural matching algorithm contains only those Web services, which are part of both preliminary result sets and are therefore able to support all input as well as all output business objects.

As one can see, we do not check that a Web service has at least one single operation able to support all business objects in the parameter list. While automating business processes, this is normally not a problem, because during transformation of a business process into an executable process model (like BPEL), a process step can be split up into several technical steps. Also, adding another operation to a Web service able to handle all business objects in one request is often possible, if the Web service is owned by the company.

The biggest disadvantage of the structural matching algorithm is the effort required for mapping business objects to technical data structures. Many of our customers have already created a comprehensive information architecture consisting of the most important business objects, but matching those business objects to technical data structures requires effort. Each customer must decide, if

this investment can be justified. As an alternative, we provide a more lightweight matching algorithm, which is described in the following sub-section.

3.5 Semantic Matching Algorithm

Not all customers are interested in creating and managing an information architecture. Therefore, we provide a second more lightweight approach for web service discovery. First, the user creates a taxonomy for functional descriptions. Each taxonomy object has a very short textual description, comparable to a tag. In addition, a more detailed description including texts and diagrams can be added so that the meaning of the tag can be illustrated for human users.

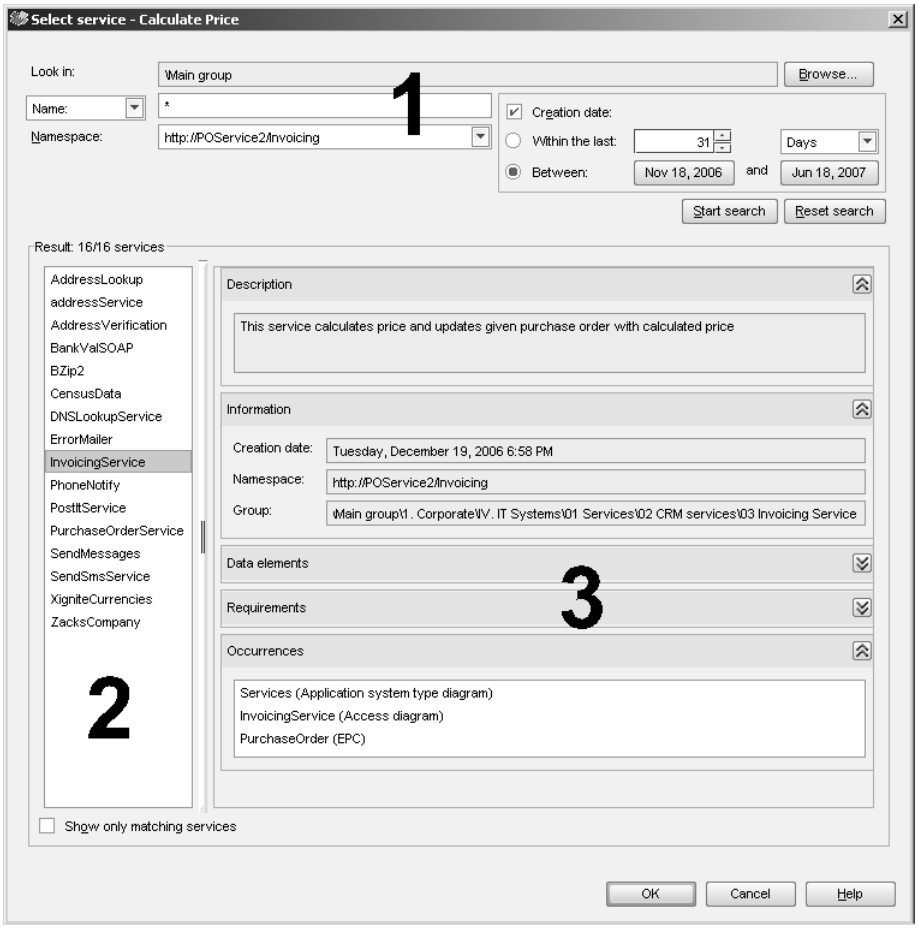

Fig. 3. Graphical User Interface of the Web Service Discovery Tool in ARIS SOA Architect

The taxonomy is used to annotate Web services by assigning the tags to them. Tags should be shared between Web services, if Web services have similar properties or functional capabilities. For example, the tag *web interface* should be added to all Web services providing a web based user interface.

