WS-BioZard: A Wizard for Composing Bioinformatics Web Services

Zhiming Wang, John A. Miller, Jessica C. Kissinger,
Rui Wang, Douglas Brewer, Cristina Aurrecoechea
University of Georgia, Athens, GA, 30602
{zhiming@cs., jam@cs., jkissing@,very@, twisted@, aurreco@}uga.edu

STUDENT TRAVEL AWARD: YES

Abstract

As the amount of biological data continues to increase, how biologists share data and analysis tools efficiently is becoming an important issue. Web service technology is a promising solution because of the advanced features it provides such as language independence, platform independence and ease of programmatic access. Recently, we have seen an increasing amount of interest in the composition of biological Web services in order to process a biological task. Although there are some research efforts towards composing biological Web services, such as BioMoby and Taverna, these existing tools are still too difficult to be widely used by the average biologist. Therefore, lowering the learning curve for Web service composition to make it easier for biologists is becoming a critical need. In this paper, we describe WS-BioZard, a comprehensive framework for addressing this need by using wizards and semantics to provide a suite of tools that support practical semi-automatic composition of Web services in the biological domain.

1. Introduction

In recent years, Web services and service composition are becoming increasingly used in bioinformatics. Web services are being used to provide access to data sources as well as software to analyze the data. Web services can provide several benefits such as language independence, platform independence and ease of programmatic access. Presently, there are numerous bioinformatics Web services provided by several organizations and university institutes or labs including the National Center for Biotechnology Information (NCBI), the European Bioinformatics Institute (EMBL-EBI), the DNA Data Bank of Japan (DDBJ) and the Protein Data Bank of Japan (PDBJ).

While individual bioinformatics Web services are useful, situations often arise where more than one service is required to perform a complete biological analysis. For example, whenever a new nucleotide sequence is annotated, biologists try to understand its functionality through searching for homologous sequences. Homologous sequences are often, but not necessarily similar at the primary amino acid sequence level but related by evolutionary descent. This task can be achieved through composing biological Web services: First a BLASTX[1] Web service could be used to identify amino acid sequences from a well-known protein sequence database. Then another Web service is used to retrieve the 'good hits' amino acid sequences. A third protein domain Web service is used to identify domains.

Without the help of Web service composition, such tasks would be time consuming and error-prone. Compared with a traditional programming based approach, the Web service composition approach has the following benefits:

- Flexibility: easy to change a Web service during the composition through drag and drop in a design tool.
- Easy to develop: as long as the design tool is easy to use.
- Easy to reuse: since the result is a standard Web service.

However, for biologists creating such biological Web service compositions using current designers is full of challenges.

When creating complex processes that involve multiple Web services, users may choose between (1) manually composing the services themselves, (2) using fully automatic composition software, or (3) using a hybrid approach which is called semi-automatic composition. When composing the services themselves, users must decide what service follows the previous one at each point by picking from potential services. An example of this can be seen in the Active BPEL Designer (*www.active-endpoints.com/*) or Taverna (*taverna.sourceforge.net*/). The automatic approach removes much of the burden from users in that all they need do is to describe the input, output and the functionality desired and then a program will produce a composition of services (such as EFlow [1]). The semi-

[1] The Basic Local Alignment Search Tool (BLAST) can be used to infer gene family information and sequences' relationship, such as function or evolution, through comparing the querying sequence to sequence databases and calculating the statistical significance of matches. BLASTX is a special BLAST program to search protein databases using a translated nucleotide query.

automatic approach sits between these two approaches. In it, users describe the desired input, output and functionality and are given suggestions and assistance to compose the services themselves.

Manual design for composition is well studied and has many useful and commercial quality tools, such as ActiveBPEL and Oracle BPEL Designer (*www.oracle.com/technology/products/ias/bpel/index.html*). However, this is not the best choice for our target user, in that, it requires familiarity with control constructs and mapping data between different services. At the same time, automatic composition is being researched and headway has been made [1, 2, 3]. The main problem with automatic composition is with the composition itself. While it relieves the user of the burden of control flow and data mapping, it cannot always produce a correct composition [4, 5]. The last approach of semi-automatic composition can produce good results while minimizing the decision making associated with service composition, such as the selection of control flow structures and mapping the outputs to the inputs.

