

**IST STREP PROJECT**  
**FP6-2004-IST-4-027517**

**2.4.9 ICT research for innovative Government**

Providing Integrated Public Services to Citizens at the National and Pan-European level with the use of Emerging Semantic Web Technologies  
(SemanticGov)

---

## D3.3: Analysis of Mediator Requirements and Mediator Implementation

---

**Editor :** Adrian Mocan, Federico M. Facca  
**Revision :** V1.0  
**Dissemination Level :** Public  
**Author(s) :** Adrian Mocan, Federico M. Facca, Sotirios K. Goudos, Nikolaos Loutas, Vassilios Peristeras, Konstantinos Tarabanis, Euthimios Kiriannakis, Theodota Papaioannou, Georgios Mpotzoris  
**Due date of deliverable :** 30.04.2007  
**Actual submission date :** 15.06.2007  
**Start date of project :** 01 January 2006  
**Duration :** 36 months  
**Organisation name of lead contractor for this deliverable :** LFUI

**Abstract:** This deliverable analyzes the conceptual requirements provide in Deliverable D2.1 and derives the set of technical solutions that need to be utilized in order to have these requirements fulfilled. Additionally, we describe the implementation details behind these solutions and we provide a comprehensive set of examples derived from the showcases identified in the SemanticGov project.

Project funded by the European Community under the FP6 IST Programme

© Copyright by the SemanticGov Consortium

The SemanticGov Consortium consists of:

Centre for Research and Technology Hellas (CERTH)	Coordinator	Greece
National University of Ireland – Digital Enterprise Research Institute (NUIG)	Partner	Ireland
Leopold-Franzens University of Innsbruck (LFUI)	Partner	Austria
University of Rome “La Sapienza”/DIS (UOR)	Partner	Italy
CAPGEMINI	Partner	The Netherlands
SOFTWARE AG	Partner	Germany
Ontotext (ONTO)	Partner	Bulgaria
ALTEC (ALTEC)	Partner	Greece
Greek Ministry of the Interior, Public Administration and Decentralization (MOI)	Partner	Greece
Region of Central Macedonia (RCM)	Partner	Greece
City of Torino (Citta Di Torino)	Partner	Italy

## History

<i>Version</i>	<i>Date</i>	<i>Modification reason</i>	<i>Modified by</i>
V0.1	11.04.2007	First cut of the table of contents	Adrian Mocan
V0.2	18.04.2007	First version of Section 3	Adrian Mocan and Federico M. Facca
V0.3	23.04.2007	Content added to Section 4	Adrian Mocan
V0.4	31.05.2007	Realignment of the document	Federico M. Facca
V0.5	01.06.2007	Compilation of the version to be sent to the reviewers	Adrian Mocan
V0.6	07.06.2007	Applying reviewer comments.	Federico M. Facca
V1.0	08.06.2007	Final refinements	Adrian Mocan

## Table of contents

<b>HISTORY</b> .....	<b>3</b>
<b>TABLE OF CONTENTS</b> .....	<b>4</b>
<b>EXECUTIVE SUMMARY</b> .....	<b>6</b>
<b>1 INTRODUCTION</b> .....	<b>7</b>
<b>2 BACKGROUND</b> .....	<b>8</b>
2.1 THE WEB SERVICE MODELING LANGUAGE.....	8
2.2 THE ABSTRACT MAPPING LANGUAGE.....	10
<b>3 TECHNICAL REQUIREMENTS FOR MEDIATORS</b> .....	<b>12</b>
3.1 PRELIMINARIES .....	12
3.2 DATA LEVEL CONFLICTS .....	14
3.2.1 DATA VALUE CONFLICTS .....	14
3.2.2 DATA REPRESENTATION CONFLICTS.....	18
3.2.3 DATA UNIT CONFLICTS .....	21
3.2.4 DATA PRECISION CONFLICTS .....	21
3.3 SCHEMA-LEVEL CONFLICTS .....	22
3.3.1 NAMING CONFLICTS.....	23
3.3.2 ENTITY IDENTIFIER CONFLICT .....	25
3.3.3 SCHEMA-ISOMORPHISM CONFLICTS.....	25
3.3.4 GENERALIZATION CONFLICTS.....	26
3.3.5 AGGREGATION CONFLICTS .....	28
3.3.6 SCHEMATIC DISCREPANCIES .....	29
<b>4 MEDIATION SERVICES AND SOLUTIONS</b> .....	<b>30</b>
4.1 THE SEMANTIC GATEWAY ARCHITECTURE.....	30
4.2 DATA MEDIATOR .....	31
4.2.1 ONTOLOGY MAPPING TOOL .....	31
4.2.2 RUN-TIME ENGINE .....	35
4.3 RESOLVING THE SEMANTIC INTEROPERABILITY CONFLICTS .....	35
4.3.1 DATA LEVEL CONFLICTS .....	35
4.3.2 DATA REPRESENTATIONS CONFLICTS .....	37
4.3.3 DATA PRECISION CONFLICTS.....	38
4.3.4 ENTITY IDENTIFIER CONFLICT.....	39
4.3.5 SCHEMA-ISOMORPHISM CONFLICTS .....	40
4.3.6 GENERALIZATION CONFLICTS .....	40
4.3.7 AGGREGATION CONFLICTS .....	41
4.4 APPLYING THE DATA MEDIATOR.....	41
<b>5 CONCLUSIONS</b> .....	<b>43</b>

