Data Integration through Service-based Mediation for Web-enabled Information Systems[1]

Yaoling Zhu, Dublin City University, School of Computing, Dublin 9, Ireland,
Phone: ++353 +1 7005620, Fax: ++353 +1 700 5442, Email: yao.zhu3@mail.dcu.ie

Claus Pahl, School of Computing, Dublin City University, Dublin 9, Ireland,
Phone: ++353 +1 7005620, Fax: ++353 +1 700 5442, Email: claus.pahl@dcu.ie

Abstract

The Web and its underlying platform technologies have often been used to integrate existing software and information systems. Traditional techniques for data representation and transformations between documents are not sufficient to support a flexible and maintainable data integration solution that meets the requirements of modern complex Web-enabled software and information systems. The difficulty arises from the high degree of complexity of data structures, for example in business and technology applications, and from the constant change of data and its representation. In the Web context, where the Web platform is used to integrate different organisations or software systems, additionally the problem of heterogeneity arises. We introduce a specific data integration solution for Web applications such as Web-enabled information systems. Our contribution is an integration technology framework for Web-enabled information systems comprising, firstly, a data integration technique based on the declarative specification of transformation rules and the construction of connectors that handle the integration and, secondly, a mediator architecture based on information services and the constructed connectors to handle the integration process.

Keywords

Web Applications, Data Integration, Software Architecture, Data Models, Information System Design

INTRODUCTION

The Web and its underlying platform technologies have often been used to integrate existing software and information systems. Information and data integration is a central issue in this context. Basic techniques based on XML for data representation and XSLT for transformations between XML documents are not sufficient to support a flexible and maintainable data integration solution that meets the requirements of modern complex Web-enabled software and information systems. The difficulty arises from the high degree of complexity of data structures, for example in business

[1] ***This chapter appears in "Software Engineering for Modern Web Applications: Methodologies and Technologies" edited by Brandon, Daniel M., Copyright 2008, IGI Global, www.igi-global.com. Posted by permission of the publisher.***

and technology applications, and from the constant change of data and its representation. In the Web context, where the Web platform is used to integrate different organisations or software systems, additionally the problem of heterogeneity arises. This calls for a specific data integration solution for Web applications such as Web-enabled information systems.

The advent of Web services and service-oriented architecture (SOA) has provided a unified way to expose the data and functionality of an information system. Web services are provided as-is at certain location and can be discovered and invoked using Web languages and protocols. SOA is a service-based approach to software application integration. The use of standard technologies reduces heterogeneity and is therefore central to facilitating application integration. The Web services platform is considered an ideal infrastructure to solve the problems in the data integration domain such as heterogeneity and interoperability (Orriens et al., 2003; Haller et al., 2005; Zhu et al., 2004). We propose a two-pronged approach to address this aim: firstly, data integration and adaptivity through declarative, rule-based service adaptor definition and construction, and, secondly, a mediator architecture that enables adaptive information service integration based on the adaptive service connectors. Abstraction has been used successfully to address flexibility problems in data processing - database query languages are a good example here.

XML as a markup language for document and data structuring has been the basis of many Web technologies. XML-based transformation languages like XSLT, the XML Stylesheet Transformation Language, XML-based data can be translated between formats. With recent advances in abstract, declarative XML-based data query and transformation languages beyond the procedural XSLT, this technology is ready to be utilised in the Web application context. The combination of declarative abstract specification and automated support of the architecture implementation achieves the necessary flexibility to deal with complexity and the maintainability of constantly changing data and system specifications.

Our objective is to explore and illustrate solutions to compose a set of data integration services. The data integration services deliver a unified data model built on top of individual data models in dynamic, heterogeneous and open environments. The presentation of this technology framework aims to investigate the practical implications of current research findings in Web information systems technology.

A lightweight mediated architecture for Web services composition shall be at the centre of our solution. Data integration is a central architectural composition aspect. The flexibility of the architecture to enable information integration is essential in order to separate the business process rules from the rest of the application logic. Therefore, the data transformation rules are best expressed at the abstract model level. We apply our solution to the Web Services platform in the context of information technology services management in the Application Service Providers ASP (on demand) business area. We focus on this context to illustrate problems and solutions. Portals, provided by ASPs, are classical examples where data might come from different sources that motivate our research. In order to consume the information, the data models and representation needs to be understood by all participants. The ASP maintains the application, the associated infrastructure, and the customer's data. The ASP also ensures that systems and data are available when needed.

The chosen area demonstrates the need to support deployment of Web service technology beyond toy examples (Stern & Davies, 2004). It is a specific, but important area due to the need to find solutions to accommodate constant structural changes in data representations. Two central themes shall be investigated:

- to identify data model transformation rules and how to express these rules in a formal, but also accessible and maintainable way are central to the data integration problem and its automation,
- service composition to enable interoperability through connector and relationship modelling based on workflow and business processes is central.

