Consistency and Modularity in Mediated Service-based Data Integration Solutions[1]

Yaoling Zhu and Claus Pahl
Dublin City University, School of Computing, Dublin 9, Ireland,

ABSTRACT. A major aim of the Web service platform is the integration of existing software and information systems. Data integration is a central aspect in this context. Traditional techniques for information and data transformation are, however, not sufficient to provide flexible and automatable data integration solutions for Web service-enabled information systems. The difficulties arise from a high degree of complexity in data structures in many applications and from the additional problem of heterogeneity of data representation in applications that often cross organisational boundaries. We present an integration technique that embeds a declarative data transformation technique based on semantic data models as a mediator service into a Web service-oriented information system architecture. Automation through consistency-oriented semantic data models and flexibility through modular declarative data transformations are the key enablers of the approach.

KEYWORDS: Data Integration, Software Architecture, Semantic Data Modelling, Mediated Architecture, Declarative Data Transformation.

INTRODUCTION

A major aim of the Web service platform is the integration of existing software and information systems (Alonso et al., 2004). Information and data integration is a central aspect in this context. Traditional techniques based on XML for data representation and XSLT for transformations between XML documents are not sufficient to provide a flexible and automatable data integration solution for Web service-enabled information systems. Difficulties arise from the high degree of complexity in data structures in many business and technology applications and from the problem of heterogeneity of data representation in applications that cross organisational boundaries.

The emergence of the Web services platform and service-oriented architecture (SOA) as an architecture paradigm has provided a unified way to expose the data and functionality of an information system (Stal, 2002). The Web services platform has the potential to solve the problems in the data integration domain such as heterogeneity and interoperability (Orriens, Yang and Papazoglou, 2003; Haller, Cimpian, Mocan, Oren and Bussler, 2005; Zhu et al., 2004). Our contribution is an integration technology framework for Web-enabled information systems comprising of

- firstly, a data integration technique based on semantic, ontology-based data models and the declarative specification of transformation rules and
- secondly, a mediator architecture based on information services and the construction of connectors that handle the transformations to implement the integration process.

[1] ***This chapter appears in "Service and Business Computing Solutions with XML: Applications for Quality Management and Best Processes" edited by P.C.K. Hung, Copyright 2009, IGI Global, www.igi-global.com. Posted by permission of the publisher.***

A data integration technique in the form of a mediator service can dynamically perform transformations based on a unified semantic data model built on top of individual data models in heterogeneous environments (Wiederhold, 1992). Abstraction has been used successfully to address flexibility problems in data processing (Rouvellou, Degenaro, Rasmus, Ehnebuske and McKee, 2000). With recent advances in abstract, declarative XML-based data query and transformation languages (Zhu et al., 2004) and Semantic Web and ontology technology (Daconta, Obrst and Smith, 2003), the respective results are ready to be utilised in the Web application context. The combination of declarative and semantic specification and automated support of architecture implementations provides the necessary flexibility and modularity to deal with complexity and consistency problems. Two central questions to the data integration problem and its automation shall be addressed in this investigation:

- How to construct data model transformation rules and how to express these rules in a formal, but also accessible and maintainable way is central.
- How integration can be facilitated through service composition to enable interoperability through connector and relationship modelling.

We show how ontology-based semantic data models and a specific declarative data query and transformation language called Xcerpt (Bry and Schaffert, 2002) and its execution environment can be combined in order to allow dynamic data transformation and integration. We focus on technical solutions to semantically enhance data modelling and adapt Xcerpt and its support environment so that it can facilitate the dynamic generation of Xcerpt query programs (in response to user requests) from abstract transformation rules.

BACKGROUND

Information integration is the problem of combining heterogeneous data residing at different sources in order to provide the user with a unified view (Lenzerini, 2002). This view is central in any attempt to adapt services and their underlying data sources to specific client and provider needs. One of the main tasks in information integration is to define the mappings between the individual data sources and the unified view of these sources and vice versa to enable this required adaptation. Fig.1 shows two sample schemas, which might represent the views of client and provider on a collection of customers, that require integration. The integration itself can be defined using transformation languages.

Information integration has the objective of bringing together different types of data from different sources in order for this data to be accessed, queried, processed and analysed in a uniform manner. Recently, service-based platforms are being used to provide integration solutions. In the Web services context, data in XML representation, which is retrieved from individual Web-based data services, needs to be merged and transformed to meet the integration requirements. Data schema integration cannot be fully automated on a syntactic level since the syntactic representation of schemas and data does not convey the semantics of different data sources. For instance, a customer can be identified in the configuration management repository by a unique customer identifier; or, the same customer may be identified in the problem management repository by a combination of a service support identifier and its geographical

location, see Fig. 1. Ontology-based semantic data models can rectify this problem by providing an agreed vocabulary of concepts with associated properties.

