# Chapter 18
# The COSMO Solution to the SWS Challenge Mediation Problem Scenarios: An Evaluation

**Camlon H. Asuncion, Marten van Sinderen, and Dick Quartel**

**Abstract** During the course of our participation in the Semantic Web Services (SWS) Challenge, we have shown how the concepts defined in the *CO*nceptual *S*ervices *MO*deling (COSMO) framework for the modeling, reasoning and analysis of services can be used to solve the Mediation Problem Scenarios of the Challenge. Along with the service-oriented refinement and composition paradigm of COSMO, our approach is also based on model-driven and goal-oriented principles where the semantic integration of applications is designed at a layer of abstraction higher than technology specifications. The objective of this paper is to evaluate our previous and current research efforts towards advancing our solution to the semantic integration of service-oriented applications, particularly, using the mediation problem scenarios of the Challenge. We do this by presenting the state of the art of our solution while reporting our experience with applying our solution to the scenarios including lessons learned and identified research challenges.

## 18.1 Introduction

In today's times, enterprises must adapt to a constant change in business demand in order to survive and stay competitive. A collaboration with other enterprises in order to add value to their products and/or services is therefore vital. Enterprise collaboration is now possible with current advances in information technology. However, achieving such collaboration effectively and efficiently is never an

C.H. Asuncion (✉) · M. van Sinderen
Center for Telematics and Information Technology (CTIT), University of Twente,
P.O. Box 217, 7500, AE, Enschede, The Netherlands
e-mail: c.h.asuncion@utwente.nl; m.j.vansinderen@utwente.nl

D. Quartel
Novay, P.O. Box 589, 7500, AN, Enschede, The Netherlands
e-mail: dick.quartel@novay.nl

M.B. Blake et al. (eds.), *Semantic Web Services*, DOI 10.1007/978-3-642-28735-0_18,

easy task. For one, enterprises are faced with system interoperability problems as a result of using legacy systems, oversupply (or the lack) of standards, heterogeneous hardware and software platforms, etc. Integration solutions also need to be flexible so that enterprises can continuously adapt to current and future changes and demands from within and outside their business environment [27].

Our work with the SWS Challenge testifies to our continued interest in designing a reusable and flexible interoperability solution. Our approach over the years has evolved into combining goal-oriented, model-driven and service-oriented design principles in designing interoperability solutions in the context of service mediation. A goal-oriented approach keeps the solution problem-oriented rather than technology-oriented: A change in the business requirements should not adversely affect the underlying technology implementation. Model-driven techniques raise the problem and solution analysis spaces to a level of abstraction that is technology independent and more suited for business-level analysis. Service-oriented principles allow integration solutions to be specified by means of service interactions; i.e., technical details can be hidden during integration design thus providing a high degree of flexibility.

This chapter presents a reflection about our experience in solving the three mediation problem scenarios of the SWS Challenge; namely, the Purchase Order Mediation Scenarios (first and second), and the Payment Problem Scenario.[1] The scenarios require the design of a Mediator service that acts as an intermediary software in reconciling message protocol and semantic data mismatches. Service mediation is ideal when systems need to interoperate but have existing and often difficult-to-change services. In particular, this paper reviews how our solution was evaluated with respect to the requirements of the SWS Challenge. We also provide an analysis of our solution in terms of its advantages and disadvantages, and the future research work resulting from this evaluation.

The rest of the chapter is structured as follows: Sect. 18.2 provides an overview the COSMO framework and its relation to the modeling, refinement and composition of services in the context of service mediation. Section 18.3 discusses our solution's the state of the art. Section 18.4 provides an account of lessons learned through an evaluation of the solution and future research challenges. Finally, Sect. 18.5 provides a summary of this chapter.

## 18.2 Approach: Service Mediation and the COSMO Framework

**Service mediation.** We define *service mediation* as *"to act as an intermediary agent in reconciling mismatches between the services of two or more systems"*[18]. We distinguish two types of mismatches: *data mismatches* which occur when

[1] http://swschallenge.org/wiki/index.php/Scenarios

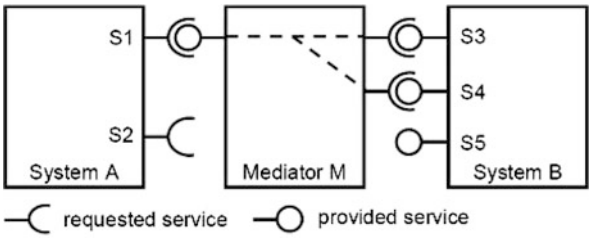


**Fig. 18.1** Service mediation as service composition [20]

systems use different information models (vocabularies) to describe the messages that are exchanged by their services; *process mismatches* which occur when systems use services that define different messages or different orderings of message exchanges. Service mediation aims at resolving these mismatches [13, 20].

