Computing and Informatics, Vol. 31, 2012, 1129–1149

DATA INTEGRATION IN MEDIATED SERVICE COMPOSITIONS

Claus Pahl, Yaoling Zhu

Dublin City University
School of Computing
Dublin 9
Ireland
e-mail: claus.pahl@dcu.ie, yao.zhu3@mail.dcu.ie

Communicated by Ondrej Habala

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 and Cloud 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. Automation is needed to enable dynamic integration and composition. Modifiability is another aim here that benefits from consistency and modularity.

Keywords: Data integration, service architecture, cloud integration, semantic data modelling, mediated architecture, declarative data transformation

1 INTRODUCTION

A major aim of the Web service platform is the integration of existing software and information systems [1]. Information and data integration is a central aspect here [26]. Traditional techniques based on XML for data representation and XSLT for transformations between XML documents are, however, not sufficient to provide flexible and automatable integration solutions 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 service-oriented architecture (SOA) as an architectural style and recently cloud computing require a unified way to expose the data and functionality of an information system [1, 23, 3]. The Web services platform has the potential to solve the problems in the service and cloud data integration domain such as heterogeneity and interoperability [12, 8, 28]. Our contribution is an integration 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 [22].

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. Abstraction has been used successfully to address flexibility problems in data processing [17]. Declarative XML-based data query and transformation languages [28] and Semantic Web and ontology technology [5] provide the basis for our approach. 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,
- 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 the declarative data query and transformation language Xcerpt [4] and its execution environment can be combined in order to allow dynamic data transformation and integration in service-based systems. 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. Automation is critical to enable dynamic integration. We analyse the maintainability of this solution.

2 DATA INTEGRATION AND TRANSFORMATION

Information integration is the problem of combining heterogeneous data residing at different sources in order to provide the user with a unified view [11, 26]. It is central in any attempt to adapt services and their underlying data sources to specific client and provider needs in the context of dynamic service composition. An important task is the definition of mappings between individual data sources and a unified view of these sources. Figure 1 shows two schemas representing the views of client and provider on a collection of customers. The integration of these hierarchically structured schemas can be defined using transformation languages.

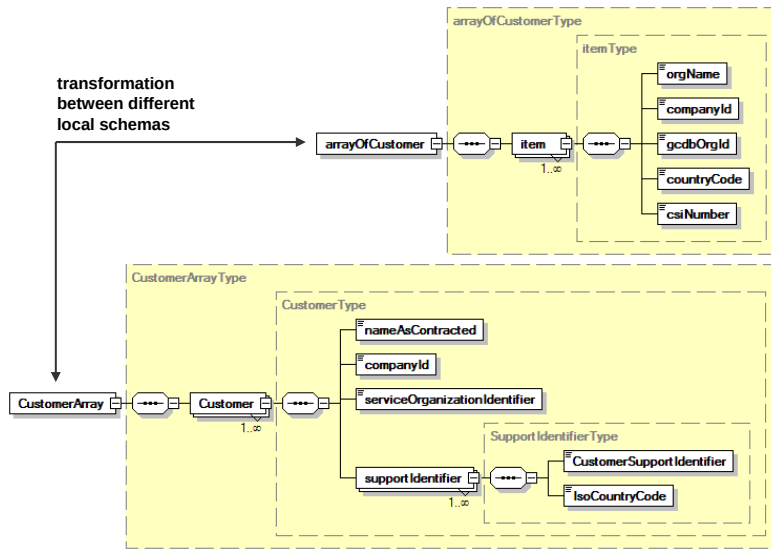

Fig. 1. Two Schema Diagrams of the Global Data Model that need to be integrated

A case study shall clarify current limitations. The Application Service Provider (ASP) business model promotes the use of software as a service SaaS in cloud computing [19]. An example of this model is information systems outsourcing, i.e. the handing over management of an enterprises information infrastructure to a third party cloud service provider. The ASP takes responsibility for managing the software application on its own infrastructure. The ASP maintains the application and ensures that system functionality and data are available when requested.

