Automated Syntactic Medation for Web Service Integration

Martin Szomszor, Terry R. Payne and Luc Moreau
School of Electronics and Computer Science
University of Southampton
Southampton, SO17 1BJ, UK
{mns03r, trp, L.Moreau}@ecs.soton.ac.uk

Abstract

As the Web Services and Grid community adopt Semantic Web technology, we observe a shift towards higher-level workflow composition and service discovery practices. While this provides excellent functionality to non-expert users, more sophisticated middleware is required to hide the details of service invocation and service integration. An investigation of a common Bioinformatics use case reveals that the execution of high-level workflow designs requires additional processing to harmonise syntactically incompatible service interfaces. In this paper, we present an architecture to support the automatic reconciliation of data formats in such Web Service worklflows. The mediation of data is driven by ontologies that encapsulate the information contained in heterogeneous data structures supplying a common, conceptual data representation. Data conversion is carried out by a Configurable Mediator component, consuming mappings between XML *schemas and* OWL *ontologies. We describe our system and give examples of our mapping language against the background of a Bioinformatics use case.*

1. Introduction

e-Science Grid applications are used to pool resources from multiple, heterogeneous resource providers. By exposing applications and data through Web Services, Web Service workflow has been adopted to encode scientific processes, allowing users to perform *in silico* science [4]. For many in the scientific community, this has resulted in stronger support for complex experimentation spanning both physical and organisational boundaries. The MYGRID[1] project is an example of such a system supporting Bioinformaticians in the construction, execution and sharing of workflows through the Taverna[2] graphical workbench.

Recent work within the MYGRID project has focused on supplying users with a richer, and more user-friendly environment to aid in the discovery and composition of services. FETA [7] has incorporated Semantic Web [2] technology into the service description policy using ontologies to capture the semantics of Web Services - essentially supplying users with conceptual definitions of what the service does using domain specific terminology. This has proven to be a valuable commodity in a system containing over a thousand services where searching over service descriptions alone is a cumbersome and tedious task.

With the introduction of Semantically annotated Web Services, workflow composition within MYGRID has shifted to a higher-level design process. While this makes workflow design more accessible to untrained users, it does lead to more complex architectual requirements. For example, the situation often arises where a user wishes to connect together two services that are conceptually compatible but have different syntactic interfaces. The current solution to this problem is entirely manual - users must insert mediation services into workflows to resolve any data incompatibilities.

This paper's contribution is a dynamic invocation framework designed to work in conjunction with existing workflow specification technologies (such as WSFL [6] and XSCUFL[3]) that provides automated data conversion when syntactically incompatible services are encountered within a workflow. This is achieved through a mapping language that links elements and attributes within XML schemas to concepts and properties in an OWL [10] ontology. By using the ontology as an intermediate representation, we are able to transform data structures between different formats using a *Translation Engine*. We present our *Configurable Mediator*, used in a Web Service workflow to harmonise data incompatibilities, and demonstrate it using a common Bioinformatics use case.

[1] www.mygrid.org.uk

[2] http://taverna.sf.net

[3] http://www.ebi.ac.uk/ tmo/mygrid/XScuflSpecification.html

This paper is organised as follows: Section 2 introduces the problem of service integration in terms of a Bioinformatics use case. In Section 3, we present the theory of our integration approach and the use of ontologies. Our Architecture is presented in Section 4 covering the invocation of services and the use of our *Configurable Mediator*. In Section 5, we give description of our mapping language, highlighting the complex mapping requirements and details of the translation process before evaluating our work in Section 6. Related work is examined in Section 7 before we conclude and give further work in Section 8.

2. Motivation and Use Case

For our use case, we examine a common Bioinformatics task: retrieve sequence data from a database and pass it to an alignment tool to check for similarities with other known sequences. According to the service-oriented view of resource access adhered to by MYGRID, this interaction can be modelled as a simple workflow with each stage in the task being fulfilled by a Web Service, illustrated in Figure 1. Many Web Services are available for retrieving sequence data. For our use case, we use the DDBJ-XML (`http://xml.ddbj.nig.ac.jp/`) and XEMBL (`http://www.ebi.ac.uk/xembl/`) services. To obtain a sequence data record, an accession number is passed as input and an XML document is returned. Such a document essentially contains the same information, namely the sequence data as a string (e.g. `atgagtga...`), references to publications, and features of the sequence (such as the protein translation). However, the format of the data returned by each provider is different - XEMBL returns an INSD[4] formatted record, whereas DDBJ-XML returns a document using their own custom format. The next stage in the workflow is to pass the sequence data to an alignment service such as the BLAST service at NCBI[5]. This service can consume a string of FASTA[6] formatted sequence data.