In this paper, we describe WS-BioZard, a new and comprehensive framework for semi-automatic Web service discovery, data mediation and service composition. It utilizes Semantic Annotations for WSDL (SAWSDL:*www.w3.org/2002/ws/sawsdl/*) and domain ontologies to add semantics to Web services. This allows us to design and implement (1) a service discovery tool with a sophisticated User Interface (UI), and (2) a multi-faceted UI Wizard that facilitates the data mediation and service composition tasks for users. The result of the composition is a BPEL executable that could be deployed as a Web service in a BPEL-compliant engine.

The remainder of this paper is structured as follows: In section 2, a brief review of existing XML-based composition languages is given. Section 3 argues for semi-automatic compositions and discusses ways to simplify the composition process from the perspective of control flow and data flow by using tools to assist the human designer. Our proposed framework and its prototype implementation are presented in sections 4, 5, 6, and 7. Section 8 discusses related work and we conclude in section 9 with a summary of the research issues addressed.

2. Review of composition languages

Currently, there are five popular workflow/process definition languages: XML Process Definition Language (XPDL), Web Services Business Process Execution Language (WS-BPEL), Web Service Choreography Definition Language (WS-CDL), Business Process Modeling Language (BPML) and Unified Modeling Language (UML). The five languages fall into two categories: those that are oriented toward Web services, such as WS-BPEL and WS-CDL, and those that are more generally for business processes/workflow, including XPDL, BPML and UML. Although Web service composition has benefited from the experiences in the workflow domain, the standards in the two domains are distinct. In comparison to business workflow, Web service composition is based on interactions in the context of a Service-Oriented Architecture (SOA). Web service composition specifies an XML grammar that can be seen as a programming language to combine control flow with invocation of the services.

There are two main categories of Web service composition: orchestration and choreography. The main difference between them is how to deal with the individual participant. A choreography model mainly focuses on the view of individual participants like a peer to peer model, while an orchestration model gives a global view of the system as seen by a particular process consisting of multiple Web service participants [6, 7].

Since their inception, there have been many languages for composing or coordinating the execution Web services. Fortunately, two of them are the main open standards of today: WS-CDL and WS-BPEL. The Web Service Choreography Definition Language (WS-CDL) is defined by the W3C as an XML-based business process modeling language that describes collaboration protocols between cooperating Web service participants. The Web Service Business Process Execution Language (WS-BPEL) is also an XML-based business process modeling language co-written by IBM, Microsoft, BEA, SUN and Oracle and standardized by OASIS. Compared with WS-CDL, WS-BPEL currently has more support from industry and wider acceptance from users. Also, there are more WS-BPEL engines and editors available, including ActiveBPEL, Eclipse BPEL, NetBeans BPEL and Oracle BPEL. This makes WS-BPEL a candidate for our framework.

There are some other related workflow languages such as YAWL [8] and XScufl [9]. XScufl is an ongoing collaborative project between EBI, IT Innovation and the Human Genome Mapping Project (HGMP) of the Medical Resource Center (MRC) and is used by Taverna. In comparison with the above main Web service composition languages, these two have a smaller tool support base.

3. Why semi-automatic composition

WS-BPEL is the industry standard orchestration language for Web services. The use of WS-BPEL for use in service composition is a good choice in that it implements thirteen of the original twenty workflow patterns [10], including the five most basic patterns: sequence, parallel split, synchronization, exclusive choice,

and simple merge. One draw back to WS-BPEL is the complexity of the language itself. It provides many rich control flow constructs commonly associated with programming languages to implement the the workflow patterns. The problem, therefore, with manual composition lies not in the ability of WS-BPEL, but the ability of the user. In our case, the user is expected to not be a programmer and will have a hard time manually composing services. Another problem that exists for our user in WS-BPEL is the data mediation facilities. In biology, it is hard to find a data type that has single representation. According to [11] there are 20 different DNA representations. This presents a problem for biological users, as they will have to perform many manual transformations, in XSLT (eXtensible Stylesheet Language Transformation) or XQuery/XPath, in WS-BPEL using the assign/copy statements.

With manual composition prohibitive to the user, we could have chosen to provide a fully automatic composition. Fully automatic planners exist and their use in Web service composition has been studied in [12, 13]. The commonality between all the planning methods is that they require preconditions and effects to be specified for correct operation, meaning they need Semantic Web Services. In the context of the planners, a precondition describes the state of the world before the invocation of a service, and an effect describes the state of the world after the invocation of a service. The good thing about automatic composition is that since the user does not have to deal with control or data flow, all the complexity of WS-BPEL can be used. The control flow is determined by the planner by matching the preconditions and effects to find services that can provide necessary transformations. The data mediation is provided by the planner trying to find a transformation, which may be provided or calculated with or without annotations.