Our contribution based on these themes is an integration technology framework for Web-enabled information systems comprising

- a data integration technique based on the declarative specification of transformation rules and the construction of connectors that handle the integration in a software system,
- a mediator architecture based on information services and the constructed connectors to handle the integration process.

We start our investigation by providing some data integration background. We then present the principles of our declarative data integration technique. The mediator architecture that realises the data integration technique for Web services is subsequently presented. A larger application scenario will then be discussed. We end with some conclusions.

BACKGROUND

Data Integration Context

The Application Service Provider or ASP business model, which has been embraced by many companies, promotes the use of software as a service. Information Systems (IS) outsourcing is defined as the handing over to third party the management of IT and IS infrastructure, resources and/or activities (Willcocks & Lacity, 1998). The ASP takes primary responsibility for managing the software application on its infrastructure, using the Internet as the delivery channel between each customer and the primary software application. The ASP maintains the application and ensures that systems and data are available when needed. Handing over the management of corporate information systems to third party application service providers in order to improve the availability of the systems and reduce costs is changing the ways that we manage information and information systems.

Information integration aims at bringing together various types of data from multiple sources such that it can be accessed, queried, processed and analysed in an integrated and uniform manner. In a large modern enterprise, it is inevitable that different parts of the organization will use different systems to produce, store, and search their critical data.

Recently, service-based platforms have been used to provide integration solutions for ASP applications. Data integration in these types of collaborating systems is necessary. This problem has been widely addressed in component-based software

development through adaptor and connector approaches (Crnkovic & Larsson, 2000; Szyperski, 2002). In the service-based Web applications context, the data in XML representation retrieved from the individual Web services needs to be merged and transformed to meet the integration requirements. The XML query and transformation rules that govern the integration may change; therefore, the programs for building up the connectors that facilitate the connection between integrated Web services and data service providers need to be adjusted or rewritten. As with schema integration, the schema-mapping task cannot be fully automated since the syntactic representation of schemas and data do not completely convey the semantics of different data sources. As a result, for both schema mapping and schema integration, we must rely on an outside source to provide some information about how different schemas (and data) correspond. 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. In this case, a transformation might be necessary; see Fig. 1 for a visualisation of the customer identifier example.

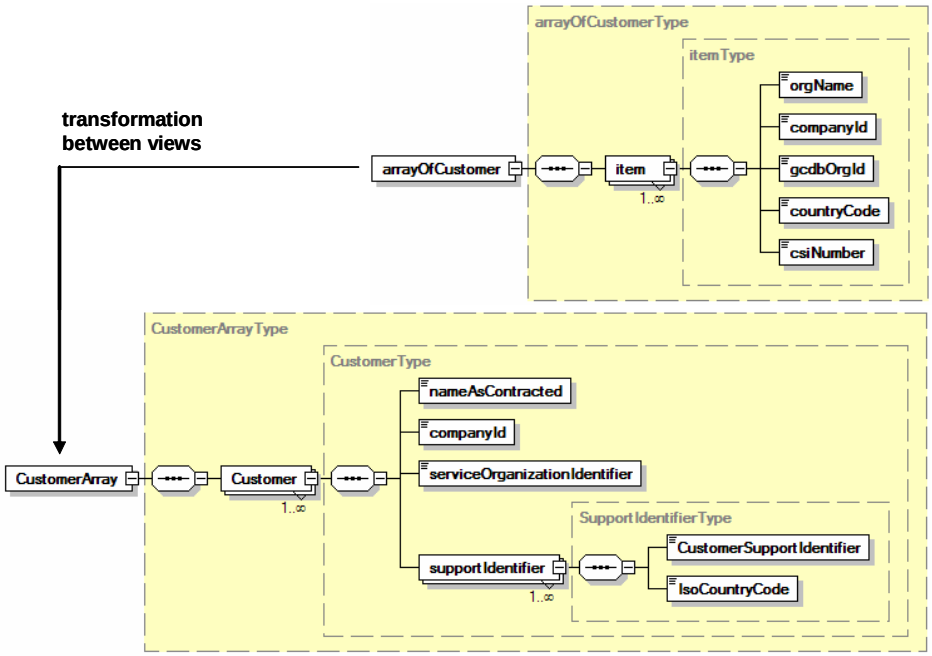

Fig. 1. Example of Data Integration in Adaptive Service Architectures - two Data Schemas that need to be transformed into one another.

Data Integration Principles

Information integration is the problem of combining heterogeneous data residing at different sources, and providing 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, as the example in Fig. 1 illustrates. The data integration itself is defined using transformation languages.

There are two major architectural approaches to the data integration problem that provide the infrastructure for the execution of transformations (Widom, 1995).

