Formalizing Ontology Reconciliation Techniques as a Basis for Meaningful Mediation in Service-Related Tasks

Patrício de Alencar Silva  
Universidade Federal de Campina Grande  
Campina Grande, PB, Brazil  
patricio@dsc.ufcg.edu.br

Cláudia M. F. A. Ribeiro  
Universidade do Estado do Rio Grande do Norte  
Natal, RN, Brazil  
claudiaribeiro@uern.br

Ulrich Schiel  
Universidade Federal de Campina Grande  
Campina Grande, PB, Brazil  
ulrich@dsc.ufcg.edu.br

ABSTRACT
With the advent of Semantic Web, the fast dissemination of ontologies to represent and share information causes a deep impact on knowledge retrieval, as a whole. In this context, the use of different ontologies to express meaning on the same application domain leads to a kind of “Tower of Babel Effect” on the Web, bringing new problems to the communication among different applications. This problem still remains in service-oriented architectures, considering that a same service can be described by the use of different ontologies and standards. Therefore, the urge of an ontology reconciliation approach arises in order to enable communication despite the differences. This paper focuses on providing a formal description of ontology reconciliation techniques, such as merging, alignment and integration, to provide better understanding of how these techniques can be used on the scope of Semantic Web Services Architecture.

Categories and Subject Descriptors
F.4.1[Mathematical Logic and Formal Languages]: Mathematical Logic – logic and constraint programming, mechanical theorem proving, model theory, proof theory, set theory
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval– information filtering, retrieval models, selection process.

General Terms
Design, Theory, Verification.

Keywords
Ontology Reconciliation, Semantic Web, Service Discovery, Service Selection, Formal Methods, Z Notation.

1. INTRODUCTION
As the spread adoption of service-oriented paradigm becomes a reality, new challenges arise in software engineering. The main problems related to Service Oriented Architecture (SOA) can be summarized by: “how” services are described, “how” services can be discovered and “how” services are invoked.

Web services technologies, at the present moment, are the most usual way to implement SOA. The Web Services Description Language (WSDL), a XML-based language, is used to describe service’s functionalities, the UDDI (Universal Description Discovery and Integration)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PIKM’07, November 9, 2007, Lisbon, Portugal.
Copyright 2007 ACM 978-1-59593-832-9/07/0011...
$5.00.

states a way to organize registries that maintain services descriptions, and SOAP regulates the way services are invoked and communicate.

Despite the benefits and effectiveness of Web services to integrate disparate systems, the process of Web service discovery is syntactic instead of semantic. The consequence is a strong dependence of form, i.e. the service requester must know details of service operations and parameters to actually be able to use it. Additionally, at the same time, as the quantity of services and documents grows over the Web, the more difficult is to find that one which entirely fulfills requester’s necessities.

An alternative and a smarter manner to discover Web resources (e.g. documents and services), is advocated by Semantic Web [1]. The Semantic Web is considered an extension of the traditional one by the addition of meaning. The benefits of meaningful description are clear. The more we know about something, the greater is the chance to identify it, i.e. adding semantics can improve the way resources are discovered on the Web. The Semantic Web Services term identifies the category of Web Services whose description was semantically enriched [2].

The enrichment of description service allows the automation of many tasks related to services, such as discovery, selection, and composition. This feature can improve the way businesses are made on the Web, since service offering is becoming more personalized. Personalization adds soundness to services and is the key to distinguish providers in a competitive marketplace [3].

However, the inclusion of meaning to service description can originate divergences from different perspectives. For example, the way service providers characterize quality of service may possibly be different from the way clients understand quality. Hence, the approximation of these different points of view is a prerequisite to negotiate personalized services on the Web [3,4]. This is a typical case of reconciliation. More specifically, it is a case of ontology reconciliation, since knowledge is structured by ontologies on the Semantic Web, and each perspective corresponds to ontology [5]. This is precisely where this paper focuses on.

