A Semantic Web Mediation Architecture

Michael Stollberg[1], Emilia Cimpian[1], Adrian Mocan[1], and Dieter Fensel[1,2]

[1] Digital Enterprise Research Institute Innsbruck (DERI Austria),
Institute for Computer Science, University of Innsbruck,
Technikerstrasse 21a, A-6020 Innsbruck, Austria
[2] Digital Enterprise Research Institute (DERI Ireland),
IDA Business Park, Lower Dangan, Galway, Ireland

Abstract. Heterogeneity is an inherent characteristic of open and distributed environments like the Internet that can hamper Web resources and Web services from successful interoperation. Mediation can be used to resolve these issues, which are critical problems in the Semantic Web. Appropriate technologies for mediation need to cover two aspects: first, techniques for handling the different kinds of heterogeneity that can occur between Web resources, and secondly logical components that connect resources and apply required mediation technique along with invocation and execution facilities. This paper presents an integrated model for mediation on the Semantic Web with special attention to Semantic Web services that is developed around the Web Service Modeling Ontology WSMO. Covering both dimensions, we explain the techniques developed for handling different types of heterogeneity as well as the components and architecture for establishing interoperability on the Semantic Web if not given a priori.

Keywords: Mediation, Heterogeneity, Semantic Web, Web Services, Mediation Techniques, Mediation Architecture

1 Introduction

Due to its design principle of decentralization, the World Wide Web is a network of decoupled, independently working computers. This makes the Web heterogeneous by nature: people create web sites and applications independently, resulting in mismatches that hamper information interchange and interoperability [4]. In consequence, the Semantic Web - envisioned for better supporting information processing and computing over the Web on basis of ontologies and Web services as an augmentation of the existing Internet [3] - will be heterogeneous as well. Techniques for handling and resolving mismatches that hamper interoperability of Web resources require mediation, which becomes a central pillar of next generation Web technologies [8].

In the early 1990ies, Wiederhold propagated so-called *mediator-orientated architectures* for heterogeneity handling in IT systems [26]. In these architectures, mediators are integrated components capable of dynamically handling heterogeneities that hamper system components from successful interoperation.

For generic, application independent mediation, the mechanisms for mismatch resolution need to work on a structural level based on declarative resource descriptions. A main merit of the Semantic Web is that resources carry semantic descriptions, which allows mediation techniques to be defined on a semantic level. Understanding Semantic Web services as an integrated technology for realizing the Semantic Web [24], OWL-S [14] defines an ontology for semantically describing Web services while remaining orthogonal to mediation [19]. In contrast, the Web Service Modeling Ontology WSMO [13] identifies mediation as a first class citizen and in consequence defines mediators as a core element of Semantic Web services.

This paper presents the mediation framework and techniques developed within WSMO as an integrated technology for handling and resolving all kinds of heterogeneity that potentially occur on the Semantic Web. In order to attain a mediator-oriented architecture in accordance to Wiederhold's conception, our approach distinguishes two dimensions: (1) the *mediation techniques* for resolving different kinds of heterogeneities that can arise within the Semantic Web, (2) *logical components* that connect resources and apply required mediation techniques; these are embedded in a software architecture for dynamic invocation and execution. Figure 1 shows the structure of the mediation model that we subsequently explicate and position within related work in this paper.

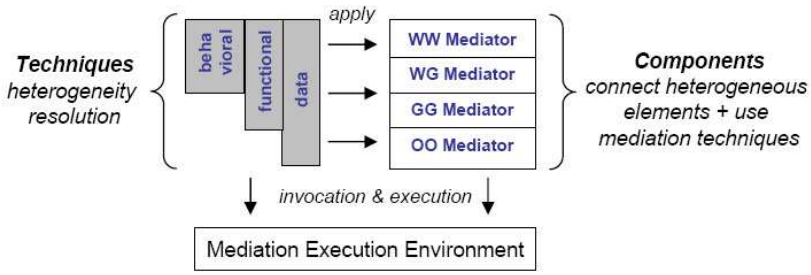

Fig. 1. Dimensions of Mediation

Throughout the paper we apply the well-studied Virtual Travel Agency use case for illustration. Referring to [25] for a detailed specification, a Web service provider `VTA` offers end-user travel services by dynamically using and aggregating other Web services. Requesters can define several different goals, e.g. buying train or flight tickets, booking hotels, as well as combination of these or similar travel related requests. For resource modelling, we use the Web Service Modeling Ontology Language WSML that provides a structural and logical language for WSMO [7].