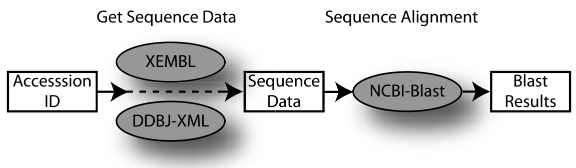

Figure 1. A simple bioinformatics task: get sequence data from a database and perform a sequence alignment on it.

Intuitively, a Bioinformatician will view the two sequence retrieval tasks as the same type of operation, expecting both to be compatible with the NCBI-Blast service. The semantic annotations attached through FETA affirm this as the output types are assigned the same conceptual type, namely a Sequence Data Record concept. However, when plugging the two services together, we see that the output from either sequence data retrieval service is not directly compatible for input to the NCBI Blast service. For example, the DDBJ-XML Service produces a DDBJ formatted XML document whereas the NCBI-Blast service consumes a FASTA formatted sequence, as shown in Figure 2.

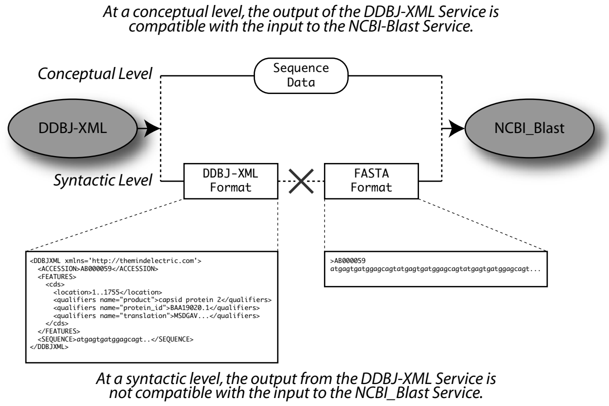

Figure 2. The output from the DDBJ-XML Service is not compatible for input to the NCBI-Blast Serivce.

We use the term *service integration* to denote the combination of two or more web services within a workflow with data passing from one service to another. In our use case, service integration occurs between the sequence data retrieval service and the NCBI-Blast alignment service. Since these two services have syntactically incompatible interfaces, successful service integration requires a translation between the two data formats assumed. We define this translation as *syntactic mediation*.

3. Ontologies for Data Integration

The service integration problem we present emanates from the variety of data formats assumed by service providers. *Data Integration* (the means of gathering information from multiple, heterogeneous sources) also addresses this problem. Building on existing data integration models [12], we present our service integration problem against a three-tier data representation model, separating the storage, structure, and meaning of information:

1. **Physical Layer - How the data is stored**
   Data can be stored in a variety of different formats: proprietary binary files, text files, XML documents and relational databases encompass the most common methods.

[4]http://www.ebi.ac.uk/embl/Documentation/DTD/INSDSeq_v1.3.dtd.txt
[5]http://www.ncbi.nlm.nih.gov/BLAST/
[6]http://www.ebi.ac.uk/help/formats_frame.html

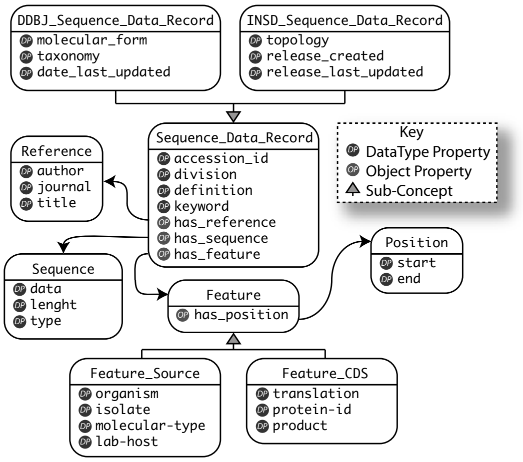

Figure 3. An ontology to describe sequence data. See `http://www.ecs.soton.ac.uk/~mns03r/ont/sequencedata` for a full listing.

2. **Logical Layer - How the data is structured**
   On top of the physical representation layer, the logical organisation of the data describes the structure of physical data elements, such as XML schema and relational database models.

3. **Conceptual Layer - What the data means**
   Above the logical layer, the conceptual model of an information source specifies what the data means using high-level language, such as an OWL ontology or description logic.