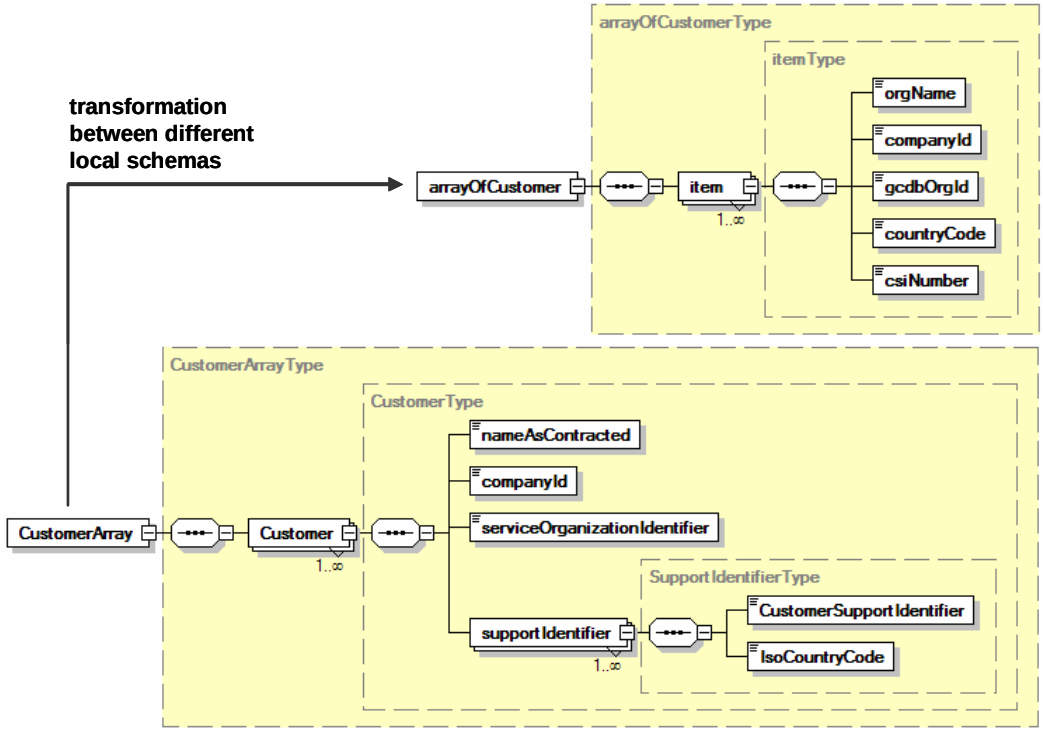

Fig. 1. Two Schema Diagrams of the Global Data Model that need to be integrated. (© 2008, Claus Pahl. Used with permission).

XSLT is the most widely used XML data integration language, but suffers from some limitations within our context due its is syntactical focus and operational language.

- Semantics: Only the syntactical integration of query and construction part of a XSLT transformation program is specified, but consistency in terms of the semantics can not be guaranteed.
- Modularity: XSLT does not support a join or composition operator on XML documents that allows several source XML documents to merged into one before being transformed.
- Maintainability: XSLT transformations are difficult to write, maintain, and reuse for large-scale information integration. It is difficult to separate the source and target parts of transformation rules as well as the filtering constraints due to its operational character without a separation of query and construction concerns.

Due to these drawbacks, we propose semantic data models and a declarative query and transformation approach providing more expressive power and the ability to automatically generate query and transformation programs as connectors for services-based data integration in Web-enabled information systems. A range of characteristics of XML query and transformation languages beyond XSLT, which have been studied and compared (Jhingran, Mattos and Pirahesh, 2002; Lenzerini, 2002; Peltier, Bezivin, and Guillaume, 2002), led us to choose the fully declarative language Xcerpt (Bry and Schaffert, 2002) as our transformation platform (Zhu, 2007).

DATA TRANSFORMATION AND CONNECTOR ARCHITECTURE

Mappings between data schemas of different participants might or might not represent the same semantical information. The Semantic Web and in particular ontology-based data domain and service models (Daconta et al., 2003) can provide input for improvements of current integration approaches in terms of data modelling and transformation validation by providing a notion of consistency, based on which an automated transformation approach can become reliable (Reynaud, Sirot and Vodislav, 2001, Haller et al., 2005). We define consistency here as the preservation of semantics in transformations.

Information Architecture

Ontologies are knowledge representation frameworks that represent knowledge about a domain in terms of concepts and properties of these concepts. We use a description logic notation here, which is the formal foundation of many ontology languages such as OWL (Daconta et al., 2003). Description logic provides us with a concise notation here to express a semantic data model. The elements of the XML data models of each of the participants are represented as concepts in the ontology. The concept `Customer` is defined in terms of its properties – data type-like properties such as a name or an identifier and also object type properties such as a collection of services used by a customer. Three concept descriptions, using the existential quantifier "∃" here, express that a customer is linked to an identification through a `supportID` property, to a name using the `custName` property, and to services using `Services`. In some cases, these properties refer to other composite concepts, sometimes they refer to atomic concepts that act as type names here. Technically, the existential quantification means that there exits for instance a name that is a customer name.

```
Customer =
     ∃ supportID . Identification  ∧
     ∃ custName . Name             ∧
     ∃ usedServices . Service

Service =
     ∃ custID . ID       ∧
     ∃ servSystem . System

System =
     ∃ hasPart . Machine
```

The ontology represents syntactical and semantical properties of a common overarching data model, which is agreed upon by all participants such as service (or data) provider and consumer. This model is actually a domain ontology, capturing central concepts of a domain and defining them semantically. This means that all individual XML data models can be mapped onto this common semantic model. These mappings can then be used to automatically generate transformations between different concrete participant data models. The overall information architecture is summarised in Fig. 2.