There are some techniques to treat ontology reconciliation. They can be grouped into two categories. The first one, named vertical approach, concerns the creation of upper-level ontologies to integrate different ontologies. The second one, known as the horizontal approach, consists of mapping relationships between ontologies. Ontology reconciliation techniques involve complex issues, therefore we adopted the Z notation [6] as a strategy to deeply investigate and describe their functionalities. We aim to demonstrate how these techniques can be used as a mediation process in service-related tasks, such as service discovery and selection.
This paper is organized as follows. Section 2 presents some issues related to the convergences and divergences that arise from the merging between Semantic Web and SOA. Section 3 is dedicated to ontology reconciliation techniques explanation and Section 4 deals with formalization of these techniques by the use of Z notation. Section 5 presents two scenarios where ontology reconciliation techniques can be used as a basis for approximate clients and providers through service discovery and selection. Finally, Section 6 presents some conclusions and points to a future work.

2. TOWER OF BABEL EFFECT IN SOA AND SEMANTIC WEB

Nowadays, the Web can be considered as the largest information repository ever known. In this environment, freedom of publication and expression leads to a rapid dissemination of pages which offers access to whatsoever resources. Simple resources include texts, videos, advertisements, while more sophisticated ones include, for instance, Web services which allows access to another resources (e.g. search engines). It implies changes on service provisioning in general.

The Semantic Web envisioning comprises the “Web revolutions”, where the information will be available with a well-defined meaning [1]. That means explicit definition of relationships among information inside several knowledge domains. For this purpose, ontologies have been used for linking information that are related somehow and somewhere on the Web. These semantic information cells can also be used to enrich service descriptions. The merging between Semantic Web and Web Services Architecture comprises a new effort in distributed information systems, named Semantic Web Services. The main purpose behind that fusion is the automation of service-related tasks, such as service discovery, selection, composition and establishment of service level agreements.

Figure 1. Domain ontology as an intersection between OWL-S and WSMO approaches

Although great expectations about Semantic Web services may exist, it is worth noticing that we live in a multiple vision world. In the context of Semantic Web, this “parallel worlds” are encapsulated by ontologies, which represent different visions about several knowledge domains. In other words, it is unrealistic to expect a global consensus between people and organizations onto a common, shared ontology. This semantic heterogeneity is considered as a kind of “Tower of Babel Effect”, bringing new problems on knowledge sharing and interchanging among different applications [5]. The same occurs in service-oriented architecture, considering that different approaches exist for service description, including OWL-S [7] and WSMO [8]. As shown in Figure 1, these models describe the same thing (i.e., the service), yet differently.

However, both approaches use domain ontologies in order to define service classifications. Functional requirement classes can be used to allow service discovery (i.e., “what” the service does?), while non-functional requirements can be used in service selection (i.e., “how” it works?). In other words, domain ontologies can be considered as a bridge upon which reconciliation can be applied. Therefore, it becomes clear the need for an ontological reconciliation approach in order to close the gap between vocabularies used by the applications. It is important noticing that semantic heterogeneity is a problem that crosses Semantic Web. It well deserves better considerations about semantic mediation in SOA, as well as agent-oriented computing, which are two important paradigms in software development and deployment on the Web [4, 9].

Substantial work has been done on ontology mapping and integration [10, 11, 12, 13]. However, more investigation must be made. For instance: (1) there is a misunderstanding and a weak consensus about what ontology reconciliation techniques really are (e.g. integration, merging or alignment techniques) [14]; (2) ontology reconciliation is a very complex computational problem and requires some precision degree in modeling algorithmic solutions [1]; (3) the formal definition of each ontology reconciliation technique is vital to apply or combine techniques in order to solve real world problems, such as the “Tower of Babel Effect” in service-oriented architectures [4,15]; (4) the effective implementation of trustworthy mediation services for ontological reconciliation also depends on correctness in modeling its behavioral and functional properties, considering for instance, the amount of preconditions in axiom analysis during alignment process [9]. All of these aspects must be treated to make the Semantic Web a reality for enterprise software development.