Since XML is used to describe the data transported to and from Web Services, we can assume a homogeneous physical layer. The data incompatibilities occurring in our use case stem from different logical organisations of data, i.e. service providers designing different XML schemas. To enable the transformation of data between different logical formats, we link conceptually equivalent elements from different logical schemas to a common concept in the conceptual layer via a set of mappings. An OWL ontology is used to describe the contents of XML schemas with a custom mapping language to specify the correspondence of XML schema elements[7] to OWL concepts and properties. Figure 3 shows a Sequence Data Record ontology we use to describe the data formats in our use case. The main concept, `Sequence_Data_Record`, has two sub-concepts: `DDBJ_Sequence_Data_Record` and `INSD_Sequence_Data_Record`. This is used to express the subtle differences between the two formats which contain additional information while sharing common properties such as `accession_id`. If a service is described as consuming a `Sequence_Data_Record`, it should be able to consume instances of either sub-concept because the necessary information will be present. Each Sequence Data Record has a `Sequence` that contains the string of sequence data, references to publications on the sequence, and a number of Features. There is a variety of sequence features; we show two common ones in this example: `Feature_Source` (where and how the sequence was gathered) and `Feature_CDS` (which shows the protein sequence translation and id).

[7]The term *elements* is used to refer to XML schema elements, attributes, and text values.

4. Architecture

Before presenting our architecture, we list the four principal requirements of the system:

R1. *The ability to harmonise data incompatibilities in Web Service workflows using syntactic mediation driven by ontologies that capture the semantics of data structures, and mappings between XML data sources and their corresponding conceptual representations.*
With this approach, users must define an ontology to describe the contents of an information source and a set of mappings to specify the translation of data to and from the conceptual representation.

R2. *A modular and composable mapping language to support sharing and reuse.*
Since service providers often expose multiple operations over subsets of the same dataset, the mapping design overhead can be reduced through a modular and composable mapping language.

R3. *Support for the invocation of arbitrary WSDL Web Services.*
Since large scale e-Science Grid applications pull resources from multiple providers into a dynamic and volatile environment, the ability to invoke unseen services is paramount because services may appear and disappear without warning. Often, it is a requirement to replace Web Services within a workflow with new ones when the original services are unavailable.

R4. *Minimise annotation overheads by utilising existing semantic annotation techniques.*
Rather than impose new annotation requirements, we build our solution on existing semantic annotation techniques, namely the association of WSDL message parts with concepts in an ontology.

To automate the process of syntactic mediation, we require a mechanism to link XML elements to OWL concepts and properties. To simplify this problem, we assume a canonical XML representation of an OWL concept instance, referred to in this paper as an OWL-$\mathcal{XI}$ (i.e OWL XML instance). In order to validate instances of OWL concepts,

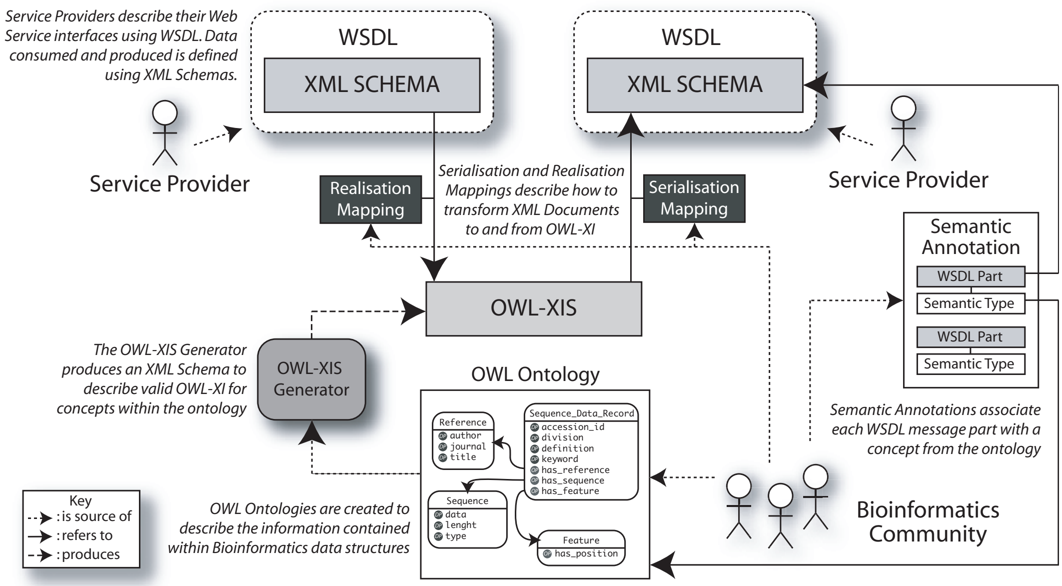

Figure 4. Architecture Overview: information sources and relationships.