Although there is a standardised OWL-based equivalent for our description logic ontology, for practical reasons a corresponding semantically equivalent XML representation is needed. The corresponding global XML schema representation for the customer element is:

```
<!ELEMENT Customer ( Service, System ) >
<!ATTLIST Customer
     supportID ID
     custName  Name >
```

Here, the principle of this mapping becomes clear: ontology concepts are mapped to XML elements and specific predefined atomic concepts serve to represent simple properties that are mapped to XML attributes. We have focused on the core elements of ontologies and XML data here to highlight the principles. Description elements of XML such as different types of attributes or option and iteration in element definition can also be captured through a refined property language. In particular the Web Ontology Language OWL provides such constructs (W3C, 2004).

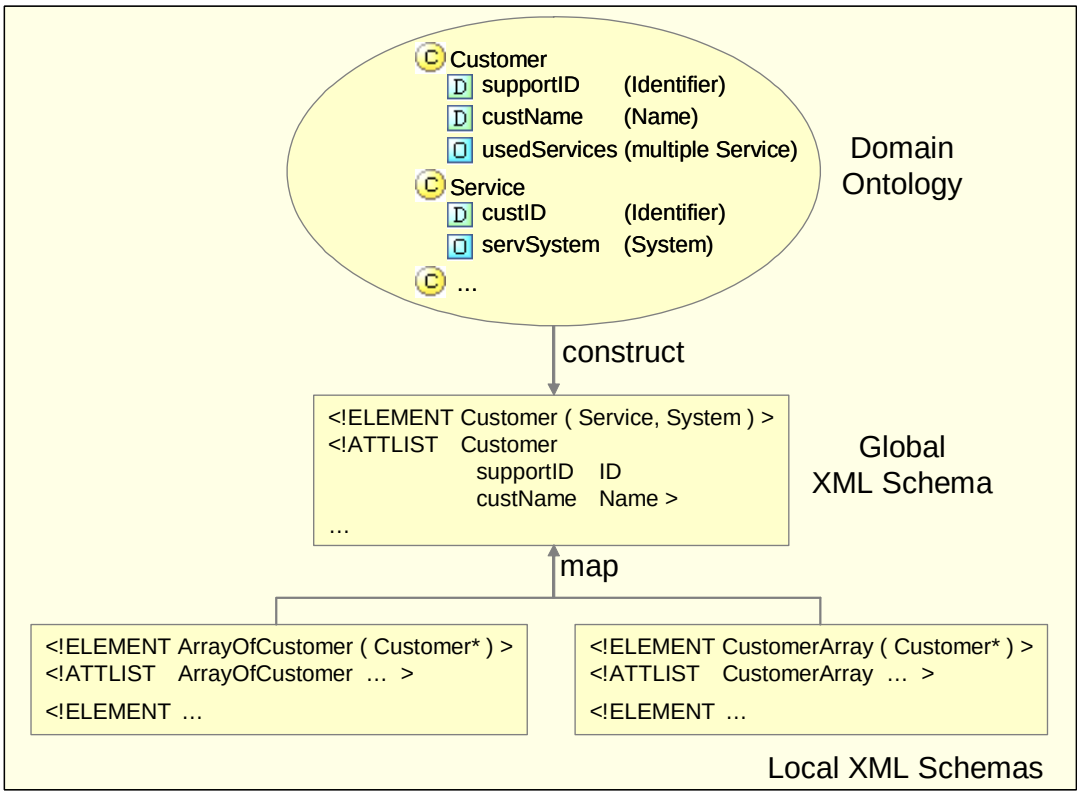

Fig. 2. Information Architecture Overview. (© 2008, Claus Pahl. Used with permission).

Transformation Rule Construction

The ontology provides a semantically defined global data model from which transformations between different participant data representations can be derived. This construction needs to address a number of specific objectives regarding the transformation rules:

- modularity of transformation rules is needed for the flexible generation and configuration of transformation rules by allowing these rules to be specific to particular data elements,
- consistency needs to be addressed for the reliable generation and configuration of transformation rules by allowing semantics-preserving rules to be constructed automatically.

Based on a data-oriented domain ontology and two given local data models (source and target, expressed as XML schemas) that are mapped onto the ontology, the rule construction process is based on three steps:

1. Define one transformation rule per concept in the ontology that is represented in the target data model.
2. Identify semantically equivalent concepts of the selected concepts in the source data model.
3. For each identified concept:
    a. determine required attributes – these are end nodes of the ontological structure,
    b. copy semantically equivalent counterparts from the source model.

A necessary prerequisite is that all concepts of the source model are actually supported by the target data model. Otherwise, the transformation definition cannot be completed.

