Mediated Data Integration and Transformation for Web Service-based Software Architectures

Yaoling Zhu
School of Computing
Dublin City University
Dublin 9, Ireland
Email: yao.zhu3@mail.dcu.ie

Claus Pahl
School of Computing
Dublin City University
Dublin 9, Ireland
Email: claus.pahl@dcu.ie

***Abstract*— Service-oriented architecture using XML-based Web services has been widely accepted by many organisations as the standard infrastructure to integrate heterogeneous and autonomous data sources. As a result, many Web service providers are built up on top of the data sources to share the data by supporting provided and required interfaces and methods of data access in a unified manner. In the context of data integration, problems arise when Web services are assembled to deliver an integrated view of data, adaptable to the specific needs of individual clients and providers. Traditional approaches of data integration and transformation are not suitable to automate the construction of connectors dedicated to connect selected Web services to render integrated and tailored views of data. We propose a declarative approach that addresses the often-neglected data integration and adaptivity aspects of service-oriented architecture.**

I. Introduction

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. The use of standard technologies reduces heterogeneity and is therefore key 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 [9], [6], [23]. However, recent research activities in Web services technology have been focused on service composition and integration rather than data aspects. We propose a two-pronged approach to address this shortcoming: 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.

Our objective is to explore solutions to compose a set of data integration services. The Business Process Execution Language (WS-BPEL) [2] shall be used as the orchestration language to build up a mediator process flow for service-based data integration. The data integration services deliver a unified data model built on top of individual data models for adaptivity in dynamic, heterogeneous and open environments.

Portals, provided by Application Service Providers (ASP), 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.

A lightweight mediated architecture for Web services composition shall be at the centre of our solution. Information 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 will apply our solution to the service-oriented architecture in general and the Web Services platform in particular in the context of information technology services management in the ASP (on demand) business area.

Data integration is a common problem in the composition of collaborating services. The chosen area, however, demonstrates the need to support deployment of Web service technology beyond toy examples [17]. 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:

- Data Model transformation. To identify data 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. Interoperability enabled through connector and relationship modelling based on workflow and business processes is central. Two main aspects can be distinguished, which represent different views on processes. The orchestration of data providers and connector services addresses the internal perspective of process composition, looking at one process only.

We start our investigation by providing some data integration background in Section II. We then present the principles of our declarative data integration technique in Section III. The mediator architecture that realises the data integration technique for Web services is presented in Section IV. We end with a discussion of related work and some conclusions.

II. Data Integration

A. Problem Context

The Application Service Provider or ASP business model, which has been embraced by many companies, promotes the use of software as a service. IS outsourcing is defined [21] as the handing over to third party the management of IT and IS infrastructure, resources and/or activities. 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 to manage information and information systems.

Data integration in collaborating systems is necessary. This problem has been widely addressed in component-based software development through adaptor and connector approaches [4], [18]. In the Web services 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, see Fig. 1, a transformation might be necessary.

Information integration aims at bringing together various types of data from multiple sources such that it can be accessed, queried, processed and analyzed 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.

B. Data Integration Principles

Information integration is the problem of combining heterogeneous data residing at different sources, and providing the user with a unified view [8]. 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; see Fig. 1.

There are two major architectural approaches to the data integration problem [19].

- The data warehousing, eager, or in-advance approach 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 amount of data from numerous information sources and most importantly for clients with the need of the most recent state of data.

The integration itself is defined using transformation languages.

C. XSLT Shortcomings

XSLT transformations – based on the most widely used language for data integration – 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. 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 adaptive services-based data integration.

XSLT does work well in terms of transforming data output from one Web service to another in an ad-hoc manner. XSLT code is 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 neededeven when a small portion of the data representation changes. XSLT does not support join of XML documents. We would in our context need to merge source XML documents into one document before it can be transformed into the document according to an over-arching schema.