3. ONTOLOGY RECONCILIATION

Ontology reconciliation is a complex task that includes three processes: merging, integration and alignment [14]. Merging is a fusion of two ontologies into a single one [12]. This process can be made progressively, by unifying several ontologies. The main purpose consists on creating a single coherent ontology that includes information from all sources. The new ontology is created from two or more existing ontologies with overlapping parts. Integration consists in building a new ontology by composing parts of other available ontologies [16]. In the same way as merging, this process results in a new ontology. The main difference between integration and merging is that only parts of the original ontologies will be integrated. In other words, the goal of integration is not to achieve a complete merging. These two techniques represent a vertical approach for ontology interoperability (vide Figure 2). The result is an “upper-level” ontology, which can be used as a global schema for query propagation among the original ontologies, besides representing a more abstract vision of the original concepts and properties.
Formal methods, based upon elementary mathematics, can be used to produce precise and unambiguous architectural descriptions, in which information is structured and presented at an appropriate abstraction level. Hence, reasoning about a specification and attempting to construct proofs about its properties greatly help detecting problems at an early stage of systems 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.

The use of formal methods for describing and verifying behavioral properties of systems is not a new approach [18]. The Z notation [6], for example, is a specification language based on the set-theory and predicate calculus is widely used in this context. Indeed, some basic characteristics of Z notation guided its choice as the formalism in this work. The first one is related to the morphism relation with Semantic Web languages such as OWL. Z is also considered as a meta-language for verifying ontological properties on the proof level [19]. The second one is related to its maturity level, recently conveyed into ISO standard. The third one concerns the availability of tools to support formal activities, such as type checking and theorem proof [15].

The fundamentals of Z notation are types sets defined at the beginning of the specification. A “given set” is the genesis of a Z specification, represented by names inside brackets, from which a formal specification begins. Enumerated sets are also permitted in Z notation. In the following, it is enumerated some of the main given sets used during formal modeling.

\[ \text{CLASS, DATATYPE, ID, INSTANCE, MESSAGE, OPERATION, PROTOCOL, RESOURCE, URI, REPORT ::= Match \mid Impasse \mid ServiceDiscovered} \mid \text{NoServiceDiscovered} \mid \text{NoServiceSelected} \mid \text{ServiceSelected} \]

Another key element of Z specification is the schema. In an analogous way, the schema can be considered as a class inside the object-oriented paradigm. As the same fashion as class, the schema includes a declarative part to encompass variables and a second part dedicated to the manipulation of variables, the predicate. Schemas in Z are used to describe both static and dynamic aspects of a system. The static aspect concerns the global state of the system and the relationships between its components, namely the invariant relationships. A rigid control over the state integrity is guaranteed by the invariants during any operation execution that changes the global state. Dynamic aspects include all the schema operations that manipulate the elements of the global state [6, 20].

\[
\begin{array}{l}
\text{SchemaName} \\
\text{Declaration Part (variant scope)} \\
\text{Predicate Part (invariant/constraints scope)}
\end{array}
\]

4.1 Merging

The merging process is composed by three schemes: (1) MergingInTotalImport, which specifies the merging between two ontologies when both imports a same “third” ontology; (2) MergingInPluginAndSubsumptionImport, when there is a partial import in one ontology and a total import in the other; and (3) MergingInIntersectionImport, when both ontologies import a same ontology partially.
Each schema is composed by another simplified schemas, for class, data type properties, object properties and individuals merging (for the sake of simplicity, some of the schemas will be suppressed). Formally, the complete merging process can be defined by the use of the symbol “∪”, in order to define a sequential composition (on the same manner as a state machine).

