A Comparison of Data and Process Mediation Approaches

Rodrigo Mantovaneli Pessoa[1], Dick A. C. Quartel[2], Marten J. van Sinderen[1]

[1] University of Twente, 7500 AE Enschede, The Netherlands
[2] Telematica Instituut, 7500 AN Enschede, The Netherlands

{r.mantovanelipessoa, M.J.vanSinderen}@ewi.utwente.nl
{Dick.Quartel}@telin.nl

Abstract. In recent years, a huge amount of effort has been invested in the area of service discovery and composition. However, surprisingly little effort is being put into the evaluation of these approaches. The SWS Challenge is an ongoing and continuous experiment in developing a common understanding of various technologies intended to facilitate the automation of mediation, composition, and discovery for Web Services using semantic annotations. The mediation scenario problems concern making a legacy order management system interoperable with external systems that use a simplified version of the RosettaNet PIP3A4 specifications. The participants are supposed to be evaluated with focus on functional coverage. However, it turned out that it is extremely difficult to assess this in an objective manner. In this paper, we describe a framework for comparison of data and process mediation approaches. As a case study, we apply our framework to perform a comparative analysis of four participants from the SWS Challenge.

Keywords: Data and Process Mediation, Enterprise Application Integration (EAI), Semantic Web Service Challenge

1 Introduction

One of the most engaging promises of Service Oriented Architectures (SOA) is to enable the construction of flexible and loosely coupled business applications, spanning over several networked enterprises capable of interconnecting their applications and share data by combining a set of services. As services mature to suit the basic building blocks of Service Oriented Architectures, the service composition paradigm is becoming one of the main concerns of the application development process. Some already raised questions related to services are: how to specify them in an expressive enough language, how to compose them, how to discover them through the distributed environment, and how to ensure their correctness.

However, the multiplicity and diversity of the proposed approaches attests a lack of consensus on the most appropriate technologies and methodologies to compose services. The Semantic Web Service Challenge is an initiative aiming to develop a common understanding of various technologies intended to facilitate the automation of mediation, discovery and composition for services using semantic annotations. The evaluation process is performed by teams composed of workshop organizers and peer participants with focus on evaluating the functional coverage, i.e. on whether a particular level of the problem could be solved by a particular approach. However it turned out that it is extremely difficult to assess this in an objective manner [1].

Motivated by this fact, we developed a framework for comparison of mediation approaches. The framework is expressed in terms of quantitative and qualitative evaluation points in order to clarify and expose different aspects involved in features supported by a method or tool. As a case study, we applied our framework to perform a comparative evaluation of four participants from the SWS Challenge, around the mediation scenario.

The remaining of this work is structured as follows: Section 2 presents the mediation scenario proposed by the Semantic Web Services Challenge. Section 3 introduces our comparison framework. Section 4 describes four different approaches for data and process mediation. In Section 5, the comparison is conducted and summarized. Finally, Section 6 presents our conclusions and defines some future research directions.

2 The Mediation Problem: Purchase Order Scenario

This session describes the static mediation scenario proposed by the Semantic Web Services Challenge. This problem centres around a simple purchase order scenario between two companies: Moon and Blue. The manufacturer Moon has signed an agreement with the company Blue, to exchange purchase order messages in RosettaNet PIP 3A4 format. RosettaNet is an industry-driven standard for B2B integration that represents an agreement on the message exchange patterns, the message content and a secure transportation mechanism among business trading partners in a supply chain network. The Blue's system has to interact with Moon's legacy system, also provided as a set of Web services, which however do not use the RosettaNet standard. The objective of the SWS Challenge is to build a system called Mediator, which compensates the differences in communication between the involved parties by solving possible data and behaviour mismatches.

The subsequent levels of the SWS Challenge addresses the mediation problem by asking its participants to, while minimizing direct intervention from programmers, effectively and quickly react to incremental changes of the application requirements built on top of the static scenario. Those solutions that were still able to tackle the problem are then ranked in different levels of adaptability.

2.1 The Static Mediation Scenario

The static scenario involves the mediation between two companies, Blue and Moon, within a stable (static) context: the protocols, the messages, and the data formats are known a priori and fixed. In the scenario discussed above, the company Moon uses two back-end (legacy) systems to manage its order processing, namely, a Customer Relation Management System (CRM) and an Order Management System (OM).

As illustrated by Figure 1, the customer Blue sends a RosettaNet order request and expects that, upon the request being submitted, the order will be processed and a purchase order confirmation will be received, acknowledging that the order was received and processed by the company Moon. Messages in RosettaNet PIP 3A4 format enable a buyer to issue a purchase order and to obtain a quick response from the provider that acknowledges which of the purchase order product line items are accepted, rejected, or pending. As mentioned before, the company Moon only offers a set of legacy Web Services that do not fit with the RosettaNet standard. The mediator is in charge of receiving a single RosettaNet message (containing all the order details) from Blue and splitting it to the various messages needed by Moon to create and handle a purchase order. In this way, the mediator will have to orchestrate a sequence of services provided by Moon and translate the set of confirmation messages into a whole RosettaNet Purchase Order Confirmation to be sent back to Blue.