However, there exists some downfalls to automatic composition. Research has shown they are good at composing sequences of services, but when we introduce the need for loops and for using the last two basic workflow patterns (Arbitrary Cycles and Implicit termination), they have not been shown to work as well [14]. The problem is that automated planning relies on the availability of complete formal representations of the domain knowledge and the individual cases that need to be resolved, i.e., their initial state and goal state need to be formally encoded. The task of formally specifying a domain in sufficient fidelity so that it can be used for automated planning presents a huge challenge. Especially for the Biology domain we can assume that complete ontologies will not be available in the near future. Incomplete domain knowledge will often result in a situation that an automated planner fails to produce a plan. In contrast, human planners can draw upon their experience with a specific domain when they compose services. This experience will often compensate for missing or erroneous ontologies. One of the more important problems for automatic composition is that the Semantic Web Service standard, SAWSDL, the standard chosen in our framework, does not currently contain precondition and effects even though its predecessor WSDL-S does. Still, specifying correct preconditions and effects can be difficult.

Having seen the shortfalls in the manual and fully automatic approaches, we have chosen a semi-automatic approach. It overcomes the weaknesses of automatic composition because, the user can handle the problem patterns and it eases the burden on the user, which is the main goal of semi-automatic approaches. We accomplish this through providing automation when possible, helping the user along with the wizards, and by reducing the number of available patterns to those deemed essential. In particular, our approach provides nine of the original twenty patterns, including the five most basic patterns. The patterns we lose, when compared to WS-BPEL, are deferred choice, interleaved parallel routing, cancel task, and cancel case. The patterns lost in our approach are of less use to our users, while we still keep much of the power of WS-BPEL.

4. Overall architecture

Figure 1 shows the overall architecture of WS-BioZard, a new comprehensive framework aiming to facilitate biologists in the task of composing Web services, including service annotation and registration, service discovery, data mediation and service composition. Web service semantic annotation and registration (Semantic Process block) is a fundamental task that is required prior to the user performing Web service discovery and composition (User Interface and Model blocks). Service providers are responsible for this task and our framework provides a design and implementation to accomplish this. Multiple technologies are used to achieve the above goal and keep the framework flexible and extensible. The following sections will further explain the design and implementation of the three software architecture modules shown in Figure 1 from top to bottom.

5. User interface module

Our user interface is very intuitive and much simpler than a standard BPEL editor. There are two parts, the Web service discovery tool and the Web service composition editor. The discovery tool is used to discover and browse the available Web services based on the criteria specified by the end user (section 5.1). The composition editor is discussed in section 5.2. This framework has been implemented on the Eclipse platform, which is discussed in section 5.3.