III. A Declarative Data Integration and Transformation Technique

A declarative, rules-based approach can be applied into the data transformation problem [9]. A study by [10] introduces the MTRANS language that is placed on top of XSLT to describe model transformations. XSLT is generated from an MTrans specification. The transformation rules are expressed in the form of MTrans, and will be parsed using a compiler generator. They argued that the transformation rules are best expressed declaratively at the abstract model level rather than at the concrete operational level to reduce the complexity of the transformation rules.

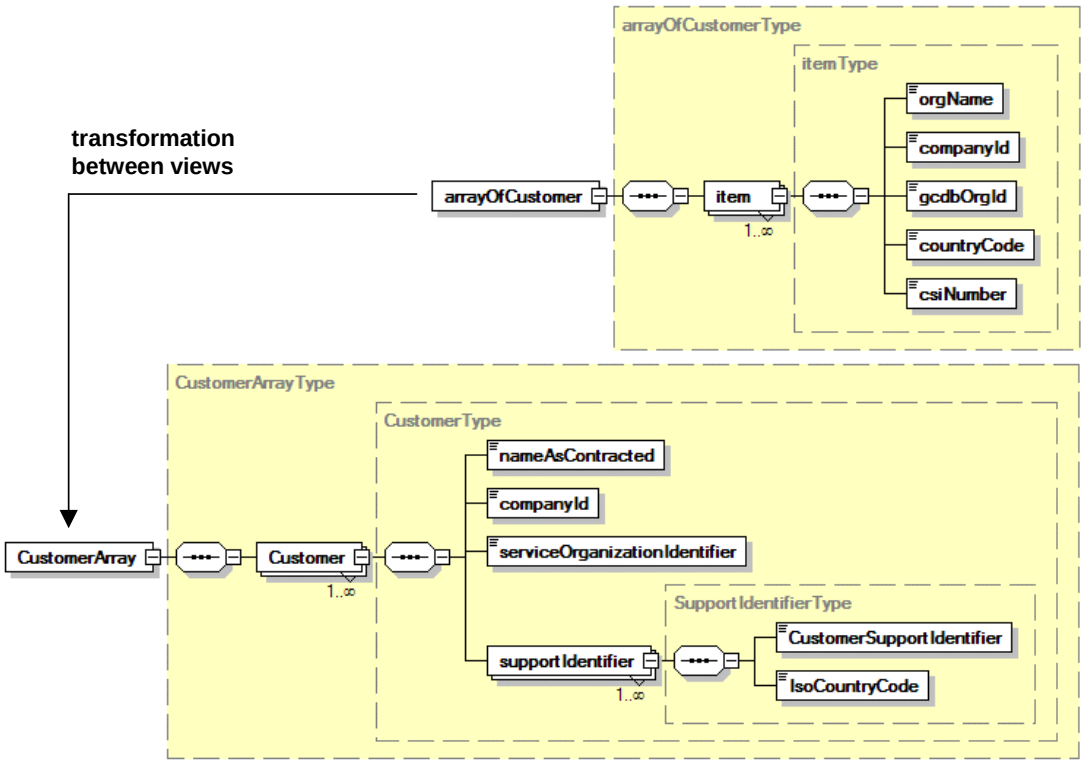

Fig. 1. Example of Data Integration in Adaptive Service Architectures – two Data Schemas that need to be transformed into one another.

A data integration engine for the Web services context can be built in BPEL. A common over-arching information model governs what types of services are involved in the composition. In [13], a business rule engine-based approach has been introduced to separate the business logic from the BPEL process.