The transformation rules based on the sample ontology for the given customer example will be presented later on once the transformation language is introduced. These could be formulated such that data integration problem depicted in Fig. 1 is formally defined. The mappings between participant data models and the data ontology define semantically equivalent representation of common agreed ontology elements in the data models. Consequently, the presented rule construction process is consistent in that it preserves the semantics in transformations.

The concrete target of this construction is the chosen declarative transformation language Xcerpt. The construction process has been expressed here in abstract terms – a complete specification in terms of transformation languages such as QVT or even Xcerpt itself would have been too verbose for this context. Declarativeness and modularity provide the required flexibility for our solution, in addition to consistency that has been addressed through the semantic ontology-based data models. The construction of transformation rules is actually only the first step in the provision of XML data integration. These transformations can be constructed prior to the customer query construction and stored in rule repositories.

Xcerpt Background

We describe Xcerpt principles and the rationale for choosing it and demonstrate how such a declarative language and its environment need to be adapted for their deployment in a dynamic, mediated service context. Xcerpt is a query language designed for querying and transforming traditional XML and HTML data, as well as Semantic Web data in the form of RDF and OWL. One of the design principles is to strictly separate the matching part and the construction part in a transformation specification, see Fig. 3. Xcerpt follows a pattern-based approach to querying XML data.

```
CONSTRUCT
   CustomerArray [
      all Customer[
         nameAsContracted[var Name],
         companyId[var CompanyId],
         serviceOrganizationIdentifier[var OrgId],
         all supportidentifier[
            CustomerSupportIdentifier [var Code],
            ISOCountryCode [var CSI]
         ]
      ]
   ]
FROM
   arrayOfCustomer[[
      item [[
         orgName[var Name],
         companyId[var CompanyId],
         gcdbOrgId [var OrgId],
         countryCode[var Code],
         csiNumber[var CSI]
      ]]
   ]]
```

Fig. 3. Declarative Query and Transformation Specification of a Customer Array Element in Xcerpt. (© 2008, Claus Pahl. Used with permission).

Fig. 3 shows a transformation example for a customer array based on Fig. 1. The structure of this specification is based on a construction part (CONSTRUCT) and a source query part (FROM). An output customer in `CustomerArray` is constructed based on the elements of an item in an `arrayOfCustomer` by using a pattern matching approach, identifying relevant attributes in the source and referring to them in the constructed output through variables such as `Name` or `CompanyID`. During transformation, these hold the concrete values of the selected (matched) elements.

Xcerpt distinguishes two types of specifications:

- Goal-based query programs, identified by the keyword GOAL, are executable query programs that refer to input and output resources and that describe data extraction and construction.
- Abstract transformation rules, identified by the keyword CONSTRUCT as in Fig. 3, are function-like transformation specifications with no output resource associated.

Xcerpt extends the pattern-based approach, which is also used in other query and transformation languages, in following ways:

- Firstly, query patterns can be formulated as incomplete specifications in three dimensions. Incomplete query specifications can be represented in depth, which allows XML data to be selected at any arbitrary depth; in breadth, which allows querying neighbouring nodes by using wildcards, and in order. Incomplete query specifications allow patterns to be specified more flexibly without losing accuracy.
- Secondly, the simulation unification computes answer substitutions for the variables in the query pattern against underlying XML terms.

Xcerpt provides a runtime environment with an execution engine at its core (Schaffert, 2004). The central problem is to embed this type of environment, which can also be found for other query and transformation languages, into a dynamic, mediated service setting.

Connector Construction and Query Composition

We have adapted Xcerpt to support the construction of service connectors, i.e. executable query and transformation programs that integrate different data services:

- In order to promote modularity and code reuse, individual integration rules should not be designed to perform complex transformation tasks – rather a composition of individual rules is preferable. The composition of rules through rule chaining demand the query part of a service connector to be built ahead of the construction part.
- The data representation of the global data model changes as element names change or elements are being removed – these should not affect the query and integration part of the rules. Only an additional construction part is needed to enable versioning of the global data model.

Modularity and incomplete query specifications turn out to be essential features that are required from a query and transformation language in our context. In order to achieve the compositionality of modular rules, a layered approach shall be taken:

- Ground rules are responsible for populating XML data in the form of Xcerpt data terms by reading XML documents from individual service providers. These ground rules are tightly coupled to individual data Web services. These rules instruct the connector where to retrieve elements of data objects.
- The Xcerpt data terms are consumed subsequently by non-ground queries based on intermediate composite rules. These rules are responsible for integrating ground rules to render data types in the global XML schema. However, these rules still do not produce output.
- Finally, the composite rules are responsible for rendering the data objects defined in the interfaces of the mediator Web services based on customer requests. The composite rules are views on top of ground and intermediate representations according to the global schema. Therefore, the exported data from a mediator Web service is the goal of the corresponding connector (a query program).

Xcerpt is a document-centric language, designed to query and transform XML documents. Therefore, ground rules, which read individual data elements from the resources, are associated to at least one resource identifier. This is a bottom-up approach in terms of data population because data is assigned from the bottom level of the rules upward until it reaches the ultimate goal of a hierarchically structured rule. These rules are defined through an integration goal (the top-level query program) and structured into sub-rules down to ground rules.