2 Mediation Levels and Techniques

The first dimension of our model is concerned with the types of heterogeneities that can occur within the Semantic Web. Each heterogeneity type requires a specific technique for mismatch resolution, which we refer to as *levels of mediation*. Extending the heterogeneity types and corresponding mediation levels first identified in [8], developing Semantic Web technology has revealed the four types of heterogeneity enlisted below. We explain this categorization and then reveal mediation techniques for each level developed around WSMO.

1. **Terminology:** Web services or other Web resources use different terminologies; e.g. one entity understands `name` to be the full name of a person, and another one defines `name` to only denote the family name. This can hamper successful interoperation on the semantic level, i.e. concerning the meaning of information.
2. **Representation Format and Transfer Protocol:** resources that interact use different formats or languages for information representation (e.g. HTML, XML, RDF, OWL, etc.), or different protocols for information transfer (e.g. HTTP, RPC, etc.); incompatibilities on this level obviously can hamper prosperous information interchange.
3. **Functionality:** specific to Web services, this refers to functionalities of a provider and a requester that do not match exactly. This enforces complex and thus expensive reasoning procedures for detecting Web services usable for a given request; the need for such expensive operations can be reduced by gaining and utilizing knowledge on the functional heterogeneities, as explained below in more detail.
4. **Business Process:** also specific to Web services, this denotes mismatches in the supported interaction behavior of Web services and clients. This can hamper successful interaction on a behavioral level for consumption or interaction of Web services.

2.1 Data Level Mediation

The first mediation level addresses the first two types of mediation identified above. As these are strongly interconnected and can be handled by similar techniques, they are consolidated as *data level mediation* [16]. This provides a general mediation technique for Semantic Web applications.

The most common type of mismatch in the Semantic Web occurs due to usage of different terminologies by entities that shall interchange information. Within ontology-based environments like the Semantic Web, this results from usage of heterogeneous ontologies as the terminological basis for resource or information descriptions. A main merit of ontologies is that such mismatches can be handled on a semantic level by so-called *ontology integration techniques* explained below in more detail. Regarding the second type of heterogeneity on representation formats and transfer protocols, a suitable way of resolving such heterogeneities is to lift the data from the syntactic to a semantic level on basis of ontologies, and then resolve the mismatches on this level [17].

Techniques Used. The central mediation techniques for the data level are semantically enabled information integration techniques. Collectively referred to as ontology integration [1], the main techniques are ontology mapping, alignment, and merging that we briefly summarize in accordance to [18].

- *Ontology mapping* involves the creation of a set of rules and axioms that precisely define how terms from one ontology relate with terms from the other ontology. These rules and axioms are expressed using a mapping language, as in the example given below. Ontology mapping refers to mapping definitions only, while none of the involved ontologies is changed or altered.
- *Ontology alignment* has the role of bringing the involved ontologies in a mutual agreement. As for the ontology mapping technique, the ontologies are kept separately but at least one of them has to be altered such as the involved ontologies are "aligned" (i.e. they match) in their overlapping parts.
- *Ontology merging* results in creation of a new ontology that replaces the original ontologies. The merging can be done either by unification (all the terms from the involved ontologies are included and mismatches between the overlapping ones are resolved) or by intersection (only the overlapping terms are included and their mismatches reconciliated).

Illustrative Example. Within the VTA use case, consider that a client uses a different ontology than the VTA Web service description. We consider the following example for illustrating one terminology mismatch handling: the ontology used by the requestor contains the concept `station`, and the one used by the provider contains the concept `route`:

```
concept station
        startLocation impliesType _boolean
        destinationLocation impliesType _boolean
        name impliesType _string
concept route
        from hasType (0 1) _string
        to hasType (0 1) _string
```

There are two terminological mismatches: (1) the attribute `startLocation` of the concept `station` corresponds to the attribute `from` in the `route` concept; (2) the attribute `destinationLocation` of the concept `station` corresponds to the attribute `to` the `route` concept. In order to allow automated processing by ontology mapping, we need to create three mapping rules: one for stating the relation between the two concepts and two for imposing the mappings between their attributes. The following shows this using an *abstract* mapping language, propagated in [21] for higher flexibility and easier maintenance of mappings.

```
Mapping(http://www.example.org/ontologies/TravelRequestOntology#station
        http://www.example.org/ontologies/TravelOfferOntology#route
        classMapping(one−way station route))
Mapping(http://www.example.org/ontologies/TravelRequestOntology#destination_Location
        http://www.example.org/ontologies/TravelOfferOntology#to
        attributeMapping( one−way
          [( station ) destination_Location => city] [( route ) to => string]))
          valueCondition( station [( station ) destination_Location => boolean] true)
Mapping(http://www.example.org/ontologies/TravelRequestOntology#start_Location
        http://www.example.org/ontologies/TravelOfferOntology#from
```

```
attributeMapping( one−way
  [( station )  start_Location  => boolean]    [( route)  from => string]))
  valueCondition( station  [( station )  start_Location  => boolean] true)
```

