A Formal Approach to Semantic Mediation in SOA: Specification and Proof

Patrício de Alencar Silva, Universidade Federal de Campina Grande (UFCG), Brazil; E-mail: patricio@dsc.ufcg.edu.br

Cláudia M. F. A. Ribeiro, Universidade do Estado do Rio Grande do Norte (UERN), Brazil, & Centro Federal de Educação Tecnológica (CEFET-RN), Brazil; E-mail: claudiaribeiro@uern

Ulrich Schiel, Universidade Federal de Campina Grande (UFCG), Brazil; E-mail: ulrich@dsc.ufcg.edu.br

José Eustáquio Rangel de Queiroz, Universidade Federal de Campina Grande (UFCG), Brazil; E-mail: rangel@dsc.ufcg.edu.br

ABSTRACT

There are many situations in service provisioning scenarios that lead to conflicts and impasses. Very often consumers and providers do not share the same perspective about quality of service. These divergences become more pronounced when the involved parties use different vocabularies to show their interests. It has become commonplace since service providers use different ontologies to describe the offered services. In this paper we discuss the inclusion of a middle layer in the Service-Oriented Architecture (SOA), composed by mediators, in order to make closer the involved sides towards a service negotiation. Considering that mediation is a complex task which crosscuts all the service-related tasks (e.g. discovery and composition), it is proposed a formal approach to describe behavioral aspects of the mediator unambiguously.

Keywords: Mediators, Semantic Web Services, Service Discovery, Service Composition, Formal Methods.

1. INTRODUCTION

Service-oriented applications are potentially source of conflict, since there are two opposite parties: the *service requester* and the *service provider*. Divergences around the maximization of satisfaction with minimal cost (by the *service requester*), against the minimization of resource usage (by the *service provider*) may create impasses that require some external help. Mediators usually play that role bringing together these parties towards an agreement that explicitly considers individual needs but possibly admit reconsiderations. The development of such complex applications commonly requires the adoption of some structured guidelines to be observed.

The Service-Oriented Architecture (SOA) and Web Services technologies (Booth, 2004) have been largely used by developers of Web applications, which functionality is divided into self-contained units, the *services*. This architecture is made up by three entities. Besides the client and the provider, the *registry* is a fundamental part of this architecture. The basic dynamic tasks of SOA entities include: (i) the publication of the service inside the registry by the provider; (ii) the subsequent search for a specific services by the client; and (iii) the consequent execution of a web service and its corresponding result handling during the invocation phase. Eventually, a composition of services can occur when there is not a single service that can fully satisfy the client needs. An opposite situation can also occur, when many services fulfill the client requirements. In this case, it is critical to decide what service actually is the best, considering the client's viewpoint.

Situations like service composition and selection naturally encompass negotiation, although SOA does not explicitly treat this aspect. Indeed, mediators are generally treated as operational elements implemented by the applications (Wiederhold, 2004). However, considering the importance of mediation activity in a service scenario, it is reasonable to put mediators in the first-level of service-oriented architectures. It is precisely what this paper deals with.

Besides specific techniques for reconciliation, the mediator needs information in order to be efficient. It includes information about client and provider requirements, as well as information about the service itself. Although the use of mediators is not new (Wiederhold, 1992), (Wiederhold & Genesereth, 1997), (Mocan et. al, 2005), there is a gap to be fulfilled in the formalization of the mediator's behavior and the precise description of information treated by the mediator. The described proposal presents an architecture that extends SOA by the inclusion of the mediator as the fourth entity at the first-level. An important characteristic of the proposed architecture is the use of ontologies (Gómez-Perez et. al, 2005) that serve as a basis for meaningful information and context-aware activities related to the mediator.

This paper is organized as follows: Section 2 briefly surveys the state of the art in mediators and semantic web services technologies, mainly ontology and its use by mediators. Section 3 depicts the elements of the proposed architecture. Sections 4 and 5 present the translation of the proposed model into a formal specification, relating its static and dynamic aspects, respectively. Finally, Section 6 presents some conclusions and an outlook on our future work.

2. MEDIATORS AND SEMANTIC WEB SERVICES

In the Semantic Web, the information is given with a well-defined meaning through ontologies, which represent formal and explicit specification of a shared conceptualization of some knowledge domain (Berners-Lee, 2001). However, it is unrealistic to expect a global consensus between people and organizations onto a common, shared ontology.