These layered rules are saved in a repository. When needed, a rule will be picked and a backward rule chaining technique for rule composition enables data objects to be populated to answer transformation requests. Rule chaining means that resulting variable bindings from a transformation rule that is used within a query program are chained with those of the query program itself. Rule chaining is used to build recursive query programs. Consistent connectors can then be constructed on the fly based on input data such as the data services and the layered rules.

```
GOAL
   Out { Resource {“file:SupportIdentifier_Customer.xml”},
         SupportIdentifier [ All var SupportIdentifier ] }
FROM
   Var SupportIdentifier -> SupportIdentifier {{}}
END

CONSTRUCT
   SupportIdentifier [var Code, optional Var Cname, Var Code]
FROM
in { Resource {“file:customer1.xml”},
     ArrayOfCustomer [[
         customer [[ optional countryName [var CName],
                     couuntryCode [var Code]
                     csiNumber [var CSI]  ]]  }
END

CONSTRUCT
   SupportIdentifier [var Code, Var Cname, optional Var Code]
FROM
in { Resource {“file:customer2.xml”},
     Customers [[ customer [[
                     countryName [var CName],
                     optional couuntryCode [var Code]
                     csiNumber [var CSI]  ]] }
END
```

Fig. 4. Transformation Specification in Xcerpt based on Goal Chaining with one Goal-based Query Program and two supporting Transformation Rules. (© 2008, Claus Pahl. Used with permission).

We apply backward goal-based rule chaining to execute complex queries based on composite rules. Fig. 4 shows an example of this pattern matching approach that separates a possibly partial query into resource and construction parts. The transformation rule maps the `supportIdentifier` element of the customer example from Fig. 1. Fig. 4 is a composite rule based on the `SupportIdentifier` construction rule at a lower level. Fig. 5 demonstrates the transformation that produces the resulting XML data for the `Customer` service. The output from the `Customer` mediator represents a customer as identified in a servicing system. In the example, rule `CustomerArray` is a composite rule, based on the `Customer` and `Service` rules, that could be used to answer a user query directly. The resource identifiers in form of variables and the interfaces for the data representation will be supplied to the connector generator. Rule mappings in the connector generator determine which queries are constructed from the repository for execution.

Rule 1: This rule produces the CustomerArray by grouping and reconstructing.

```
CONSTRUCT
   CustomerArray [[
      all var customer,
      all var supportidentifier,
      all var services [[
         var customerName,
         all var system [[ var systemId,  all var machine ]]
      ]]
   ]]
FROM
   Customer [[ var customer, var supportidentifier ]]
AND
   Service [[var services [[ var system [[ var machine]] ]] ]]
```

Rule 2a: This rule gets Customer data terms according to the global data model.

```
CONSTRUCT
   Customer[[ var customer, all var supportidentifier ]]
FROM
   arrayOfCustomer[[ var customer, var supportidentifier ]]
```

Rule 2b: This rule gets Service data terms according to the global data model.

```
CONSTRUCT
   Service [[ var service [[ var system [[ var machine]] ]] ]]
FROM
   arrayOfService [[
      var service [[ var system[[ var systemId ]] ]]
   ]]
AND
   Machine [[ var machine, var systemId ]];
```

Rule 3: This construct rule gets Machine data terms.

```
CONSTRUCT
   Machines [[
      all machine-of-system [[var machine]],
      var systemId
   ]]
FROM
   machineItem [[ var machine, var systemId ]]
```

Fig. 5. The Composite Rules for Customer Transformation in Xcerpt. (© 2008, Claus Pahl. Used with permission).

THE MEDIATED SERVICE INTEGRATION ARCHITECTURE

We propose a mediated service-based architecture for the integration of XML data in Web service-based information systems. The major aims of the proposed mediated software architecture for the integration and mediation of XML data in the context of Web services are threefold: improved modifiability through declarative rule-based query programs, improved reusability of declarative integration rules through automated connector construction, and improved flexibility through dynamic generation of consistent, i.e. semantics-preserving connectors.

Service-based Mediator Architectures

A declarative, rule-based approach can be applied to the data transformation problem (Orriens et al., 2003, Peltier et al., 2002). The difficulty lies in embedding a declarative transformation approach into a service-based architecture in which clients, mediators, and data provider services are composed (Garcia-Molina et al., 1997). A data integration engine can be built in the Web service business process execution language WS-BPEL. In (Rosenberg and Dustdar, 2005), a business rule engine-based approach is introduced to separate the business logic from the executable WS-BPEL process, which demonstrates that one of our objectives can be achieved (Rouvellou et al., 2000). These rules, stored in a repository, can be used to dynamically create executable query and transformation programs using a consistency-guaranteeing connector or integration service as the mediator. These integration services are the cornerstones of a mediator architecture that processes composite client queries that possibly involve different data sources provided by different Web services (Wiederhold, 1992). Mediators in an architecture harmonise and present the information available in heterogeneous data sources (Stern and Davies, 2003). This harmonisation comes in the form of an identification of semantic similarities in data while masking their syntactic differences. Figs. 1 and 2 have illustrated an example whose foundations we have defined in terms of an ontology in order to guarantee consistency for transformations.