2.2 Functional Level Mediation

Heterogeneities on the functional level arise when the functionality provided by a Web service does not precisely match with the one requested by a client. For instance, in our VTA scenario a requester defines a goal for purchasing a ticket to travel from Innsbruck to Vienna without specifying the type of ticket (i.e. for a bus, train, or plane); an available Web service offers train tickets from Innsbruck to Vienna. Here, the Web service is only usable for solving the request under the condition that the ticket is a train ticket.

We expect situations like this to be the common case for Web service usage. In order to determine the usability of a Web service for a given request - commonly referred to as functional discovery, a central reasoning task for automated Web service usage - complex and thus expensive reasoning procedures are required [11]. As this hampers efficiency of Semantic Web technologies with regard to Web scalability, we use so-called Δ-relations for denoting functional heterogeneities and allow omitting or reducing the need for such expensive operations.

Techniques Used. The central technique for functional level mediation are Δ-relations that denote the explicit logical relationship between functional descriptions of Web services and goals. Functional descriptions are a central pillar of comprehensive Web service description frameworks like OWL-S and WSMO. Defined as conditions on pre- and post-states in some first-order logic derivate, they provide a black box description of normal runs of a Web service omitting information on how technical service invocation [12].

Following [9], the desired relationship can most adequately be described as the logical difference between functional descriptions. For two given functional descriptions ϕ and ψ as first-order logic formulas, the Δ-relation between is defined as $\Delta_{\phi,\psi} = (\phi \wedge \neg\psi) \vee (\neg\phi \wedge \psi)$; this means that Δ contains those elements that are models for either ϕ or ψ and not common to them. A Δ-relation defines a symmetric relation between ϕ and ψ; when concatenating $\Delta_{\phi,\psi}$ with either ϕ or ψ we obtain logical equality with the respective other formula. This allows definition of beneficial algorithmic procedures for omitting or reducing the need of expensive reasoning operations in functional Web service discovery. We refer to [23] for details on this technique.

Illustrative Example. For purpose of illustration, we consider the Δ-relation between functional descriptions of the goal and the Web service in the example outlined above. The following gives the WSMO element definitions for (1) the postcondition of the goal capability (capabilities denote functional descriptions in WSMO [13]), and (2) the capability postcondition of the VTA Web service description.

```
goal _"http://www.example.org/goals/goal1"
  capability
   postcondition
   definedBy
   ?x memberOf ticket[passenger hasValue "Michael Stollberg,
       origin hasValue innsbruck, destination hasValue vienna,
       date hasValue 2006−01−20].

webService _"http://www.example.org/webservices/ws1"
  capability
   postcondition
   definedBy
   ?x memberOf trainTicket[passenger hasValue ?pass,
       origin hasValue ?ori, destination hasValue ?dest,
       date hasValue ?date] and
    ?pass memberOf person and
    ?ori memberOf city and ?dest memberOf city
    ?date memberOf date and (?date >= currentdate).
```

The Δ-relation between the postconditions is given below. It basically states defines all tickets that are not tickets to be models of Δ, and so forth for the attribute value types. Computable by the above formula, this explicates the Δ-relation to denote the logical difference between the source and target component.

```
?x memberOf ticket and not(?x memberOf trainTicket) and
?x[passenger hasValue ?pass,
    origin hasValue ?ori, destination hasValue ?dest,
    date hasValue ?date] and
?pass memberOf person and
?ori memberOf city and ?dest memberOf city and
?date memberOf date.
```

2.3 Process Level Mediation

The third mediation technique is concerned with mismatches on the behavioral level that can occur during the Web service consumption or interaction. For instance, at some point during the consumption of a Web service S by a requester R, R expects an acknowledgement while S waits for the next input; so, the interaction process between R and S runs into a deadlock situation.

Within the WSMO framework, this mediation level refers to the interaction behavior described in the so-called interfaces of a Web services [22]. These specify the interaction behavior supported or expected by the Web service for consuming its functionality (choreography), and for interacting with other Web services that are aggregated in order to achieve the service functionality (orchestration). WSMO defines a formal description language that integrates ontologies with Abstract State Machines [5] for representing the dynamics of service interface descriptions.

Techniques Used. Business process level mismatches can occur in every interaction a Web service is involved in. These heterogeneities can be resolved by inspecting the individual business processes of the entities that interact and trying to establish a valid process for interaction on basis of pre-defined mediation operations on business processes. [6] presents a prototype that supports the patterns for process level mismatch resolution shown in Figure 2.