- Data warehousing is an eager or in-advance approach that gathers data from the appropriate data sources to populate the entities in the global view. A data warehousing approach to integration is suitable for data consumers wanting to access to local copies of data so that it can be modified and calculated to suite the business needs by nature.
- In contrast, the mediated approach extracts only data from export schemas in advance. A mediated approach to integration is suitable for information that changes rapidly, for service environments that change, for clients in need tailored data, and for queries that operate over large amounts of data from numerous information sources and most importantly for clients with the need of the most recent state of data.

XSLT Shortcomings

XSLT is the most widely used language for XML data integration, but these XSLT transformations are difficult to write and maintain for large-scale information integration. It is difficult to separate the source and target parts of the rules as well as the filtering constraints. The verbosity of XML makes manual specifications of data and transformations difficult in any case. With this difficulty in mind, we propose a declarative query and transformation approach yielding more expressive power and the ability to automatically generate query programs as connectors to improve the development of services-based data integration in Web-based information systems.

XSLT does work well in terms of transforming data output from one Web service to another in an ad-hoc manner. XSLT code is, however, difficult to write and almost impossible to reuse in a large enterprise integration solution. The syntactical integration of the query part and construction part of a XSLT transformation program is hard to read and often new programs are needed even when a small portion of the data representation changes. XSLT does not support the join of XML documents. We would in our context need to merge several source XML documents into one document before it can be transformed into another document according to an over-arching general schema.

A DECLARATIVE DATA INTEGRATION AND TRANSFORMATION TECHNIQUE

A declarative, rules-based approach can be applied into the data transformation problem (Orriens et al., 2003). A study by Peltier et al. (2001) introduces the MTRANS language that is placed on top of XSLT to describe data model transformations. XSLT is generated from an MTrans specification. The transformation rules are expressed in the form of MTrans and then parsed using a generator. Peltier et al. argue that the data transformation rules are best expressed declaratively at the abstract model level rather than at the concrete operational level in order to reduce the complexity of the transformation rules.

A data integration engine for the Web services context can be built in the Web service business process execution language WS-BPEL, which is another example for the

benefits of abstraction in transformation and integration. A common over-arching information model governs what types of services are involved in the composition. In (Rosenberg & Dustdar, 2005), a business rule engine-based approach has been introduced to separate the business logic from the executable WS-BPEL process.

These two examples illustrate current work in this context. Now, a detailed discussion shall elicit the specific requirements for service-based information integration.

Requirements for Mediated Integration

The flexibility of the architecture in which information integration is to be realised is essential in order to separate the business logic from the rest of the application logic. Therefore, the data transformation rules are best expressed at an abstract business model level. These rules, stored in a repository, can be used to dynamically create XSLT-based transformations using a 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. We start our investigation by discussing the properties of suitable integration and transformation languages.

XML data might be provided without accompanying schema and sometimes is not well-formed; XML data often contains nested structures. Therefore, transformation techniques need more expressive power than traditional database languages such as relational algebra or SQL. The characteristics of an XML query language have been studied extensively (Jhingran et al., 2002; Lenzerini, 2002; Peltier et al., 2002). However, these investigations often focus on the features to query an XML or semi-structured data repository in the spirit of database query languages rather than constructing a new XML document in the context of the data integration. The following principles, which are inspired by the data integration literature such as (Lenzerini, 2002), aim to provide a comprehensive requirements list.

- The language should support both querying and restructuring XML Data.
- The language must enable the generation of query programs by other programs.
- The language should be capable of expressing the following operations in addition to the ones existing in database languages (such as projection, selection, and joins): restructuring (constructing a new set of element instances based on variable bindings and the global schema), combination (merging two or more element instances into one), and reduction (to express transformation rules that exclude parts of the data from the result).
- Compositionality is an essential feature for an XML query and transformation language to support query composition.

A rule-based, declarative language enables developers to concentrate on the integration logic rather than on implementation details and enables the required compositionality and expressiveness.

Most XML and semi-structured data query languages have been proposed to extract XML data from the XML databases or the Web. A comparative analysis of existing languages has been done by Reynaud et al. (2001). A language is generally designed to suit the needs for a limited application domain such as database querying or data integration; some languages are designated only for semi-structured data that predated the XML-format. A query language should be able to query data sources using

complex predicates, joins and even document restructuring. We add the following criteria specifically for the context of Web-based data integration:

- Join. The language must support joins of multiple XML data sources. A join condition is necessary to compare attributes or elements in any number of XML documents. In data integration systems, data is most likely to come from more than one source.
- Data Model. The queries and their answers are the instances of a data model. Sometimes, a rich data model is needed to support the functionality of some query languages. The underlying framework plays a major role in determining a data model for a query language.
- Incomplete Query Specification. XML and semi-structured data is not as rigid as relational data in term of schema definitions and data structure. Therefore, it is important that a query language is capable of expressing queries in incomplete form, such as by using wildcard and regular expressions - also called partially-specified path expressions.
- Halt on Cyclic Query Terms. If a language supports querying with incomplete query specification by wildcard and regular expression, it might cause termination problems. Therefore, features to detect cyclic conditions are required.
- Building New Elements. The ability to construct a new node added to the answering tree is an important feature for data integration systems.
- Grouping. Grouping XML nodes together by some conditions by querying the distinct values is another important feature in data integration. Some languages use nested queries to perform grouping operations; in contrast, some more powerful languages have built-in constructors.
- Nested Queries. Nested queries are common in relational database languages for joining different data elements by their values. In logic-based languages, the construction part and the selection part are separated.
- Query Reduction. Query reduction allows users to specify what part of the elements or what nodes in the query conditions will be removed from the resulting XML tree.

A number of potential candidates shall briefly be discussed in the context of these requirements:

- XQuery is a W3C-suppported query language that aims at XML-based database systems. XQuery is an extension of XPath 2.0 adding functionalities needed by a full query language. The most notable of these functionalities are support of sequences, the construction of nodes and variables, and user-defined functions.
- UnQL - the Unstructured Query Language - is a query language originally developed for querying semi-structured data and nested relational databases with cyclic structures. It has later been adapted to query XML documents and data. Its syntax uses query patterns and construction patterns and a query consists of a single select or traverse rule that separates construction from querying. Queries may be nested, in which case the separation of querying and construction is abandoned. UnQL was one of the first languages to propose a pattern-based querying (albeit with subqueries instead of rule chaining).
- XML-QL uses query patterns and path expressions to select data from XML sources. These patterns can be augmented by variables for selecting data.

XML-QL uses query patterns containing multiple variables that may select several data items at a time instead of path selections that may only select one data item at a time. Furthermore, variables are similar to the variables of logic programming, i.e. joins can be evaluated over variable name equality. Since XML-QL does not allow one to use more than one separate rule, it is often necessary to employ subqueries to perform complex queries.

The shortcomings of these widely known and used languages in the context of the given requirements and the language comparisons have led us to choose a fully declarative language called Xcerpt (Bry & Schaffert, 2002) that satisfies all criteria that we have listed earlier on. However, other recently developed and well-supported transformation languages such as ATL and QVT are similarly suitable candidates. While QVT satisfies the criteria, it is currently not as well supported through tools and accessible tutorial material.

Xcerpt is a query language designed for querying and transforming both data on the standard Web (e.g. XML and HTML data) and data on the Semantic Web (e.g. RDF data). Xcerpt not only allows one to construct answers in the same data formats as the data queries like XQuery, but also allows further processing of the data generated by this same query program. One of the design principles is to strictly separate the matching part and the construction part in a query. Xcerpt follows a pattern-based approach to querying the XML data. A similar approach has been proposed in the languages UnQL and XML-QL. However, Xcerpt has extended the pattern-based approach in the following aspects. Firstly, the query patterns can be specified by incomplete query 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 the pattern specifications to be specified in a more flexible manner but without losing accuracy. Secondly, the simulation unification computes answer substitutions for the variables in the query pattern against underlying XML terms - similar to UnQL, but strict unification is used in UnQL.

Declarative Transformation Rules

We have adapted Xcerpt to support the construction of the service connectors, which is our central objective:

- From the technical point of view, in order to promote code reuse, the individual integration rules should not be designed to perform the transformation tasks alone. The composition of rules and rule chaining demand the query part of service connector to be built ahead of the construction part of the service connector.
- From the business point of view, the data presentation 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 logic. Only an additional construction part is needed to enable versioning of the global data model.

Grouping and incomplete query specifications turn out to be essential features.

Xcerpt is a document-centric language which is designed to query and transform XML and semi-structured documents. Therefore the ground rules, which read data