The ASP context is the environment in which we have developed our solution and evaluated its characteristics. Our evaluation is based on a joint project with a major ASP [27], who provides standard ERP solutions to hundreds of customers in Europe and Asia. Recently, a service-based platform has been implemented by the company to provide an ASP integration solution. The case study is part of the ASP's Customer Intelligence Framework (CIF). CIF provides portal-based ac-

cess for customers and managers to a reporting system for the ASP's on-demand services. The reporting system consists of analyser objects that access content in the form of requests logs, outage tracking data and customer life cycle information. The aim is to support ad-hoc, cross-content adaptable user queries. Change and evolution problems are omnipresent for this company due to a large number of ASP clients, the variety of services offered by the ASP and the range of data associated to services. Maintainability is consequently a particular focus of our investigation. Only maintainability can provide a long-term cost-effective integration solution.

XSLT is the most widely used XML data integration language, which is also the basis of an existing system in the ASP of our case study. However, an XSLT solution suffers from some limitations within our context due to its 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 cannot be guaranteed.

Modularity: XSLT does not support a join or composition operator on XML documents that allows several source XML documents to be merged into one before being transformed, i.e. does not adequately support complex data structures and their integration.

Maintainability: XSLT transformations are difficult to write, maintain and reuse for large-scale information integration. It is difficult to separate source and target parts of transformation rules as well as the filtering constraints due to its operational character without separation of query and construction concerns.

Due to these drawbacks, we propose a two-pronged approach consisting of 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 [9, 11, 16], led us to choose the fully declarative language Xcerpt [4] as our transformation platform [27].

Recently, service platforms are used to provide integration solutions. In the Web services context, data in XML representation, which is retrieved from individual data services, needs to be merged and transformed. Data schema integration cannot be fully automated on a syntactic level since the syntactic representation of data schemas does not convey the semantics of different data sources. For instance, a customer can be identified by a unique customer identifier; or, the same customer may be identified by a combination of a service support identifier and its geographical location (see Figure 1). Ontology-based semantic data models can rectify this problem by providing an agreed vocabulary of concepts with associated properties.

3 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 [5] 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 [8]. We define consistency here as the preservation of semantics in transformations.

We introduce here the semantic information architecture that defines and integrates the individual schemas in use (Section 3.1). We show how transformation rules between the schemas can be automatically constructed (Section 3.2) in terms of the transformation language Xcerpt (Section 3.3). Connectors, i.e. implemented transformations that connect to data sources, are then addressed in Section 3.4.

3.1 Information Architecture

Ontologies are knowledge representation frameworks that represent a domain in terms of concepts and their properties. We use a simplified OWL syntax to avoid the verbosity of XML-OWL [5]. It provides us with a concise notation 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 name or identification and also object type properties such as services used by a customer. Three concept descriptions using the existential quantifier exist here, and 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  = exits supportID . Identification     and
            exits custName . Name                and
            exits usedServices . Service

Service   =  exists custID . ID                  and
             exists servSystem . System