The need for an intermediary, hereafter denoted as *Mediator*, is imposed by the assumption that the mediated services can not be changed. The definition abstracts, however, who will perform the Mediator role, e.g., some of the existing systems or a 'third' system. We approach the design of the Mediator as a composition problem: each service that is requested by some system has to be composed from one or more services that are provided by the other systems and, possibly, by the same system. Figure 18.1 illustrates this for the case of two systems. Mediator M offers a mediation service that matches requested service S1 of system A by composing services S3 and S4 that are offered by system B. The Mediator should provide such a mediation service for each service that is requested by systems A and B [19].

We have developed an *integration framework* [19] that supports the design, implementation and validation of mediation services. Our framework consists of the following elements: the COSMO *conceptual framework* for modeling and reasoning about services, *languages* to express service models using COSMO, *techniques* to analyze the interoperability and conformance of service models, *transformations* from service design to service implementation level, *tools* supporting the editing, analysis and transformation of service models and a *methodology* for developing mediation services.

**The COSMO framework.** COSMO defines concepts to support the modeling, reasoning and analysis of services. In COSMO, we define a service as *"the establishment of some effect (or value) through the interaction between two or more systems"*. Figure 18.2 provides a graphical description of the COSMO framework.

We distinguish four service aspects, i.e., *information*, *behavior*, *structure*, and *quality*, representing categories of service properties that need to be modeled. The structure aspect is concerned with modeling the systems that provide or use services, and their interconnection structure. The interconnection structure comprises (amongst others) the interfaces at which services are offered. The behavioral aspect is concerned with the activities that are performed by systems as well as the relations among these activities. The information aspect is concerned with modeling the information that is managed and exchanged by systems. The quality aspect is concerned with modeling the non-functional characteristics of

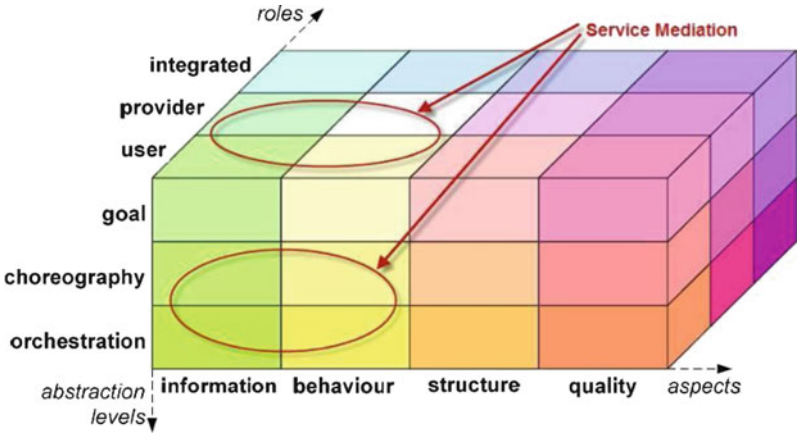


**Fig. 18.2** The COSMO framework

services. These qualities often play an important role in the selection of services. Examples of quality aspects are the "cost" associated with a service or the "response time" of a service.

Besides service aspects, we distinguish three global abstraction levels at which a service can be modeled; namely, *goal*, *choreography* and *orchestration* level. A model at a goal level describes a service as a single interaction, where the interaction result represents the effect of the service as a whole. A model at choreography level refines the model at goal level by describing the service as a set of multiple related, more concrete interactions. A model at orchestration level describes the implementation of the service using a central coordinator that invokes and adds value to one or more other services.

Finally, we distinguish different roles of the systems involved in a service: the *user*, *provider* and *integrated roles*. The integrated role abstracts from the distinction between a user and provider by considering interactions as joint actions, thereby focusing on what the user and provider have in common.

Currently, our mediation solution mainly considers choreographies and orchestrations from the behavior and information aspect, and by distinguishing between a user and provider role. Furthermore, services are modeled close to the level at which they are described using WSDL, while abstracting from technology details [18, 23].