we have built an XML schema generator to produce OWL-$\mathcal{XIS}$ (OWL XML instance schemas) that validate OWL-$\mathcal{XI}$. This component utilises the JENA[8] API to compute concept hierarchies and produce XML schemas that mirror them. With an OWL-$\mathcal{XIS}$ in place, the transformation of XML documents to and from an OWL-$\mathcal{XI}$ can be viewed as an XML to XML translation process. To distinguish between these translations, we define the terms *conceptual realisation* - denoting the transformation from XML to OWL-$\mathcal{XI}$, and *conceptual serialisation* - for the transformation from OWL-$\mathcal{XI}$ to XML. An OWL-$\mathcal{XIS}$ for the ontology presented in Figure 3 can be found at `http://www.ecs.soton.ac.uk/˜mns03r/ont/sequencedata.xsd`.

Our architecture is built around the existing MYGRID infrastructure. We assume service providers expose services using WSDL descriptions with data structures specified using XML schemas. We show the relationship between these existing WSDL descriptions, their semantic Annotations which relate them to concepts within a Bioinformatics ontology, and the serialisation and realisation mappings in Figure 4. Dotted lines represent the source of information and solid lines denote references (e.g. a serialisation mapping references elements in the WSDL description and the OWL-$\mathcal{XIS}$). In Figure 4, three information providers are shown: two separate service providers (upper left and upper right) and the Bioinformatics community (bottom right). Each service provider supplies a WSDL description of their service, the Bioinformatics community collectively supply the Bioinformatics ontology, semantic annotations for each service and the serialisation and realisation mappings.

To illustrate the mechanics of our system and the interface to the Configurable Mediator (C-Mediator), we continue using our use case from Section 2. For demonstration purposes, we use the DDBJ Sequence retrieval service and the NCBI-Blast service. In Figure 5, we give a visual representation of the workflow execution and syntactic mediation. XML schemas for datasets and OWL-$\mathcal{XIS}$, as well as the serialisation and realisation mappings correspond with those presented in Figure 4. Beginning at the upper left of the diagram (marked 1), the workflow Input (accession id) is used to create an input message for the DDBJ service. The Dynamic WSDL Invoker (DWI) calls the service using SOAP encoding over HTTP transport. The output message, containing the full sequence data record, is then passed to the C-Mediator to be converted into the correct format for input to the NCBI-Blast service. The C-Mediator is comprised of three components; two instances of a *Translation Engine* and a *Mediation-KB*. During the first half of syntactic mediation (2), the sequence data record is transformed to an instance of a `DDBJ_Sequence_Data_Record`. This transformation is performed by the Translation Engine which consumes XML schemas for both the source and target representation as well as the `DDBJ-XML->Seq-Data-Ont` mapping. The output of this transformation is an OWL-$\mathcal{XI}$ representing the Sequence Data Record. This is passed to the Mediation-KB which imports the individual into a JENA inference model to perform reasoning (e.g. calculate concept hierarchies). From the perspective of our use case, reason-