Semantic heterogeneity represents a typical case in which mediation can be applied. In order to reconcile ontologies, it is necessary to analyze the mismatches between them (Sheth et. al, 2003). Mismatches might be present at a conceptual level, as well as at the terminological, taxonomical or purely syntactic levels (Hameed et. al, 2003). Different applications have different ontologies, different semantics, and different knowledge and data stores (Laskey et al., 2006). It is necessary to detect and resolve such discrepancies among them. Correspondences may have to be established among the source ontologies, and overlapping concepts need to be identified.

In the context of SOA, intelligent and active use of information requires a class of software modules that mediate between the service requester and service provider. Mediation simplifies, abstracts, reduces, merges and explains data (Wiederhold, 2004). Interoperability problems inevitably emerge from highly heterogeneous Web service descriptions (Paolucci et. al, 2004). For example, Web services functionalities may be described by different ontologies (e.g. WSMO or OWL-S), can use different protocols or they may have been designed with different goals in mind (Paolucci et. al, 2002). Semantic mediators identify implicit similarities, by the use of ontology reconciliation techniques, such as merging, alignment or integration (Hameed et. al, 2003). Therefore, mediators form a distinct middle layer, making the service requester queries independent of the service description semantics from registries. The translations needed in such layer and what form will have the modules supporting this layer are somehow interrelated.

Since there will eventually be a great variety of mediators, service requesters have to be able to choose among them. Alternate metamediators will have to exist that merely provide access to ontologies that describe available mediators and its properties. Since mediation occurs in different contexts and levels, metamediators synthesize intermediary mediations in order to compose a global mediation. Moreover, the use of ontologies to describe knowledge related to mediators themselves can improve the reuse of mediation functions. Applications may compose their subtasks as much as possible by acquiring information from the set of available mediators. Additionally, unavailable information may motivate the creation of new mediators (Wiederhold & Genesereth, 2004).

A lot of work has been done on mediation systems (Mocan et. al, 2005), (Garcia-Molina et. al, 1997), (Yan et al., 1997), (Tomasic et al., 1998). However, a proper conceptual setting for this task is largely missing. Formal methods, based upon elementary mathematics, can be used to create precise, unambiguous architectural descriptions, in which information is structured and presented at an appropriate level of abstraction. Hence, reasoning about a specification and attempting to construct proofs about its properties can help to detect problems at an early stage of system development. The process of constructing proofs leads to a better understanding about the requirements upon a system, and can assist in identifying any hidden assumptions.

3. AN ONTOLOGY-BASED ARCHITECTURE FOR MEDIATION

The conceptual elements of the extended SOA architecture proposal can be observed in Figure 1.

The first-level entities, i.e. *Service Requester*, *Service Provider*, *Mediator* and *Registry* are highlighted. The majority of other concepts are grouped into contexts. In the service requester context, the desired *functional requirements* are represented by the goal concept, while the user's subjective notion of quality is defined by *QoS Preferences* (what "includes" the service), *QoS Constraints* (what "excludes" undesirable services), and *QoS Priorities* (what "distinguishes" similar services). On the other hand, in the service provider context, the *capability* concept represents the provided functionality, while *QoS offered* abstracts a subset of *non-functional requirements* related to the service provisioning. It is important noting that the same service may be offered in different levels of quality, which causes an impact over resource allocation.

In a typical SOA scenario, the service requester accesses the registry in a direct manner, in order to discover potential services that accomplish the desired functionality. In this case, the mediator represents the entity that makes easy this search. This reconciliation process is made on the ontological level. Considering that ontologies are composed basically by related classes and properties, the role of mediator comprises an automatic verification of intersections in the concepts associated with these elements.

Mediators, as considered in this work, adopt a win-win approach to reach some agreement between the involved parties. Thus, in order to remain neutrality, mediators could be placed apart, in a namely semantic web server (Ribeiro et al., 2004), (Ribeiro et al., 2006), avoiding possible tendentious decisions that could possibly beneficiate either clients or providers. Another aspect to be considered is the fact that the whole mediation may be actually a composition of minor mediation processes. Hence, the concept of mediation in the proposed architecture abstracts several intermediary conflict resolutions, in different levels of complexity.