Considering two source ontologies, \( O_1 \) and \( O_2 \), the imported ontology \( O_3 \), the set of relationships propagated from the other relationships amongst classes, properties and individuals from \( O_1 \) to \( O_1 \) and \( O_2 \), denominated \( P_{1\text{propagation}} \cup P_{2\text{propagation}} \), respectively, so the ontology \( O_3 \) generated by merging application will be:

1. \( O_3 \), if and only if \( O_1 = O_1 \) and \( O_2 = O_2 \) are true propositions;
2. \( O_3 \), if and only if \( O_1 \subseteq O_1 \) and \( O_2 \subseteq O_2 \) are true propositions;
3. \( O_3 \), if and only if \( O_2 = O_1 \) and \( O_1 = O_2 \) are true propositions;
4. \( O_3 \cup (P_{1\text{propagation}} \cap P_{2\text{propagation}}) \), if and only if \( O_1 \cap O_2 \subseteq O_1 \) and \( O_1 \cap O_2 \subseteq O_2 \).

These cases are illustrated in Figure 4 (suppressing the case 2, commutative with the case 3). In the first case (vide Figure 4a), where \( O_1 \) importation is total in \( O_1 \) and \( O_2 \), the result ontology \( O_1 \) is equivalent from \( O_2 \). Considering that the merging process is different from integration one by the fact that concept merging eliminates duplications or redundancy, the merging in this case is total, without any loss. In the second case (the third one is commutative, considering domain and range inversion), one of the source ontologies fully imports \( O_2 \), while the other one imports \( O_2 \) partially. Considering that \( O_2 \) is a proper set of \( O_2 \), the result ontology aggregates \( O_2 \) and \( O_2 \) extension (vide Figure 4b). On the fourth and last case (vide Figure 4c), the source ontologies import \( O_1 \) partially. The relationships propagation amongst \( O_1 \) and \( O_2 \) produces new semantic links, which can also contain similarities and differences. The set of concepts and properties from \( O_1 \) and \( O_2 \), which are connected with the imported concepts somehow, were denominated \( P_{1\text{propagation}} \) and \( P_{2\text{propagation}} \), respectively. Hence, the core ontology \( O_3 \) will be composed by the common concepts and properties from \( O_1 \) and by the intersections among the propagated relationships in \( O_2 \) and \( O_2 \). This is the most complex case in ontology merging process.

The IndividualMergingInIntersectionImport schema contains two input variables (\( O_1?\) e \( O_2? \), for representation of two source ontologies), an output variable (CoreOntology!) and imports a set of ontologies for axiomatic manipulation and calculus (\( \Delta \)Ontologies, for state changing, and \( \Xi \)IndividualAxioms, for operations that do not change the system state). The predicate defines the preconditions which must be concatenated in order to express if the imported ontology (i.e. the shared concepts) represents all the input ontologies. In this case, the result ontology corresponds to the imported ontology, considering that the last one represents the intersection between the source ontologies. Analogous situation occurs for properties and individuals. The entire process is defined by the composite schema Ontology Merging.
4.2 Alignment

The ontology alignment process generates a set of mappings between the source ontologies. These mappings (vide MappingSet schema) consist in a set of associations started by the user, considering that alignment is a semi-automatic process, where the user indicates which concepts, in two different ontologies, are equivalent or similar (vide Figure 5). In general, the EquivalentFrom axiom is used for starting the first association (in the case of “individuals” it is used the SameAs axiom). The EquivalentFrom axiom is often used because it represents the strongest relationship between classes and properties. The composite schema OntologyAlignment specifies the alignment of classes, properties and individuals. This technique can be used in service selection, where the goal is to select quality of service parameters in order to fulfill the user needs, by considering non-functional properties of the services.