[8]http://jena.sourceforge.net/

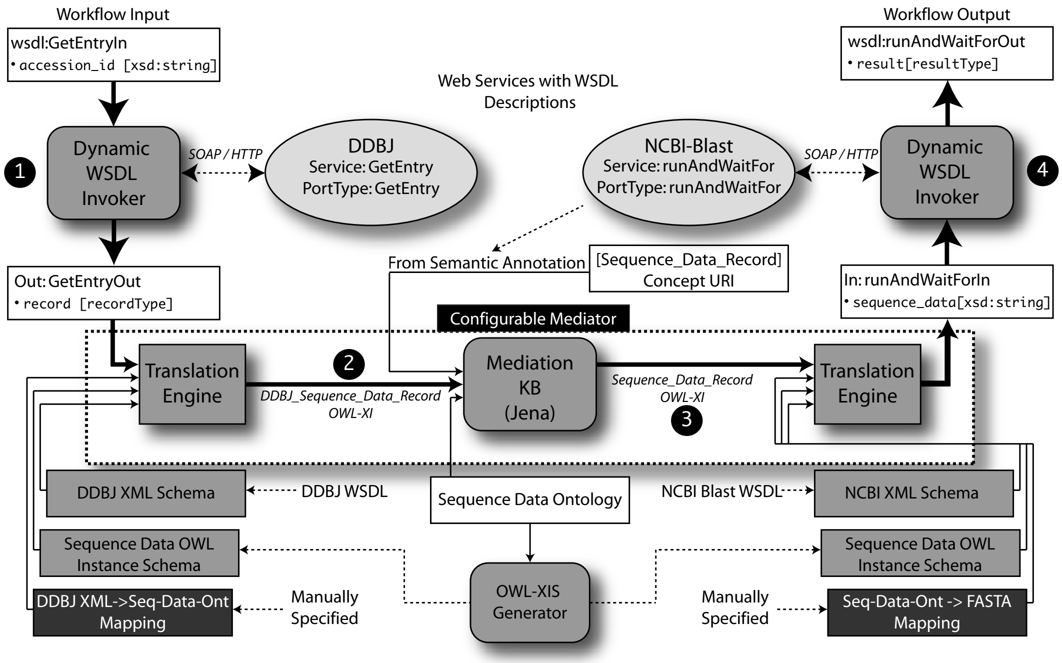

Figure 5. The Configurable Mediator

ing is required because the output concept of the DDBJ-XML service (`DDBJ_Sequence_Data_Record`) is subsumed by the input concept of the NCBI-Blast service (`Sequence_Data_Record`). Therefore, an instance of the `DDBJ_Sequence_Data_Record` concept is compatible for input to the NCBI-Blast service - this can only be deduced using the ontology definition and a reasoning engine such as JENA. The second half of syntactic mediation is to convert this OWL-$\mathcal{XI}$ to a different representation, in this case FASTA format (3). Again, the Translation Engine is used to achieve this, consuming the `Seq-Data-Ont->FASTA` mapping along with the relevant XML schemas. The output of mediation stage is then used to create an input message for the NCBI-Blast service which is called by the DWI (4). The service invocation output message, containing the results of the sequence alignment, is then passed back as the workflow output.

5. Mapping Language

A single Web Service may offer a number of different operations, each having different inputs and outputs. Often, different operation's input and output types overlap, in effect reusing types defined in a global schema. For example, the DDBJ-XML service in our use case offers many operations over sequence data records. When passing an accession id as input, the user can retrieve records from different databases (e.g. SWISS and EMBL) or different parts of the record such as the isolated sequence data or a particular sequence feature. Because of this schema reuse, we design our mapping language to be modular and composable to minimise design effort. Therefore, transformations are specified using a set of mappings which describe the relationship between either single, or groups of elements and attributes. In this Section, we describe the requirements of our mapping language against our use case datasets, provide an overview of the transformation mechanics, and give a small example. A full and formal specification of our mapping language is in preparation.

We stated in Section 4 that we simplify the transformation requirements for conceptual realisation and conceptual serialisation by assuming a canonical XML representation of XML concept instance (OWL-$\mathcal{XI}$). Examination of use case datasets reveals that the mapping requirements are complex, as we illustrate in Figure 6 with a subset of the use case transformation. We describe these mappings below and form a list of requirements:

1. **Single element to element mapping**
   The mapping language should enable to association of elements from the source document to the destination document. In Figure 6, The `<DDBJXML>` element is mapped to the `<DDBJ_Sequence_Data_Record>` element.

2. **Element contents mapping**
   The `<ACCESSION>` element and its text value are mapped to the `<accession_id>` element.

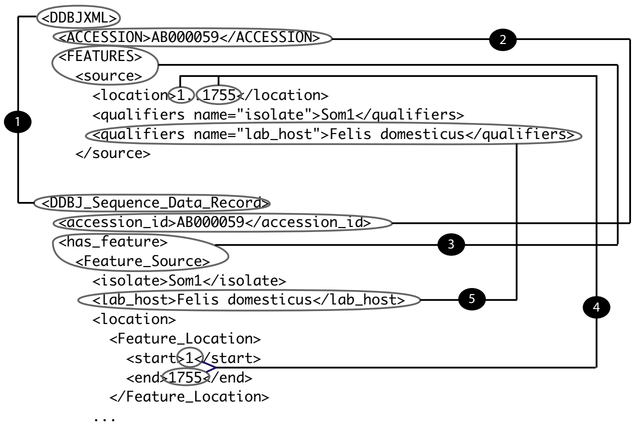

Figure 6. Mapping of Elements between a DDBJXML **Sequence Data Record and a** DDBJ-Sequence-Data-Record OWL-$\mathcal{XI}$.