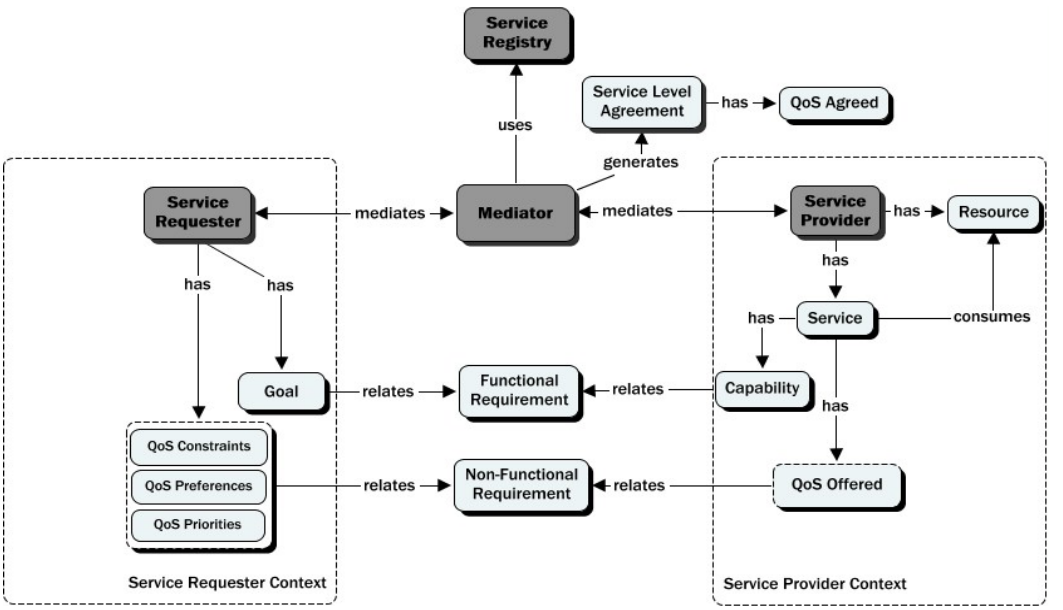
Despite knowing the importance of the role of discussions around implementation issues, this paper focuses on some universal aspects of mediators. For instance, mediators make use well-defined reconciliation laws, valid in some context or jurisdiction, in order to approximate different perspectives and possibly generating a contract that consolidates the conditions related to service provisioning. Hence, any mechanism that makes use of semantics (e.g. semantic web services, intelligent agents or matchmakers) could be used to implement mediators.

4. TRANSLATING THE CONCEPTUAL MODEL INTO A FORMAL SPECIFICATION

The use of formal methods in describing and verifying properties and behavior of systems is not new (Dong et al., 2002), (Dong et al., 2004). The Z notation (Spivey, 2004), for example, is a specification language based on set theory and predicate calculus. Some basic characteristics of Z notation guided its choice as the formalism in this work. The first one is related to its maturity level, recently conveyed into ISO standard (ISO, 2002). The second one refers to the availability of tools to support formal activities, such as type checking, theorem proving and animation of formal specification (Saaltink, 1997).

The fundamentals of Z notation and types are sets defined at the beginning of specification. A *given set* is a powerful abstract element of Z, represented by names inside brackets, from which a formal specification begins. *Enumerated* sets are also permitted in Z notation. In the following, some of the main given sets used to describe the related architectural elements are presented.

Figure 1. The extended service oriented architecture proposal



[Class, Instance, DataType, Parameter, Protocol, Resource, Registry, Preconditions, Postconditions]

Level ::= High | Medium | Low

OperationMode ::= Notification | OneWay | SolicitResponse | RequestResponse

OntologyReconciliationType ::= Alignment | Merging | Integration

ReconciliationResultType ::= ExactMatch | PluginMatch | SubsumesMatch | IntersectionMatch | Impasse

Other key element of Z specification is the *schema*. In analogous way, schema can be considered as class inside object-oriented paradigm. Like a class in OO paradigm, the schema includes a declarative part to encompass variables and a second part dedicated to the manipulation of variables, the predicate.

Schema

Variables

Predicate

Schemas in Z are used to describe both static and dynamic aspects of a system. The static aspects concern the global state of the system and the relationships between its components, namely the invariant. A rigid control over state integrity is accomplished by the invariant during any operation that changes the state. Dynamic aspects include all operations that manipulate the elements of the state (Spivey, 2004).