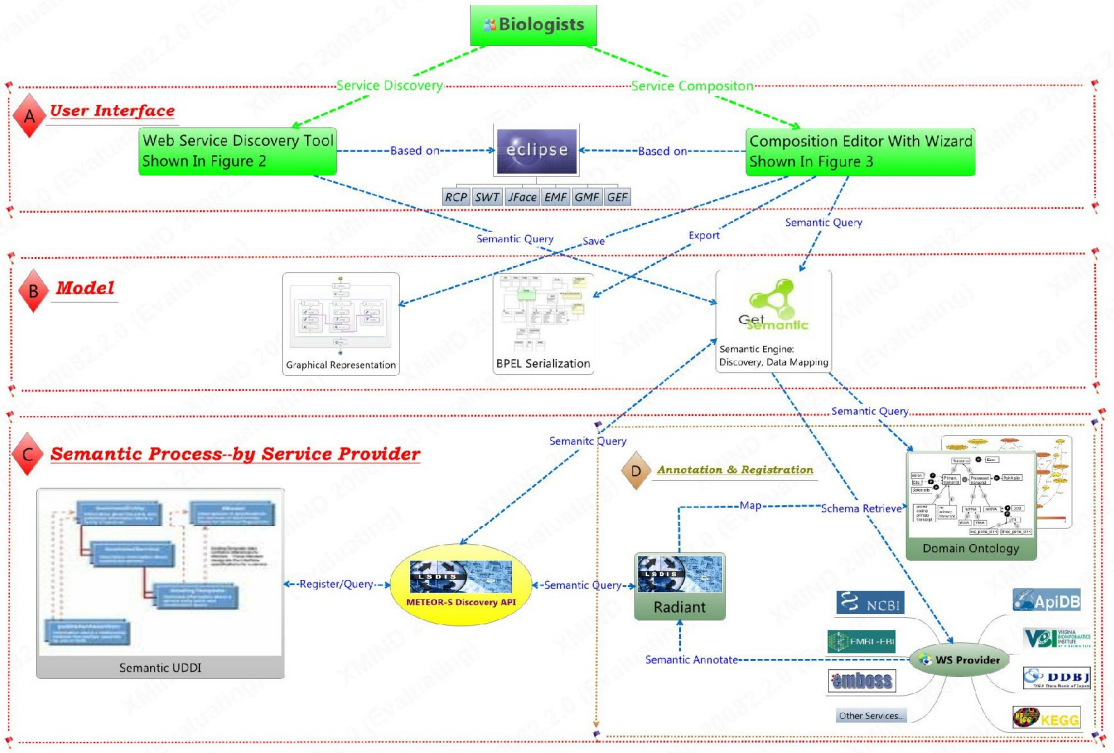

Figure 1. Architecture

Of importance to the user community, this framework can run as a stand-alone Java application without the Eclipse workbench. This will be ideal for those users who do not have the technological background to install the different packages used in our framework. With just one click the application will be ready to use.

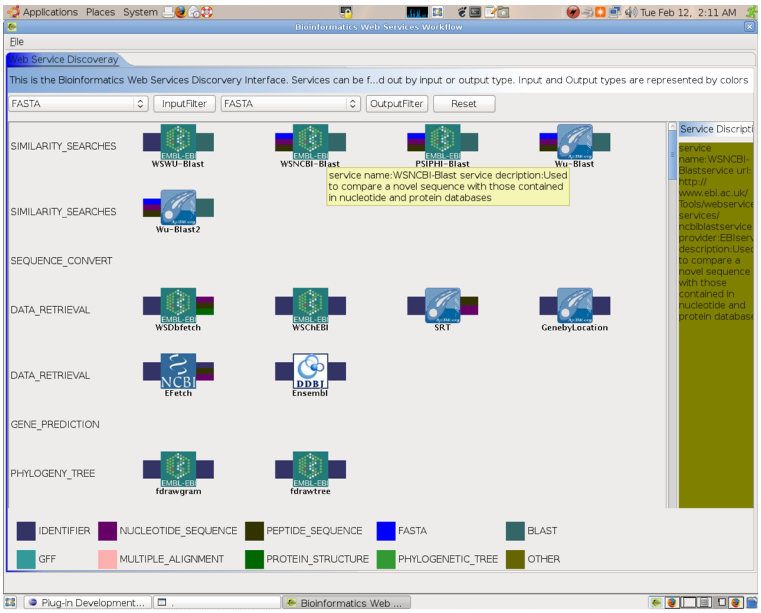

Figure 2: Interface of Web Service Discovery

5.1. Web service discovery tool

Through the intuitive interface shown in Figure 2, biologists can query and view all the available Web services by many different criteria, such as sequence type, function type and input/output types. The result is dynamically generated based on queries to a semantic UDDI registry, which contains all the registered WSDLs with SAWSDL annotations. When a service is selected, all the incompatible services are automatically filtered out. Web services in each cell are dynamically populated according to the semantic UDDI registry. Whenever the mouse hovers on a service, the description of this service will popup. The color coded tabs attached on the middle icon represent input and output types, and they are dynamically generated based on the domain ontology. Each color represents a type, and the same color between two services' output and input means that the two services could be composed together (i.e., there is at least a partial match).

5.2. Web service composition editor

In this subsection, we discuss how the Composition Editor utilizes a multi-faceted wizard to assist biologist in composing Web services. We then discuss how our editor simplifies the service composition process.

5.2.1. A multi-faceted wizard. A wizard is a user interface element assisting end users to finish a specific task. A well-defined wizard should hide much of the complexity involved in the task from the user. At the same time, it should provide a supplementary rather than substitutive way to accomplish the task to avoid restricting functionality for advanced users [15]. The end user is led through a sequence of dialogs to collect all the necessary information and user input for the task. A general purpose wizard is too complex and difficult to implement. However, a wizard for a specific domain can be very useful and can lower a user's burden, especially when performing a complex task [16]. Restricting the wizard to a domain such as bioinformatics allows the wizard to utilize one or a few related domain ontologies (i.e., complex and risky alignment of diverse ontologies can be avoided).