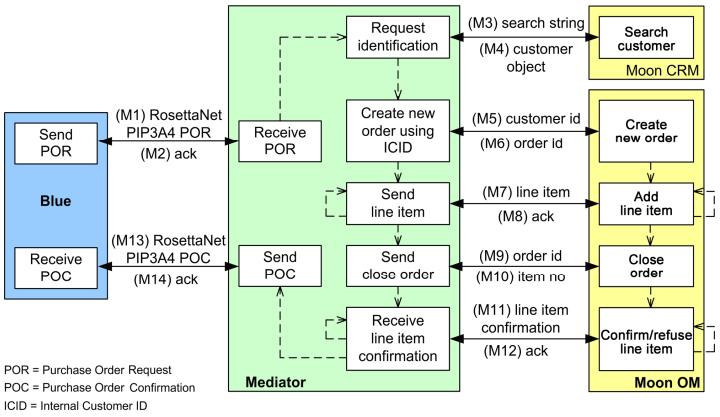

Fig. 1. – Mediation Scenario Overview.

At first, the Mediator receives a Purchase Order Request message from the customer Blue. The Purchase Order Request message is synchronously confirmed by an Acknowledgement of Receipt message. However, in order to orchestrate Moon to process a RosettaNet purchase order, several steps have to be made.

First, the customer needs to be identified by sending a search string to Moon's CRM system. The internal costumer identification number is obtained by invoking the *searchCustomer* operation. As a next step, the creation of a new order is requested by sending the costumer identification number to Moon's OM system invoking and invoking the *createNewOrder* operation, which returns the id of the newly created order. After a new order is created, Moon's OM system expects all order lines to be

added one by one by invoking *addLineItem* operation (possibly for many times). Finally, once all the line items are submitted, Moon OM system is requested to close the order (*closeOrder* operation) and returns the number of items that has been received. Subsequently, Moon's OM system confirms the status of each order line, which is acknowledged synchronously the mediator. After all order lines have been confirmed, a RosettaNet PIP3A4 Purchase Order Confirmation message is sent to Blue and confirmed synchronously by an Acknowledgement of Receipt message

3 The Comparison Framework

This section describes our framework for comparison of mediation approaches. In order to develop our framework, we use the DESMET method [2], a comprehensive methodology for assisting organisations and academic institutions to plan and execute unbiased and reliable evaluation exercises. This method identifies such an evaluation as a qualitative or subjective evaluation and enables the framework to be expressed in terms of a set of common (mandatory and/or desirable) features supported by a method or tool.

Quantitative or objective evaluations are based on identifying the expected benefits and drawbacks of a new method or tool in measurable terms. Qualitative or subjective evaluations assess the appropriateness of a method/tool in terms of the features provided by the method/tool, the characteristics that distinguish this method/tool from others, support offered by the method/tool supplier and its training requirements. This type of analysis is usually based on the identification of the requirements that potential users have for performing a particular task and the mapping of those requirements to features that a method/tool (intend to support that task) should possess. The main activities involved in carrying out a feature analysis are [2]:

1. Select a set of candidate method/tools to evaluate.
2. Decide upon the required properties or features of the item being evaluated.
3. Prioritise those properties or features with respect to the requirements of the method/tool users.
4. Decide the level of confidence that is required in the results and therefore select the level of rigour required of the feature analysis.
5. Agree on a scoring/ranking system that can be applied to all the features.
6. Allocate the responsibilities for carrying out the actual feature evaluation.
7. Carry out the evaluation to determine how well the methods/tools being evaluated meet the criteria that have been set.
8. Analyse and interpret the results.
9. Present the results to the appropriate decision-makers.

As shown in Figure 2, our framework involves both qualitative and quantitative elements, structured into five main features: *data mediation*, *process mediation*, *correctness*, *suitability of design concepts* and *level of effort required to drive changes*. Under *data mediation* and *process mediation* features, we consider both design time and runtime aspects of the mediation task. The first refers to the design support provided by each approach as well as the steps needed to implement each solution, whereas the second refers to characteristics concerning their execution.