The *Ontology* type was composed by other types including power sets (P) of *Class*, *DatatypeProperties* (a partial function of *Class* in *DataType*), *ObjectProperties* (a relation between two classes) and *Individuals* (a function of *Class* in *Instance*). The *Mediator* is represented as a schema made up by *Description*, which is typed as an *Ontology* for mediators; a power set of *Techniques* and a *Context*, which represents a relation between two ontologies, i.e. the conceptualizations of involved partners.

Ontologies are built layer on layer. Since the proposed model relies upon them, it is necessary checking and validating its axioms, through the definition of Z semantics for the ontology language. This semantic model serves as a reasoning environment for verification using Z/EVES (Saaltink, 1997), which offers some powerful commands for theorem (e.g. *prove*, *rewrite* or *reduce*). In this section, only the static properties of ontologies are checked (Figure 2), which can be well captured by axiomatic definitions. Dynamic properties (e.g. *discovery* and *composition* phases) are detailed in Section 5. For the sake of space and simplicity, several ontology properties from the proposed axiomatic model were shortened.

5. GENERATING OPERATION SPECIFICATIONS

5.1 Mediation and Service Discovery

In general, service discovery comprises the matchmaking between *goals* (from the service requester context) and service *capabilities* (from the service provider context). The semantics of *goal* element, as well as Web service *capabilities* can be represented by a set of concepts described in a *functional requirement ontology*. The semantic mediator verifies possible similarities on the conceptual level. In order to consider goals and capabilities to match on the semantic level, these elements have to be interrelated somehow. Precisely spoken, we expect that some set-theoretic relationship between them has to exist. The most basic set-theoretic relationships that one might consider are the following:

- Exact match: goal = capability
- Subsumes match: goal c capability
- Plugin match: capability c goal
- Intersection match: goal I capability ë 0
- Non-match: goal I capability = 0

These set-theoretic relationships basically provide the basic means for formalizing an intuitive understanding of a match between goals and Web services in the real world. For this reason, they have been extracted from some extent already in the literature (Paolucci et al., 2002), (Paolucci et. al., 2004). The ideal situation can be represented when an *exact match* between requester desires and provider offerings occurs. Here the semantic mediator identifies that the offered service functionality coincides perfectly with the service requester goals (see Figure 3.a).

Subsumes match (Figure 3.b) occurs when the capabilities that are advertised by the service provider form a superset of relevant objects for the requester as specified in the goals. In other words, the service might be able to fulfill the desired

Figure 2. Formal definitions and proofs related to ontology and mediator concepts

theorem *IsObjectPropertyDisjoint*
local *ObjectPropertyDisjointness*
ObjectProperty ObjectProperty
◁objP1 ObjectProperty objP2
ObjectProperty▷
objP1 dom *ObjPropInstantiation*
objP2 dom *ObjPropInstantiation*

proof of *IsObjectPropertyDisjoint*
prove by reduce;
true

theorem *IsSubDatatypeProperty*
local *SubDatatypePropertyOf*
DatatypeProperty DatatypeProperty
◁dtP1 DatatypeProperty dtP2
DatatypeProperty▷
dtP1 dom *DtPropInstantiation*
dtP2 dom *DtPropInstantiation*

proof of *IsSubDatatypeProperty*
simplify;
prove by reduce;
true

theorem *isEquivalentObjectProperty*
local *EquivalentObjectProperty*
ObjectProperty ObjectProperty
◁objP1 ObjectProperty
objP2 ObjectProperty▷
objP1 dom *ObjPropInstantiation*
objP2 dom *ObjPropInstantiation*