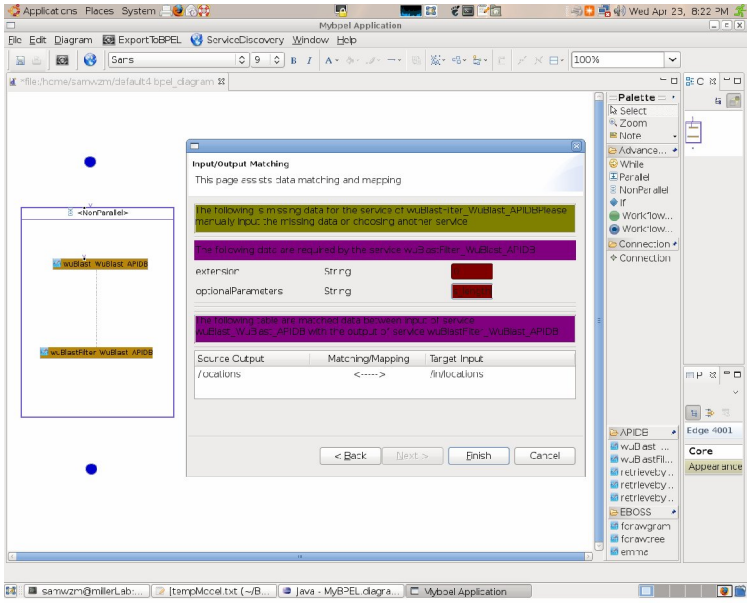

Figure3: WS Composition Editor with Wizard

In our framework, the role of the wizard is to be a biologist's assistant. For example, a wizard can automatically pop up when a biologist tries to connect any two services, providing information about service input, output and functionality. Furthermore, a wizard can be used to check the services mapping/matching as well as assisting data mediation. Because of the above benefits provided by our wizard, we believe that our architecture significantly reduces the complexity of bioinformatics Web services composition. Figure 3 shows the wizard, assisting in data mapping/matching in our Web service composition editor.

5.2.2. Simplifying service composition. Since BPEL is a very general orchestration language, most BPEL editors will reflect this through an interface with all the 13 BPEL constructs/workflow patterns. This makes the editor too complex to be willingly used by the average biologist. In our composition editor, multiple approaches are used to lower the learning curve for biologists.

First, we make the palette more intuitive through separating the Web service selection from the flow control activities, as well as moving BPEL activities such as "assign, copy and invoke" from the palette to the wizard. Multiple commonly used workflow templates are provided to assist biologists.

Second, whenever there is a connection between two services, a wizard will popup and collect information from a user to create BPEL activities, such as "assign", "copy" and "invoke". Whenever the connection is reoriented or deleted, the corresponding actions will be taken to make the correct change. All these activities are transparent to the end user.

Third, the Web service information in the palette is dynamically populated by querying the semantic UDDI registry. Whenever our application starts running, it will check all the available services in the semantic UDDI registry. Each possible Web service selection represents a method of the Web service for three reasons:

- Eventually a method will be chosen to invoke the service.
- Many Web services include multiple methods. The end user is often more concerned about the method, instead of the overall Web service itself.
- Automatic data matching/mapping can only be achieved after the methods are finalized.

5.3. Eclipse platform

In our architecture, the low-level user interface technologies are based on Eclipse's Standard Widget Toolkit (SWT), Jface and Graphical Editing Framework (GEF). On the high level, we use Eclipse Modeling Framework (EMF) and Eclipse Graphical Editing Framework (GEF).

The Eclipse EMF project is a modeling framework that provides a code generation facility. The model can be defined by XML, UML, Java classes or Ecore, providing a Java code generation function. Eclipse GEF uses many design patterns, such as Model-View-Controller (MVC), Command, and Chain of Responsibility, to provide quite powerful and flexible functionality. However, it is not easy to use. The Eclipse GMF project aims to improve the usability of UI applications through combining Eclipse EMF with GEF. In this visual development approach, much of the source code will be automatically generated based on the model and can also be customized by the developer.

Using these new high-level Eclipse frameworks, a developer can create sophisticated UI's with a fraction of the effort that would have been required in the past. This also allows the developer to focus on the application logic and on making software easy to maintain and extend.

6. Model module

There are three components in the Model module. The composition designer needs support from all three components, while the discovery tool only needs support from the semantic engine component. A graphical representation is used to save the composition designer results to the disk, which can be done automatically by Eclipse GMF. BPEL serialization uses the Eclipse BPEL API to generate BPEL. The model part of Eclipse BPEL project (*www.eclipse.org/bpel/*) is used as our model implementation. It is an open source project and implements all the features of WS-BPEL.