## 18.3 Solution: Applying the COSMO Framework

As previously mentioned, we have developed an integration framework that consists of, among others, a *methodology* for developing mediation services. We discuss in this section a brief overview of the previous and current methodologies, hereafter termed version 1 and version 2, respectively. Separating the solutions

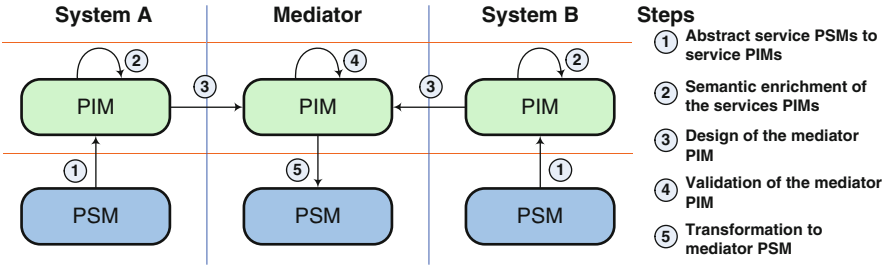


**Fig. 18.3** The COSMO methodology for service integration [12]

into versions permits a more focused and targeted evaluation. Version 1 of the methodology largely draws from the service modeling, refinement, reasoning and analysis concepts of COSMO combined with Model Driven Architecture (MDA) principles. Version 2 seeks to extend the previous through a combination of goal-oriented requirements engineering and business rules approach to specification of the integration requirements. Version 1 of the methodology is used to solve the two Purchase Order Mediation Scenarios (cf. [12, 16, 19, 20, 22]), and version 2 of the methodology is used to solve the Payment Problem Scenario (cf. [1–3]).

### *18.3.1 Version 1*

We have shown in [12, 16, 19, 20, 22] that in version 1 of the methodology (shown in Fig. 18.3), we start by "lifting" service description described in the WSDLs to a platform independent level which means, in MDA, transforming Platform Specific Models (PSM) to Platform Independent Models (PIM). Doing so avoids unnecessarily complicating the design space and thus provides more opportunity for business domain experts to be involved in the design. Business domain experts do not need to understand WSDLs to design the integration solution.

The second step involves semantically enriching PIM information which cannot be automatically derived from the WSDL. This is done because WSDLs do not inherently provide interaction protocols (i.e. how the sequence of message execution is specified), therefore, this information is supplemented through some text documentation in natural language, stakeholder interviews, or even code inspection. This enrichment is done so that the PIM is designed completely and precisely allowing better reasoning and generation of the mediation solution later on.

The third step involves the actual design of the Mediator at the PIM level. This usually involves splitting the integration solution in two areas which may be done in parallel: generating the behavior and information models. The information model unifies the differences in the data representations and interpretations between systems. The behavior model composes requested and provided mismatching service by relating their operations (i.e. matching the input of an operation call to the output of another and their constraints). The Mediator PIM is currently specified and designed using the Interaction System Design Language (ISDL) [17].

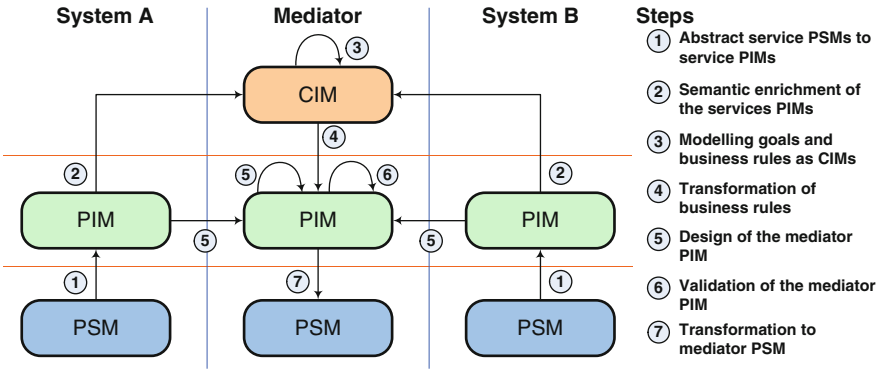


**Fig. 18.4** The extended service mediation methodology [2]

The fourth step involves validation of the composed service Mediator using techniques such as interoperability assessment and simulation of generated behavior and information models [14, 15].

The fifth and sixth steps include transforming the designed PIM models of Mediator to PSM by mapping the integration solution to a specific service computing platform [5]. This is done by transforming the Mediator PIM in ISDL to the Mediator PSM using Business Process Execution Language (BPEL).[2]

### *18.3.2 Version 2*

As version 1 of the methodology is only limited to the PIM level, we have looked into extending the methodology to the CIM (Computation-Independent Model) level. This is done so that we can abstract further in separating the business requirements from their technical implementation (i.e. essentially separating the "why" from the "how"). We argue that doing so gives business domain experts the opportunity to better understand, specify, and validate integration requirements without having to deal with their technical implementations. To do this, we treat requirements as goals and specifying them in some goal modeling language at the CIM layer. In turn, these goals are refined into rules which are encapsulated as services and incorporated into the Mediator service [1–3]. Figure 18.4 describes the extended methodology.