from the document resources, are tied with at least one resource identifier. This is a bottom up approach in terms of data population because the data are assigned from the bottom level of the rules upward until the rule application reaches the ultimate goal of a complex, hierarchically structured rule. These rules are defined through an integration goal at the top level and structured into sub-rules down to ground rules, which address individual data elements.

```
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. 2. Declarative Query and Transformation Specification of Customer Array Element in Xcerpt.

Fig. 2 shows a transformation example for a customer array based on Fig. 1. Fig. 1 is a graphical illustration of XML-based data structures. The upper structure provide the data schema of the input document; the lower structure is the target data schema that a transformation needs to map onto. The graphical representation allows us to avoid the verbosity of XML-based data representations for this investigation. 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. For instance, the `Name` variable is used to declare `nameAsContracted` and `OrgName` as semantically equal elements in both representations that are syntactically different.

This original Xcerpt approach is unfortunately not feasible in an information integration solution because the resource identifiers can not be hard coded in the ground rules in our setting. A wrapper mechanism has been developed to pass the resource identifiers from the goal level all the way down to the ground rules. In addition to the original Xcerpt approach, we propose a mediator-based data integration architecture where the Xcerpt-based connectors are integrated with the client and provider Web services. WS-BPEL code is generated by a transformation generator within the mediator service (see Fig. 4 below, which is explained in a separate section).

Implementation of Connector Construction

The construction of Xcerpt-based connectors, which specify integration through declarative rules, can be automated using rule chaining. Ground rules are responsible for querying data from individual Web services. Intermediate composite rules are responsible for integrating the ground rules to render data types that are described in global schemas. The composite rules are responsible for rendering the data objects described in the interfaces of the mediator Web services based on demand. Therefore, exported data from a mediator service is the goal of the corresponding connector (i.e. a query program), see Fig. 3. Fig. 1 defines again the respective input and output data schemas. The `CONSTRUCT .. FROM` clauses in Fig. 3 define the individual rules. Here, information from `ArrayOfCustomers` and `Customers` is selected to construct the `SupportIdentifier`.

```
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

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

Fig. 3. Transformation Specification in Xcerpt based on Goal Chaining.

We apply backward goal-based rule chaining in this adapted implementation to execute complex queries based on composite rules. Fig. 3 shows an example of this pattern matching-based approach that separates a possibly partial query based on resource and construction parts. This transformation rule maps the `supportIdentifier` element of the `customer` example from Fig. 1. Fig. 3 is a composite rule based on the `SupportIdentifier` construction rule at a lower level.

These rules are saved in a repository. When needed, a rule will be picked and the backward rule chaining enables data objects to be populated to answer transformation requests. This architecture will be detailed in the subsequent section.

MEDIATOR ARCHITECTURE

Motivation

Zhu et.al. (2004) 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. We propose, based on (Haller et al., 2005; Wiederhold, 1992; Sheth & Larson, 1990; Zhu et al., 2004), a service-oriented data integration architecture to provide a unified view of data on demand from various data sources. A service-oriented data integration architecture is different from business process integration as the latter is concerned with integrating the business process rather than data. The proposed integration architecture uses Web services to enable the provision of data on demand whilst keeping the underlying data sources autonomous.

There is consequently a need for mediators in an architecture that harmonise and present the information available in heterogeneous data sources (Stern & Davies, 2003). This harmonisation comes in the form of identification of semantic similarities in data while masking their syntactic differences; see Fig. 1. Relevant and related data is then integrated and presented to a higher layer of applications. The sourcing, integration, and presentation of information can be seen as logically separated mediator rules for integration, implemented by mediator services - which shall form the basis for the presented mediator architecture.

Garcia-Molina et.al. (1997) identify that the following requirements are essential in order to build a mediator architecture. Firstly, it must be based on a common data model that is more flexible than the models commonly used for the database management systems. Secondly, it must be supported by a common query language. Finally, there must be a tool to make the creation of new mediators and mediator systems more cost-effective than building them from scratch.

Architecture Definition

The mediator architecture transforms local XML documents into documents based on a global schema. Fig. 4 illustrates this architecture with a few sample information services - Customer Data, E-business System, Request Logging and Analysis Service - that a client might access. The data integration engine is built based on a composition of individual services using WS-BPEL, where component invocation orders are predefined in the integration schemas. These service orchestrations are defined by specifying the order in which operations should be invoked.