An important contribution in this project is the design and implementation of the semantic engine. It handles the semantic requests from the UI module, providing information to assist service discovery and data mapping. It connects with the Semantic Web service registry and other components of the Semantic Process block to accomplish his function. Currently, commercial Web service technology lacks full support for service discovery, service invocation/interoperation, service negotiation and service composition. Adding semantics to Web services is the most promising solution for this problem. The objective of Semantic Web Service research is to provide a knowledge representation of individual services in order to allow automatic machine processing. Adding semantics to Web services can provide the benefits of better discovery and better data mediation, which will be discussed in section 6.1 and 6.2.

The Semantic Annotations for WSDL and XML Schema (SAWSDL) W3C Recommendation, chosen in our framework, is the first step by W3C to add semantic support to Web services technology by adding semantic annotations to WSDL components..

6.1. How SAWSDL facilitates service discovery

Since there is a large number of biological Web services available online, locating the relevant ones manually would be time consuming. Providing a tool to find the most relevant service based on input, output and functionality automatically is very useful. The METEOR-S Discovery API (see section 7.2) is used to discover Semantic Web Services based upon SAWSDL annotations. The algorithm can find the most relevant Web service through calculating semantic Web service property similarity, based on concepts, semantic relations, and their common and distinguishing features. A Web service's input, output, and functionality are considered during the similarity calculation. Furthermore, this algorithm can calculate similarity between semantic Web services annotated by different ontologies. We have designed and implemented an algorithm to find the most relevant services for the end user.

6.2. How SAWSDL facilitates schema/data mapping

Once the end user is ready to compose Web services from a known set, we have developed an algorithm to find out if two Web services match or not. If the match is close enough, a means for providing missing information can be given manually by the end user at design-time (see the wizard interface in Figure 3).

A bottom up algorithm is implemented to do data mediation between WS1 with WS2. Suppose WS1's output annotates to class C_o in the ontology and WS2's input annotates to class C_i (annotation using properties is also supported). There are three possible data matching results:

1) Input is the same class, an equivalent or a super class of the output
2) The input and output have overlapping properties.
3) The input and output have no common properties.

If discovery was done properly, case 3 should not occur. Case 1 can be handled fully automatically, while case 2 is handled by our data mediation wizard.

The following is the pseudo code for this data mediation algorithm:

IF $C_i = C_o$ or $C_i \equiv C_o$ or $C_i \in$ superclass(C_o) //case 1
  $O = XPath_{lift\text{-}o}(X_o)$
  $X_i = XPath_{lower\text{-}i}(O)$
ELSEIF $C_i \sqcap C_o \neq \square$ //case 2
  X_i = Data Mediation Wizard(X_o)
ELSE //case 3
  Reject WS2

For case 1, the tool will automatically generate the BPEL <assign> and <copy> elements based on the XPath annotation. For case 2, the wizard will popup and let the end user type in expressions for missing values. Then the tool will generate the <assign> and <copy> elements.

7. Semantic process module

The semantic engine described in section 6 relies on the semantics provided by the Semantic Process module. There are two main components in this module. In the first component, the semantic information is annotated and the service is registered. In the second, the semantic information stored in a UDDI registry can be queried through an API (see Figure 1).

7.1. Annotation and registration component

The first and most fundamental procedure in our framework is to annotate the Web services. This procedure provides the key support for service discovery and data mapping. Multiple tools can achieve this task including Radiant and WSMO Studio. Radiant is an open source Eclipse plug-in project developed in LSDIS lab at the University of Georgia. Since Radiant can not only annotate, but also publish the annotated service to a UDDI registry with the assistance of the discovery API, it is used in our framework.

Ontology plays a fundamental part for semantics by providing the formal definition and relationship between concepts. Because of the complexity of biology, diversity of biological terms and lack of an universal naming convention, ontology is particularly important in the biological domain. In our framework, we chose the Sequence Ontology (SO) (*www.sequenceontology.org/*) as our domain ontology. A joint effort by genome annotation centers, SO includes a set of terms and relationships used to describe the features and attributes of biological sequences. We have extended it to fit our needs. However, our framework can support multiple domain ontologies to provide broader semantics.