System    =   exists 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 Figure 2. For practical reasons a corresponding semantically equivalent XML representation to our notation is needed. The corresponding global XML schema representation for the customer element is as follows:

```
<!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.

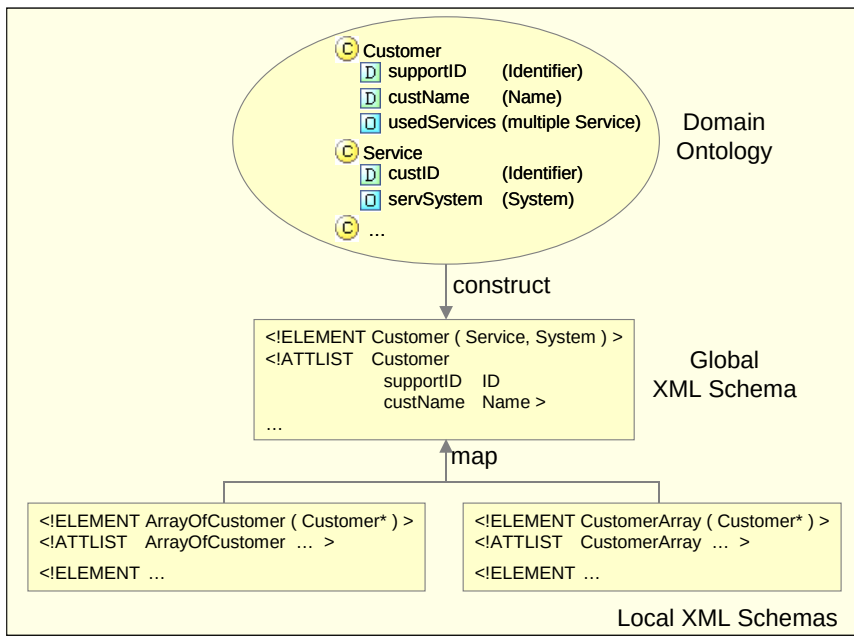

Fig. 2. Information architecture overview

3.2 Transformation Rule Construction

The ontology provides a semantically defined global data model from which transformations between different data representations can be derived. This construction has a number of specific requirements regarding transformation rules:

- modularity of transformation rules, i.e. rules specific to particular data elements of the information architecture that can be combined according to the structure of data, to support flexible rule generation and configuration,
- consistency 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:

- Define one transformation rule per concept in the ontology that is represented in the target data model.
- Identify semantically equivalent concepts of the selected source model concepts.
- For each identified concept:

  a) determine required attributes – these are end nodes of the ontological structure – and
  b) copy semantically equivalent counterparts from the source model.

This illustrates that rule generation automation is possible. A necessary prerequisite is that all concepts of the source model are actually supported by the transformation rules, mapping each to the target data model. The taxonomic hierarchy that the ontology defines is the basis of modular rule definitions – there is a rule for each concept in the taxonomic hierarchy. Consistency of rules is demonstrated using (ontology-proven) semantically equivalent concepts.

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 the data integration problem from Figure 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. 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.

3.3 Xcerpt Background

We describe Xcerpt principles and the rationale for choosing it and demonstrate how such a declarative language needs to be adapted for deployment in a dynamic, mediated service environment. 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. It separates the matching part and the construction part in a transformation specification, see Figure 3. Xcerpt follows a pattern-based approach to querying XML data. The Xcerpt platform includes a runtime environment with an execution engine at its core [18].

```
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 transformation specification of a customer array element in Xcerpt

Figure 3 shows a transformation example for a customer array based on Figure 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 CONSTRUCT as in Figure 3, are function-like transformation specifications with no output resource associated.

Xcerpt extends classical pattern-based matching: [..] is used for ordered exact sub-term patterns and [[..]] for ordered partial sub-term patterns, possibly part of a quantified expression (all/exists). 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 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.

3.4 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 demands 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 linked 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).

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. This modular approach allows us to deal with arbitrarily complex data structures. Modularity entails maintainability and scalability as well.

```
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],
                     countryCode [var Code]
                     csiNumber [var CSI]  ]]  }
END
```

Fig. 4. Transformation specification based on goal chaining with goal-based query program

We apply backward goal-based rule chaining to execute complex queries based on composite rules. Figure 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 Figure 1. Figure 4 is a composite rule based on the SupportIdentifier construction rule at a lower level. Figure 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 Customer and Service, that could be used to answer a user query directly. The resource identifiers in the 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.

4 THE MEDIATED SERVICE INTEGRATION ARCHITECTURE

We propose a mediated service-based architecture for the transformation of XML data in Web service-based information systems. The major aims of the proposed

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

architecture for the integration and mediation of XML data in the context of Web services are threefold: improved modifiability through declarative and modular 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. After introducing the mediator architecture in Section 4.1, we will demonstrate how the transformation rule generation from the previous section and execution techniques can be integrated into composed Web service processes through a mediated approach. In Section 4.2, we show how connectors are generated in this architectural setting.

4.1 Service-based Mediator Architectures

A declarative, rule-based approach can be applied to the data transformation problem [12, 16]. The difficulty lies in embedding a declarative transformation approach into a service-based architecture in which clients, mediators, and data provider ser-

vices are composed [7, 14]. The 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. Mediators in an architecture harmonise and present the information available in heterogeneous data sources [21]. This harmonisation comes in the form of an identification of semantic similarities in data while masking their syntactic differences. Figures 1 and 2 have illustrated an example defined in terms of an ontology in order to guarantee transformation consistency.

Zhu et al. [28] and Widom [24] argue that traditional data integration architectures 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 [8, 28]. We develop an architecture where broker-style 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 is part of the Integration Service and which executes the generated query/transformation program.

The system architecture is illustrated in Figure 6 with a few sample information services from an application service provider (ASP) scenario – Customer Data, E-business System, and Request Analysis services.