**REFERENCES ..... 44**

## Executive summary

Interoperability issues are one of the most challenging problems in modern information systems that are spread over different peers and rely on heterogeneous information and process models. This is a core issue especially for e-Government information systems that require communication with different cross-border services. The main goal is to seamlessly provide service integration to the user (citizen) of the service. Our analysis uses basic concepts of the generic public service object model of the Governance Enterprise Architecture (GEA) and the Web Service Modeling Ontology (WSMO) to express the semantic description of the e-services and by this the data model for the underlying ontologies. Based on the above we present a mediation infrastructure part of the overall system's Semantic Gateway that is capable of resolving semantic interoperability conflicts at a pan-European level. We provide several examples to illustrate both the technical needs to solve such semantic conflicts and the solution we have adopted in our approach.

This deliverable analyzes the conceptual requirements provided in Deliverable D2.1 and derives the set of technical solutions that need to be utilized in order to have these requirements fulfilled. Additionally, we describe the implementation details behind these solutions and we provide a comprehensive set of examples derived from the showcases identified in the SemanticGov project.

# 1 Introduction

The Web has been continuously growing in the past decade and its growth conveyed to the wide use of information systems based on modern Web technologies (e.g. Web services). The distributed nature of the Web has led to the urgent necessity of integrating and allowing communication between arbitrary informative systems based on heterogeneous technologies and different data models. These integration problems are often referred to as interoperability conflicts and a lot of efforts are currently ongoing to propose the silver bullet to these issues. One of the most promising solutions, which opens new opportunities and in the same way new challenges, is the use of semantic technologies [14], [23].

Also public administration are facing nowadays this problem, especially in the European context where a number of different public administrations due to the laws of the European Community needs to integrate their e-services to provide better and cross border services to the European citizens. Because of this reason the European community itself is pushing the research efforts to find a solution for interoperability issues among cross-border e-services by financing a number of projects. In this deliverable we present a mediation infrastructure able to solve interoperability issues based on semantic technologies, under the conceptual model of Web Service Modeling Ontology [14]. The analysis of the interoperability issues solved by our framework is based on the classification in D2.1 [6], grounded to the public administration domain through the model for service provision provided by the Governance Enterprise Architecture [11].

We analyze a comprehensive set of semantic interoperability conflicts and draw the technical solutions we have adopted in order to solve them. These technical solutions are discussed mainly in terms of the alignment needed between the conflicting ontologies as well as in terms of the accompanying infrastructure required to properly use such alignments. Further on, we re-iterate through these types of semantic conflicts and show how the previously described solutions are implemented and delivered to the human domain or integration expert. That is, we propose a set of semi-automatic methods and a graphical tool that can be used by the human user to semi-automatically create mappings and mapping rules that are to be eventually automatically executed during run-time mediation processes.

The document is organized as follows: Section 2 gives an overview of the background concepts adopted through the paper. In Section 3 we briefly review the classification of semantic interoperability conflicts and we present example solution of the various types of interoperability conflicts adopting a Semantic Gateway. Section 4 presents the current status of the implementation of our framework and finally in Section 5 we draw some conclusion and point some future directions.