7.2. METEOR-S Discovery API

The METEOR-S Discovery API [18] provides a semantic layer for Web services on top of a traditional UDDI registry. The API provides semantic equivalents to two of the most important operations in UDDI, publishing and discovering. Publishing is done with Web services that were annotated according to the SAWSDL standard storing all annotations present in the SAWSDL document. The API's discovery mechanism works against an ontology and is able to provide services that are compatible based on reasoning over the ontology.

8. Related work

In this section, we discuss related work in the areas of semi-automatic composition in general, as well as, more specifically Web service composition for bioinformatics.

Sirin et al. [19] present a Web service composer with the help of OWL-S. The main focus of their work is to filter out the non-compatiable services at each composition step, thus helping the user to select the appropriate services. While [19] offers filtering of inappropriate Web services it does not check for composition validity nor offers suggestions for partial plans, while it imposes backward chaining for the composition. On the plus side, it does provide a standard output format (OWL-S) and offers a graphical modeling environment that includes the execution of the composition. Similarly, Xu et al. [20] present Dynamic Semantic Association (DSAC) which utilizes the Dynamic Semantic (DS) and Semantic Association (SA) in service matching. This work is based on SRO (an OWL-DL ontology) and focuses on utilizing a service matchmaking algorithm to provide users with candidate services to select in an interactive fashion. This work supports filtering inappropriate services and achieves sequence control flow, based on a graphical user interface. However, this work cannot suggest partial plans, check the validity of the composition, or utilize a planning strategy. The composition result is not a standard and cannot be executed.

The two main projects focusing on bioinformatics Web services composition are myGrid and BioMoby. The myGrid project is part of the UK government's e-Science program [21], and aims to provide a comprehensive middleware suite specifically to support data intensive in-silico experiments in biology. The core components are based on Web service technology, which can be adopted in a “pick and mix” way for developers and tool builders to form, execute, manage and share discovery experiments. As a subproject of myGrid, Taverna is a bioinformatics Web service composition editor using the XScufl language with a friendly GUI. Compared with a general BPEL editor, it is much easier and more intuitive to use. However, it has two main drawbacks. First, the workflow generated by it can not be run outside of Taverna as a standard Web service, which limits its functionality and usability. Second, the user needs to manually handle data mapping using a scripting languages such as the Java-based Bean Shell language. This feature provides high flexibility with the cost of making Taverna somewhat hard to use by the average biologist. BioMoby [22] is an open source research project which implements an architecture for the discovery and distribution of biological data through Web services; data and services are decentralized, but the availability of these resources, and the instructions for interacting with them, are registered in a central location called MOBY Central. MOBY Central provides an object-driven registry query system with object and service ontologies. The strength of this initiative is in the service registration, discovery and transaction process with semantic information. However the service provider must implement a common interface and some libraries in order to registry its service to BioMoby. BioMOBY is limited in scope, in that it does not include workflow generation. For this functionality developers tend to use Taverna [23].

9. Conclusions

With more and more bioinformatics Web services and analysis tools available now, the need for composing workflows to achieve complex tasks is increasing.

Unfortunately, for biologists composing Web services is still too complex to be widely practiced. Our framework addresses this problem and uses multiple technologies to lower the complexity of service composition. First, Web services are annotated with Radiant using SAWSDL, and then registed in a semantic UDDI registry to assist service discovery and data mediation. Second, we implement a semantic Web service discovery tool to help biologists discover and browse the available services. Third, a multi-faceted wizard is used in a friendly and intuitive Web services composition editor to assist biologists. This wizard can hide most of BPEL's complexity from biologists. Furthermore, with the help of the METEOR-S Discovery API and the methods described in sections 6.1 and 6.2, the wizard can help biologists achieve data mediation, which is a huge challenge in biological Web service composition domain. The service composition can be persistently stored in BPEL format and can be executed in a standard BPEL engine. The result of the service composition itself is a standard Web service that can be reused by other applications or Web services.

Acknowledgments

Z.W. was supported by NIH R01 AI058515 to J.C.K. We would like to thank all the ApiDB project team members for their help in this study. J.A, J.C.K, D.B and C.A has been funded in whole or in part with Federal funds from the National Institute of Allergy and Infectious Diseases, National Institutes of Health, Department of Health and Human Services, under Contract No. HHSN266200400037C.

References

[1] F. Casati, S. Ilnicki, and L. Jin. Adaptive and dynamic service composition in EFlow. In Proceedings of 12th CAiSE, June 2000.