A. 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, will 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 bt 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 [7], [8], [11]. However, these investigations mainly focus on the features on how 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 are inspired by the literature 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 additional 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 features 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

```
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.

been proposed to extract XML data from the XML databases or the Web. A comparative analysis of the existing languages has been done [12]. 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 the semi-structured data that predated the XML-format. A query language should be able to query data source using complex predicates, joins and even document restructuring. We add the following criteria specifically for the context of data integration:

- Join. The language must support joint of multiple XML data sources. A join condition is necessary to compare attributes or elements in any number of XML document. 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 underling framework plays a major role to determine a data model for a query language.
- Incomplete Querying 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 wild-card and regular expression, it might cause termination problems. Therefore, features to detect cyclic conditions is required.
- Building New Elements. The ability to construct a new element or a new node adding 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 operation; in contrast, some more powerful languages have the built-in constructors.
- Nested Queries. Nested queries are common in relational database languages to join between 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.

We have chosen, based on this requirements list and the language comparisons, the Xcerpt language [3] – a declarative rule-based XML query language as the basis for our solution.

B. Declarative Transformation Rules

We have adapted Xcerpt to support the construction of the adaptive service connectors – which is our central objective. We need to adapt Xcerpt to specifically address the needs of adaptive service connectors:

- From the technical point of view, in order to promote the 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 built ahead of the construction part of service connector.
- From business point of view, the data presentation of the global data model changes as element names change or

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

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

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

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

elements 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 specification will turn out to be essentially features.

Xcerpt is a document-centric language which is designed to query and transform XML and semistructured documents, therefore the ground rules that 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 it reaches a goal.

Fig. 2 shows a transformation example for a customer array based on Fig. 1. 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.

This original Xcerpt approach is 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 by us to pass the resource identifiers from the goal level all the way down to the ground rules.

In addition, we propose a mediator-based data integration architecture where the Xcerpt-based connectors are integrated with the BPEL-based Web services.

C. Implementation of Connector Construction