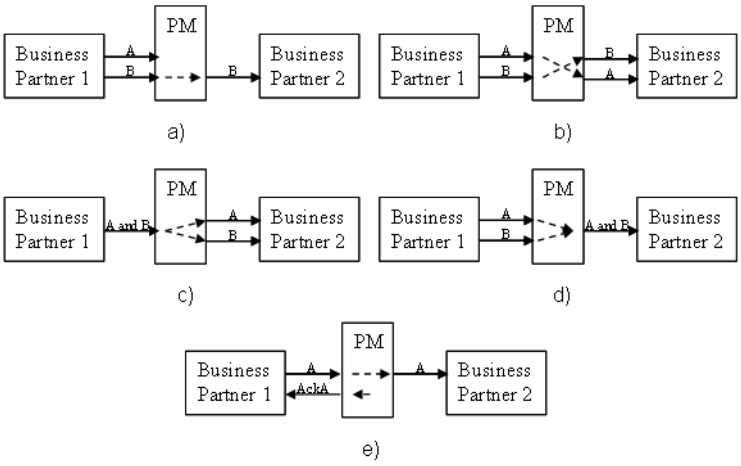

Fig. 2. Process Mediation Patterns - (a) Unexpected Message Stop, (b) Order Inversion, (c) Splitting, (d) Merging, (e) Dummy Acknowledgement

Illustrative Example. The following exemplifies how the process mediation patterns can be applied for resolving a communication mismatch in the VTA scenario. A requestor R wants to first send information about the travel date, followed by the start location and end location of the trip; the provider P wants to receive first the route, and then the data of the trip. R and P use the ontologies with the concepts introduced in Section 2.1.

Figure 3 gives an overview of this situation. The interaction between the requester and the provider is initialized by an outgoing message from R with content of type `date`. But P expects an incoming message with a `route`. The Process Mediator inspects the interaction behavior of R, determining that the second and third outgoing messages contain instances of `station` that can be mediated to `route` by the mappings defined above. Hence, the Process Mediator applies the order inversion pattern in order to hold the first message from R, and then - after data level mediation - uses the merging pattern; now, the information can be submitted to P in the expected order and terminology.

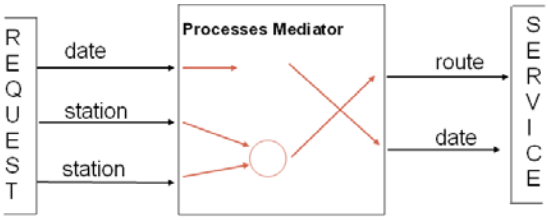

Fig. 3. Example for Process Level Mediation

3 Mediator Component Specification

The second dimension of our mediation technology deals with the logical components that utilize the presented mediation techniques in order to resolve mismatches. With respect to the dynamic and evolving nature of the Semantic Web, essential design principles for a comprehensive mediation architecture are *minimality*, i.e. modularized mediation in distinct components, and *strong decoupling* with respect to reusability of mediation facilities [8]. The following describes how this is realized within the concept of mediators in WSMO, explaining the conceptual model and the explicit logical definition of mediator components.

3.1 Mediator Typology

WSMO defines four top level notions and provides a structure for semantic description of each [13]. Understood to be the general elements of Semantic Web service technology, these are *Ontologies* that provide the formal terminology definition for the domain of discourse, *Goals* that specify the objective a client wants to achieve, *Web services* as the functionality implementation accessible over the Web, and *Mediators* for resolving possibly occurring mismatches.

Four different types of mediators are distinguished that appear to be applicable within the Web service usage process [24]. The mediator type is indicated by a prefix denoting the type of the source and the target component; each mediator type applies those mediation technique required for resolving the heterogeneities that can possibly occur between the connected components. Figure 4 gives an overview of the WSMO mediator typology further explained below.

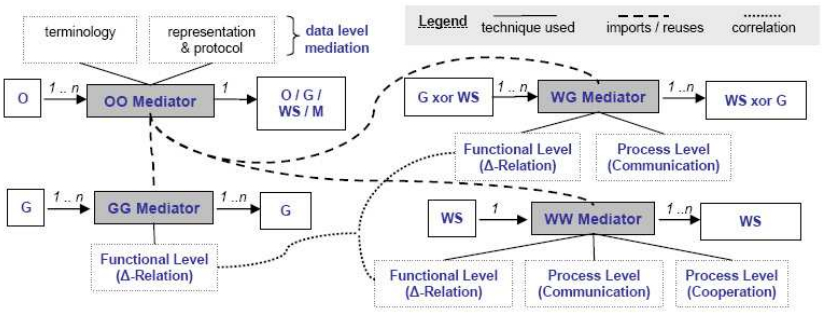

Fig. 4. WSMO Mediator Topology

OO Mediators provide the data level mediation component. The source elements are ontologies or other OO Mediators, while the target can be any WSMO top level element. The only mediation technique used is data level mediation. OO Mediators provide a general mediation component for ontology-based applications; the other mediator types are specific for Web services.