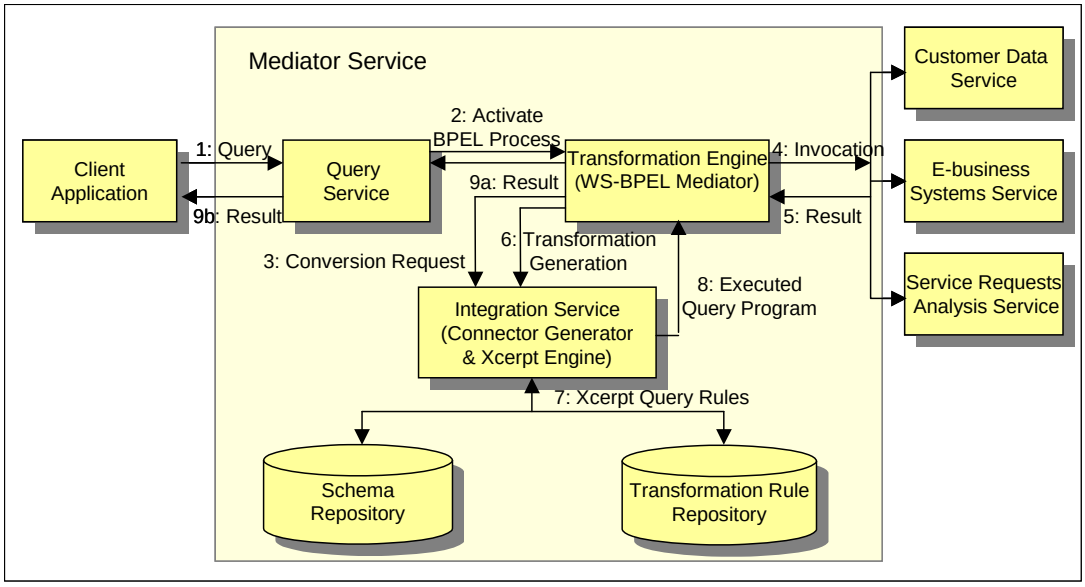

Fig. 4. Mediator Architecture for Adaptive Service-based Information Systems with sample Information Services.

The proposed Web services-based mediator architecture, Fig. 4, contains the following components:

- Schema Repository: Each object within the model is a logical representation of the entity and will often be populated with data sourced from more than one repository. The advantage of having a unified view of data is to make sure that the customers will have a consistent view of data and to avoid duplication.
- Information Services: These provide source data retrieved from the underling data repositories to clients and other services. The signature of the Web service interfaces such as input parameters and data output is agreed in advance by business domain experts from both client and provider sides. The benefit of asking the data sources to provide a Web service interface is to delegate the responsibility and cut down the effort spent on developing data access code and understanding the business logic.
- Data Integration and Mediation Services: A common data model can be implemented as an XML schema. Two basic approaches have been proposed for the mappings between the export schemas and the federated schema - called global-as-view and local-as-view in (Lenzerini, 2002). The former approach defines the entities in the global data model as views over the export schemas whereas the latter approach defines the export schemas as views over the global data model. In this work, a data integration service will be treated as a mediator in the mediator architecture. We introduce a novel approach to ease and improve the development of the mediators. There are two quite different styles of transformation: procedural, with explicit source model traversal and target object creation and update, and declarative, with implicit source model traversal and implicit target object creation. Therefore, an approach based on a declarative rule markup language to express the data transformation rules and a rule engine have been chosen. The mapping should be conducted at the abstract syntax mappings level, leaving the rendering of the result to a separate step at runtime to the BPEL engine.

- Query Component: The query service is designed to handle inbound requests from the application consumer side. The application developers build their applications and processes around common objects and make successive calls to the mediated Web services. Therefore, the interfaces of individual Web service providers are transparent to the application customers; they may send any combinations of the input parameters to the query service. In order to facilitate these unpredicted needs, the query service has to decompose the input messages into a set of pre-defined WS-BPEL flows. Normally a BPEL flow belongs to a mediator that delivers a single common object. Occasionally, two or more mediators need to be bundled together to deliver a single object.

Each of the components can in principle be offered as a service by a (potentially different) provider. Within the composite mediator service, both transformation and connector generation services are separated and only loosely coupled.

Developer Activities

The architecture in Fig. 4 explains the runtime view from the client and user perspective. In order to complete the picture, the development perspective shall also be addressed. Fig. 5 illustrates development activities, looking at the developers of architecture, rules, and services - and their respective activities. A number of actors including service provider engineers, application software engineer, integration business analyst, integration software architect, and integration software engineer are distinguished. These are associated to the activities they are involved in. In particular the integration team is involved with Xcerpt-based rule definition and application. Activities are also related among themselves. The participation of different roles from possibly different organisations (application customer, service provider, integration team) demonstrates the need for common understanding and maintainability of the integration problem, which can be achieved through abstract and declarative rule specifications (here in Xcerpt format), shared by service provider developers, integration business analysts, and integration software developers.

APPLICATION SCENARIO AND DISCUSSION

The presented data integration technique and the mediated architecture are complemented by an incremental, evolutionary process model. Some pragmatic aspects of this process shall now be addressed. In the proposed architecture, the unified data model (over-arching schema) is maintained manually. The schema for large enterprise integration solutions might consist of a large number of data aspects. From the development point of view, it is only reasonable to deliver the data integration services on a phased basis such as one data aspect for one release cycle. A mediator consists of the following components: the individual provided Web services, a WS-BPEL workflow, and one or more service connectors, as illustrated in Fig. 4. Mediators in our solution are used to deliver these data aspects according to the unified schema. This schema is available to the customers so that these can decide which mediator to call based on the definition of the unified schema.