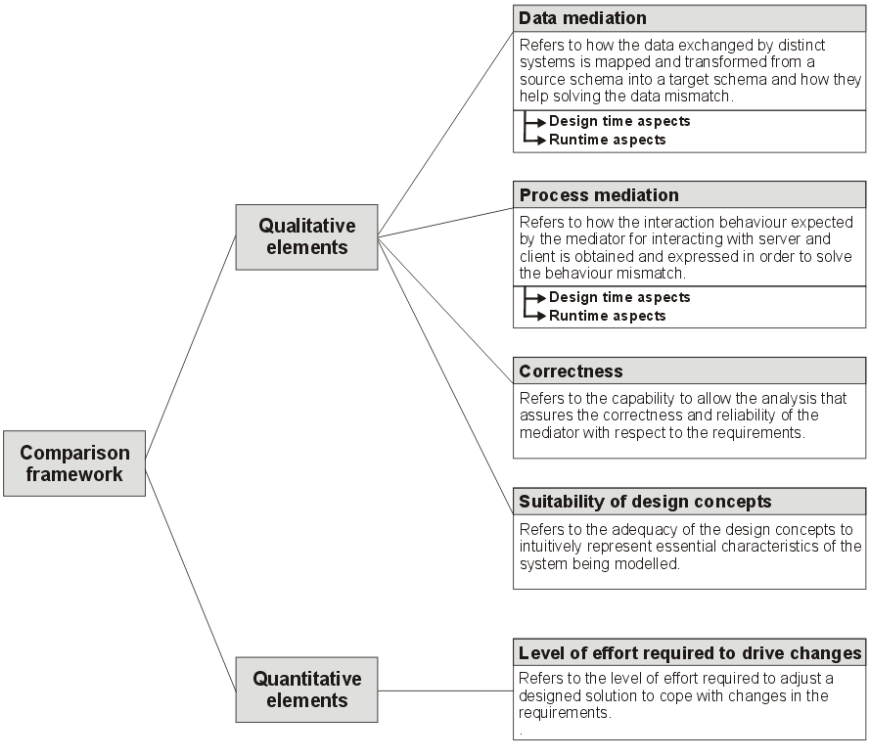

Fig. 2. – The Comparison Framework Elements.

In addition to the separation between quantitative and qualitative evaluations, there is another dimension to an evaluation: the way in which the evaluation is organised. DESMET has identified three rather different ways of organising an evaluation exercise, including: formal experiment (where many subjects are asked to perform a variety of tasks using the different methods/tools under investigation), case study (where each method/tool under investigation is tried out on a real project) or a survey (where subjects that have used a specific method/tool on past are asked to provide information about the method or tool). As mentioned before, as a case study we have adopted the mediation scenario proposed by the SWS Challenge, where different mediation approaches addressing the same real world problem scenario have been peer reviewed and documented.

4 Data and Process Mediation Approaches

In this section, we briefly describe different approaches proposed to address the mediation scenario offered by the SWS Challenge. Based on past studies, we have selected four well-documented approaches which have shown some distinctions in their realization.

4.1 WSMO, WSML and WSMX

The DERI (Galway and Innsbruck) team based its solution on the Web Service Modelling eXecution environment (WSMX) [7]. WSMX is a reference implementation of the Web Services Modelling Ontology (WSMO) [6] and operates using the Web Services Modelling Language (WSML) [8]. The approach incorporates four core elements that are needed to represent semantic web services and related issues: ontologies, that provide the common terminology used by other WSMO elements, services that are requested, provided, and agreed upon by requesters and providers, goals that represents a desire that a client delegates (which should be solved by services), and mediators, which deal with interoperability problems between different WSMO elements.

During design time, the design and implementation of adapters, creation of WSMO ontologies and services, rules for lifting/lowering, and mapping rules between ontologies are carried out for the RosettaNet, OMS and CRM systems. The run-time phase involves discovery, selection and execution of the appropriate services to mediate the interaction between Blue and Moon systems. The general view of the approach is shown in Figure 3.

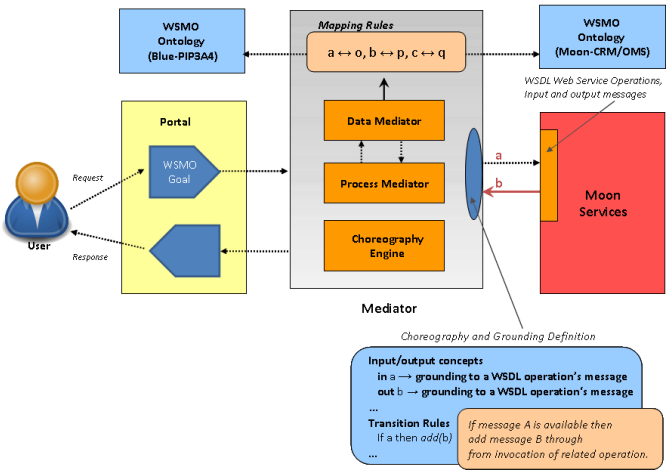

Fig. 3. – General view of the approach

Initially, ontologies describing the information model used by each involved party are manually designed, after careful analysis of the schemas for the RosettaNet

messages and the WSDL service descriptions offered by CRM and OMS systems. In the given scenario, both Blue and Moon use different information models and the data mediation is accomplished through mappings between RosettaNet and CRM/OMS ontologies. In particular, a mapping can specify that classes from two ontologies are equivalent while transformation rules use logical expressions to unambiguously define how the data encapsulated in an instance of one class can be encapsulated in instances of the second class. During run time, if there is a need for data to be mediated, the source instances are provided to the data mediation component, which has the role to derive the target data instances from the source data instances.