[2] H. Schuster, D. Georgakopoulos, A. Cichocki, and D. Baker. Modeling and composing service-based and reference process-based multi-enterprise processes. In Proceeding of 12th International Conference on Advanced Information Systems Engineering (CAiSE), Stockholm, Sweden, June 2000. Springer Verlag.

[3] F. Casati, M. Sayal, and M.-C. Shan. Developing e-services for composing eservices. In Proceedings of 13th International Conference on Advanced Information Systems Engineering(CAiSE), Interlaken, Switzerland, June 2001. Springer Verlag.

[4] R. Hull, M. Benedikt, V. Christophides, and J. Su. E-service: A look behind the curtain. In Proceeding of the 22th PODS, San Diego, USA, June 2003.

[5] J. Rao and X. Su. "A Survey of Automated Web Service Composition Methods". In Proceedings of the First International Workshop on Semantic Web Services and Web Process Composition, SWSWPC 2004, San Diego, California, USA, July 6th, 2004.

[6] C. Peltz. "Web Services Orchestration and Choreography", *Web Services Journal*, Volume 03--07, July 2003, pp. 30-35

[7] Ranjit Mulye. "METEOR-S Process Design and Development Tool" Master These

[8] W. M. P. van der Aalst and A. H. M. ter Hofstede. YAWL: yet another workflow language. *Information Systems*. Volume 30, Issue 4, pp. 245-275, 2005.

[9] E. Pignotti, P. Edwards, A. Preece, G. Polhill and N. Gotts. Semantic Workflow Management for E-Social Science. Proceedings of the Third International Conference on eSocial Science, October 2007.

[10]Workflow Patterns Standards Evaluation, www.workflowpatterns.com/evaluations/standard/index.php

[11] C.A. Goble, R. Stevens, G. Ng, S. Bechhofer, N.W. Paton, P.G. Baker, M. Peim, and A. Brass. Transparent Access to Multiple Bioinformatics Information Sources. *IBM Systems Journal*. Special issue on deep computing for the life sciences, 40(2):532 -- 552, 2001.

[12] S. McIlraith, and T. Son. Adapting Golog for Composition of Semantic Web Services. In Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR2002), pp. 482—493, 2002.

[13] Dan Wu, Bijan Parsia , Evren Sirin , James Hendler , and Dana Nau, "Automating DAML-S Web Services Composition Using SHOP2", ISWC 2003, LNCS 2870, pp. 195-210, 2003.

[14] B. Srivastava, J. Koehler, "Web service composition: Current solutions and open problems", ICAPS 2003 Workshop on Planning for Web Services, 28—35, 2003

[15] L. T. Santos, P.A. Roberto, M.A. Gonçalves, H. F. Laender: Design, Implementation, and Evaluation of a Wizard Tool for Setting Up Component-Based Digital Libraries. *Research and Advanced Technology for Digital Libraries*.

[16] C. Petrie. The World Wide Wizard of Open Source Services. 2007, IEEE International Conference on Web Services (ICWS' 2007)

[17] M. Nagarajan, K. Verma, A.P. Sheth and J.A. Miller, "Ontology Driven Data Mediation in Web Services," *International Journal of Web Services Research*, Vol. 4, No. 4. 2007.

[18] K. Verma, K. Sivashanmugam, A.P. Sheth, A. Patil, S. Oundhakar and J.A. Miller, "METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services," *Journal of Information Technology and Management*, Special Issue on Universal Global Integration, Vol. 6, No. 1. 2005.

[19] E. Sirin, B. Parsia, and J. Hendler. Filtering and Selecting Semantic Web Services with Interactive Composition Techniques. *IEEE Intelligent Systems*, 19:42–49, 2004.

[20] M. Xu, J. Chen, Y. Peng, X. Mei and Ch. Liu. A Dynamic Semantic Association-Based Web Service Composition Method. Web Intelligence, Issue18-22, 2006.

[21] R.D. Stevens, A.J. Robinson, and C.A. Goble. my Grid: personalized bioinformatics on the information grid. Bioinformatics, 19(90001):302i–304, 2003.

[22] M.D Wilkinson and M. Links. BioMOBY: an open source biological web services proposal. Brief. Bioinform., 3, 331–341, 2002.

[23] Y.C. Song, E. Kawas, B.M. Good, M.D. Wilkinson and S.J. Tebbutt. DataBiNS: a BioMoby-based data-mining workflow for biological pathways and non-synonymous SNPs. Brief. Bioinform., 23, 780-782, 2007.