![Figure 5. Class alignment with equivalence relationship](image)

\[
\begin{align*}
\text{MappingSet} & = \{ \text{Ontologies}, \text{ClassAlignments}, \text{ClassMappings}, \text{ObjectPropertyMappings}, \text{DataTypePropertyMappings}, \text{IndividualMappings} \} \\
\text{Ontologies} & \rightarrow \text{Ontology} \\
\text{ClassAlignments} & \rightarrow \text{CLASS} \\
\text{ClassMappings} & \rightarrow \text{CLASS} \\
\text{ObjectPropertyMappings} & \rightarrow \text{ObjectProperty} \\
\text{DataTypePropertyMappings} & \rightarrow \text{DataTypeProperty} \\
\text{IndividualMappings} & \rightarrow \text{Individual} \\
\end{align*}
\]

\[
\forall c1, c2: \text{CLASS}; \text{objp1}, \text{objp2}: \text{ObjectProperty}; \text{dtp1}, \text{dtp2}: \text{DataTypeProperty}; i1, i2: \text{Individual} \\
\begin{align*}
\text{c1 ClassMappings} & \rightarrow c2 \\
\text{class} & \equiv (c1 \in \text{O1}; \text{Classes} \land c2 \in \text{O2}; \text{Classes}) \\
\text{objp1 ObjectPropertyMappings} & \rightarrow \text{objp2} \\
\text{DataTypeProperty} & \rightarrow \text{O1}; \text{ObjectProperties} \land \text{objp2} \in \text{O2}; \text{ObjectProperties} \\
\text{ObjectProperties} & \rightarrow \text{O1}; \text{ObjectProperties} \land \text{objp2} \in \text{O2}; \text{ObjectProperties} \\
\text{DataTypes} & \rightarrow \text{objp1 DataPropertyMappings} \land \text{dtp2} \\
\text{DataType} & \rightarrow \text{O1}; \text{DataTypes} \land \text{dtp2} \in \text{O2}; \text{DataTypes} \\
\end{align*}
\]

\[
\begin{align*}
\text{ObjectPropertyIntegration} & \rightarrow \text{CLASS} \\
\text{ObjectPropertyAlignment} & \rightarrow \text{ObjectProperty} \\
\text{DataTypePropertyAlignment} & \rightarrow \text{DataTypeProperty} \\
\text{IndividualAlignment} & \rightarrow \text{Individual} \\
\end{align*}
\]

4.3 Integration

The integration process represents the simplest operation for core ontologies creation. In this case, a generalized union of concepts, properties and individuals occurs, with no analysis about implicit intersections. In other words, blocks of previously existent ontologies are reused, even if conceptual divergences may exist. Ontology integration can be useful when conceptual blocks already exist for the desired purpose. For instance, a service level agreement (SLA) must contain a provider section and a client section, for the purpose of documentation of the involved profiles. Considering that several ontologies for user profile description exist, integration may be used in order to promote fast instantiation of electronic contracts in service-oriented applications.

\[
\begin{align*}
\text{OntologyIntegration} & \equiv \text{ClassIntegration} \\
\text{Ontology} & \equiv \text{OntologySet} \\
\text{ObjectProperty} & \equiv \text{ObjectProperty} \cup \text{ObjectProperty} \\
\text{DataTypeProperty} & \equiv \text{DataProperty} \cup \text{DataProperty} \\
\text{Individual} & \equiv \text{Individual} \\
\end{align*}
\]
5. USAGE SCENARIOS

5.1 Service Discovery

Although ontology reconciliation techniques can be applied in several contexts, this paper focuses on the task of service discovery. Considering that the main goal in this phase consists in retrieve lots of services that will enable the user to choose the best one, the integration technique can be applied for concepts aggregation from different service descriptions [21]. Furthermore, the composition of a core ontology allows querying for concepts in more abstract terms, with progressive mapping toward more specific ones, by exploring object properties (relating class to class) and basic axioms (e.g. equivalence).