## 2 Background

This section gives an overview of background concepts that form the basis of the work presented in this deliverable. In particular, the classification of the interoperability conflicts (see Section 3) is based on the work carried on in D2.1 [6]. Furthermore we adopt the GEA [11] concepts to provide examples in the cross-border eGovernment services domain. Regarding the technology adopted to solve the interoperability issues our research relies on the WSMO [14] conceptual framework for semantic Web services and the examples reported in the next sections are based on this framework and its modeling language (i.e. Web Service Modeling Language – WSML [1], [2]). Data mapping examples rely on WSML and on the Abstract Mapping Language proposed in [4], [5]. Finally the implementation relies on the components of the SemanticGov architecture for the runtime support and on SemanticGov User Tools, a semantic Web services design toolkit for WSMO, for the design time support (as depicted in [15]).

The next two subsections give an overview of the Web Service Modeling Language and of the Abstract Mapping Language due to their important role in the solutions we present in Section 3 and Section 4. Details about the WSMO and GEA models can be found in the state of the art report in D1.2 [24].

### 2.1 The Web Service Modeling Language

The Web Service Modeling Language (WSML) [1], [2] offers a set of language variants for describing WSMO elements that enable modelers to balance between expressiveness and tractability according to different knowledge representation paradigms.

WSML makes a clear distinction between the modeling of conceptual elements (e.g. Ontologies, Web Services, Goals and Mediators) and the specification of the logical definitions (the conceptual syntax and the logical expression syntax, respectively). The conceptual syntax is developed from the user's perspective, in order to hide the complexity of the underlying logics. The user can still have access to the full power of logic formalism by using the logical expression syntax. In the rest of this subsection the main elements of the syntax used to model ontologies are presented.

The conceptual syntax of WSML is a frame-like style. The information about classes, attributes, relations and their parameters, instances and their attribute values are specified in one large syntactic construct, instead of being divided in a number of atomic chunks.

**Listing 1.** An ontology fragment modeling concepts for the Italian PA (keywords are evidenced in **bold**)

```

1  concept City
2      name ofType (1 1) _string
3      hasCountry ofType Country
4
5  concept Country
6      name ofType (1 1) _string
7      isoCode ofType (1 1) CitizenshipCode
8
9  instance Italy memberOf Country
10     name hasValue "Italy"
11     isoCode hasValue IT
12
13 concept Person
14     hasFirstName ofType (1 1) _string
15     hasSecondName ofType (0 *) _string
16     hasSurname ofType (1 1) _string
17     hasGender ofType (1 1) Gender
18     hasCitizenship implyType Citizenship
19     hasMaritalStatus ofType (1 1) MaritalStatus
20     hasBirthday ofType (1 1) _dateTime
21     hasBirthplace ofType (1 1) City
22

```

---

```

23  instance anna memberOf Person
24      hasFirstName hasValue "Anna"
25      hasSurname hasValue "Nibioli"
26      hasGender hasValue F
27      ...
28      hasBirthplace hasValue Bergamo
29      hasMaritalStatus hasValue Celibe
30
31  concept Address
32      streetName ofType (1 1) _string
33      streetNumber ofType (1 1) _int
34      city ofType (1 1) City
35      country ofType (1 1) Country
36      zipCode ofType (0 1) _string
37
38  concept ResidenceAddress subConceptOf Address
39
40  relation lives (ofType Person, ofType ResidenceAddress)
41  relationInstance anna_address lives(anna, annaAddress)
42
43  axiom italianCitizienRule
44      definedBy
45          ?x[hasBirthplace hasValue ?y] memberOf City
46          and ?y[hasCountry hasValue ?z] and ?z=Italy
47          implies ?x [hasCitizenship hasValue italianCitizenship].

```

---

Listing 1 shows a fragment from the “Italian Citizen” ontology, an ontology developed for the SemanticGov project. This ontology models some generic concepts related to the Italian PA domain: like `Person`, `Country`, `City`, `Citizenship`, `Address`, `MartialStatus`, and so on. At lines 13 to 21 the concept `Person` is being defined together with its attributes. In WSMML the attributes are defined locally to concepts using the keyword `ofType` or `impliesType`. In the first case it is imposed that the value taken by that attribute has to be of the specified type (otherwise a constraint violation is signaled by the reasoner). When using `impliesType` the reasoner will infer that the attribute value is of the specified type (this is similar with the OWL properties behavior [17]). Note that concepts can be organized in sub-class hierarchies by using the `subConceptOf` keyword (line 38).