In WSMO, requestors of a service express their objectives as goals, which are high level descriptions of concrete tasks. From this point of view, a WSMO goal description consists of a requested capability and requested interfaces. The former shall specify the objective to be achieved in terms of a capability from the client perspective. The latter is intended to specify the communication behaviour for automated Web service usage supported and required by the client. A goal template is a generic objective description that is defined at design time and a goal instance denotes a concrete client request that is created at runtime by instantiating a goal template with concrete values. One advantage of this approach is that the requestor only has to provide a declarative specification of what it wants, and does not need to have a fixed relation with the Web Service or to browse through an UDDI registry for finding Web Services that provide the appropriate capability.

In order for this goal to be accomplished, the requestor has to find an appropriate Web Service which may fulfil the required task. Similar to the way the requestor declares its goal, every Web Service has to declare its capability (that is, what it is able to accomplish) in terms of its own ontology. A WSMO Web service description consists of two central parts. At first, the capability describes the overall functionality provided by a Web service in terms of pre-conditions, assumptions, post-conditions, and effects; these are logical expressions, specified e.g. in WSML. Secondly, the interfaces describe the interaction behaviour supported by a Web service. To cope with impossibility of service requester and provider to communicate with each other due to heterogeneity of their communication protocols, WSMO introduces the mediator concept, which has the task of overcoming the heterogeneity problems, both at data level and at behaviour level.

The WSMX process representation is similar with the WSMO choreography definition, which representation is based on Abstract State Machines (ASM), consisting of states and guarded transitions. A state is described by the WSMO ontology and the guarded transitions (transition rules) are used to express changes of states by means of transition rules. It falls into process execution based on underlying rich knowledge base formalism where an ASM is used to abstractly describe the behaviour of the mediator. In the utilized Abstract State Machines (ASM), the domain ontology constitutes the underlying knowledge representation and transition rules (specified in terms of logic formulas) describe how the state changes when a transition is executed. For the purposes of the SWS Challenge, the provided solution has the assumption that the invocation order is unimportant, but that is not the case: there is an order in which the operations should be correctly invoked.

At this point, both Blue and Moon back-end systems have semantically rich descriptions of the information models and behaviour (choreography) of both

systems. This, along with additional mappings between the ontologies of the Blue and Moon systems, allows both choreographies to "connect" at run-time and resolve process interoperability issues (mediate between both choreographies). One of the main advantages of the WSMX-based integration is the strong partner de-coupling. As opposed to traditional centralized solution (when a central workflow would solve this integration problem), this approach enables the automatic adaptation when changes to service descriptions are introduced. In contrast, solutions based on a central workflow would additionally require changes to the workflow type definition.

4.2 SWE-ET: Semantic Web Engineering Environment and Tools

The team composed of Politecnico di Milano and CEFRIEL based its solution on the SWE-ET [3] framework. SWE-TE is a framework for designing and developing Semantic Web Service applications, based on existing models for the specification of business processes (such as BPMN [4]) combined with Web engineering models for designing Web applications (such as WebML [5]), with strong emphasis on graphical process modelling.

The approach aims to lead the designer from the process modeling to the running Web application by producing some intermediate artifacts (BPMN models, data models, hypertext models). Such models are enriched by imported ontological descriptions and transformed into a WSMO specification: the ontology is derived from the process model, data model, and hypertext model; the service capability description is derived from the hypertext model; and the choreography information is derived from the process model and the hypertext model. Later, the execution is delegated to a Semantic Execution Environment (e.g. WSMX). Figure 4 provides an overall picture of the approach.

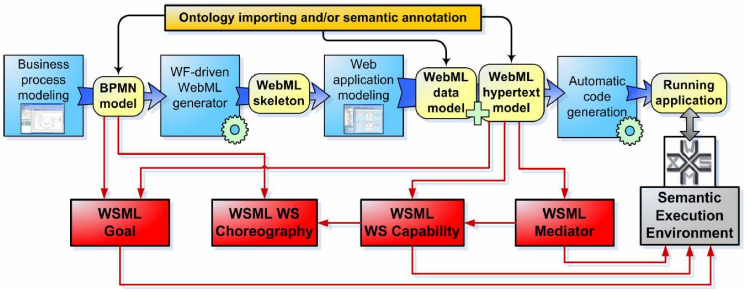

Fig. 4. – Overall picture of the approach.

The specification of the mediator consists of a set of models: the application data model (an extended Entity-Relationship model), one or more hypertext models (i.e., providing different site views for different types of users), expressing the navigation paths and the page composition of the Web application; and the presentation model, describing the visual aspects of the pages.

Initially, the RosettaNet message schemas and the service descriptions offered by Moon systems were analysed and a corresponding data model was manually obtained from it. The WebML data model is the standard Entity-Relationship (E-R) model and

the conversion from RosettaNet messages is handled by Adapter units that use XSLT for transforming messages in an XML format compatible with WebML's internal data format (WSML). In the same way conversion to and from Moon legacy messages are handled by proper XSLT stylesheets that act as templates for SOAP messages and that are then populated by runtime queries.