For example, consider two tourism domain ontologies, $O_1$ and $O_2$, which import concepts from a third ontology $O_3$, as shown in Figure 6. The class DepartureArriveInformation extends the original concept from the imported ontology through an equivalence axiom (equivalentFrom). The class Point2PointInformation extends the same concept from $O_2$ through a generalization axiom (isSuperClassOf). These relationships propagate from the imported concepts in $O_3$ to the range ones in $O_1$ and $O_2$. The core ontology generated by a merging process contains the source concepts from $O_1$ and a partial intersection between these concepts. Then, the core ontology represents a global schema which aggregates common concepts from the source ontologies. Some preconditions are specified in ClassMergingInIntersectionImport schema, which defines the case of partial concept importation in $O_1$ and $O_2$.

Figure 6. Relationship propagation in service discovery

Considering that the desired service functionality may be described by different domain ontologies, the merged ontology represents a global schema which aggregates more abstract concepts from the source ontologies. Hence, the ontology merging process antecedes the service discovery one. It is worth noticing that property or individual merging may generate too much specific core ontologies. It reduces the number of retrieved services. This refinement level can be applied in service selection process, by querying for most specific individuals.

The intersection between the relationship propagation in the domain ontologies depends on the linking axioms. For instance, the equivalentFrom axiom expresses a similarity level greater than the isSubClassOf one. The MergingInIntersectionImport schema specifies some of the preconditions to be considered in the process of inference of new relationships between the source ontologies. This operation corresponds to the case where the two source ontologies extend the imported one. The schema was reduced for the sake of simplicity and demonstration of the mentioned concepts. A complete model and formal demonstration of all the operations can be found in [22].

The input/output variables ($O_1?$, $O_2?$ and CoreOntology!, respectively) are given in the declarative part in the schema. In the declarative part, the preconditions related to the merging process are defined. The ClassAxiomSet variable represents the class axioms which will be included in the generated core ontology. The $c_1$ class belongs to $O_3$ ontology (instantiated by the class OriginDestinationInformationType); the $c_2$ class belongs to $O_2$ (instantiated by the class DepartureArriveInformation); the $c_3$ class belongs to the $O_3$ ontology (instantiated by the class Point2PointInformation).

The following condition expresses the existence of relationships of equivalence and generalization, as shown in Figure 6. The set comprehension is completed by the attribution of a new specification relationship (by the ClassAxiomSet.isSubClassOf set), between the classes $c_2$ and $c_3$.

\[
\text{GoalOrientedServiceDiscovery} \equiv \begin{array}{l}
\exists \text{PERSON/E} \\
\text{Goals?}: \text{FunctionalRequirement} \\
\text{ServiceSet!}: \text{ID} \rightarrow \text{ServiceDescription} \\
O_1!, O_2!: \text{Ontology}
\end{array}
\]

\[
\exists \text{GoalClasses, CapabilityClasses: P CLASS} \\
\text{GoalClasses} = \{ \text{fr: FunctionalRequirement} \mid \text{fr = Goals?} \} \\
\text{CapabilityClasses} = \{ \text{Registy: ServiceRegistry; id: ID; SD: ServiceDescription; fr: FunctionalRequirement} \mid \text{Registy \in Registries \land (id, SD) \in ServiceDescription; fr = SD \land Capability \land fr: Class} \} \\
\land \text{O1! \in OntologySet} \land \text{fr: Class} \\
\land \text{O1!. Classes = GoalClasses \land O2!. Classes = CapabilityClasses}
\]

\[
\text{ClassMergingInIntersectionImport} \equiv \begin{array}{l}
\Delta \text{Ontologies} \\
\exists \text{ClassAxioms} \\
O_1?!, O_2?!, \text{CoreOntology!: Ontology}
\end{array}
\]

\[
\exists \text{ClassAxiomSet: ClassAxioms; ImportedOntology: Ontology} \\
\land \text{ClassAxiomSet \in CoreOntology!. OntologicalClassAxioms} \\
\land \text{O1! \in OntologySet} \land \text{O2? \in OntologySet} \\
\land \text{ImportedOntology \in OntologySet \land CoreOntology! \notin OntologySet}
\]

\[
\text{ServiceDiscovery} \equiv \text{SyntacticalServiceDiscovery} \\
\land \text{(GoalOrientedServiceDiscovery \land InputOrientedServiceDiscovery) \land OntologyMerging}
\]
5.2 Service Selection

When multiple services fulfill the requester’s needs, selection process must be applied in order to aid the user in choosing the most appropriate one. This process is based on quality of services parameters (i.e. non functional requirements). These requirements can be defined by specific QoS ontologies or by domain ontologies, which may provide a more specific characterization about quality parameters related to the offered services. Following the example previously given, the functional properties can be specialized by non functional ones, as shown in Figure 7.