GG Mediators connect WSMO goals, i.e. both the source and target are goals. The mediation techniques used are (1) data mediation by usage of OO Mediators, and (2) functional level mediation on basis of Δ-relations that precisely define the logical relationship between source and target goals. As outlined in Section 2.2, the purpose of the latter is increasing the efficiency of functional discovery.

WG Mediators connect Web services and goals in case a Web service is not usable for solving a goal a priori. WG Mediators can be defined in two directions: either the source elements are one or more Web Services and the target is a Goal, or the other way around. The used mediation techniques are (1) data level mediation by usage of OO Mediators, (2) functional level mediation for establishing usability of a Web service for solving a Goal if not given a priori, and (3) process level mediation for resolving potential mismatches on the communication level between the source and target component.

WW Mediators connect Web services that interact but are not compatible a priori. Its source and target components are Web services. The related mediation techniques are (1) data level mediation by usage of OO Mediators, (2) functional level mediation for handling functional heterogeneities, and (3) process level mediation for resolving mismatches between the source and target service with respect to communication and coordination of interaction. Most commonly, the source component of a WW Mediator is a Web service W that aggregates other Web services $W_1 \dots W_j \dots W_n$ in its orchestration, and the target component is one of the aggregated Web services W_j.

3.2 Logical Specification

A main feature of the WSMO framework is that it defines the description structure of its elements as a meta-layer ontology, following OMG's Meta-Object Facility [13]. This allows explicit meta-model definitions of elements and their interrelation, thereby supporting semantic validation of element definitions and providing an unambiguous specification for execution.

The meta-model ontological description of WSMO mediators consists of a superclass `mediator` that is refined within the distinct mediator types. It defines the source and target component of a mediator, the mediation techniques used for mismatch resolution, the imported other mediators, and non-functional properties as the means for element descriptions used in WSMO. While referring to [15] for detailed meta-model definitions of each WSMO mediator type, the following inspects a concrete mediator definition in detail in order to explicate the presented model.

The listing below shows a WG mediator from the VTA usage scenario that connects the Web service `ws1` and the Goal `goal1` for ticket purchasing as introduced in Section 2.2. Apart from the Δ-relation for functional level mediation, imagine that the goal and the Web service use heterogeneous ontologies, so we need to apply data level mediation. Therefore, we use an OO Mediator `oom1` that contains the mapping definitions outlined in Section 2.1. As the facility for executing the mappings, the data mediator provided in WSMX is used (see next

section); this is defined in the `mediationService` description slot. Moreover, process level mismatches might occur when the goal and Web service start interacting. Hence, the `mediationService` description slot indicates that the WSMX Process Mediator (also see next section) is used for handling these.

```
wgMediator _"http://www.example.org/mediators/wgm1"

 source _"http://www.example.org/webservices/ws1"
 target _"http://www.example.org/goals/goal1"

 importsOntology {_"http://www.example.org/ontologies/TravelRequestOntology"
    _"http://www.example.org/ontologies/TravelOfferOntology"}

 usedMediators _"http://www.example.org/mediators/oom1"

 deltarelation
 definedBy
  ?x memberOf ... // omitted here (see section 2.2)

 mediationService {_"http://www.wsmx.org/datamediator"
    _"http://www.wsmx.org/processmediator"}
```

This example reveals that WSMO mediators explicitly specify elements that are needed in order to establish interoperability between Web services if not given a priori. Apart from the source and target components, all mediation definitions (i.e. mappings and Δ-relations) are explicitly specified as well as the components used for executing the mediation. Consequently, WSMO mediators provide a specification framework for mediation definition and execution whereby each element definition is modularized and decoupled to the maximum possible extent in order to achieve a flexible mediation technology.

In conclusion, the most important feature of mediator components is that they are *minimal* and *modular* components, meaning that each mediator only covers a minimal aspect of heterogeneity handling while several mediators might be used within a specific application scenario. Additionally, the model of WSMO mediators exhibits the following properties:

- OO Mediators provide a data level mediation component generally applicable for the Semantic Web; all data level mismatches are handled by OO Mediators via re-use in the Web service specific mediator types;
- In case that the same Goals and Web services are connected by GG, WG, and WW Mediators, specific logical correlations exist between the Δ-relations defined in the respective mediators.