The construction of Xcerpt-based connectors 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 the customers' on demand. Therefore, exported data from a mediator service is the goal of the corresponding connector (query program), see Fig. 3.

We apply backward goal-based rule chaining in our 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.

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.

IV. Mediator Architecture

A. Motivation

Zhu et. al. [23] 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 [6], [20], [14], [23], 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. 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 and adaptivity, implemented by mediator services – which shall form the basis for our mediator architecture.

Garcia-Molina et.al. [5] identify that the following requirements are essential in order to build 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 from scratch.

B. Architecture Definition

The mediator architecture (Fig. 4) transforms local XML documents into documents based on a global schema. The data integration engine is built based on WS-BPEL, where component invocation orders are predefined in the integration schemas. Service orchestrations are defined by specifying the order in which operations should be invoked.

The proposed Web services based mediator architecture will contain the following components [16]:

- 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.
- Web 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 parameter 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 [8]. 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 project, 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, a declarative rule markup language based approach 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 SOAP messages into a set of pre-defined 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.

C. Software Process Model

The architecture in Fig. 4 explains the runtime view from the client and user perspective. Fig. 5 is the development perspective, looking at the developers of the architecture and their tasks. This is part of the development process model. It 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.

D. Application

Our approach is based on an incremental, evolutionary process model. Some pragmatic aspects of this process shall now be addressed.

In our proposed architecture, the unified data model (overarching 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 one release cycle. Mediators in our solution are used to deliver these data aspects according to the unified schema. This schema is available to the customers

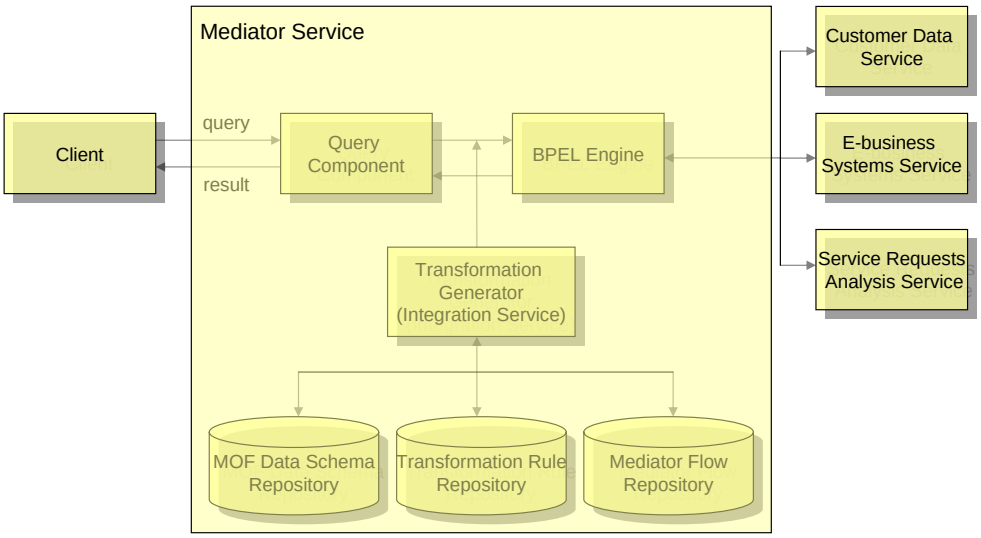

Fig. 4. Mediator Architecture for Adaptive Service-based Information Systems with sample Information Services (Customer Data, E-business System, Request Logging and Analysis Service).

also so that customers can decide which mediator to call based on the definition of the unified schema. A mediator consists of the following components: the individual provided Web services, a BPEL workflow, and one or more service connectors, see Fig. 4.

The focus of our study is not on the automatic composition of Web services, rather than 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 our BPEL process flow, we can taken a static approach with respect to the orchestration of the involved Web services. These can be orchestrated together in form of a BPEL flow built in advance.