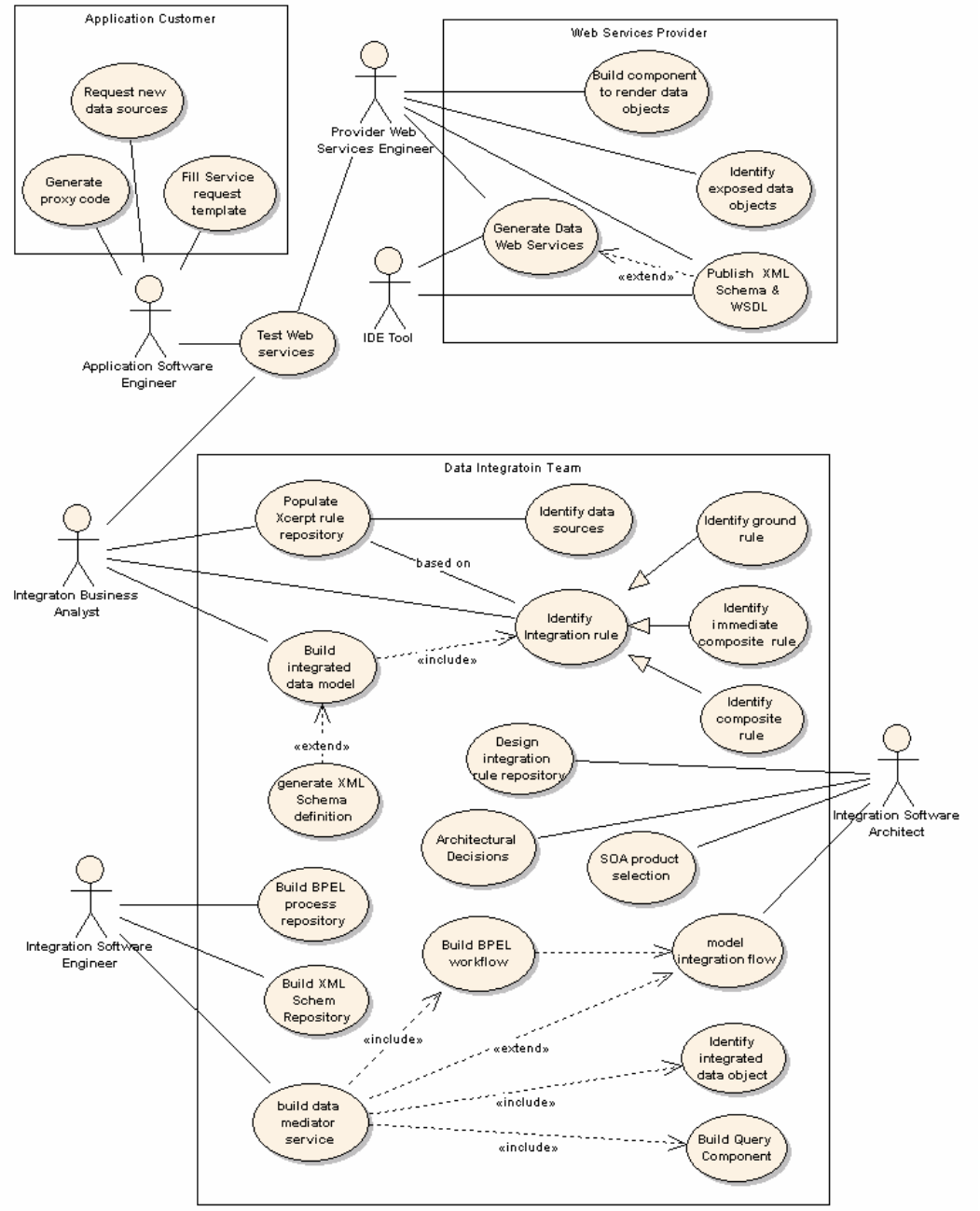

Fig. 5. Overview of different Developer Roles and their activities involved.

The focus of this investigation is not on the automatic composition of Web services, rather on how the data output from multiple Web services can be automatically integrated according to a global data model and sent back to users. Therefore, in terms of the WS-BPEL process flow, a static approach with respect to the orchestration of the involved Web services can be taken. These can be orchestrated together in form of a WS-BPEL flow built in advance.

During the development phase, the mappings between the global model and the local models will be expressed at the abstract model level, for instance in the widely used

MOF (Meta Object Facility) framework for modelling language definition. Model transformation between different metamodels can then be automatically carried out. The inputs are the source XML schema definitions and the transformation rules. The output is an XSLT transformation file.

In the proposed process model illustrated in Fig. 5, the unified data model and the creation of rules are the responsibility of the business solution analysts, not necessarily the software architect. The rules are merely mappings from the elements exposed by Web service providers to the elements in the unified data model. We assume here that the semantic similarity is determined manually. In the literature on data model transformation, the automation of the mapping is often limited to transforming the source model and the destination model rather than integrating more than one data model into a unified data model. Even in the case of source to destination model mapping, the user's intervention is needed to select one from more than one sets of mappings that are generated. In our proposed architecture, the service connectors can be generated on the fly by rule composition. The sacrifice is that semantic similarity is not taken into consideration.