4 Reference Implementation

In order to demonstrate the implementability of the presented mediation framework and technology, the following outlines its realization within the Web Service Execution Environment WSMX [10], a reference implementation of the overall WSMO framework (homepage: www.wsmx.org).

4.1 The Web Service Execution Environment WSMX

The Web Service Execution Environment WSMX is a platform for automated discovery, selection, composition, invocation and execution of Semantic Web services. In order to enable automated usage of Semantic Web services, WSMX takes a WSMO goal specification as input and dynamically utilizes components required for resolving the goal.

The WSMX architecture depicted in Figure 5 consists of two types of services: application services and base services. The former provide components for central reasoning tasks for Semantic Web services like discovery, as required for automated goal resolution on the Problem Solving Layer. The base services offer low-level support such as reasoning or semantic based storage/retrieval mechanisms. For instance, the Process Mediator service may use a reasoner when analyzing candidate web services, previously retrieved from the repository. Dependent on the concrete goal to be solved and on available Web services, WSMX invokes respective application services.

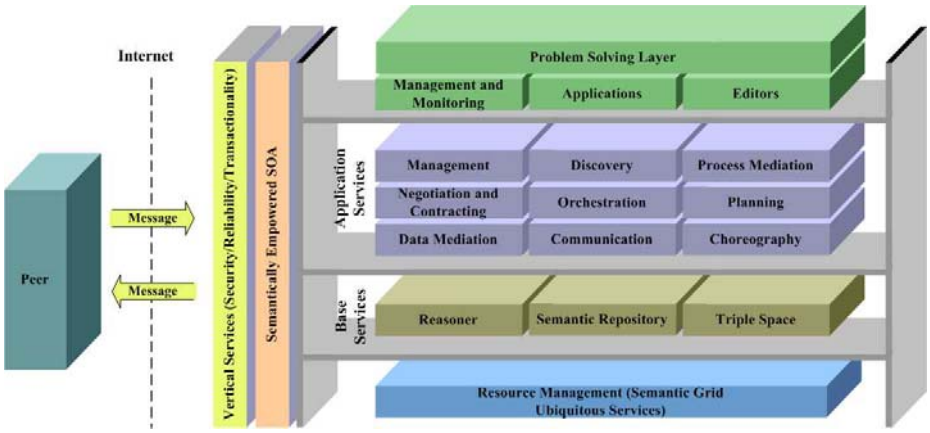

Fig. 5. WSMX Architecture

While functional level mediation will be incorporated in future versions of the WSMX discovery component, implementations for mediation on the data level and the process level exist. Below, we depict the realization of these components with respect to the functionality they offer and how WSMX dynamically invokes them when needed for resolving a goal. Note that the mediation components are only invoked when needed for mismatch handling during the goal resolution process. Nevertheless, the components are integrated into the WSMX architecture, thereby supporting dynamic mediation usage if needed.

4.2 Dynamic Mediation Invocation

The WSMX Data Mediator [16] is invoked in two situations: during the discovery phase and during the communication phase. The need for data mediation is necessary when the ontologies of the goal and of the candidate or selected web service are different - in both the discovery or the communication phase. For data level heterogeneity handling, it uses the ontology mapping technique described above to resolve the mismatches that can appear between two given ontologies. The mappings between ontologies are created in a semi-automatic manner during design time and stored in a persistent storage. That is, these mappings are retrieved during run-time by WSMX and applied on the incoming data (i.e. ontology instances) to transform it from the terms of one ontology in the terms of another ontology (this process in known as instance transformation). The same mappings can also be used for determining which concepts from the mapped ontologies are semantically related (and how). The former functionality is required to enable the process level mediation (it solves the data heterogeneity for the communication stage), while the latter is required to enable the functional level mediation (solves the data heterogeneity that appears in the functional descriptions).

The WSMX Process Mediator [6] works on the behavior interface descriptions of goals and the Web services (i.e. WSMO choreographies) to determine if the communication is interrupted by behavioral mismatches. As a consequence, the process mediator acts as an intermediary and maintains instances of the two choreographies analyzing what messages are expected and in what order. Following this analysis, the order of messages might be change by delaying, suppressing or even generating of fake messages as described in Section 2.3. It is worth noting that the analysis of sent and expected messages is done with the support of the data mediation level when the two choreographies use different ontologies.