Exposing data sources as services is only the first step towards building a SOA solution. Without a service integrator, the client 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 implements the mediator process that handles the internal messaging of the architecture. The query service decomposes input messages into a set of pre-defined WS-BPEL processes.

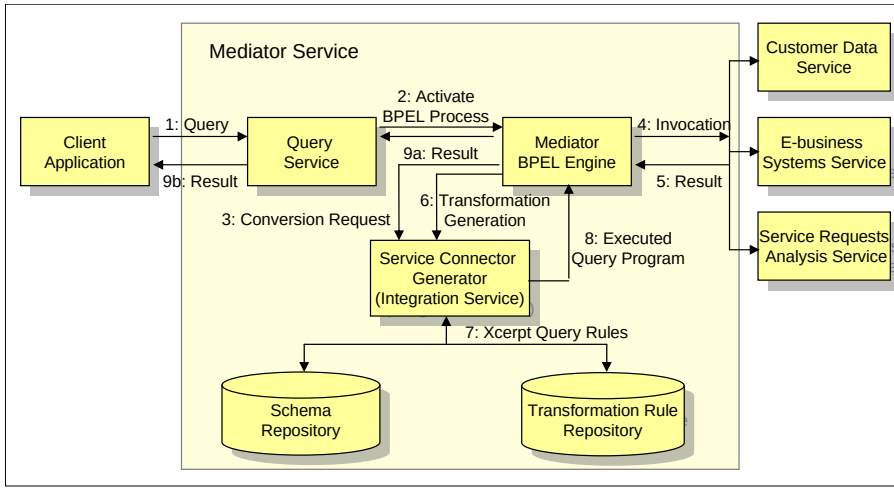

Fig. 6. Component view of a mediator service with interactions

Mediator (BPEL) engine. A mediator engine is itself a WS-BPEL process. The mediator delivers data according to the 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 Web 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 application.

In our ASP scenario, the data sources are under the control of the ASP which runs the mediator architecture, but advanced solutions where semantic matching is used to incorporate external sources have also been investigated [5].

4.2 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 retrieved using the data services. The process begins by expanding each composite query according to the definitional data mappings stored in the rule repository. The rule chaining mechanism in Xcerpt converts the goal query and all supporting queries into 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 resource 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 in our solution are not hard-coded in ground rules 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. This extension is needed where rules are decoupled from 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.

5 EVALUATION

We now provide a qualitative evaluation of our techniques, analysed using a prototype we developed in conjunction with a major application service provider in the context of its on-demand service infrastructure.

5.1 Evaluation Aims

Consistency and modularity are the two central aspects of our investigation. The effectiveness of the proposed integration architecture in terms of change and evolution requirements shall be evaluated. Maintainability shall be evaluated using an analytic evaluation method. Change scenarios are defined and used to elicit and evaluate the modifiability goal. We also looked at scalability, which shall be evaluated based on the same architectural analysis method as maintainability, since scalability and maintainability are related changeability aspects. Consistency and modularity are expected to be central factors in achieving the desired degree of maintainability and scalability. However, the consequences of such a solution on other quality attributes shall also be empirically evaluated. We analyse the performance of the developed prototype and compare the prototype to the existing solution.

5.2 Modifiability Analysis and Evaluation Scenarios

The Architecture-Level Modifiability Analysis (ALMA) provides a framework to evaluate the maintainability and scalability of software application architectures [2]. Scalability is an important requirement from a service-level integration perspective. Part of the current software development and maintenance process at the ASP are regular internal assessments by in-house software architects and inspections on the software architecture performed by external experts. The definition of realistic change scenarios as part of this process is crucial in this context. These activities have led to the definition of three change scenarios for the modifiability evaluation focussing on changes in specific aspects:

Scenario 1 [Business rules]. Clients often change the services requested from the ASP, which requires changes to integration rules at the mediator level. Business rules change more often than the data model.

Scenario 2 [Data source providers]. Structural changes in the data provider service architecture lead to changes in the communication at element level. Two scenarios may happen in terms of changes of ground rules – source or target schema changes. One is that source attributes of a data transformations from one Web service provider to another change; the other is that the name of attributes of a data object in the unified data model changes or the ones in the data model of a Web service provider change.