proof of *isEquivalentObjectProperty*
prove by reduce;
true

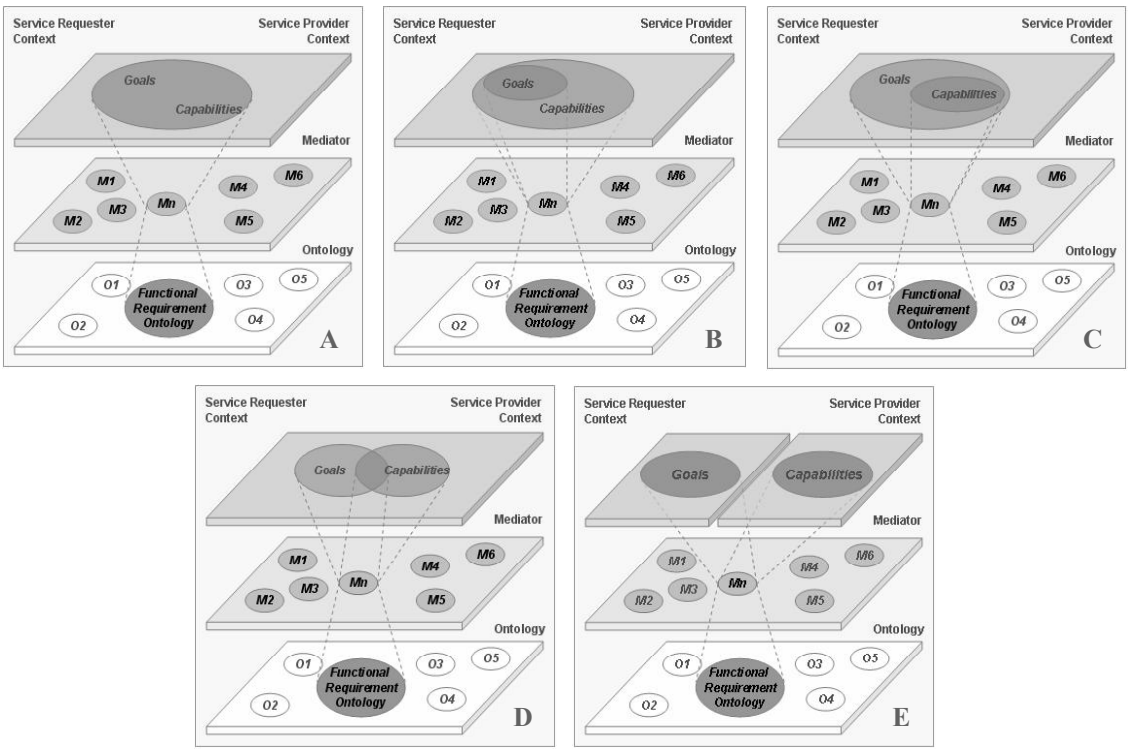
⋮

ObjectProperty: Class Class
DatatypeProperty: Class DataType

Individual: Class Instance
Ontology
Classes: Class
ObjectProperties: ObjectProperty
DatatypeProperties: DatatypeProperty
Individuals: Individual

Mediator
Description: Ontology
Techniques: OntologyReconciliationType
Context: Ontology Ontology

Figure 3. The set-modeling approach to ontology-based mediation: (A) Exact Match; (B) Subsumes Match; (C) Plugin Match; (D) Intersection Match; (E) Impasse



functionality. However it is possible that the service delivers objects that are irrelevant for the requester. When the service capabilities form a subset of requester goals, as illustrated in Figure 3.c, then a *plugin match* occurs. In other words, the service in general is not able to provide all the desired functionality, but there is a guarantee that no irrelevant objects will be delivered by the service.

There is a case when the set of service capabilities and the set of requester goals have an *intersection match* (Figure 3.d). Thus, the service is not able to deliver all the objects that are relevant for the requester, but at least one such element can be delivered. In this case a composition of services may be an alternative in order to achieve the desired functionality. Figure 3.e illustrates the case when the service capabilities description and requester goals are disjoint. That means there is no semantic link. Therefore, in the context of ontology reconciliation, it is also considered an *impasse* situation.

As mentioned before, semantic mediation is directly associated with ontology reconciliation techniques. This process involves the alignment of the two basic ontology structures: *classes* and *properties*. All the matching cases were formalized by two operation schemas, respectively *Goal2ServiceClassAlignment* (between classes), and *Goal2ServicePropertyAlignment* (between properties). Since these