This required changes to the third step of version 1 where goal models are used to specify the requirements of the integration solution at the CIM layer. As Fig. 18.4 shows, the goal model depicts only the motivations of the integration (represented by the single rounded square under the Mediator column). Although the collaborating enterprises may have their own goals, we are only concerned with the goal of

[2] http://docs.oasisopen.org./wsbpel/2.0/wsbpel-v2.0.pdf

the integration. Currently, we model the goals using Architectural Modelling of Requirements (ARMOR) [21] an extension of ArchiMate [10] that seeks to add a goal-oriented requirements engineering aspect to architectural modeling.

The higher-level goals are then refined into lower-level goals which are then mapped to existing services identified in Step 1. Furthermore, business rules that constrain some aspect of the integration may also be derived from the lower-level goals. If no existing service can realize or satisfy these business rules, they may have to be transformed into an executable form, exposed as a service (as described in the fourth step) and integrated into the behavior model of the Mediator.

The fourth step transforms the business rules derived from the goal model into their equivalent rule specifications at the different layers of the MDA stack. At the CIM layer, we use Attempto Controlled English (ACE) [7] to specify the business rules in a near-natural controlled English language. These rules are then transformed into an XML-based rule specification for added rule interoperability at the PIM layer using Rule Markup Language (RuleML).[3] Finally, at the PSM layer, we transform these business rules into an executable form and expose them as a service to constrain the behavior of the Mediator using Java Expert System Shell (Jess) [6]. The generated Jess rules are then wrapped into one batch file and deployed as a Web service. This approach essentially separates the business rules of the Mediator from the business logic they constrain [8]. The next step involves the design of the integration specification at the PIM layer as described in Step 3 of version 1 (now, Step 5). The rest of the steps as outlined in version 1 follows as usual thereafter.

## 18.4 Lessons Learned

This section discusses the results of our solution as evaluated against the requirements of the SWS Challenge. The SWS Challenge evaluation criteria mostly focus on how the solution can cope with changes when new requirements are introduced. This section describes how our solutions respond to such changes. We also compare of COSMO with other approaches. Finally, we discuss the advantages and disadvantages of the solution.

### *18.4.1 Evaluation Results*

#### 18.4.1.1 Solving the Purchase Order Scenarios with Version 1

This section describes the required changes to version 1 of the solution when additional requirements are introduced from the second Purchase Order Scenario.

[3] http://ruleml.org/

The second Purchase Order Scenario is essentially an extension of the first. In particular, it requires the addition of Moon's Product Management System (PMS) to manage production capabilities requiring additional new services to be modeled in the Mediator PIM. Further details can be found in [16] as to how the changes to the solution have been carried out. Essentially, when changes are introduced at a higher abstraction level, model reuse can be done. Already existing abstract models need not be entirely redesigned or replaced. Furthermore, the PSM can simply be regenerated based on the existing abstract PIM model (and in fact, to another technology implementation of choice).

To illustrate, when the `checkProductionCapability` and `confirmOrder` services from Moon's PMS are added according to the requirements of the second Purchase Order Scenario, the steps outlined in Sect. 18.3.1 can be repeated minimally to accommodate such requirements: Step 1 requires the same abstraction of WSDL data and behavior information to UML and ISDL, respectively. The other PIMs of the first Purchase Order Scenario are simply reused. Step 2 requires no further behavior enrichment is necessary since the new services can be invoked independently of each other. However, new information mapping relations will have to be defined. Step 3 requires an update on the behavior and information models to reflect the changes in the requirements. This means that the new services will also need to be represented as complementary operation calls. The services of the first scenario are also reused. This only requires adding new relations between the services of the first and second scenarios. Step 4 requires that the same techniques be used to validate (e.g. simulation or conformance assessment) and analyze whether the integration solution still allows the systems to interoperate. Furthermore, we can also reuse the already existing information mappings of the first scenario to add new mappings between the Mediator PIM and Moon's PMS. Finally, Step 5 simply requires the reuse of the transformations between the Mediator PIM in ISDL to its an executable PMS in BPEL. This step is done automatically requiring only the addition of the endpoints of the newly added services.

#### 18.4.1.2 Solving the Payment Problem Scenario with Version 2

This section describes how version 2 of the solution complies when changes are made in the authorization requirements of the Payment Problem Scenario. When changes are required in the Mediator PIM in ISDL, the same adjustments to the solution must be made as described previously in Sect. 18.4.1.1. However, version 2 is much more flexible when changes are confined within the requirements related to payment authorization since such requirements are separated, encapsulated and specified as business rules.