Scenario 3 [Integration rules]. Caused by data model changes, new integration rules might need to be added. This scenario might happen on two occasions: one is that customers request new mediator Web services, the other is that a new Web service provider is integrated. In any case, it will introduce new integration rules at the top level of the global data model.

These scenarios have been defined in conjunction with software architects and analysts at the ASP. These address maintainability primarily, but as the addition of new services or rules indicates, also refer to scalability.

5.3 Comparison Results

To demonstrate effectiveness in terms of modifiability, we compare our solution with a traditional solution. We have evaluated our application service provider (ASP) solution after the release of a first prototype implementation of our architecture and have compared it with the existing traditional XSLT-based and ad-hoc WS-BPEL-based solution that has been already in place at the ASP – see Section 2.

Existing system: The architecture of the existing system is based on a mediator service at the core, which is activated by a query service and which invokes the XSLT transformation program, which in turn accesses a schema repository.

Solution prototype: The architecture of the prototype was presented in Figure 6. It differs from the traditional one in that the mediator functionality is split over several individual services over two layers.

The ALMA impact analysis identifies if an architectural element is affected by a change scenario directly or indirectly. The investigation of the three scenarios has led to the following observations based on a comparative and quantitative analysis for the existing system and the prototype:

Scenario 1 [Business rules]. A problem that traditional data transformation languages such as XSLT have is that a set of business rules to render unified entities in the global data model are intertwined, although the business rules are separated from the application logic. Our approach is to have a number of targeted mediator processes rather than only one to support the unified virtual information view. This supports an incremental definition as more mediator Web services can be built to answer users queries. The unified virtual view is subject to changes and additions as the analysis of the information sources proceeds.

- Effect on existing architecture: transformations;
- Effect on proposed integration architecture: composite rules;
- How achieved: automation through connector construction at runtime.

In our architecture, we have implemented a declarative rule-based approach in which rules are represented in Xcerpt and saved separately in a rule repository. The business rules are composed at runtime during the construction of the service connectors by the connector generator. Therefore, changes of business rules do not affect the rest of the business and application logic. In contrast, the query and the construct part of the XSLT transformation queries are tightly coupled. Therefore, it is difficult to automate the construction process of a transformation file in the traditional solution.

Scenario 2 [Data source providers]. This relates to changes of a data term when one of the source XML documents is replaced by a new source.

- Effect on existing architecture: transformations and architecture;
- Effect on proposed integration architecture: ground rules, maybe some immediate rules;
- How achieved: modularity, since query part and construct part of an integration rule are separated.

Data term changes only affect the population of one data term in our case. In case of name changes of attributes on both sides, the data terms remain untouched, i.e. the query terms do not need to be changed. The only element to change is the construct term of a business rule. If one of the data terms at the lower level changes, then we only need to update the directly referenced query term. The construct term in an integration rule is not affected. In order

to handle the same scenario for XSLT, another new version of the entire XSLT transformation file needs to be created.

Scenario 3 [Integration rules]. In the third change scenario, the immediate composite rules can be leveraged from the existing ones since it is likely that the new Web service provider shares some common entities.

- Effect on existing architecture: transformations and architecture;
- Effect on proposed integration architecture: new version of composite rules, or reuse or addition of ground and immediate rules;
- How achieved: integration rule repository and independent data services: the connector generator injects no code into the integration flow.

This scenario demonstrates that change of the Xcerpt integration rules can occur in the following two scenarios: one is to build a new mediator service; the other is to build a new version of the mediator process.

5.4 Discussion

The impact resulting from each change scenario is only local in our proposed solution, whereas in traditional solutions the entire transformation set-up including the software architecture can be affected. The declarativity and modularity of the transformation rules and the separation of architectural concerns such as connector generation and execution (which is Xcerpt-specific) from the mediation process as such (which is Xcerpt-independent) into different architectural layers are the contributors to a maintainable solution in our case. Consistency, achieved here through abstract integration models and rules, is crucial for a software architect to control changes locally. The semantic information architecture simplifies changes at the local level. The key benefit of guaranteed consistency is, however, increased automation. The success factors for maintainability and scalability that emerge are modularity and consistency based on an abstract, declarative approach.