Figure 4. Operation schema related to goal to service conceptual alignment

$$
\begin{aligned}
&\textit{Goal2ServiceClassAlignment}\\
&\Delta \textit{ARCHITECTURE}\\
&\textit{sr: ServiceRequester; m: Mediator; sp: ServiceProvider; s: Service; fr1,}\\
&\textit{fr2: FunctionalRequirement; o1, o2: Ontology; c1, c2: Class; i1,}\\
&\textit{i2: Individual}\\
&\quad \textit{sr} \quad \textit{ServiceRequesters} \quad \textit{sp} \quad \textit{ServiceProviders} \quad \textit{m} \quad \textit{Mediators} \quad \textit{s} \quad \textit{sp . Services}\\
&\quad \textit{c1} \quad \textit{o1 . Classes} \quad \textit{c2} \quad \textit{o2 . Classes} \quad \textit{i1} \quad \textit{o1 . Individuals} \quad \textit{i2} \quad \textit{o2 . Individuals}\\
&\quad \textit{fr1 = o1} \quad \textit{fr2 = o2} \quad \textit{o1} \quad \textit{sr . Goal} \quad \textit{o2} \quad \textit{s . Capability}\\
&\quad \textit{m . Context = m . Context} \quad \textit{dom}(o1 \cup o2)\\
&\quad \textit{dom}(c1 \cup c2) \quad \textit{EquivalentClass} \quad \textit{dom}(i1 \cup i2) \quad \textit{SameAs}\\
&\quad \textit{ReconciliationResult = ExactMatch}\\
&\qquad \textit{dom}(c1 \cup c2) \quad \textit{SubClassOf} \quad \textit{ReconciliationResult = PluginMatch}\\
&\qquad \textit{dom}(c2 \cup c1) \quad \textit{SubClassOf} \quad \textit{ReconciliationResult = SubsumesMatch}\\
&\qquad \textit{dom}(c1 \cup c2) \quad \textit{ComplementOf} \quad \textit{ReconciliationResult = IntersectionMatch}\\
&\qquad \textit{dom}(c1 \cup c2) \quad \textit{DisjointWith} \quad \textit{dom}(i1 \cup i2) \quad \textit{DifferentFrom} \quad \textit{ReconciliationResult = Impasse}\\
&\quad \textit{ServiceRequesters}' = \textit{ServiceRequesters}\\
&\textit{ServiceProviders}' = \textit{ServiceProviders}\\
&\textit{ServiceLevelAgreements}' = \textit{ServiceLevelAgreements}\\
&\textit{Mediators}' = \textit{Mediators} \quad \textit{m}\\
&\textit{Registries}' = \textit{Registries}
\end{aligned}
$$

operations are very similar, only the class alignment is showed in Figure 4. The symbol (Δ) means that this operation changes the global state of the system. The predicate section instantiates some local variables, including ontological structures such as *DatatypeProperties* and *ObjectProperties*. The functional requirements are assigned to distinct ontologies followed by a logical concatenation of reconciliation preconditions.

The schema closes with a state change, by the inclusion of a new local mediator that was created to deal with the service discovery context, according to the *DatatypeProperties* and *ObjectProperties*. Considering these formal statements, the entire discovery phase could be represented by a sequential composition of the two before mentioned operation schemas. The following statement summarizes this aspect.

DiscoveryPhase ; *Goal2ServiceClassAlignment* ; *Goal2ServicePropertyAlignment*

5.2 Mediation and Service Composition

The partial response to the required functionality justifies the use of composite services. Composing Web services requires the description of each service, so that other services can understand its features and learn how to interact with.

A generalized model for Web service description is described with more details in (Medjahed et al., 2003). Figure 5 illustrates that, basically, the service description includes: *domain* information, represented by domain ontologies; a set of *operations,* which include aspects related to the message interchange; *bindings,* that defines message format and protocol details for service invocation; and *capability*, that describes the business functionalities offered by the service operations. Each element in the service purpose refers to the business functionality offered by a specific operation. In order to serve as a basis for workflow models that determine the behavior of composite services, other elements such as *inputs, outputs, preconditions* and *postconditions* are defined. Finally, the provider's notion of quality of service is represented by *QoSOffered* (a function of a set of non-functional requirements in a level of quality).

Service composition can also occur in different levels, from the binding level to the QoS level. A major issue when defining a composite service is whether its component services are *composable* (Medjahed et al., 2003). For example, it would be difficult to invoke an operation if there were no mapping between the parameters requested by this operation (e.g., data types, number of parameters) and those transmitted by the service requester. Hence the service composition is a complex task that involves both syntactic (e.g. binding) and semantic features (QoS Offered).