Zhu et al. (2004) and Widom (1995) argue that traditional data integration approaches such as federated schema systems and data warehouses fail to meet the requirements of constantly changing and adaptive environments. With the support of Web service technology, however, it is possible to encapsulate integration logic in a separate component as a mediator Web service between heterogeneous data service providers and consumers. Therefore, we build a connector construction component as a separate integration service, based on (Szyperski, 2002; Haller et al. 2005, Zhu et al., 2004, Rosenberg and Dustdar 2005). We develop an architecture where broker or mediator functionality is provided by a connector generator and a transformation engine:

- The connector construction is responsible for providing connectors based on transformation rules to integrate and mediate XML documents. The connector construction generates, based on schema information and transformation rules, an executable service process that gathers information from the required resources and generates a query/transformation program that compiles and translates the incoming data into the required output format.
- The process execution engine is responsible for the integration of XML data and mediation between clients, data providers and the connector component. The execution engine is implemented in WS-BPEL and shall access the Xcerpt runtime engine, which executes the generated query/transformation program.

The connector construction component is responsible for converting the client query, dynamically create a transformation program based on stored declarative transformation rules, and to pass all XML data and programs to the execution engine. The system architecture is explained in Fig. 6 with a few sample information services from an application service provider scenario – Customer Data, E-business System, and Request Analysis Service.

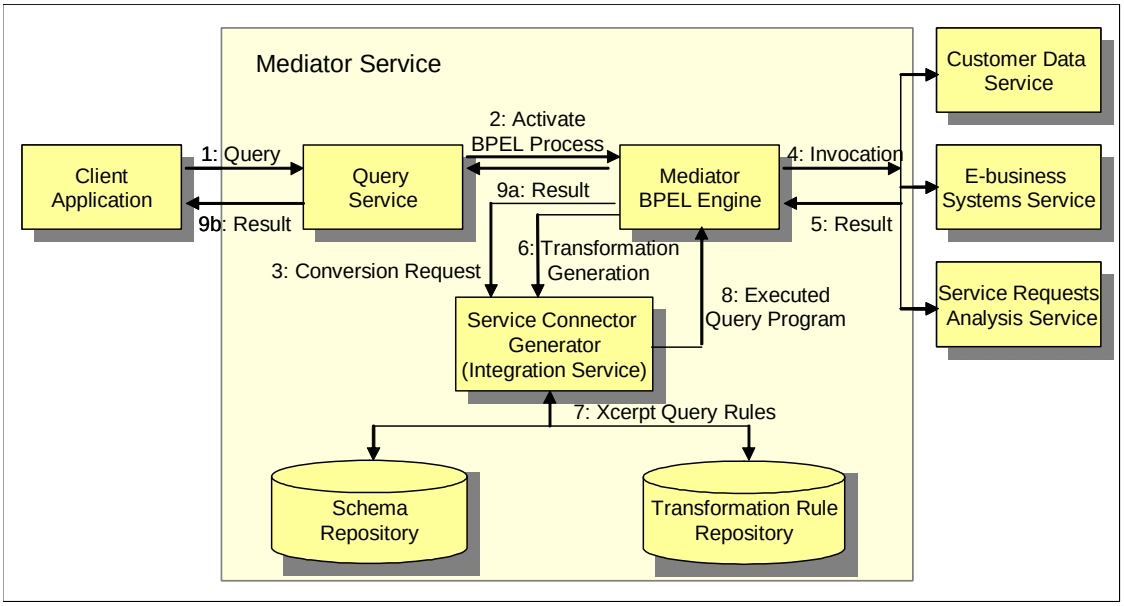

Fig. 6. Component View of a Mediator Web Service with Interactions. (© 2008, Claus Pahl. Used with permission).

Exposing data sources as services is only the first step towards building a SOA solution. Without a service integrator, the data user needs to understand each of the data models and relationships of service providers. The mediator architecture has the following components:

- Query service. The query service is responsible for handling inbound requests from the application consumer side and transferring outbound results back. The WS-BPEL process engine handles the internal messaging of the architecture. The query service decomposes input messages into a set of pre-defined WS-BPEL processes.
- Mediator (BPEL) engine. A mediator engine is itself a WS-BPEL process. Mediators deliver data according to a global schema. The schema may consist of various data entities for large enterprise integration solutions.
- Connector generation service. This component is responsible for generating connectors for transforming messages both entering the WS-BPEL engine from service clients and leaving the WS-BPEL engine from data provider services according to the global data model.

The active components, provided as information services, are complemented by two repositories:

- Transformation rule repository. The repository allows the reuse of rules and can support multiple versions of service providers and mediator services.
- Schema repository. The repository stores the WSDL metadata and the XML schema information for the Web service providers and the mediator Web service. The schema information is used to validate the XML documents at runtime before they are integrated and returned to the client applications.