Version 2 of the solution was evaluated during the workshop against the different requirements that the `authorize` operation of Blue's Management Department System may have which included the following: Deny any request for authorization outright. Change the amounts that can be authorized by an authority. Return an authority that does not follow the original sequence of authorities. For example,

the rule engine expects that `Arnold Black` is the next subsequent authority, but when the `authorize` operation returns `Peter Petrelli`, the response is still `denied`.

These changes in requirements did not trigger any change to the solution's code or data since they do not affect the interface constraints. However, the following change in requirements can introduce a change in the solution artifacts as they affect the logic behind the rules. The solution, however, is still flexible as the changes are confined only within the rule specifications. This was not evaluated during the workshop itself but through our own evaluation efforts: Adding other authorization response codes. Deleting an authorization response code. Adding or removing an authority from the list. Changing the original sequence of authorizations.

### *18.4.2 Comparison with Other Approaches*

This section provides a comparison between other approaches that have solved the mediation problems and COSMO. In particular, we compare COSMO with Web Services Modelling Ontology (WSMO) [24], SWE-ET/WebML [4], and jABC [25]. This comparison is taken from a more detailed version in [11]. As a comparison criteria, we make use of DESMET [9] – a comprehensive methodology for planning and executing unbiased and reliable evaluation exercises. DESMET is a qualitative form of evaluation allowing the comparison to be described using a set of *features*. Feature analysis involves identifying requirements and mapping them to features that a method (or tool) must have to support such requirements.

We focus on the qualitative aspect of the comparison by using four features: *data mediation*, *process mediation*, *correctness* and *suitability of design concepts*. Data mediation refers to how an exchange data is mapped and transformed from a source schema into a target schema to solve the data mismatch. Process mediation refers to how the interaction behavior is expressed to solve the behavior mismatch. Data and process mismatches are evaluated against design and runtime aspects, as well. Correctness refers to how correct and reliable the Mediator is with respect to the requirements. Finally, suitability of design concepts mean how adequate the design concepts are in characterizing the system being modeled. Figure 18.5 shows the summary of the comparison between WSMO, WebML, jABC and COSMO.

WSMO uses the notion of goals, mediators, services and ontologies. The Mediator, a core concept, performs an ontology-to-ontology mediation through the use of adapters that provide mapping rules between ontologies. Goals specify requested capability and requested interfaces. The goal-oriented paradigm allows service discovery, selection of appropriate services, invocation of services, and composition of services for a common task.

WebML uses Business Process Modelling Notation (BPMN) to first specify a workflow at a high level of abstraction which is then transformed into hypertext diagrams that represent a service invocation order – a design concept that is adapted to model the mediation solution. Furthermore, in WebML, Adapter units that transform XML messages into WebML's internal ontology perform data mediation.

| | | WSMO | WebML | jABC | COSMO |
|---|---|---|---|---|---|
| **Data mediation** | **Design time aspects** | Ontologies manually created from analyzing the RosettaNet messages and WSDL service descriptions.<br><br>Ontology to Ontology mappings. | ER-model manually created from analyzing the RosettaNet messages and WSDL service descriptions.<br><br>XML to Ontology mappings. | SIBs and hierarchical parameters automatically generated from WSDL service descriptions.<br><br>XML to SIB parameters mapping. | Ontologies partially generated from RosettaNet messages and WSDL service descriptions.<br><br>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.<br><br>Behaviour modelled as Abstract State Machines by means of transformation rules. | Defining BPMN model, hypertexts and constraints.<br><br>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.<br><br>Behaviour modelled in terms of control flow graphs based on fork/join parallelism. | Defining a workflow explicitly describing the behaviour of the mediator.<br><br>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). |

**Fig. 18.5** A comparison between COSMO, WSMO, WebML, and jABC (Taken from [11])

jABC uses basic service types, called SIBs (Service-Independent Building Blocks), from WSDL service descriptions which are then used to design behavior models, called SLGs (Service Logic Graphs), by composing the reusable SIBs into (flow-)graph structures. The behavior models can then be analyzed early on for

correctness. Formal verification of the models allows simplification of debugging complex processes, reducing development time, and increasing robustness. Data mediation is achieved by mapping the messages structures from the WSDL into hierarchical SIB parameters. Finally, the Mediator can be made executable through model-to-code transformations.