After modeling the data structures, a high level Business Process Modelling Notation (BPMN) model is created representing the mediator. This model formalizes the orchestration of the Moon Web services and defines states pertaining to the mediation process as by the SWS Challenge specification. The BPMN notation allows one to represent all the basic process concepts such as data and control flow, activity, actor, conditional/split/join gateways, event and exception management, and others. BPMN activities can be grouped into pools, and one pool contains all activities that are to be enacted by a given process participant. The elements of the workflow model (e.g., activity, names, and lanes) are extracted as semantic concepts and used as additional piece of the ontology. If a lane is identified as a mediator at the BPMN level, the basic information about the design of the mediation can be extracted from high-level BPMN description of the interactions (in particular, basic information about possible choreography, interface and parameters of the service).

Then, the BPMN model is used to automatically generate a WebML skeleton that is manually refined. The WebML [5] service model allows one to define different hypertexts (e.g., for different types of users or for different publishing devices), called *site views*. A site view is a graph of *pages*, allowing users from the corresponding group to perform their specific tasks. *Pages* consist of connected *units*, representing publishing of atomic pieces of information, and operations for modifying the underlying data or performing arbitrary business actions. Units are connected by links, to allow navigation, parameter passing, and computation of the hypertext from a unit to another. The WebML conceptual model offers standard workflow units to model control flow and has been extended with Web service units to describe Web services interactions. These units correspond to the WSDL classes of Web service operations, including request-response and one-way operations. Distributed processes can be obtained by combining the workflow units and Web services units. The language is extensible, allowing for the definition of customized operations and units.

Once the business process has been designed, workflow constraints must be turned into navigation constraints among the pages of the activities of the hypertext and into data queries on the workflow metadata for checking the status of the process, thus ensuring that the data shown by the application and user navigation respect the constraints described by the specification.

Then, the WSMO description of the mediator can be derived from the WebML diagrams. This specification can be used to generate a working Web Service providing mediation between Blue and Moon Web Service.

4.3 jABC/jETI Framework

The jABC/jETI solution is realized within the jABC framework [9], an environment for model-driven service orchestration based on lightweight process coordination. jABC originated in the context of the verification of distributed systems and use SLGs

(Service Logic Graphs) as choreography models, allowing users to easily develop services by composing reusable building blocks into (flow-)graph structures. These basic building blocks are called SIBs (Service Independent Building Block) and the development process is supported by an extensible set of plug-ins that provide additional functionality.

SIBs have one or more edges (branches), which depend on the different outcomes of the execution of the functionality represented by the SIB. Each SLG model can be wrapped into a single coarser-grained SIB, and may be used on another hierarchical level of modelling. Similarly, each SIB can be refined into an own model, showing a more detailed view on the represented feature. The provided model driven design tools allow modelling the mediator in a graphical high level modelling language and supports the derivation of an executable mediator from these models. Figure 5 shows an overview of the described approach.

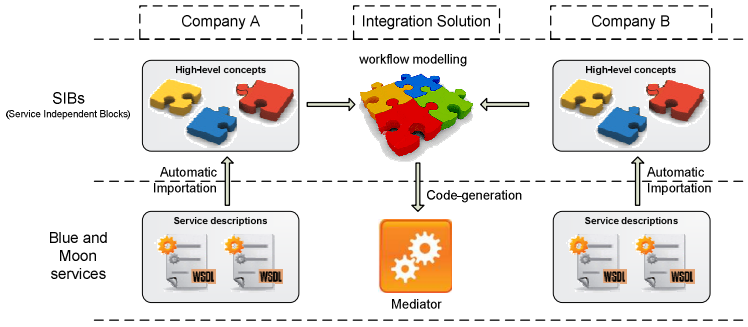

Fig. 5. – Approach Overview.

Initially, the corresponding SIBs are automatically generated from the WSDL descriptions of the web services provided by the Moon legacy system. At this step, the SIB generator extracts the information about the functions defined in the WSDL service descriptions and creates a SIB for each function. The structure prescribed by the original WSDL service descriptions and RosettaNet Schemas is then mapped into the structure of the SIB parameters, using the pre-existing graphical user interface of the jABC. As a result, the messages are created within the SIBs according to the structure prescribed by the original WSDL descriptions, which is reflected and mapped into the hierarchical parameter structure of the SIBs.

These parameters and the SIB branch labels are visible to the model checker, which allows automatically proving global compliance constraints on the business logic of an SLG. These constraints are expressible in mu-calculus and its derivatives, a family of modal (temporal) logics. Additionally, arbitrary relations between data elements can be provided as local checking expressions, with the expressiveness of Java. This facility allows expressing and checking pre and post conditions.

Next, the mediator is manually modelled as a workflow, by dragging and dropping elements from the palettes of standard and generated SIBs. The modelling activity can then be complemented by analysis, verification and simulation techniques, provided by a set of plug-ins. At this point, the mediation model consists of a structured coordination graph and is interpreted by the tracer plug-in as a flow graph with one or more distinguished start nodes.