The achievements in terms of maintainability, as demonstrated through ALMA, but also limitations and drawbacks have also been looked at using empirical methods. There is often a trade-off between maintainability and performance in software systems. Two factors inevitably decrease performance in our solution: firstly, levels of indirection as part of the architectural separation of mediation, connector generation and transformation, and, secondly, dynamic connector generation based on rules and schemas stored in the respective repositories. Our prototype has, however, demonstrated that these two factors together in general do not exceed 15–20 % of the overall transformation time compared to the traditional architecture, which is acceptable in most ASP situations.

6 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, often Cloud services-based approach. However, the question of handling information integration in a flexible, automated 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-based software architectures (SaaS). 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.

Our contribution deals with complex data structures and still provides maintainability and scalability through modularity – using automatically generated layered rules guided by the taxonomic hierarchy of the underlying ontology. We have demonstrated how to seamlessly integrate this semantic data transformation technique into a Web service architecture as a mediator service. Our investigation is based on our experience with infrastructures of a multinational on-demand service provide.

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 [25]. 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 active service management has not yet been explored in sufficient detail in the context of legacy integration and re-engineering. Our solution provides a contribution towards the evolutionary perspective of software development and integration.

A possible extension of our approach is the utilisation of the semantic knowledge that is available to represent all services involved in their functionality as semantic Web services [13]. Abstract service descriptions can be derived from the semantic properties of the data they provide, process, or consume. Karastoyanova et al. [10], for instance, discuss a middleware architecture to support semantic data mediation based on semantically annotated services. Their investigation demonstrates how our 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.

REFERENCES

[1] Alonso, G.—Casati, F.—Kuno, H.—Machiraju, V.: Web Services – Concepts, Architectures and Applications. Springer Verlag 2004.

[2] Bengtsson, P.—Lassing, N.—Bosch, J.—Vliet, H.: Architecture-Level Modifiability Analysis (ALMA). Journal of Systems and Software, Vol. 69,, 2004, No. 1, pp. 129–147.

[3] Buyya, R.—Broberg, J.—Goscinski, A.: Cloud Computing – Principles and Paradigms. Wiley 2011.

[4] Bry, F.—Schaffert, S.: Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. Proceedings Intl. Conference on Logic Programming, LNCS 2401, Springer-Verlag 2002.

[5] Daconta, M. C.—Obrst, L. J.—Smith, K. T.: The Semantic Web – A Guide to the Future of XML, Web Services, and Knowledge Management. Wiley and Sons 2003.

[6] Gal, A.: The Health Problems of Data Integration. In OTM Workshops, Lecture Notes in Computer Science, Springer 2008, Vol. 5333, p. 65.

[7] Garcia-Molina, H.—Papakonstantinou, Y.—Quass, D.—Rajaraman, A.—Sagiv, Y.—Ullman, Y. D.—Vassalos, V.—Widom, J.: The TSIMMIS approach to mediation: Data models and languages. Journal of Intelligent Information Systems, Vol. 8, 1997, No. 2, pp. 117–132.

[8] Haller, A.—Cimpian, E.—Mocan, A.—Oren, E.—Bussler, C.: WSMX – A Semantic Service-Oriented Architecture. Proc. Intl. Conf. on Web Services (ICWS) 2005.

[9] Jhingran, A. D.—Mattos, D.—Pirahesh, N. H.: Information Integration: A Research Agenda. IBM System Journal 41, 2002, No. 4.

[10] Karastoyanova, D.—Wetzstein, B.—Van Lessen, T.—Wutke, D.—Nitzsche, J.—Leymann, F.: Semantic Service Bus: Architecture and Implementation of a Next Generation Middleware. Proceedings of the Second International Workshop on Service Engineering (SEIW) 2007.

[11] Lenzerini, M.: Data Integration: A Theoretical Perspective. Proceedings – Principles of Database Systems Conference (PODS '02), pp. 233–246.

[12] Orriens, B.—Yang, J.—Papazoglou, M.: A Framework for Business Rule Driven Web Service Composition. In: Jeusfeld, M. A. & Pastor, O. (Eds).: Proceedings of ER 2003 Workshops, LNCS 2814, Springer-Verlag 2003, pp. 52–64.

[13] Pahl, C.—Zhu, Y.: A Semantical Framework for the Orchestration and Choreography of Web Services. International Workshop on Web Languages and Formal Methods (WLFM 2005).