It is worth noticing that the user can be able to define QoS constraints (i.e. what “excludes” undesirable services) about the desired quality level, besides QoS preferences (i.e. what qualifies desired services) and QoS priorities (i.e. what “distinguishes” similar services in terms of quality). When multiple QoS ontologies are used in order to describe the same quality parameters related to the same domain, class alignment is necessary. This process allows the verification of correlation relationships amongst the non-functional requirements (i.e. convergences and conflicts). This process can be used in the service negotiation process, before the establishment of service level agreement. These QoS parameters can be formalized as follows:

- NonFunctionalRequirement
  - Id: ID
  - Class: CLASS
  - Domain: Ontology

In this case, the domain of the relationships QoSProvided and QoSPreferences is the type NonFunctionalRequirement (defined by classes in some ontology) and the range is of LEVEL type (defined by a power set ‘P’ from the DATATYPE). It explains the use of property alignment in the selection process. Specific data types vary according to implementation details. Therefore, the set defined by LEVEL is an abstraction for all the possibilities related to the use of different data types related to the attribute and its modeling properties.

The DataTypePropertyAlignment schema defines some of the preconditions considered when new relationships are inferred between the source ontologies. The input/output variables (O1?, O2? and MapSet!) are given in the declarative part of the schema. Here, preconditions related to the alignment process are defined. The MapSet variable is defined by the schema MappingSet, previously defined, which contains a set of relationships generated amongst classes, properties and individuals from the two source ontologies. The dp1 property belongs to O1 ontology, the dp2 property belongs to O2 and the dp3 belongs to O3. The following condition verifies the existence of equivalence relationships, as shown in Figure 7. The set comprehension ends with the attribution of a new axiom between the source ontologies (by the set defined by MapSet!.DataTypePropertyMappings), between dp1 e dp3 properties.

```
 DataTypePropertyAlignment
 +Ontologies
 +DataTypePropertyAxioms
 O1?, O2?: Ontology
 MapSet!: MappingSet
 Report!: REPORT

 MapSet!. DataTypePropertyMappings
 = { dtp1, dtp2, dtp3: DataTypeProperty;
     EquivalentDataTypeProperty: DataTypeProperty }

MapSet!. ClassMappings = {} ⇒ Report! = Impasse
MapSet!. ClassMappings ≠ {} ⇒ Report! = Match
```

Figure 7. DataTypeProperties alignment in QoS parameters selection
6. CONCLUSIONS AND FUTURE WORK

Ontology reconciliation is a core theme on the Semantic Web. A formal approach can be useful in order to provide a deep understanding in how it can be used in the context of service-oriented architectures. Considering that clients and providers do not share the same perspectives about the same service very often, these perspectives become more pronounced when different vocabularies are used.

In terms of future work we intend to investigate the application of ontology merging, alignment and integration in dynamic service composition, negotiation, monitoring and on the establishment of service level agreements. The main goal is the reasoning on the properties of these techniques, in mathematical abstraction level, so that a conceptual framework can be provided, decoupled from specific implementation issues. Furthermore, it will be investigated how the proposed model can be refined in terms of executable code, envisioning deployment of trustworthy semantic mediators for automatic ontology reconciliation, mainly in the context of e-commerce applications, where there is a strong need for a mediation-centered approach in order to close the gap between the different vocabularies used by the service clients and providers.

7. REFERENCES