5 Related Work

We are not aware of any other comprehensive model for mediation on the Semantic Web, as most existing approaches for heterogeneity handling only address partial aspects of mediation. However, while related work on the distinct mediation techniques is discussed elsewhere (see references in Section 2), the following examines works on mediation architectures and positions our approach therein.

An early approach for realizing a mediation technology that follows Wiederhold's propagation has been presented in the MedMaker project in the mid 1990s [20]. The approach is based on a proprietary, not ontology-based description language for resources called the Object Exchange Model (OEM), and a Mediator Specification Language (MSL), which are both defined in first-order logic. The latter is used for specifying rules that integrate heterogeneous OEM resource descriptions, thereby enabling information interchange between heterogeneous resources. The referenced paper further presents a system implementation Mediator Specification Interpreter (MSI) that is capable of reading and

executing MSL specifications. This work can be seen as a predecessor of data level mediation as realized in OO Mediators (see Section 2). OEM refers to ontologies, respectively WSMO descriptions of goals and Web Services, while MSL refers to ontology mapping languages for data level mediation.

A more recent approach concerned with the formal specification of mediators as software components is presented in [2]. Addressing process level mediation, the approach proposes eight basic mediation patterns - four for bilateral communication and four for the multilateral mediation patterns, along with combinations and refinements of the basic patterns. However, all these basic patterns as well as their combinations and refinements are defined as hard-coded Abstract State Machines, and pre-defined predicates, obtaining in this way an inflexible, rigid model. In our approach we aim at being more flexible and support extensions of the process mediation patterns addressed.

Concerning the needs for mediation within Semantic Web services, the Web Service Modeling Framework WSMF - the conceptual basis of WSMO - distinguishes three levels of mediation [8]: (1) Data Level Mediation - mediation between heterogeneous data sources; (2) Protocol Level Mediation - mediation between heterogeneous communication protocols, and (3) Process Level Mediation - mediation between heterogeneous business processes. While we have adopted these levels and realized respective semantically techniques for mismatch resolution, the framework presented here introduces functional mediation as a novel level. On basis of Δ-relations that explicitly denote the logical relationship between functional descriptions of goals and Web services, this allows increasing the efficiency of Semantic Web service technologies.

6 Conclusions and Future Work

In this paper we have presented an integrated technology for mediation on the Semantic Web developed around WSMO. Heterogeneity being an inherent characteristic of the Web and hence its successors, the presented approach covers all aspects relevant for heterogeneity handling on the Web while remaining open to future developments on mismatch resolution techniques.

The first dimension of the mediation model identifies the types of heterogeneity that potentially can occur on the Semantic Web - that is general Semantic Web applications and Web services in particular. With respect to the suitable techniques for mismatch resolution, we distinguish three levels of mediation: the data level, the functional level, and the process level. For each of these, we have outlined ongoing developments for semantically enabled mismatch resolution techniques. The second dimension of our model deals with components for heterogeneity handling for which we provide WSMO mediators. Defined as logical elements, the four types of WSMO mediators allow explicitly specifying the elements and components for establishing interoperability if not given a priori, whereby each mediator remains a minimal and modularized element itself. In order to demonstrate the realizability of the presented model, we have outlined its implementation within WSMX.

The presented approach realizes Wiederhold's conception of mediator-oriented architectures as follows. While Semantic Web and especially Semantic Web services by definition have declarative resource descriptions, we have presented semantically enabled mediation techniques that allow general purpose, application independent heterogeneity handling and resolution. Furthermore, WSMO mediators provide unambiguous logical definitions of the mediation components that can be executed dynamically with respect to the goal that is to be solved.

In conclusion, we consider the presented mediation model to be sufficient for the Semantic Web as it defines architectural components that applies appropriate mediation facilities for the types of heterogeneity that can appear between the core elements of Semantic Web service systems. The main merit of this model is that each mediator is *minimal* (i.e., it covers only a minimal aspect of heterogeneity handling), and *modular* (i.e., several mediators are combined for specific application purposes). This enables reuse of mediation facilities and eases their maintenance within dynamic and evolving environments like the Internet.

Acknowledgements

This material is based upon works supported by the EU funding under the DIP project (FP6 - 507483), and by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131. The authors would like to thank the members of the WSMO working group (www.wsmo.org) for fruitful input and discussion.