COSMO uses ontologies as the underlying information model allowing analysis as to 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 behavior is specified as a workflow explicitly modeling interactions between services, operation calls and causality relations between then.

### *18.4.3 Advantages and Disadvantages*

Some of the salient *advantages* of the solution include the following:

**Use of abstract models.** In version 1, the use of model-driven and goal-oriented techniques to design the integration solution at varying abstraction levels provides flexibility to the solution. Here, the integration problem is solved at a higher level of abstraction bringing the solution closer to the problem domain and less on the solution technologies. Abstraction captures the semantics of the integration problem and the proposed solution without having to deal with technology-specific implementations yet, allowing decision makers to choose their own implementation technology later. Abstraction also captures changes in the requirements; i.e., the abstract solution specification is updated and a new solution implementation is generated without adversely affecting the implementation. Should the underlying implementation technology changes, the abstract requirements can still be reused.

**Separation of business rules.** In version 2, the solution allows the separation of business rules from the business process they constrain. The more dynamic aspects of the requirements are specified as business rules while keeping the more stable parts in the business process. Flexibility is again achieved since the dynamic parts of the business process are separated from the more stable ones so that a change between the either of them does not affect the other adversely. Furthermore, this separation allows business rules to be identified, changed, and managed better. Combined with model-driven techniques, we specify rules at various abstraction layers which again brings the integration solution closer to the problem domain. For example, low-level goals when operationalized as a controlled language can be validated better by business domain experts. Separating the business rules also allows their migration. For example, should there be some activities in the business

process that change often, these activities can be migrated and specified as rules. Conversely, should there be rules that turn out not to change too often, they can then be migrated as an activity in a business process.

**Better involvement of non-technical business domain experts.** As a result of the abstracted view on the various specifications of the solution, non-technical business domain experts can participate better in integration solution by describing the goals of the integration, specifying and validating the business logic of the Mediator PIM, deriving the semantic mapping between the information models, and verifying the order of service invocations. This involvement is important in arriving at a more complete, accurate, and reliable integration solution.

**Better requirements specification.** With the use of goal-oriented requirements engineering, the solution provides better opportunity to specify the requirements of the integration solution. This can lead to a more precise, accurate, and complete specification of the requirements. Specifying the requirements in terms of goals also allows *early* verification without waiting for the implementation. With goals and business rules specifying the requirements at a higher abstraction, business domain experts have a better way of validating the requirements in a more structured manner as the models are intuitively understandable.
Some of the salient *disadvantages* of the solution include the following:

**Manual modeling of integration solution.** Although our solution uses information models to specify entities and their mappings, we still require designers to discover and represent these mappings from documentation, stakeholder interviews, or code inspection. This is a manual and error-prone process which requires understanding the meaning of all information models to be integrated. A large part work in the manual design of the Mediator is currently done during the design of the Mediator PIM in ISDL, data- and process-wise. Data-wise, matching the *semantic equivalence* between message elements is done by drawing on domain knowledge to determine the correct mapping which is then manually specified in a domain specific language. Data elements belonging to different vocabularies are semantically equivalent if they have similar meanings. Process-wise, the composition and refinement of the Mediator service is also done manually by determining the causal dependencies, including the mapping the sequence of invocations, between interaction contributions. Automating the data and process matching remains a challenging task since the knowledge to do this depends highly on the stakeholder requirements and domain knowledge.

**Rule transformations.** Version 2 of the methodology is limited in terms of the transformations between ACE, to RuleML and to Jess. Our prototype for transforming RuleML to Jess is rather specific only to the Payment Problem Scenario. Although we strive to create just one XSLT stylesheet to transform all the four RuleML Authority rules, we have not explored the possibility of applying our transformation to other possible rule structures. While doing the scenario, our experience has been that once a different Discourse Representation Structure (DRS)

is generated by the ACE Parser Editor (APE)[4] for a given rule stated in ACE, necessary (and in fact time-consuming) changes need to be done to the XSLT stylesheet as well. Research related to rule transformation is lacking.

**Limited goal-modeling methodology.** ARMOR does not currently provide a standard "way of working" in the modeling of goals. More mature goal modeling approaches (e.g. KAOS [26]), propose a set of heuristic steps which include among others some best practice development techniques, and an extensive collection of formally-driven goal refinement techniques. Furthermore, ARMOR should be improved to allow translation of modeled business rules into a controlled language such as ACE. At the moment, although ARMOR supports business rules as one of its constructs, automatic translation is currently not supported. We have to manually specify the business rules in ACE.