3. **Multiple element mapping**
   A <FEATURES> element containing a <source> element is mapped to a <has-feature> element containing a <Feature-Source> element.

4. **String manipulation support**
   The <location> element has text containing the start and end position, delimited by a ".." . Each of these positions must be mapped to separate elements in the destination document.

5. **Predicate support**
   The contents of the <qualifiers> element should be mapped differently depending on the value of the name attribute - in the case of Mapping 5, when the string equals "lab-host" the value is mapped to the <lab-host> element.

To give an overview of the transformation mechanics, we supply a simple example, shown in Figure 7 where the upper layer shows the desired transformation. Our translation approach is recursive, starting from the root node of the source document, mappings are applied to create elements in the destination document. For example, Stage 1 in Figure 7 identifies the element <a> containing the two <b> elements and constructs an <x> element containing two <y> elements using the mapping a/b -> x/y. After Stage 1, a recursion is made on element <b> so each <b> element's value is inserted in the <y> element's in the destination document according to the mapping b/$ -> y/$ (where $ denotes text value). Example mappings to describe the translation in Figure 6 are given in Figure 8.

Mapping 1 maps the <ACCESSION> element and its text value. Mapping 2 associates the <Features>/<Source> elements and Mapping 3 maps the lab_host qual-

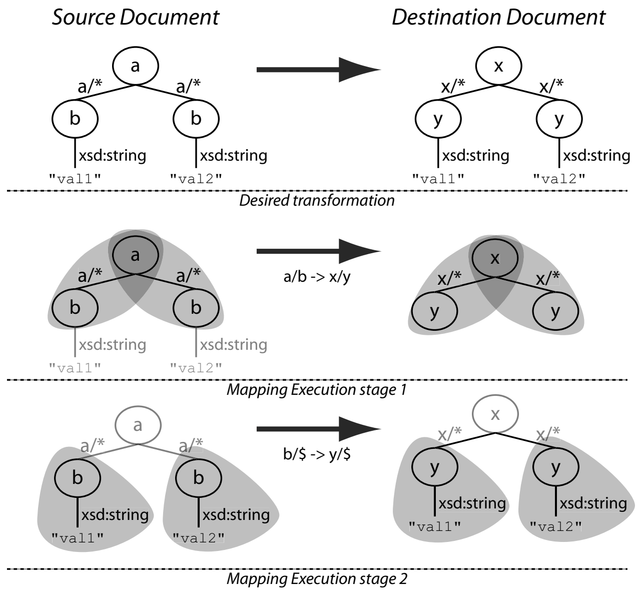

Figure 7. The execution of a mapping.

```
<binding xmlns="http://www.ecs.soton.ac.uk/~mns03r/mapping/ddbj-to-ont-mapping"
     xmlns:sns="http://jaco.ecs.soton.ac.uk/schema/DDBJ"
     xmlns:dns="http://jaco.ecs.soton.ac.uk/ont/sequencedata">

<mapping id="1">
 <source match="sns:DDBJXML/sns:ACCESSION"/>
 <destination create="dns:DDBJ_Sequence_Data_Record[join]/dns:accession_id[branch]/"/>
 <mapping>
  <source match="sns:ACCESSION/$"/>
  <destination create="dns:accession_id[join]/$"/>
 </mapping>
</mapping>

<mapping id="2">
 <source match="sns:DDBJXML/sns:FEATURES/sns:source"/>
 <destination create="dns:DDBJ_Sequence_Data_Record[join]/
                      dns:has_feature[branch]/dns:Feature_Source[branch]"/>
</mapping>

<mapping id="3">
 <source match='sns:source/sns:qualifiers[sns:qualifiers/sns:name/$ = "lab_host"]'/>
 <destination create="dns:Feature_Source[join]/dns:lab-host[branch]"/>
 <mapping>
  <source match="sns:qualifiers/$"/>
  <destination create="dns:lab-host[join]/$"/>
 </mapping>
</mapping>

<mapping id="4">
 <source match="sns:location/$^[^.]+"/>
 <destination create="dns:Location[join]/dns:start[branch]/$"/>
</mapping>

<mapping id="5">
 <source m atch="sns:location/$[^.]+$"/>
 <destination create="dns:Location[join]/dns:end[branch]/$"/>
</mapping>

</binding>
```