References

1. V. Alexiev, M. Breu, J. de Bruijn, D. Fensel, R. Lara, and H. Lausen. *Information Integration with Ontologies*. Wiley, West Sussex, UK, 2005.
2. A. Barros and E. Börger. A Compositional Framework for Service Interaction Patterns and Interaction Flows. Technical report, 2005. http://sky.fit.qut.edu.au/ dumas/ServiceInteractionPatterns/InteractionPatternsFormalisation.pdf.
3. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5), 2001.
4. Tim Berners-Lee. *Weaving the Web*. Harper, San Francisco, USA, 1999.
5. E. Börger and R. Stärk. *Logical Foundations of Artificial Intelligence*. Springer, Berlin, Heidelberg, 1987.
6. E. Cimpian and A. Mocan. WSMX Process Mediation Based on Choreographies. In *Proceedings of the 1st International Workshop on Web Service Choreography and Orchestration for Business Process Management at the BPM 2005, Nancy, France*, 2005.
7. J. de Bruijn (ed.). The Web Service Modeling Language WSML. WSML Deliverable D16.1 final version 0.2, 2005. available from http://www.wsmo.org/TR/d16/d16.1/v0.2/.
8. D. Fensel and C. Bussler. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), 2002.
9. D. Fensel and R. Straatman. The Essence of Problem-Solving Methods: Making Assumptions to Gain Efficiency. *International Journal of Human-Computer Studies*, 48(2):181–215, 1998.

10. A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - A Semantic Service-Oriented Architecture. In *Proceedings of the International Conference on Web Service (ICWS 2005), Orlando, Florida*, 2005.
11. U. Keller, R. Lara, H. Lausen, A. Polleres, and D. Fensel. Automatic Location of Services. In *Proceedings of the 2nd European Semantic Web Conference (ESWC 2005), Crete, Greece*, 2005.
12. H. Lausen. Functional Description of Web Services. Deliverable D28.1v0.1 Oct 20 2005, WSML Working Group. online: http://www.wsmo.org/TR/.
13. H. Lausen, A. Polleres, and D. Roman (eds.). Web Service Modeling Ontology (WSMO). W3C Member Submission 3 June 2005, 2005. online: http://www.w3.org/Submission/WSMO/.
14. D. Martin (ed.). OWL-S: Semantic Markup for Web Services. W3C Member Submission 22 November 2004, 2004. online: http://www.w3.org/Submission/OWL-S.
15. A. Mocan, E. Cimpian, and M. Stollberg (eds.). WSMO Mediators. Deliverable D29, 2005. Most recent version available at: http://www.wsmo.org/TR/d29/.
16. A. Mocan (ed.). WSMX Data Mediation. WSMX Working Draft D13.3, 2005. available at: http://www.wsmo.org/TR/d13/d13.3/v0.2/.
17. M. Moran and A. Mocan. Towards Translating between XML and WSML based on mappings between XML Schema and an equivalent WSMO Ontology. In *Proc. of the WIW 2005 Workshop on WSMO Implementations, Innsbruck, Austria*, 2005.
18. N. Noy. Semantic Integration: a Survey of Ontology-based Approaches. *ACM SIGMOD Record*, 33(4):65–70, 2004.
19. M. Paolucci, N. Srinivasan, and K. Sycara. Expressing WSMO Mediators in OWL-S. In *Proceedings of the workshop on Semantic Web Services: Preparing to Meet the World of Business Applications held at the 3rd International Semantic Web Conference (ISWC 2004) , Hiroshima, Japan*, 2004.
20. Y. Papakonstantinou, H. Garcia-Molina, and J. D. Ullman. MedMaker: A Mediation System Based on Declarative Specifications. In *Proceedings of the 12th International Conference on Data Engineering*, pages 132–141, 1996.
21. F. Scharffe and J. de Bruijn. A language to specify mappings between ontologies. In *Proc. of the Internet Based Systems IEEE Conference (SITIS05)*, 2005.
22. J. Scicluna, A. Polleres, and D. Roman (eds.). Ontology-based Choreography and Orchestration of WSMO Services. Deliverable D14, 2005. available at: http://www.wsmo.org/TR/d14/.
23. M. Stollberg, E. Cimpian, and D. Fensel. Mediating Capabilities with Delta-Relations. In *Proceedings of the First International Workshop on Mediation in Semantic Web Services, co-located with the Third International Conference on Service Oriented Computing (ICSOC 2005), Amsterdam, the Netherlands*, 2005.
24. M. Stollberg, C. Feier, D. Roman, and D. Fensel. Semantic web services - concepts and technology. In N. Ide, D. Cristea, and D. Tufis, editors, *Language Technology, Ontologies, and the Semantic Web*. Kluwer Publishers (to appear), 2006.
25. M Stollberg and R. Lara (eds.). WSMO Use Case "Virtual Travel Agency". Deliverable D3.3, 2004. available at: http://www.wsmo.org/2004/d3/d3.3/v0.1/.
26. G. Wiederhold. Mediators in the architecture of the future information systems. *Computer*, 25(3):38–49, 1994.