Cardinality constraints can be specified as at lines 14 to 21: `hasGender` has cardinality 1 and `hasSecondName` has cardinality 0 or more. Generally WSMML allows the specification of relations with arbitrary arities, while some WSMML variants restrict the arity to 2; the parameters’ domains are specified by using `ofType` or `impliesType` (see line 40). One can also specify the super-relation of a relation by using the keyword `subRelationOf`.

Instances of concepts can be defined as exemplified at lines 9-11 and 23-29. It is also possible to create instances of relations as well (line 41).

A way of using the logical expression syntax in WSMML is by the means of axioms (lines 43 to 47). The `italianCitizienRule` axiom states that if a `Person` has as birthplace (`hasBirthplace` with a given value  $y$ ) and the birthplace is a city situated in a country (`hasCountry` with value  $z$ ) and this country is Italy then the person has the Italian citizenship.

WSMML has direct support for different types of concrete data, namely strings, integers, and decimals, which correspond to the XML Schema [25] primitive data types and they can then be used to construct complex data types.

WSMML comprises different formalisms, most notably Description Logics and Logic Programming. Three main areas can benefit from the use of formal methods in service descriptions: *Ontology description*, *Declarative functional description of Goals and Web services*, and *Description of dynamics*. WSMML defines a syntax and semantics for ontology descriptions. The underlying logic formalisms are used to give a formal meaning to ontology descriptions in WSMML, resulting in different variants of the language:

**WSMML-Core** is based on the intersection between the Description Logic *SHIQ(D)* [18] and Horn Logic. It has the least expressive power of all the WSMML variants. The main features of the

language are concepts, attributes, binary relations and instances, as well as concept and relation hierarchies and support for data types.

**WSML-DL** captures the Description Logic  $\mathcal{SHIQ}(\mathbf{D})$ , which is a major part of the (DL species of) Web Ontology Language OWL [17].

**WSML-Flight** is an extension of WSML-Core which provides a powerful rule language. It adds features such as meta-modeling, constraints and non-monotonic negation. WSML-Flight is based on a logic programming variant of F-Logic [19] and is semantically equivalent to Datalog with inequality and (locally) stratified negation. WSML-Flight is a direct syntactic extension of WSML-Core and it is a semantic extension in the sense that the WSML-Core subset of WSML-Flight agrees with WSML-Core on ground entailments.

**WSML-Rule** extends WSML-Flight with further features from Logic Programming, namely the use of function symbols, unsafe rules, and unstratified negation under the well founded semantics.

**WSML-Full** unifies WSML-DL and WSML-Rule under a First-Order umbrella with extensions to support the non-monotonic negation of WSML-Rule.

## 2.2 The Abstract Mapping Language

One of the most common techniques in resolving the semantic conflicts is to create an *alignment* between these ontologies which formally specifies how they relate. An alignment normally consists of a set of *mappings* where a mapping is either a unidirectional or bidirectional statement that usually captures the semantic relationships existing between entities part of these ontologies [20]. Mappings can be expressed either directly in an ontology representation language (e.g. as logical statements) or in specialized mapping language [4], [5]. In the first case the mappings can be directly seen and used as mapping rules (i.e. they can be evaluated in a reasoning engine), while in the second case grounding has to be applied to the mappings in order to obtain rules in a concrete language.

We chose to express the mappings in the Abstract Mapping Language proposed in [4] and elaborated in [5] because it does not commit to any existing ontology representation language. Later, a formal semantic has to be associated with it and the mappings to be grounded to a concrete language (e.g. WSML).

A mapping in the Abstract Mapping Language has the following form<sup>1</sup>:

```
mapping ::= 'Mapping(mappingId [{annotation}] {statement})'
```

The `mappingId` is a unique identifier of the mapping, while `annotation` is a free form explanatory text providing a textual description of the mappings. The mapping language statements are briefly described below:

- `classMapping` specifies mappings between classes in the source and the target ontologies. Such a statement can be conditioned by class conditions (`attributeValueConditions`, `attributeTypeConditions`, `attributeOccurrenceConditions`) and it has the following form:

```
statement ::= 'classMapping('one-way'|'two-way' [{annotation}]
               classExpr classExpr {innerCM_AttributeMapping}
               {classCondition} ['logicalExpression'] )'

classCondition ::= 'attributeValueCondition(attributeId comp
                                             (individualId | dataLiteral))'
```