At the development phase, the mappings between the global model and the local models will be expressed in at the abstract model level, with a language based on the MOF specification. Model transformation between different metamodels (MOF allowing to unify the metamodels definition) can 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 architecture, 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 have taken the approach in 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 more data models into a unified data model. Even in the case of source to destination model mapping, the users intervention is needed to select one from more than one sets of mapping 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 integration rules are created at the higher level than the Xcerpt query program itself as below, 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, that could be used to answer a users query directly. 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 rules mapping in the transformation generator serves 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 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 services – 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 targets two services, based on several composite integration and transformation rules.

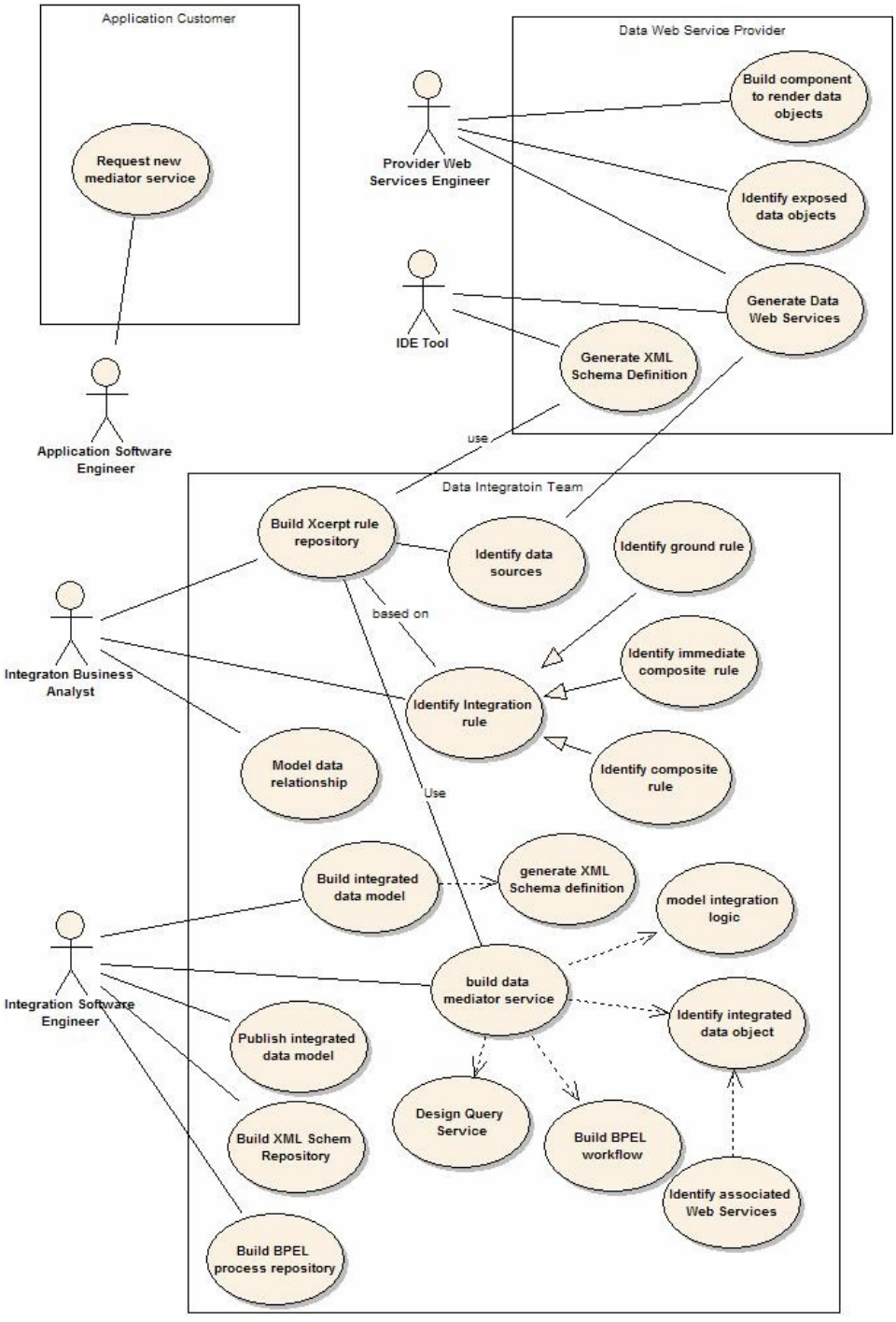

Fig. 5. Development of a Mediated ASP infrastructure – Overview of different Developer Roles involved.

V. Related Work

In contrast to XSLT, XQuery is a W3C-suppported query language aims at a syntax and semantics making it convenient for 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 querying XML. Its syntax uses query patterns and construction patterns and a query consists of a single select or traverse rule that separate 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, as in our case).

XML-QL is that uses query patterns and path expressions to select data from the XML sources. Such patterns can be augmented by variables for selecting data. One of the main characteristics of XML-QL is that it 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 have led us to choose Xcerpt as a language that satisfies all criteria that we have listed earlier on. Another candidate for this context that has emerged recently is the Query View Transformation (QVT) framework, which is another declarative model transformation language. While QVT satisfies the criteria, it is currently not as well supported through tools and accessible tutorial material.

VI. 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 adaptive service management has not yet been fully exploited. Our proposal aims to explore information integration technologies for adaptive 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 a 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 architecture from the deployment point of view.

Adaptivity in service-based software systems is emerging as a crucial aspect beyond the discussed area of service-based portals 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 semantical information. The recently widely investigated semantic Web services field, with ontology-based domain and service models, shall provide input for some planned extensions in this direction.

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 reengineering legacy systems and adopting service-oriented architecture to manage the information technology services [22]. 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 reengineering.

References

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

[2] BPEL. Business Process Execution Language for Web Services Version 1.1. [Online] Available from: http://www.ibm.com/developerworks/library/ws-bpel/. 2006.

[3] Bry, F. and Schaffert, S. Towards a Declarative Query and Transformation Language for XML and Semistructured Data: Simulation Unification. In Proc. Int. Conf. on Logic Programming. LNCS 2401, Springer-Verlag. 2002.