**Limited tooling support.** Although there is an available tool support in terms of designing the goal models in ARMOR, specifying a valid ACE sentence using APE, transforming from ACE to RuleML and Jess (albeit prototypical), and from ISDL to BPEL, designing and simulating the Mediator PIM using Grizzle and Sizzle, deploying services in BPEL, and finally specifying information model mappings using Tizzle, an integrated environment where one can perform all the design, implementation and validation activities in one place is still absent. Such an integrated environment is important to reduce errors, development time, and provide debugging and deployment facilities.

### *18.4.4 Conclusions and Future Work*

Aside from addressing the disadvantages described earlier; i.e., providing a more automated support for the integration design and implementation, improving the semantic equivalence between rule transformations, and developing an integrated tooling environment. Some future work include:

**Support for non-functional properties.** Our current work focuses solely on functional properties of integration requirements. Non-functional properties play an important role in the design and implementation of the integration solution. These non-functional properties can include, for example, security, response time, and economic value of provided and requested services.

**Validation between CIM and PIM goal models.** In version 2 of the methodology, we still have to explore the validation between the designed goal models at the CIM layer and their implementations in the PIM and PSM layers; i.e., we still need formal ways to verify if whether the overall effect of the Mediator service, when executed, does indeed satisfy the integration goal.

[4] http://attempto.ifi.uzh.ch/site/tools/

**Use of other rule-based technologies.** We shall be investigating the potential use of other rule-based specification such as the Semantics of Business Vocabulary and Business Rules (SBVR) as the controlled language to specify rules at the CIM layer. Unlike ACE, SBVR is more expressive: rules can be specified in other languages through a speech community. It also allows sharing of business vocabulary through a semantic community. It also supports some formalisms such as modalities which may add flexibility to specifying business rules. Other potential rule-based specifications include the W3C's Rule Interchange Format (RIF) and OMG's Production Rule Representation (PRR).

## 18.5 Summary

This chapter reviews the state of the art and evaluation of the COSMO framework in solving the Service Mediation Problems Scenarios of the SWS Challenge. We have demonstrated the use of goal-oriented requirements specification, coupled with model-driven techniques in the service-oriented design of integration solutions.

Several advantages can be drawn: Using model-driven techniques allows requirements to be specified at varying levels of abstraction bringing the solution closer to the problem domain. Separating business rules from the business process they constrain permits flexibility as those parts of the solution that change more often are isolated from those that do not. Several disadvantages also need to be addressed: the design largely remains manually performed, albeit the availability of tools. There is a need to develop better selection and transformations between rule-based technologies. Finally, there is still a need to develop an integrated development environment where technically and/or non-technically oriented designers can share participate better in the integration design.

Our participation in the SWS Challenge has been beneficial in improving, designing and validating the concepts behind the COSMO framework. Indeed, the Challenge has been an appropriate venue for exchanging and appreciating different solutions from other participants. Furthermore, this evaluation has allowed us to do a self reflection especially in terms of dealing with the limitations of our approaches and the future solutions at hand. Finally, a possible improvement that can be to the evaluation process is towards generalizing the results between various approaches. This could be possible since all approaches try to solve the same problems; therefore, a criteria for generalization may well be achievable.

**Acknowledgements** The authors are grateful to Rodrigo Mantovaneli Pessoa, Teduh Dirgahayu, and Stanislav Pokraev whose earlier works have been used in this book chapter.