The taxonomy must be carefully designed and managed. For example, not every user should be able to extend the taxonomy by creating new tags. Instead, reuse of existing tags must be enforced. Tags must also be described in a way that users have a clear understanding of their meaning.

The taxonomy is also used during business process modelling to express what kind of functionality is needed to automate a business process step. Tags are assigned to a business process step for this purpose.

The semantic matching algorithm first extracts all tags assigned to the business process step to be automated. The extracted tags describe the service request. Afterwards, we extract the tags assigned to each Web service and compare this list to the service request. This way we can discover those web services able to support the service request. This algorithm is much simpler compared to the structural matching algorithm. For example, the algorithm does not support decomposition of tags, so a Web service will not be discovered, if it has only a more general tag assigned as specified in the service request. We do not see this as a drawback, because this discovery algorithm is meant to be lightweight and easy to understand.

3.6 Web Service Assessment and Refinement

The final result set consists of all Web services discovered either by the structural matching algorithm or by the semantic matching algorithm. The matching results are shown to the user in a graphical user interface. A screenshot can be seen in figure 3. The screen design consists of three parts, which are marked in the screenshot with the numbers 1–3.

During the assessment phase the user further refines the result set. For example, the user can search the descriptions and names of the Web services with a string search or he can filter the list of Web services according to their namespace. He can also filter the list according to the date the web services were imported into ARIS. Those refinement settings are done in part 1 of the screen design as shown in figure 3.

The current result set can be seen in part 2 of the screen design. This part also allows switching between a list of all Web services and the list with matching Web services. This is useful in case the matching algorithms did not return a satisfying discovery result.

The user has to assess if a Web service fulfils the service request. This assessment cannot be done based on the name of a Web service. Therefore, additional information is shown in part 3 of the screen design for the currently selected Web service. For example, all business objects supported by the Web service are shown as well as the textual description. It is also possible to see in which other contexts the Web service is used.

Finally, the user selects a Web service and confirms this selection. The dialog closes and the Web service is automatically attached to the business process step. Now that a Web service is assigned to the business process step, this process step is automated from a design point of view. If all steps in a business process are supported by Web services, the model can be transformed to BPEL as we have shown in [26].

4 Example

This section provides an example to better illustrate our discovery approach. The example mentions two different roles – a business analyst and an IT architect. The business analyst has no IT background, but instead experience in business process modelling. The IT architect has SOA know-how and is able to use typical SOA middleware products and standards. In reality, there are usually more roles involved, but we tried to make the example not too complicated.

A fictitious company defines an internal business process for organising business trips. If such a business trip has to be done by car, the employee has to use a company car, if available. Only if no company car is available, the employee is allowed to rent a car from a defined car rental company.

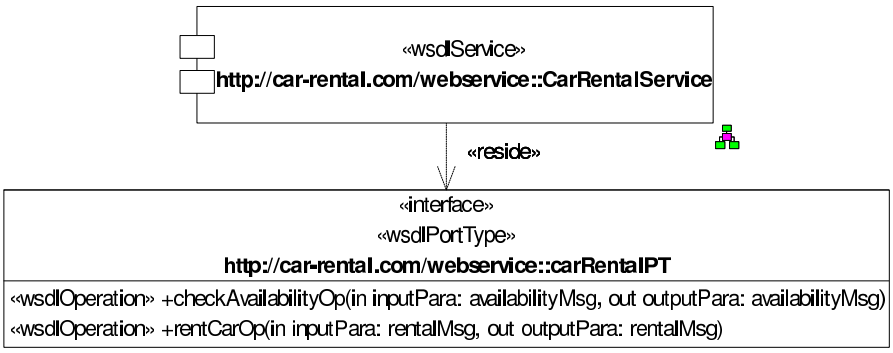

Fig. 4. Web Service as UML Component Diagram

The car rental company provides a Web service for this purpose. In order to be able to use this Web service in business process modelling, the Web service must be made available in ARIS. An IT architect imports the Web service. The content of the WSDL file is visualised as an UML component diagram as shown in figure 4.