To export the mediator as a Web service, the composite and hierarchical SLG of the mediator is first transformed into a single SIB, using the *subgraph* feature of the jABC. This creates a Graph-SIB that represents the corresponding SLG. Its implementation is the argument SLG, executable within the jABC Tracer, the interpreter (or a virtual machine) for SLGs. The tracer is able to execute the mediation model comparable to a standard debugger in *run mode* or *step mode* and using *breakpoints* or *pause* to stop the execution. However, to provide a Web service mediator that is completely independent of the jABC, the code generator plug-in is used to obtain executable source code from the Graph-SIB. This code is then deployed on a server using the AXIS framework, this way making the functionality accessible to other users and generating a WSDL description that contains all the necessary information to access the deployed service as a web service.

4.4 COSMO Framework

This approach proposes the use of the COSMO framework [12] for service modelling and refinement in order to raise the level of abstraction at which problems such as mediation and integration of legacy systems are usually solved. In terms of Model Driven Architectures, this means that platform-specific (service) models (PSMs) of Blue and Moon are transformed into platform-independent (service) models (PIMs) by removing all platform-specific details. Next, the approach adds additional semantics to the service PIMs of Blue and Moon in order to make them more precise (e.g. the semantics of service requests and the relations among service operations are explicit modelled). In this way, the solution of the mediation problem is captured in the service PIM of the Mediator. In the final step, a concrete implementation (the mediator PSM) is derived from this PIM by adding technology-specific details. The approach is illustrated in Figure 6.

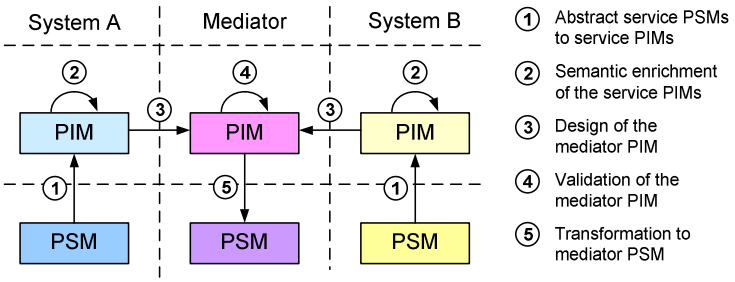

Fig. 6. – General view of the approach

First, to cope with the problem of data mismatches, the platform-independent information models of the Blue and Moon, hereby referred to as domain-specific ontologies and expressed in OWL, were partially derived using the *types* section of the WSDL descriptions of Moon and Blue systems. The platform-independent behaviour models are partly derived using the *interface* section of the WSDL descriptions of Moon and Blue. These behaviour models are expressed using Interaction System Design Language (ISDL) and the lifting of the interface section of WSDL to ISDL is supported by an integrated editor and simulator for ISDL.

A WSDL *types* section defines only the syntax of the messages that are exchanged between the service provider and its users. Therefore, some further manual work is required to define the semantics of these messages (e.g. hidden assumptions should be made explicit by defining new classes and relations among them). Next, mappings between classes, properties and individuals from Blue's and Moon's domain-specific ontologies are defined.

A WSDL *interface* section defines only its constituent messages and message exchange patterns in a *single* operation. Hence, the complete behaviour model should also define the relationships between the different operations. Since these relationships are not part of the WSDL descriptions they have to be derived from the informal textual descriptions as provided in the mediation scenario. In this way, the integrated behaviour model, describing the possible message exchanges between Blue's and Moon's services, is manually refined from combining concepts provided by the COSMO framework and defining the relationships between their executions.

The core concept underlying the COSMO framework constitutes the *interaction* concept, which represents an activity in which the involved systems produce some common result in cooperation. An interaction is defined by a composition of two or more *interaction contributions*, which represent the participation (or responsibility) of each system involved in the interaction. Consequently, an interaction is considered an atomic activity that either occurs and establishes the same result for all involved systems, or does not occur for any of the systems and therefore does not establish a (partial) result. Additionally, the *action* concept provided by the framework models an activity performed by a single entity and the *causality relations* model how actions and interaction contributions depends on other actions or interactions contributions.

Once the integration solution is specified at the business service layer, it can be early subjected to various analysis and simulation techniques. This is done by applying horizontal transformations to the service model, which are related to transform the service behaviour into a formal specification, which can be then tested and verified to assure the correctness of the derived design with respect to its specification.

After the validation and simulation of the interaction models specified at the business service layer, an IT integration solution can be semi-automatically derived by applying a number of model transformations and refinements. In this step, the behaviour model of the mediator is transformed into a BPEL specification. However, before this mapping can be applied a preparatory step is needed in which the behaviour model of the mediator is annotated with marks and possibly restructured. Marks are used to add implementation details (e.g.: interaction contributions should be marked to indicate whether they have to be mapped onto an invoke, receive or reply activity in BPEL). Furthermore, information about partner links and invoked web services (e.g., namespace URI and endpoint address) may have to be provided.