## References


1. C.H. Asuncion, Goal-driven service mediation solution, Master's thesis, University of Twente, 2009
2. C.H. Asuncion, M.E. Iacob, M.J. van Sinderen, Towards a flexible service integration through separation of business rules, in *14th IEEE International Enterprise Computing Conference*, Vitoria, IEEE Computer Society, 2010
3. C.H. Asuncion, D.A.C. Quartel, S.V. Pokraev, M.E. Iacob, M.J. van Sinderen, Combining goal-oriented and model-driven approaches to solve the payment problem scenario, in *8th Semantic Web Service (SWS) Challenge Workshop*, Eindhoven, 2010
4. M. Brambilla, I. Celino, S. Ceri, D. Cerizza, E. Della Valle, F. Facca, A software engineering approach to design and development of semantic web service applications, in *The Semantic Web – ISWC 2006*. LNCS, vol. 4273 (Springer, Berlin/Heidelberg, 2006), pp. 172–186
5. T. Dirgahayu, D.A.C. Quartel, M.J. van Sinderen, Development of transformations from business process models to implementations by Reuse, Technical report, CTIT, University of Twente, Enschede, 2007
6. E. Friedman, *Jess in Action: Rule-Based Systems in Java* (Manning Publications Co., Greenwich, 2003)
7. N.E. Fuchs, U. Schwertel, R. Schwitter, Attempto controlled english – not just another logic specification language, in *8th International Workshop on Logic Programming Synthesis and Transformation* (Springer London, 1990), pp. 1–20
8. M.E. Iacob, D. Rothengatter, J. van Hillegersberg, A health-care application of goal-driven software design. Appl. Med. Inform. **24**(1–2), 12–33 (2009)
9. B. Kitchenham, S. Linkman, D. Law, DESMET: a methodology for evaluating software engineering methods and tools. J. Comput. Control Eng. **8**(3), 120–126 (1997)
10. M. Lankhorst, *Enterprise Architecture at Work: Modelling, Communication and Analysis* (Springer, Berlin, 2009)
11. R. Mantovaneli Pessoa, D.A.C. Quartel, M.J. van Sinderen, A comparison of data and process mediation approaches, in *2nd Workshop on Enterprise Systems and Technology, I-WEST*, vol. 1, ed. by J. Cordeiro, M.J. Sinderen van, B.B. Shishkov (INSTICC Press, Portugal, 2008), pp. 48–63
12. S.V. Pokraev, *Model-driven semantic integration of service-oriented applications*, PhD thesis, University of Twente, Enschede, 2009
13. S. Pokraev, M. Reichert, M.W.A. Steen, R.J. Wieringa, Semantic and pragmatic interoperability: a model for understanding, in *Open Interop Workshop on Enterprise Modelling and Ontologies for Interoperability*, CEUR-WS, vol. 160, Porto, 2005, pp. 1–5
14. S.V. Pokraev et al., A method for formal verification of service interoperability, in *IEEE International Conference on Web Services*, Chicago, Sep 2006, IEEE Computer Society, 2006, pp. 895–900
15. S.V. Pokraev, D.A.C. Quartel, M.W.A. Steen, M.U. Reichert, Requirements and method for assessment of service interoperability, in *4th International Conference on Service Oriented Computing*, Chicago. LNCS, Springer, 2006, pp. 1–14
16. D.A.C. Quartel, M.J. Van Sinderen, Modelling and analysing interoperability in service compositions using COSMO. Enterp. Inf. Syst. **2**(4), 347–366 (2008)
17. D. Quartel, L.F. Pires, M. van Sinderen, On architectural support for behaviour refinement in distributed systems design. J. Integr. Des. Process Sci. **6**(1), 1–30 (2002)
18. D.A.C. Quartel, M.W.A. Steen, S.V. Pokraev, M.J. van Sinderen, COSMO: a conceptual framework for service modelling and refinement, Inf. Syst. Front. **9**(2–3), 225–244 (2007)
19. D.A.C. Quartel, S.V. Pokraev, R. Mantovaneli Pessoa, M.J. van Sinderen, Model-driven development of a mediation service, in *12th International IEEE Enterprise Computing Conference*, Munich, IEEE Computer Society, 2008, pp. 117–126
20. D. Quartel, S. Pokraev, T. Dirgahayu, R.M. Pessoa, M. van Sinderen, Model-driven service integration using the COSMO framework, in *7th Workshops Semantic Web Services Challenge*, Stanford Logic Group Technical Reports, Karlsruhe, 2008, pp. 77–88

21. D.A.C. Quartel, W. Engelsman, H. Jonkers, M.J. van Sinderen, A goal-oriented requirements modelling language for enterprise architecture, in *13th IEEE International Enterprise Computing Conference*, Auckland, IEEE Computer Society, 2009, pp. 3–13
22. D.A.C. Quartel et al., Model-driven development of mediation for business services using COSMO. Enterp. Inf. Syst. **3**(3), 319–345 (2009)
23. D. Quartel, T. Dirgahayu, M. Van Sinderen, Model-driven design, simulation and implementation of service compositions in COSMO. Int. J. Bus. Process Integr. Manag. **4**(1), 18–34 (2009)
24. D. Roman et al., Web service modeling ontology. Appl. Ontol. **1**, 77–106 (2005)
25. B. Steffen, T. Margaria, R. Nagel, S. Jörges, C. Kubczak, Model-driven development with the jABC, in *Hardware and Software, Verification and Testing*. LNCS, vol. 4383 (Springer, Berlin/Heidelberg, 2007), pp. 92–108
26. A. van Lamsweerde, Goal-oriented requirements engineering: a guided tour, in *5th IEEE International Symposium on Requirements Engineering*, Toronto, 2001, pp. 249–262
27. M.J. van Sinderen, Challenges and solutions in enterprise computing. Enterp. Inf. Syst. **2**(4), 341–346 (2008)