The data integration rules are created at the higher level than the Xcerpt ground query programs themselves, as the following schematic example demonstrates (Fig. 3 shows an example of a composite rule like A below):

Rule A: $A(a, b) := B(a, b), C(b)$
Rule B: $B(a, b) := D(a), E(b)$
Rule C: $C(b) := E(b), F(b)$

Each of the above rules would be implemented in the Xcerpt language. In the above example, rule *A* is a composite rule, based on *B* and *C*. This could be used to answer a user's query directly, but internally referring to subordinated rules dealing with the extraction and transformation of specific data aspects. The resource identifiers in form of variables and the interfaces for the data representation such as version number of the unified data model will be supplied to the transformation generator. The rule mappings in the transformation generator serve as an index to find the correct Xcerpt queries for execution. As a result, a query program including both query part and construction part is being executed to generate the XML output, which is sent back to the transformation generator.

In terms of examples, we have so far only addressed complex transformations based on compositional rules within data provided by one Web service - the customer information service. Queries could of course demand to integrate data from different services. For instance, to retrieve all services requests by a particular customer would target two services, based on several composite integration and transformation rules.

FUTURE TRENDS

Adaptivity in service-based software systems is emerging as a crucial aspect beyond the discussed area of service-based ASP infrastructures and on-demand information systems. Adaptability of services and their infrastructure is necessary to reconcile integration problems that arise in particular in dynamic and changing environments.

We have excluded the problem of semantic interoperability from our investigation. Mappings between schemas might still represent the same semantic information. The recently widely investigated semantic Web services field, with ontology-based domain and service models, can provide input for some planned extensions in this direction (Haller et al., 2005).

Re-engineering and the integration of legacy systems is another aspect that we have not addressed. The introduction of data transformation techniques for reengineering activities can improve the process of re-engineering legacy systems and adopting service-oriented architecture to manage the information technology services (Zhang & 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 exploited in sufficient detail in the context of transformation and re-engineering.

CONCLUSIONS

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

The presented framework utilises information integration technologies for service-oriented software architectures. The crucial solutions for the information integration problem are drawn from mediated architectures and data model transformation, allowing the data from local schemas to be transformed, merged and adapted according to declarative, rule-based integration schemas for dynamic and heterogeneous environments. We have proposed a declarative style of transformation, with implicit source model traversal and implicit 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.

REFERENCES

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

Barrett, R., Patcas, L.M., Murphy, J. and Pahl, C. (2006). Model Driven Distribution Pattern Design for Dynamic Web Service Compositions. International Conference on Web Engineering ICWE'06 (pp. 129-136).

BPEL Coalition (2006). *Business Process Execution Language for Web Services Version 1.1*. Available from http://www.ibm.com/developerworks/library/ws-bpel/.

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, Springer-Verlag.

Crnkovic, I. & Larsson, M. (2000). A Case Study: Demands on Component-based Development. *Proceedings 2nd International Conference on Software Engineering*. pp. 23–31. ACM Press.

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), March 1997, pp. 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 2005*.

Jhingran, A.D., Mattos, D. & Pirahesh, N.H. (2002). Information Integration: A research agenda. *IBM System Journal* 41, no. 4, special issue on Information Integration. Available from www.research.ibm.com/journal/sj/414/jhingran.pdf.

Lenzerini, M. (2002). Data integration: A theoretical perspective. *Proceedings Principles of Database Systems Conference PODS'02*, ACM. 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, 2003. 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. Malaga, Spain. Elsevier. Electronic Notes in Theoretical Computer Science.

Pahl, C. (2005). Layered Ontological Modelling for Web Service-oriented Model-Driven Architecture. European Conference on Model-Driven Architecture ECMDA'2005. Springer-Verlag, LNCS Series.

Pahl, C. and Zhu, Y. (2006). Semantical Framework for the Orchestration and Choreography of Web Services. International Workshop on Web Languages and Formal Methods WLFM'05. Newcastle upon Tyne, UK. Elsevier ENTCS Series.

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*.

Peltier, M., Ziserman, F. & Bezivin. (2002). On levels of model transformation. *Proceedings XML Europe Conference 2002*, pp. 1–17, Paris, France, Graphic Communications Association.

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 of the 7th International IEEE Conference on E-Commerce Technology*.

Sheth A. P. & Larson J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, vol. 22, No. 3, pp. 183.

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

Stern, A & Davis, J. (2003). A Taxonomy of Information Technology Services: Web Services as IT Services. *Proc. First International Conference on Service Oriented Computing.*

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

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

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

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

Willcocks, L. P. & Lacity, M. C. (1998). The Sourcing and Outsourcing of IS: Shock of the New? Willcocks, L. P. & Lacity, M. C. (eds) *Strategic Sourcing of Information Technology: Perspectives and Practices*.Wiley.

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

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*.