5 Comparison

Table 1 summarizes, according to our framework, the profiles of the proposed solutions, which are commented and described in more detail below:

		WSMO	WebML	jABC	COSMO
Data mediation	**Design time aspects**	Ontologies manually created from analyzing the RosettaNet messages and WSDL service descriptions. Ontology to Ontology mappings.	ER-model manually created from analyzing the RosettaNet messages and WSDL service descriptions. XML to Ontology mappings.	SIBs and hierarchical parameters automatically generated from WSDL service descriptions. XML to SIB parameters mapping.	Ontologies partially generated from RosettaNet messages and WSDL service descriptions. Ontology to Ontology mappings
	Runtime aspects	Mappings execution on the instance level.	Mappings execution on the instance level.	Reflected into the hierarchical parameter structure of the SIBs.	Mappings execution on the instance level.
Process mediation	**Design time aspects**	Defining services capability, choreography interfaces and goal templates. Behaviour modelled as Abstract State Machines by means of transformation rules.	Defining BPMN model, hypertexts and constraints. Behaviour specified at a high level of abstraction is transformed into a hypertext model for further manual refinement.	Defining a workflow explicitly describing the behaviour of the mediator. Behaviour modelled in terms of control flow graphs based on fork/join parallelism.	Defining a workflow explicitly describing the behaviour of the mediator. Behaviour modelled in terms of interactions, operation calls and causality relations.
	Runtime aspects	Execution based on abstract state machines and transformation rules defined by choreography.	WebML model is transformed into a WSMO specification and execution is delegated to a Semantic Execution Environment (WSMX).	Model-to-code transformations are defined to generate the implementation code and the execution tree is defined as the unfolding of the marking graph of the mediator.	Simulator tool able to execute the behaviour models. In addition, the Mediator was transformed into a BPEL process and its execution delegated to a BPEL engine.
Behaviour correctness		No explicit support.	No explicit support.	Formal verification capability based on temporal logic formulas expressed in mu-calculus.	Formal verification capability based on ISDL techniques.
Suitability of design concepts		Appropriate (mediators, goals, services and ontologies).	Sufficient, but not intuitive (pages, units, hypertexts, and links).	Appropriate (Service Independent Building Blocks and hierarchical parameters).	Appropriate (Goals, operations and Interactions).
Level of effort required to drive changes* ***Assessed by peer reviews at the SWS workshops.**		Low (level 3)	Low (levels 3)	Medium (level 2)	Not evaluated

Table 1: Comparison of the described approaches.

The profiles presented in Table 1 illustrate that, while the primary aim of the four approaches summarized above is to solve the mediation problem described by the SWS Challenge, their realization differ in several important aspects.

The WSMO approach reflects its four top elements by explicitly modelling goals, mediators, services and ontologies. Ontology-to-ontology mediation is achieved through the design and implementation of adapters specifying mapping rules between ontologies. The approach stresses the importance of the mediators, treated as first class citizens, as the core concepts to describe elements that overcome interoperability problems. Goals are described as requested capability and requested interfaces. From the perspective of a Goal description, the capability describes the functionality that the owner of the Goal wishes to achieve from a Service. Analogously, the capability of a Service describes the functionality offered by that service. The approach focus was on modelling semantically enhancing Web Services description, services requests (expressed as goals) and mediators. The adopted goal-oriented paradigm facilitates the Web Service's discovery by a potential client, the selection of the most appropriate service for a certain task, the actual invocation of a service and the composition of multiple services for accomplishing a common task.

On the other hand, the other approaches focus more on the modelling of the mediator internal logics. The WebML approach starts modelling a BPMN workflow, specified at a high level of abstraction. This model is then transformed into hypertext diagrams, representing the service execution chains, and need to be refined later by the designer. The design concepts provided by the hypertext diagram, originally developed in the context of conceptual modelling of Web pages and applications, were adapted to the mediation purpose and showed to be sufficient to model a mediation solution, but not in an intuitive way. The data mediation is handled by Adapter units that are configured by a proper XSLT stylesheet that transforms messages in an XML format compatible with WebML's internal ontology format.

The jABC approach automatically imports basic service types (called SIBs, Service-Independent Building Blocks) from the WSDL service descriptions. The designer is then responsible for the specification of the behaviour models, defined as SLGs (Service Logic Graphs), by composing the reusable building blocks into (flow-)graph structures. Behavioural properties of the modelled business logic can be expressed as logic formulas and the provided model, which describes the mediator behaviour, can be analysed in early stages of the design process to check the correctness with respect to its specification. Formal verification capability of the service models is greatly appreciated since it simplifies debugging complex processes directly on the model, possibly reducing development cycle time and increasing robustness of the system. The approach handles data mediation by mapping the structure prescribed by the original WSDL service description into hierarchical SIB parameters (additional semantic properties attached to the SIBs). A derivation of an executable mediator from these models is obtained by applying model-to-code transformations.