The proposed formal model aggregates the composition preconditions in two phases: *OperationCompositionPhase* and *ServiceCompositionPhase* (see Figure 5). The first one relates the mediation process in the reconciliation of the syntactic features (e.g. mode and message composability). The second one relates semantic features, in which an instance of a mediator verifies possible intersections in binding and domain aspects of the services, according to the context and reconciliation laws (e.g. *EquivalentClass* and *SameAs* axioms)

The entire *CompositionPhase* is defined by a schema that aggregates a sequential composition between *OperationCompositionPhase* and *ServiceCompositionPhase*.

CompositionPhase ; *OperationCompositionPhase* ; *ServiceCompositionPhase*

6. CONCLUSION AND FUTURE WORK

This paper presents an extended SOA proposal that explicitly includes the mediator as first-level entity. The main purpose of the architecture is to argue and reason about conceptual aspects related to the role of the mediator in service discovery and composition. It was investigated the use of ontological reconciliation techniques which serves as a basis for meaningful mediation. A formal set-based approach to describe the mediator behavior and related concepts was presented.

In particular, we investigate and analyze what kind of statements need to be formally checked and proven about relations between mediation and other service related tasks. These formal statements are called *proof obligations*. At the design stage, a proof can show not only that a design is correct, but also *why* it is correct. The additional insight that this affords can be invaluable: as requirements evolve and the design is modified, the consequences are easier to investigate. At the implementation stage, a proof can help to ensure that a piece of code behaves according to the specification. The practice of proof leads to better specifications.

In terms of future work, we intend to investigate and analyze what kind of statements need to be formally checked and proven in order to relate the mediation activity to the other service related tasks, including service selection, negotiation, agreement and monitoring. More specifically, we intend to extend the proposed formal model towards proof obligations in which QoS information can be used by the mediator to improve the process of reconciliation in the ontological level. These proof obligations, that basically comprise the formulation of theorems and automatic reasoning and simulation of the proposed architectural properties, will

Figure 5. Formal statements related to the operation and service composition phases

OperationCompositionPhase
ΔARCHITECTURE

op1, op2: Operation; m: Mediator; o1, o2: Ontology;
c1, c2: Class; i1, i2: Individual
m Mediators
(op1, op2) isModeComposableWith
(op1 . Input, op2 . Output)
isMessageComposableWith
(op1 . Output, op2 . Input)
isMessageComposableWith
o1 = op1 . Domain o2 = op2 . Domain
c1 o1 . Classes
c2 o2 . Classes
i1 o1 . Individuals
i2 o2 . Individuals
m . Context = m . Context (o1, o2)
(c1, c2) EquivalentClass (i1, i2) SameAs
ReconciliationResult = ExactMatch
(c1, c2) SubClassOf (c2, c1) SubClassOf
ReconciliationResult = PluginMatch
(c1, c1) ComplementOf
ReconciliationResult = IntersectionMatch
ServiceRequesters' = ServiceRequester
ServiceProviders' = ServiceProvider
ServiceLevelAgreements' = ServiceLevelAgreement
Mediators' = Mediator m

ServiceCompositionPhase
ΔARCHITECTURE

op1, op2: Operation; s1, s2: Service; m: Mediator;
o1, o2: Ontology; c1, c2: Class; i1, i2: Individual
m Mediators
op1 s1 . Operations
op2 s2 . Operations
(s1, s2) isBindingComposableWith
c1 o1 . Classes c2 o2 . Classes
i1 o1 . Individuals i2 o2 . Individuals
o1 = s1 . Domain o2 = s2 . Domain
op1 . Domain = s1 . Domain
op2 . Domain = s2 . Domain
m . Context = m . Context (o1, o2)
(c1, c2) EquivalentClass (i1, i2) SameAs
ReconciliationResult = ExactMatch
(c1, c2) SubClassOf
ReconciliationResult = PluginMatch
(c1, c2) ComplementOf
ReconciliationResult = IntersectionMatch
ServiceRequesters' = ServiceRequester
ServiceProviders' = ServiceProvider
ServiceLevelAgreements' = ServiceLevelAgreement
Mediators' = Mediator m

serve as the basis for a most complete framework for mediation in several levels in the context of service negotiation and provisioning.

REFERENCES

Berners-Lee, T., Hendler, J., & Lassila, O. (2001, May). The semantic web. *Scientific American, 284(5),* 34–43.