Connector Generation

The construction of a service connector means to generate an executable Xcerpt query program by composing each Xcerpt query with the corresponding transformation rules. In an Xcerpt

query program, there is only one goal query, which will be processed first. The goal query is made up of composite transformations rules that in turn are made up of ground rules that read XML data from external resources. The process begins by expanding each composite query according to the definitional data mappings that are stored in a rule repository. The rule chaining mechanism in Xcerpt needs the goal query and all supporting queries in one query program at runtime.

The Xcerpt runtime engine reads XML-based resources and populates them into data terms before the query terms can start to evaluate them. The drawback is that all resources identifiers have to be specified inside a query program rather than be passed into a query program as parameters. Consequently, we adapted the Xcerpt approach to processing transformation requests in an information integration solution. The resource identifiers are not hard-coded in ground rules in our setting in order to achieve the desired loose coupling to achieve flexibility and reusability. These resource identifiers are invisible to the connector construction service. Xcerpt does not support automatic query program construction by default, although it provides the necessary backward rule chaining technique to evaluate a chain of queries.

We have developed a wrapper mechanism to pass the resource identifiers from the goal level down to the ground rules. Therefore, as an extension to the original Xcerpt approach, a mediator-based data integration architecture is needed where the rules are decoupled from the resources and the only the generated Xcerpt-based connectors are integrated with the client and provider Web services. WS-BPEL code that coordinates the mediation and transformation process is generated by a connector generator for transformations within the mediator service.

FUTURE TRENDS

Integration has currently been investigated from a static perspective looking at existing systems integration. We discuss emerging needs to address this as part of software evolution and legacy systems integration. Another current trend is the increasing utilisation of semantic enhancements, such as ontologies and reasoning frameworks, to support integration. We address briefly attempts of using service ontologies, which would complement the presented ontology-based information architecture.

Re-engineering and the integration of legacy systems is an aspect that goes beyond the integration context we described – although the application service provider (ASP) context is a typical example of a field where ASPs currently convert their systems into service-based architectures (Seltsikas and Currie, 2002). The introduction of data transformation techniques for re-engineering activities can improve the process of re-engineering legacy systems and adopting service-oriented architecture to manage the information technology services (Zhang and Yang, 2004). Business rules often change rapidly – requiring the integration of legacy systems to deliver a new service. How to handle the information integration in the context of service management has not yet been explored in sufficient detail in the context of transformation and re-engineering.

The utilisation of the semantic knowledge that is available to represent the services that make up the mediator architecture is another promising direction that would increase flexibility in terms

of dynamic composition. The functionality and quality attributes of Web services can be in terms of one of the widely known service ontologies such as OWL-S or WSMO (Payne and Lassila, 2004). Abstract service descriptions can be derived from the semantic properties of the data they provide, process, or consume. Some progress has been made with respect to semantics-based service discovery and composition; the interplay between semantic data integeration and semantic service integration needs a deeper investigation. Karastoyanova et al. (2007), for instance, discuss a middleware architecture to support semantic data mediation based on semantically annotated services. Their investigation demonstrates how your semantic data mediation can be incorporated into a service-based middleware architecture that supports SOA-based development. However, the need to have an overarching semantic information architecture also becomes apparent, which supports our results.

CONCLUSIONS

The benefit of information systems on demand must be supported by corresponding information management services. Many application service providers are currently modifying their technical infrastructures to manage and integrate information using a Web services-based approach. However, the question of handling information integration in a flexible and modifiable way in the context of service-based information systems has not yet been fully explored.

The presented framework utilises semantic information integration technologies for XML data in service-oriented software architectures. The crucial solutions for the information integration problem are drawn from mediated architectures and data model transformation, allowing the XML data from local schemas to be consistently transformed, merged and adapted according to declarative, rule-based integration schemas for dynamic and heterogeneous environments. We have proposed a declarative style of transformation based on a semantic, ontology-based data model, with implicit source model traversal and target object creation. The development of a flexible mediator service is crucial for the success of the service-based information systems architecture from the deployment point of view. Our solution based on the query and transformation language Xcerpt is meant to provide a template for other similar languages. One of our central objectives was to introduce an integration solution from a technical perspective.

A number of extensions of our approach would strongly benefit its flexibility. Essentially, we plan to address the trends outlined in the previous section. Systems evolution and legacy system integration shall be addressed through a more transformation systems-oriented perspective on integration. We are also working on an integration of service ontologies and general data-oriented domain ontologies for service-oriented architectures.

REFERENCES

Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web Services – Concepts, Architectures and Applications*. Berlin, Germany: Springer Verlag.

Barrett, R., Patcas, L., Murphy, J., & Pahl, C. (2006). Model driven distribution pattern design for dynamic web service compositions. In: *International Conference on Web Engineering ICWE'06*, 10-11 Jul 2006, Palo Alto, US.

Bry, F., & Schaffert, S. (2002). Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. *Proceedings Intl. Conference on Logic Programming. LNCS 2401*, (pp. 255-270). Heidelberg, Gerrnany: Springer-Verlag.

Daconta, M.C., Obrst, L.J., & Smith, K.T. (2003). *The Semantic Web – a Guide to the Future of XML, Web Services, and Knowledge Management*. Indianapolis, USA: Wiley & Sons.

Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, Y. D., Vassalos, V., & Widom, J. (1997). The TSIMMIS approach to mediation: Data models and languages. *Journal of Intelligent Information Systems. 8(2)*, 117-132.

Haller, A., Cimpian, E., Mocan, A., Oren, E., & Bussler, C. (2005). WSMX - a semantic service-oriented architecture. *Proceedings Intl. Conference on Web Services ICWS 200 5,* (pp. 321-328).

Jhingran, A.D., Mattos, D., & Pirahesh, N.H. (2002). Information Integration: A research agenda. *IBM System Journal*, 41(4) , 55-62.

Karastoyanova, D., Wetzstein, B., van Lessen, T., Wutke, D., Nitzsche, J., & Leymann, F. (2007). Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. *Proceedings of the Second International Workshop on Service Engineering SEIW 2007*, (pp. 347-354).

Lenzerini, M. (2002). Data integration: A theoretical perspective. *Proceedings Principles of Database Systems Conference PODS'02*, (pp. 233-246).

Orriens, B., Yang, J., & Papazoglou, M. (2003). A Framework for Business Rule Driven Web Service Composition. Jeusfeld, M.A. & Pastor, O. (Eds). *Proceedings ER'2003 Workshops, LNCS 2814*, (pp. 52-64). Heidelberg, Germany: Springer-Verlag.

Pahl, C. (2002). A Formal Composition and Interaction Model for a Web Component Platform. *ICALP'2002 Workshop on Formal Methods and Component Interaction. Elsevier Electronic Notes on Computer Science ENTCS*, Vol. 66, No. 4.

Pahl, C. and Zhu, Y. (2006). A Semantical Framework for the Orchestration and Choreography of Web Services. *Proceedings of the International Workshop on Web Languages and Formal Methods (WLFM 2005). Electronic Notes in Theoretical Computer Science* 151(2):3-18.

Pahl, C., Giesecke, S., & Hasselbring, W. (2007). An ontology-based approach for modelling architectural styles. In: *The European Conference on Software Architecture ECSA'2007*. 24-26 Sept 2007, Madrid, Spain.

Payne, T. and Lassila, O. (2004). Semantic Web Services. *IEEE Intelligent Systems*, 19(4), 14-15.

Peltier, M., Bezivin, J, & Guillaume, G. (2001). MTRANS: A general framework, based on XSLT, for model transformations. *Proceedings of the Workshop on Transformations in UML WTUML'01*. Retrieved 21 July 2008 from: http://citeseer.ist.psu.edu/581336.html

Reynaud, C., Sirot, J.P., & Vodislav, D. (2001). Semantic Integration of XML Heterogeneous Data Sources. *Proceedings IDEAS Conference 2001*, (pp. 199–208).

Rosenberg, F., & Dustdar, S. (2005). Business Rules Integration in BPEL - A Service-Oriented Approach. *Proceedings 7th International IEEE Conference on E-Commerce Technology*, (pp. 476- 479).

Rouvellou, I., Degenaro, L., Rasmus, K., Ehnebuske, D., & McKee, B. (2000). Extending business objects with business rules. *Proceedings 33rd Intl. Conference on Technology of Object-Oriented Languages*, (pp. 238-249).

Schaffert, S. (2004). *Xcerpt: A Rule-Based Query and Transformation Language for the Web*. PhD Thesis, University of Munich.

Seltsikas, P., & Currie, W.L. (2002). Evaluating the application service provider (ASP) business model: the challenge of integration. *Proceedings 35th Annual Hawaii International Conference 2002*. 2801 – 2809.

Stal, M. (2002). Web Services: Beyond Component-based Computing. *Communications of the ACM, 45 (10)*, 71-76.

Stern, A., & Davis, J. (2004). Extending the Web services model to IT services. *Proceedings IEEE International Conference on Web Services*, (pp. 824-825).

Szyperski, C. (2002). *Component Software: Beyond Object-Oriented Programming – 2nd Ed*. New York, USA: Addison-Wesley.

W3C – the World Wide Web Consortium. (2004). *The Semantic Web Initiative*. Retrieved March 9, 2008 from http://www.w3.org/2001/sw.

Widom, J. (1995). Research problems in data warehousing. *Proceedings of 4th International Conference on Information and Knowledge Management*, (pp. 25-30).

Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer, 25*, 38-49.

Zhang, Z., & Yang, H. (2004). Incubating Services in Legacy Systems for Architectural Migration. *Proceedings 11th Asia-Pacific Software Engineering Conference APSEC'04*, (pp. 196-203).

Zhu, Y. (2007). *Declarative Rule-based Integration and Mediation for XML Data in Web Service-based Software Architectures*. M.Sc. Thesis. Dublin City University.

Zhu, F., Turner, M., Kotsiopoulos, I., Bennett, K., Russell, M., Budgen, D., Brereton, P., Keane, J., Layzell, P., Rigby, M., & Xu, J. (2004). Dynamic Data Integration Using Web Services. *Proceedings 2nd International Conference on Web Services ICWS'2004*, (pp. 262-269).