Figure 8. An example mapping - The full mapping for our use case can be found at http://www.ecs.soton.ac.uk/~mns03r/mapping/ddbj-to-ont-mapping.xml.

ifier. Mapping 3 contains a predicate evaluation `[sns:qualifiers/sns:name/$ = "lab_host"]` to ensure the `qualifier` element is mapped correctly. Mappings 4 and 5 map the contents of the `location` element to two different elements in the destination. A regular expression is attached to the string selection statements (e.g. `"$^[^.]+"`) to separate the string values.

6. Evaluation

This paper demonstrates a proof of concept solution to the service integration problem associated with multiple service providers assuming different data formats. Our system has been tested against the use case presented in Section 2 using a Sequence Data ontology and a set of serialisation and realisation mappings. The requirements presented in Section 4 are met as follows:

R1. The Configurable Mediator is able to translate conceptually equivalent XML documents between different logical organisations to resolve workflow data incompatibilities.

R2 . Our mapping language adheres to conventional XML namespace declarations and supports document inclusion. Document translations are expressed in terms of the relationship between XML schema elements. The Translation Engine interprets these statements at runtime so service operations that reuse XML schema definitions are supported by one set of mappings.

R3 . The Dymanic WSDL Invoker is able to invoke previously unseen WSDL services. The current implementation supports the most widely used invocation methods (i.e. SOAP encoding over HTTP transport).

R4. By extending existing ontologies used to semantically annotate Web Services with more detailed ontologies that capture the semantics of the data content, we are able provide the necessary information to support our mediation approach without imposing a new annotation policy.

7. Related Work

We position related work against the three-tier data representation model presented in Section 3. The TAMBIS project [13] provides a data integration framework that operates in the molecular biology domain. TAMBIS supports the gathering of information from varying data sources through a high-level, conceptually driven query interface. In this system, information sources are typically proprietary flat file structures, the outputs of programs, or the product of a services, none of which share a common query interface. A molecular biology ontology, expressed using a description logic, is used in conjunction with functions specifying how each concept is accessed within a data source to deliver an advanced querying interface. TAMBIS supports data integration across all three layers of the three-tier model, but only one direction, namely from *physical* up to *conceptual*. For Web Service integration, two way translation is required so data can be converted between different logical representations.

Moreau *et al* [9], have also investigated the need to integrate data from heterogeneous source, in this case, within the Grid Physics Network, GriPhyn[9]. Like the bioinformatics domain, data source used in physics Grids range across a variety of legacy file formats. To provide a homogeneous access model to these varying data sources, Moreau *et al* proposes a separation between logical and physical file structures. This allows access to data sources to be expressed in terms of the logical structure of the information, rather than the way it is physically represented. To achieve this, XML schema is used to express the logical structure of an information source, and mappings are used to relate XML schema elements to their corresponding parts within a physical representation. The XML Data Type and Mapping for Specifying Datasets (XDTM) prototype provides an implementation which allows data source to be navigated using XPATH. This enable users to retrieve and iterate across data stored over multiple, heterogeneous sources. While this approach is useful when amalgamating data from different *physical* representations, it does not address the problem of data represented using different *logical* representations. Our service integration problem arises from the fact that different service providers use different logical representations of conceptually equivalent information.

The SEEK project [3] specifically addresses the problem of heterogeneous data representations in service oriented architectures. Within their framework, each service has a number of ports which expose a given functionality. Each port advertises a *structural type* which defines the input and output data format by a reference to an XML schema type. If the output of one service port is used as input to another service port, it is defined as *structurally valid* when the two types are equal. Each service port can also be allocated a *semantic type* which is specified by a reference to a concept within an ontology. If two service ports are plugged together, they are *semantically valid* if the output from the first port is subsumed by the input to the second port. Structural types are linked to semantic types by a registration mapping using a custom mapping language based on XPATH. If the concatenation of two ports is semantically valid, but not structurally valid, an XQUERY transformation

[9] http://griphyn.org/

can be generated to integrate the two ports, making the link *structurally feasible*. The SEEK system provides data integration between different logical organisations of data using a common conceptual representation, the same technique that we adopt. However, their work is only applicable to services within the bespoke SEEK framework. The architecture we present is designed to work with arbitrary WSDL Web Services annotated using semantic Web techniques.