Booth, D. (ed.). (2004, February 11). Web Service Architecture,W3C Working Group Note. Latest version available at http://www.w3.org/TR/ws-arch/

Dong, J. S., Sun, J., & Wang, H. (2002, October). Z Approach to Semantic Web. In *International Conference on Formal Engineering Methods (ICFEM'02)*, Shanghai, China, LNCS, Springer-Verlag, 156-167.

Dong, J. S., Lee, C. H., Li, Y.F., & Wang, H. (2004, May). Verifying DAML+OIL and beyond in Z/EVES. In *Proceedings of the 26th International Conference on Software Engineering.*

Garcia-Molina, H. et. al. (1997). The TSIMMIS Approach to Mediation: Data Models and Languages. In *Journal of Intelligent Information Systems*, 117-132.

Gómez-Perez, A., Fernándes-López, M, & Corcho, O. (2004). *Ontological Engineering: with exemple from the areas of Knowledge Management, e-commerce and the Semantic Web*. Springer-Verlag,

Hameed, A., Preece, A., & Sleeman, D. (2003). Ontology Reconciliation. In Staab, S., and Studer, R. (eds.). *Handbook on Ontologies in Information Systems*, Springer- Verlag, 231-250.

ISO/IEC 13586:2002 – Z formal specification notation – Syntax, type system and semantics. Latest version available at http://www.iso.org

Laskey, K. J., Costa, P. C. G., & Laskey, K. B. (2006, November). A Proposal for a W3C XG on Uncertainty Reasoning for the WWW. Accepted to the *Proceedings of the second workshop on Uncertainty Reasoning for the Semantic Web (URSW 2006)*, to be held at the Fifth International Semantic Web Conference (ISWC 2006), Athens, GA, USA.

Medjahed, B., Bouguettaya, A., & Elmagarmid, A. K. (2003, November). Composing web services on the semantic web. *VLDB Journal*, *12,* 333-351.

Mocan, A., Cimpian, E., Stollberg, M., Scharffe, & F., Scicluna, J. (eds.). (2005, September 16). WSMO Mediators, Working Draft , latest version available at http://www.wsmo.org/TR/d29/

Paolucci, M., Kawamura, T., Payne, T. R., & Sycara, K. (2002, June). Semantic Matching of Web Services Capabilities. In *Proceedings of the First international Semantic Web Conference on the Semantic Web*. Horrocks, I., and Hendler, J. A. (eds.). *Lecture Notes In Computer Science*, *2342*., London: Springer-Verlag, 333-347.

Paolucci, M., Srinivasan, N., & Sycara, K. (2004). 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.

Ribeiro, C. M. F. A., Rosa, N. S., & Cunha, P. R. F. (2004, March). Towards a Model for Personalized Communication Services. In *Proceedings of The IEEE 18th International Conference on Advanced Information Networking and Application* (AINA 04). *IEEE Press*, *2*, 99-102.

Ribeiro, C. M. F. A. (2006, April). An Ontological Approach for Personalized Services, In *Proceedings of The IEEE 20th International Conference on Advanced Information Networking and Applications* (AINA 2006). *IEEE Press (2006)*, *2*, 5 pp.

Saaltink, M. (1997) The Z/EVES System. In Bowen, J. P., Hinchey, M. G., and Till, D. (eds.), ZUM'97: Z Formal Specification Notation, *LNCS*, Springer-Verlag, 72- 85.

Sheth, A., Thacker, S., & Patel, S. (2003, May). Complex relationships and knowledge discovery support in the InfoQuilt system. *The VLDB Journal*, *12*, 2–27.

Spivey, J. M. (1992). *The Z Notation: A Reference Manual*. Prentice Hall International, 2nd edition.

Tomasic, A., Raschid, L., & Valduriez, P. (1998). Scalling Access to Heterogeneous Data Sources with DISCO. In *IEEE Transactions on Knowledge and Data Engineering*, *10*, 808-823.

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

Wiederhold, G., & Genesereth, M. (1997, September-October). The Conceptual Basis for Mediation Services. *IEEE Expert: Intelligent Systems and Their Applications*, *12*, 38-47.

Yan, L., Özsu, M. T., & Liu, L. (1997, June). Accessing Heterogeneous Data through Homogenization and Integration Mediators. In *Proceedings of the 2nd IFCIS International Conference on Cooperative Information Systems*, South Carolina, USA, 130-139.