[14] Pahl, C.—Giesecke, S.—Hasselbring, W.: An Ontology-Based Approach for Modelling Architectural Styles. Proc. European Conference on Software Architecture (ECSA) 2007, Springer-Verlag, LNCS Series.

[15] Payne, T.—Lassila, O.: Semantic Web Services. IEEE Intelligent Systems, Vol. 19, 2004, No. 4.

[16] Peltier, M.—Bezivin, J.—Guillaume, G.: MTRANS: A General Framework, Based on XSLT, for Model Transformations. Proceedings of the Workshop on Transformations in UML (WTUML), 2001.

[17] Rouvellou, I.—Degenaro, L.—Rasmus, K.—Ehnebuske, D.—Mckee, B.: Extending Business Objects with Business Rules. Proceedings of 33rd Intl. Conference on Technology of Object-Oriented Languages 2000, pp. 238–249.

[18] Schaffert, S.: Xcerpt: A Rule-Based Query and Transformation Language for the Web. Ph. D. Thesis, University of Munich 2004.

[19] Seltsikas, P.—Currie, W. L.: Evaluating the Application Service Provider (ASP) Business Model: The Challenge of Integration. Proceedings of 35th Annual Hawaii International Conference 2002, pp. 2801–2809.

[20] Stal, M.: Web Services: Beyond Component-Based Computing. Communications of the ACM, Vol. 45, 2002, pp. 10, pp. 71–76.

[21] Stern, A.—Davis, J.: Extending the Web services model to IT services. Proceedings of IEEE International Conference on Web Services 2004, pp. 824–825.

[22] Subasu, I. E.—Ziegler, P.—Dittrich, K. R.—Gall, H.: Architectural Concerns for Flexible Data Management. In: Workshop on Software Engineering for Tailor-made Data Management (SETMDM) 2008, EDBT Workshops, pp. 34–39.

[23] Wang, M. X.—Bandara, K. Y.—Pahl, C.: Integrated Constraint Violation Handling for Dynamic Service Composition. Proc. of 2009 IEEE International Conference on Services Computing.

[24] Widom, J.: Research Problems in Data Warehousing. Proceedings of 4th International Conference on Information and Knowledge Management 1995.

[25] Zhang, Z.—Yang, H.: Incubating Services in Legacy Systems for Architectural Migration. Proceedings of 11th Asia-Pacific Software Engineering Conference (APSEC) 2004, pp. 196–203.

[26] Ziegler, P.—Dittrich, K. R.: Data Integration – Problems, Approaches, and Perspectives. In J. Krogstie, A. L. Opdahl and S. Brinkkemper: Conceptual Modelling in Information Systems Engineering, Springer 2007, pp. 39–58.

[27] Zhu, Y.: Declarative Rule-Based Integration and Mediation for XML Data in Web Service-Based Software Architectures. M. Sc. Thesis, Dublin City University 2007.

[28] Zhu, F.—Turner, M.—Kotsiopoulos, I.—Bennett, K.—Russell, M.—Budgen, D.—Brereton, P.—Keane, J.—Layzell, P.—Rigby, M.—Xu, J.: Dynamic Data Integration Using Web Services. Proc. Intl. Conference on Web Services (ICWS) 2004.

Claus Pahl is a Senior Lecturer at Dublin City University's School of Computing, where he is the leader of the Software and Systems Engineering group. He has graduated from the Technical University of Braunschweig and has obtained a Ph. D. from the University of Dortmund. He has published more than 200 papers in various journals, books, conference and workshop proceedings. He is on the editorial board of four journals and is a regular reviewer for journals and conferences in the area of Web and software technologies. He is the principal investigator of several research projects in Web software engineering. His research interests cover a broad spectrum from service- and component technologies in software engineering to infrastructure and development technologies for Web applications.

Yaoling Zhu has recently finished his postgraduate research at the School of Computing at Dublin City University with an M. Sc. by Research. He is a graduate in Computer Science from the Zhengzhou Institute of Engineering, China. He has extensive experience in the software sector, working for several years as a senior software engineer for multinational companies such as Oracle, where he has been working on e-business outsourcing and Web service technologies in Oracle's European Development and Technology Centre. His research focuses on data integration problems in Web-based software systems.