[4] Crnkovic, I and Larsson, M. A Case Study: Demands on Component-based Development. IN: Proc. 2nd International Conference on Software Engineering. pages 2331. ACM Press. 2000.

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

[6] Haller, A., Cimpian, E., Mocan, A., Oren, E. and Bussler, C. WSMX - a semantic service-oriented architecture. In Proc. Intl. Conf. on Web Services ICWS 2005. 2005.

[7] Jhingran, A.D. and Mattos, D. and Pirahesh, N.H. Information Integration: A research agenda. IN: IBM System Journal 41, no. 4, special issue on Information Integration [Online]. Available from : www.research.ibm.com/journal/sj/414/jhingran.pdf. 2002.

[8] Lenzerini, M. Data integration: A theoretical perspective. Proc. Principles of Database Systems PODS'02, ACM. pp. 233-246. 2002.

[9] Orriens, B. and Yang, J and Papazoglou, M. A Framework for Business Rule Driven Web Service Composition. IN: ER 2003 Workshops, Jeusfeld, M.A. & Pastor, O. Ed(s). LNCS 2814, pp. 52-64, 2003. Springer-Verlag Berlin Heidelberg. 2003.

[10] Peltier, M. and Bezivin, J and Guillaume, G. 2001. MTRANS: A general framework, based on XSLT, for model transformations. IN: WTUML01, Proceedsings of the Workshop on Transformations in UML .Genova, Italy. 2001.

[11] Peltier, M. and Ziserman, F. and and Bezivin. 2002. On levels of model transformation. IN: XML Europe 2002, pages 117, Paris, France, Graphic Communications Association. 2002.

[12] Reynaud, C. and Sirot, J.P. and Vodislav, D. Semantic Integration of XML Heterogeneous Data Sources. IN: Proc. IDEAS, pages 199208, 2001.

[13] Rosenberg, F., Dustdar, S. Business Rules Integration in BPEL - A Service-Oriented Approach. IN: Proceedings of the 7th International IEEE Conference on E-Commerce Technology. Munich, Germany. 2005.

[14] Sheth A. P. and Larson J. A. Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Computing Surveys, vol. 22, No. 3, pp. 183. 1990.

[15] Seltsikas, P. and Currie, W.L. Evaluating the application service provider (ASP) business model: the challenge of integration. IN: Proceedings of the 35th Annual Hawaii International Conference on 7-10 Jan 2002 Page(s):2801 2809. 2002.

[16] Stern, A and Davis, J. A Taxonomy of Information Technology Services: Web Services as IT Services. First International Conference on Service Oriented Computing, Trento, Italy. 2003.

[17] Stern, A. and Davis, J. 2004. Extending the Web services model to IT services. IN Proceedings IEEE International Conference on Web Services Page(s):824 - 825. 2004.

[18] Szyperski, C. Component Software: Beyond Object-Oriented Programming 2nd Ed. Addison-Wesley. 2002.
[19] Widom, J. Research problems in data warehousing. IN: Proceedings of 4th International Conference on Information and Knowledge Management. 1995.
[20] Wiederhold, G. Mediators in the architecture of future information systems. IEEE Computer, Volume 25. March 1992, pp. 38-49. 1992.
[21] Willcocks, L. P. and Lacity, M. C. The Sourcing and Outsourcing of IS: Shock of the New? IN: Strategic Sourcing of Information Technology: Perspectives and Practices.Willcocks, L. P. and Lacity, M. C. (eds) Chichester: Wiley. 1998.
[22] Zhang, Z. and Yang, H. Incubating Services in Legacy Systems for Architectural Migration. IN: Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC'04). Busan, Korea. pp. 196-203. 2004.
[23] Zhu, F. and Turner, M. and Kotsiopoulos, I. and Bennett, K. and Russell, M. and Budgen, D. and Brereton, P. and Keane, J. and Layzell, P. and Rigby, M. and Xu, J. Dynamic Data Integration Using Web Services. IN: 2nd International Conference on Web Services (ICWS 2004. San Diego, California, USA. 2004.