Besides using this technical information, the IT architect annotates the web service with tags from the company wide taxonomy to describe the service semantically. The company wide taxonomy is defined prior and will not be changed by the IT architect. In addition, the IT architect might add who is responsible

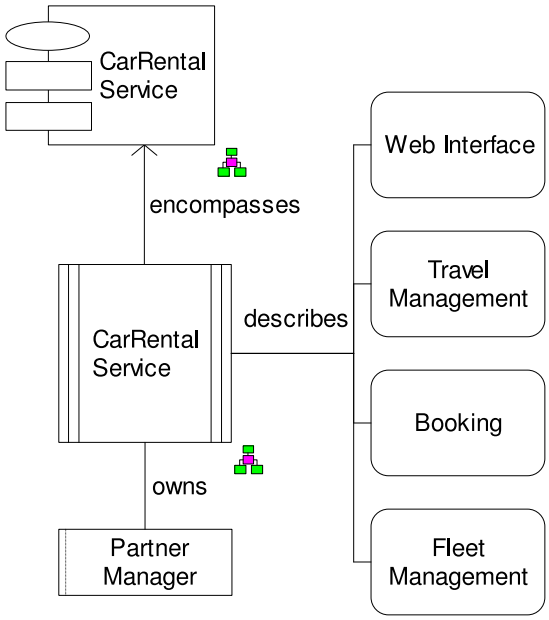

Fig. 5. Annotated Web Service

for this Web service. Figure 5 shows the annotated web service. There are four tags describing the Web service and an owner is defined, as well.

If the company has an information architecture, the IT architect maps the message types used by the Web service to the belonging business objects. This can be a complex task and he might have to consult business analysts to identify the correct business objects. The mapping is done in different diagrams, which are not shown.

A business analyst models the business process described at the beginning of this section. Figure 6 shows a small part of the business process. The business analyst creates a business function and connects it with the input and output business objects. In addition, the business expert specifies requirements by relating the business function to tags from the company wide taxonomy. In reality, companies have either an information architecture or a company wide taxonomy, but not both.

The business analyst wants to automate the business function using a web service. He selects the business function and starts the integrated discovery tool. The discovery tool evaluates the content of the business process by extracting all input and output business objects and extracting the tags connected to the business function. This information is the input for the semantic and structural matching algorithms as described in section 3. The results are shown to the business analyst in the graphical user interface discussed in section 3.6 and shown in figure 3. The business analyst selects a Web service after assessing the different

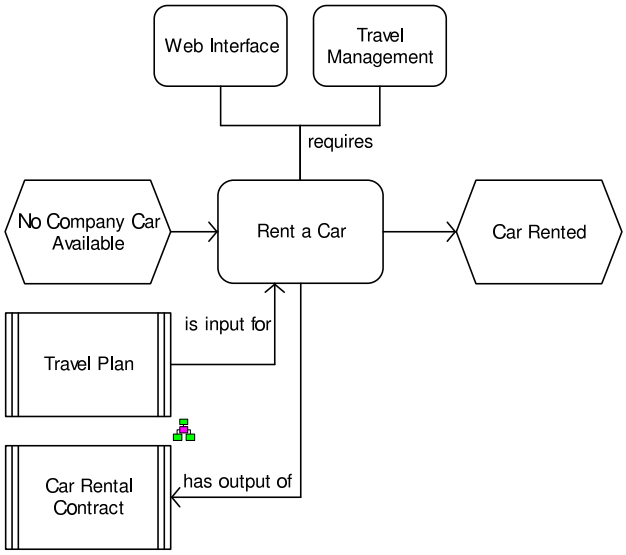

Fig. 6. Business Process without Web Service

choices. The Web service is automatically added to the business process as shown in figure 7. The symbol of the business function is changed as well to visualise that this process step is now automated by a Web service.

The resulting business process cannot be executed directly, because different technical information is missing. An IT analyst uses our EPC to BPEL transformation [26] to generate a corresponding BPEL model. This BPEL model must be further refined, for example selecting correct Web service operations or defining technical exception handling.

The example given in this section shows that a business analyst is able to automate business processes by discovering matching Web services. In order to select a Web service the business analyst does not need IT knowledge. On the other hand, an IT expert implementing a business process gets a detailed specification for his work.