Hull *et al* [5] have also investigated the service integration problem within the MYGRID application. They dictate that conversion services, or *shims*, can be placed in between service whenever some type of translation is required - exactly as the current MYGRID solution. They explicitly specify that a shim service is *experimentally neutral* in the sense that it has no side-effect on the result of the experiment. By enumerating the types of shims required in bioinformatics Grids and classifying all instances of shim services, it is hoped that the necessary translation components could be automatically inserted into a workflow. However, their focus is not on the translation between different data representation, rather the need to manipulate data sets; extracting information from records, finding alternative sources for data, and modifying workflow designs to cope with iterations over data sets.

8. Conclusions and Future Work

In this paper, we have used a bioinformatics Grid application to show the problem of data integration in open, service oriented architectures. With OWL ontologies in place to capture the semantics of a data source, we can use instances of these ontology concepts as an intermediate representation to support the conversion of data between different formats. Our mapping language has been specially designed to be modular, to support sharing and reuse, as well as expressive to cope with the complicated mappings required in Bioinformatics data structures. We have implemented novel features which allow two-way integration across the conceptual and logical layers of the three-tier data model, supporting the conversion of data between logical formats using the conceptual layer as a common model.

While our Architecture has been designed to fit within the existing MYGRID application, its principles apply to any Grid or Web Services architecture. When incorporating the use of Semantic Web technology, namely the association of WSDL message parts with concepts from an ontology, we have followed existing practices such as those used by the FETA system, OWL-S [8], WSMO [11], and WSDL-S [1].

Within our current architecture, it is assumed that the serialisation and realisation mappings are known at execution time. To fully automate the mediation process, we intend to develop a mapping registry which supports users in the uploading and sharing of mappings and provides a query interface to retrieve mappings. With such a registry in place, it would also be possible to infer the semantics of a Web Service by finding existing mappings within the registry for the data types consumed and produced.

9. Acknowledgment

This research is funded in part by EPSRC myGrid project (reference GR/R67743/01).

References

[1] R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, and A. S. K. Verma. Web service semantics - WSDL-S. Technical report, UGA-IBM, 2005.

[2] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 34 – 43, 2001.

[3] S. Bowers and B. Ludascher. An ontology-driven framework for data transformation in scientific workflows. In *Intl. Workshop on Data Integration in the Life Sciences (DILS'04)*, 2004.

[4] C. Goble, S. Pettifer, R. Stevens, and C. Greenhalgh. Knowledge Integration: In silico Experiments in Bioinformatics. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure Second Edition*. Morgan Kaufmann, November 2003.

[5] D. Hull, R. Stevens, and P. Lord. Describing web services for user-oriented retrieval. 2005.

[6] F. Leymann. Web services flow language (WSFL 1.0), May 2001.

[7] P. Lord, P. Alper, C. Wroe, and C. Goble. Feta: A lightweight architecture for user oriented semantic service discovery. In *The Semantic Web: Research and Applications: Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece*, pages 17 – 31, Jan. 2005.

[8] D. Martin, M. Burstein, G. Denker, J. Hobbs, L. Kagal, O. Lassila, D. McDermott, S. McIlraith, M. Paolucci, B. Parsia, T. Payne, M. Sabou, E. Sirin, M. Solanki, N. Srinivasan, and K. Sycara. OWL-S: Semantic markup for web service. Technical report, The OWL Services Coalition, 2003.

[9] L. Moreau, Y. Zhao, I. Foster, J. Voeckler, and M. Wilde. XDTM: the XML Dataset Typing and Mapping for Specifying Datasets. In *Proceedings of the 2005 European Grid Conference (EGC'05)*, Amsterdam, Nederlands, Feb. 2005.

[10] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. OWL web ontology language semantics and abstract syntax. Technical report, W3C, 2004.

[11] D. Roman, H. Lausen, and U. Keller. D2v1.0. web service modeling ontology (WSMO), September 2004. WSMO Working Draft.

[12] J. F. Sowa and J. A. Zachman. Extending and formalizing the framework for information systems architecture. *IBM Syst. J.*, 31(3):590–616, 1992.

[13] R. Stevens, C. Goble, N. W. Paton, S. Bechhofer, G. Ng, P. Baker, and A. Brass. Complex Query Formulation Over Diverse Information Sources in TAMBIS. In Z. Lacroix and T. Critchlow, editors, *Bioinformatics: Managing Scientific Data*. Morgan Kaufmann, May 2003.