Similarly to the WSMO approach, the COSMO approach employs ontologies as the underlying information model. This allows for reasoning to assess whether the relations defined between classes and properties are violated at the instance level or if a common interaction result can be established by matching input and output services parameters. Based on the selected match, the signature for the required data

transformation can be obtained automatically. In particular, the approach focuses in applying reasoning techniques to automate parts of the mediator design process. The mediator behaviour is specified as a workflow explicitly modelling interactions between services, operation calls and causality relations between then. Formal verification and analyse of the behaviour models is also supported. The simulator tool is able to execute the behaviour models by performing real web service invocations and incorporating the results that are returned by web services into the simulation. In addition, the Mediator was transformed into a BPEL process and its execution delegated to a BPEL engine.

The level of effort required to adapt each mediator solution to cope with the new changes proposed to the mediator scenario has been assessed by peer reviews at the SWS workshops. For practical reasons, these assessments were adopted and incorporated in our comparison study. There are four possible levels of success that evaluate the transition of the designed solution from one problem level to another. The initial mediation scenario, described in section 2, corresponds to level 0 (static mediation). On top of this static scenario were added various levels, each corresponding to a general kind of problem, and each with sublevels of complexity. In this sense, a higher evaluation success level indicates a better solution to the problem level transition. Since the COSMO team only participated in the first edition of the workshop, their solution has not been assessed by peer review yet.

6 Conclusion

In this paper, we have presented a framework for comparison of data and process mediation approaches. The proposed framework establishes a common set of criteria that provide basic guidelines for the evaluation process, enabling a more comprehensive understanding of existing mediation approaches by exploring and making more explicit their possibilities and limitations. In order to assess the features and aspects defined in our framework, the DESMET method for Feature Analysis has been used. This type of analysis identifies an evaluation as a quantitative or qualitative evaluation. In particular, our framework involves both objective and subjective elements and the assessment to which the approaches provide the required features was based on literature review and personal opinion.

As a case study, we applied our framework to perform a comparative analysis of four approaches aimed to solve the mediation problem described by the SWS Challenge. The mediation scenario is pretty close to a real world integration problem involving data and process mediation and has showed to be complex enough to stress the compared solutions. In addition, by applying our framework, we could expose and evidence the advantages and drawbacks of each approach and show that their realization differs in several important aspects.

With our framework, we hope to help the SWS Challenge community by describing and comparing these approaches and providing a comprehensive overview about the underlying concepts, assumptions and promising practices of each approach, including methods, principles and techniques involved in data and process mediation tasks.

Acknowledgments

This work is part of the Freeband A-MUSE project. Freeband is sponsored by the Dutch government under contract BSIK 03025.

References

1. Petrie, C., Margaria, T., Käuster, U., Lausen, H., Zaremba, M. (2007): SWS Challenge: status, perspectives and lessons learned so far. In Proceedings of the 9th International Conference on Enterprise Information Systems (ICEIS2007), Special Session on Comparative Evaluation of Semantic Web Service Frameworks, Funchal, Madeira-Portugal.
2. Kitchenham, B. (1996): DESMET: A method for evaluating Software Engineering methods and tools Technical Report TR96-09. Department of Computer Science, University of Keele, Staffordshire.
3. M. Brambilla, I. Celino, S. Ceri, D. Cerizza, E. Della Valle, F. M. Facca (2006): A Software Engineering Approach to Design and Development of Semantic Web Service Applications, In Proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, GA, USA, 5-9 November 2006, LNCS 4273, pp. 172-186.
4. White S. A. (2004). Business Process Modeling Notation (BPMN), BPMI.org, http://www.bpmi.org/ bpmi-downloads/BPMN-V1.0.pdf
5. S. Ceri, P. Fraternali, and M. Matera (2002): Conceptual Modeling of Data-Intensive web Applications. IEEE Internet Computing, 6(4).
6. D. Roman, U. Keller, L. Lausen, J. de Bruijn, R. Lara, M. Stollberg, A. Polleres, C. Feier, C. Bussler, and D. Fensel (2005): Web Service Modeling Ontology, Applied Ontologies, vol. 1, pp. 77-106.
7. Mocan, A., Moran, M., Cimpian, E., and Zaremba, M. (2006): Filling the gap - extending service oriented architectures with semantics. In ICEBE, pp. 594-601. IEEE Computer Society."
8. de Bruijn, J. , H. Lausen, A. Polleres, D. Fensel (2006) The Web Service Modeling Language WSML: An Overview. In Proceedings of the 3rd European Semantic Web Conference (ESWC 2006), Budva, Montenegro: Springer, LNCS 4011.
9. Steffen, B., Margaria, T., Nagel, R., Jörges, S., and Kubczak, C. (2006). Model-Driven Development with the jABC. In Proceedings of Haifa Verification Conference, LNCS N.4383. Springer Verlag.
10. Müller-Olm, M., Schmidt, D., and Steffen, B. (1999). Model-checking: A tutorial introduction. In SAS, 6th In: Static Analysis Symposium, LNCS N.1694, pages 330–354. Springer Verlag.
11. Dick A. Quartel , Maarten W. Steen , Stanislav Pokraev , Marten J. Sinderen (2007): COSMO: A conceptual framework for service modelling and refinement, Information Systems Frontiers, v.9 n.2-3, p.225-244.