5 Conclusions

In this article we presented a Web service discovery tool for business process automation. The tool is completely integrated in the world market leading tool for business process management ARIS and the belonging ARIS method. In contrast to other publications, our approach clearly separates between the different abstraction levels by not mixing technical data with technology independent business processes. The Web service discovery tool is structured around the three discovery phases: matching, assessment, and selection. The intended audience are non technical users like business analysts. The tool does not confront them with unnecessary technical details. The matching algorithms used discover

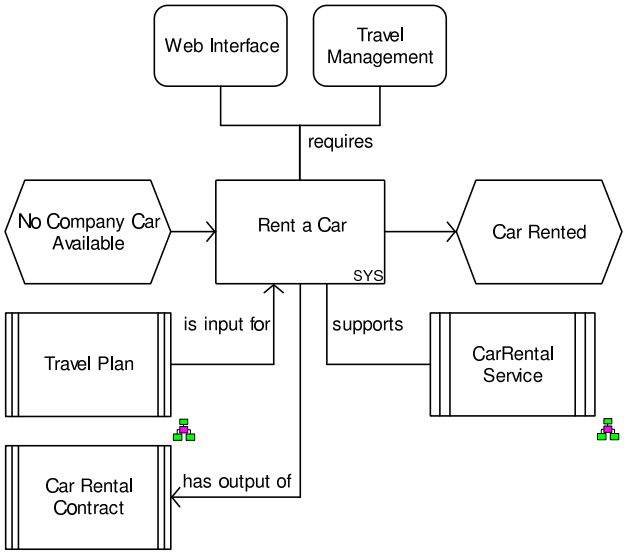

Fig. 7. Business Process with Web Service

a set of matching services. The result set can be assessed and refined by the user or the user can switch to a list with all available Web services if needed. For each Web service we provide additional information so that the user can make an informed decision while selecting a Web service.

In order to discover Web services, we use a structural matching algorithm and a semantic matching algorithm. In both cases, the service request is extracted from the business process model. No user input is required for defining the service request, which simplifies the overall tool usage. The structural matching algorithm identifies all Web services able to support the business objects modelled at the business process step to be automated. This is possible based on a mapping of Web service message types to business objects. The semantic matching algorithm requires that Web services and the business process step are tagged using a taxonomy. A Web service is considered to match semantically, if it has at least all tags also attached to the business process step.

So far, the tool was already deployed by several customers. Many technical oriented users were fascinated by the structural matching algorithm, but it seems that business oriented users like the semantic matching algorithm more. However, at the current point we have not received enough feedback from the field to make any final judgement. We already initiated a survey among the first users, but the results are not available yet.

Even though our customers perceive our current solution for Web service discovery as good, there is still room for improvements. For example, the structural matching algorithm does not scale very well, because we are not able to use any optimisation techniques like pre-indexing or caching. This is not a problem inherent in our algorithms, but related to the technological framework we

have to use. It is our challenge for the next months to find optimisation tricks to overcome those problems. Another point of improvement is to allow a more sophisticated semantic matching. For example, it should be possible to create a tag hierarchy so that Web services are discovered, even if a more general tag was assigned to them. Also, it should be possible to express that two tags cannot be used together, because they contradict each other. However, we do not intend to provide complete ontology modelling and matching possibilities in the near future, because the tool must be easy and intuitive to use even without any special training. We also plan to extend our web service discovery concept to a more general service discovery concept. Basically, the service concept can be used to describe any kind of business function. We will broaden the definition of the service concept so that it better aligns with the service concept as defined in OASIS' SOA Reference Model [3]. For example, a service must not be implemented using software at all. We will extend our discovery approach to cover such business services as well.

Our most important contribution is to bring Web service discovery to a non-technical audience. Web service discovery can now be done by business analysts. Even though there is still room for improvements, we are confident that our tool helps bridging the gap between business and IT.

6 Acknowledgement

A first prototype of the discovery tool was partly funded by the German federal ministry of education and research within the public research project OrViA (http://www.orvia.de/). The literature review as well as preparing this paper was supported by the EU Commission within the integrated research project SUPER (http://www.ip-super.org/). We like to thank the German federal ministry of education and research and the EU Commission for this opportunity!

References

1. Scheer, A.W., Thomas, O., Adam, O.: Process modelling using event-driven process chains. In Dumas, M., van der Aalst, W.M.P., ter Hofstede, A.H.M., eds.: Process-Aware Information Systems. Wiley, Hoboken, New Jersey, USA (2005) 119–146
2. Smith, H., Fingar, P.: Business Process Management: The Third Wave. 1st edn. Meghan-Kiffer Press, Tampa, FL, USA (2003)
3. MacKenzie, C.M., Laskey, K., McCabe, F., Brown, P.F., Metz, R.: Reference model for service oriented architecture 1.0. Technical report, OASIS (July 2006) http://www.oasis-open.org/committees/download.php/19361/soa-rm-cs.pdf.
4. McGovern, J., Sims, O., Jain, A., Little, M.: Enterprise Service Oriented Architectures. Springer, Dordrecht, The Netherlands (2006)
5. Alves, A., Arkin, A., Askary, S., Barreto, C., Bloch, B., Curbera, F., Ford, M., Goland, Y., Guizar, A., Kartha, N., Liu, C.K., Khalaf, R., König, D., Marin, M., Mehta, V., Thatte, S., van der Rijn, D., Yendluri, P., Yiu, A.: Web services business process execution language (bpel) version 2.0. Technical report, OASIS (April 2007)

6. Blechar, M.: Magic quadrant for business process analysis market, 2h07. Technical report, Gartner (June 2007)
7. Peyret, H.: The forrester wave: Enterprise architecture tools, q2. Technical report, Forrester (April 2007)
8. Scheer, A.W.: ARIS - Business Process Frameworks. 3rd edn. Springer, Berlin, Germany (1999)
9. Foster, I., Kesselman, C., Tuecke, S.: The anatomy of the grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications **15**(3) (2001) 200–223
10. Weiss, G.: Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. MIT Press (1999)
11. Clement, L., Hately, A., von Riegen, C., Rogers, T.: Uddi version 3.0.2. Technical report, OASIS (October 2004) http://www.oasis-open.org/committees/uddi-spec/.
12. Fuger, S., Najmi, F., Stojanovic, N.: ebxml registry information model version 3.0. Technical report, OASIS (May 2005) http://docs.oasis-open.org/regrep-rim/v3.0/.
13. Szyperski, C.: Component Software: Beyond Object-Oriented Programming. Addison-Wesley (1997)
14. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web service description language (wsdl) 1.1. Technical report, W3 Consortium (March 2001) http://www.w3.org/TR/wsdl.
15. Mitra, N., Lafon, Y.: Soap version 1.2. Technical report, W3 Consortium (April 2007) http://www.w3.org/TR/soap.
16. Ramasamy, V.: Syntactical & semantical web services discovery and composition. In: The 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE'06). (2006)
17. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press (1998)
18. Zhuang, Z., Mitra, P., Jaiswal, A.: Corpus-based web services matchmaking. In: Workshop on Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing, held in conjunction with The Twentieth National Conference on Artificial Intelligence (AAAI '05), Pittsburgh, PA, USA (July 2005)
19. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer (2006)
20. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: Owl-s: Semantic markup for web services. Technical report (2004) http://www.daml.org/services/owl-s/.
21. Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.T., Sheth, A., Verma, K.: Web service semantics (wsdl-s) version 1.0. Technical report, W3 Consortium (November 2005) http://www.w3.org/Submission/WSDL-S/.
22. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic matching of web services capabilities. In: The Semantic Web - ISWC 2002: First International Semantic Web Conference. LNCS 2342/2002, Sardinia, Italy, Springer, Germany (June 2002)
23. Kritikos, K., Plexousakis, D.: Semantic qos metric matching. In: 4th European Conference on Web Services (ECOWS). (December 2006) 265–274
24. Wang, Y., Stroulia, E.: Flexible interface matching for web-service discovery. In: Web Information Systems Engineering (WISE). Proceedings of the Fourth International Conference on. (2003) 147–156

25. Kokash, N., van den Heuvel, W.J., D'Andrea, V.: Leveraging web services discovery with customizable hybrid matching. In Dan, A., Lamersdorf, W., eds.: Service-Oriented Computing (ICSOC 2006). Proceedings of the Fourth International Conference on. LNCS 4294, Berlin, Germany, Springer (2006) 522–528
26. Stein, S., Ivanov, K.: EPK nach BPEL Transformation als Voraussetzung für praktische Umsetzung einer SOA. In Bleek, W.G., Raasch, J., Züllighoven, H., eds.: Software Engineering 2007. Volume 105 of Lecture Notes in Informatics (LNI)., Hamburg, Germany, Gesellschaft für Informatik (GI) (March 2007) 75–80