<sup>1</sup> An abstract syntax expressed in EBNF is used. The content between ``` and `'` literally appears in the mapping language, elements between `{` and `}` can have multiple occurrences and elements between `[` and `]` are optional.

```

classCondition ::= 'attributeTypeCondition('attributeId comp classExpr ')
classCondition ::= 'attributeOccurrenceCondition('attributeId ')

comp ::= (less-than | less-than-or-equal | greater-than | greater-than-or-equal |
contains | starts-with | ends-with | matches | includes | includes-strictly |
empty)

```

The `attributeValueCondition` specifies for what values of a certain attribute the given mapping holds where the relationship between the attribute and values is given by a comparator as specified in [5]. The `attributeTypeCondition` indicates the type (range) a given attribute should have, while `attributeOccurrenceCondition` only imposes that a given attribute has to be present in the in the source or the target concept definitions for that mapping to hold (see Listing 3).

- `attributeMapping` specifies mappings between attributes. Such statements can appear together with or outside `classMapping` and can be conditioned by attribute conditions (`valueConditions`, `typeConditions`).

```

innerCM_AttributeMapping ::= 'attributeMapping('one-way'|two-way' [{annotation}]
attributeExpr attributeExpr {attributeCondition}

statement ::= 'attributeMapping('one-way'|two-way' [{annotation}]
attributeExpr attributeExpr
{attributeCondition} ['logicalExpression'])

attributeCondition ::= 'valueCondition('comp (individualId | dataLiteral) ')

attributeCondition ::= 'typeCondition(' comp classExpr ')

```

The `attributeCondition` are similar with the `classCondition`, with the difference that the first are applied on attributes, while the second are applied on classes. They specify the value or the type a certain attribute should have in order for the mapping to hold (see Listing 3).

- `classAttributeMapping` specifies mappings between a class and an attribute (or the other way around) and it can be conditioned by both class conditions and attribute conditions.

```

statement ::= 'classAttributeMapping('one-way'|two-way' [{annotation}]
(classExpr attributeExpr)|(attributeExpr classExpr)
{classCondition} {attributeCondition} ['logicalExpression'])

```

Since this type of mappings involves classes as well as attributes, both `classCondition` and `attributeCondition` are allowed (see Listing 23).

- `instanceMapping` states a mapping between two individuals, one from the source and the other from the target.

```

statement ::= 'instanceMapping('[{annotation}] individualId individualId ')

```

The `classExpr` and `attributeExpr` can be class identifiers or attribute identifiers, respectively, but also more complex expressions on these identifiers, e.g. conjunctions, disjunction, negation, etc. The class and attribute identifiers take the form of the identifiers used in the given ontology languages the abstract mapping language is grounded to. For example, if there are WSMML ontologies to be mapped these identifiers are IRIs [2]. The `logicalExpressions` represent extra refinements that can be applied to particular mappings.

Detailed examples are presented in the next section.

### 3 Technical Requirements for Mediators

This section shows how interoperability conflicts can be solved from a technical point of view adopting semantic technologies. The resolution of the conflicts is presented through a set of examples in cross-border eGovernment services domain and the relative discussions on the approach and the technical artifacts required in order to address these conflicts. The section starts with a preliminary discussion followed by the analysis of the two main types of conflicts: *Data Value Conflicts* and *Schema Level Conflicts*.

#### 3.1 Preliminaries

The definition and resolution of interoperability problems across the information systems of European Public Administration (PA) agencies is the necessary precondition for successfully developing Pan-European Public Services (PEPS). Information systems interoperability is an active research field for decades but has attracted a lot of interest during the past few years, e.g. [7], [8].

According to [9] the use of the semantic gateway is proposed as a means to overcome semantic interoperability problems. The semantic gateway provides a set of services which aim to harmonize the meaning of the information exchanged between PAs from different Member States (MS). In order to fulfil its objective the semantic gateway needs to have access to context data in the MSs, as well as to other kinds of data, including metadata translation tables (i.e. mappings and mapping rules).

As by D2.1 [6], for the discussion of the different types of interoperability conflicts that may arise in cross-border e-Government Services we refer to the work of Park et al. [10], who classify semantic interoperability conflicts regardless of the application domain. Furthermore, we use the core GEA concepts [11] to help us to instantiate the classification of Park et al. in the cross-border eGovernment services domain (see Section 3.1 and following ones for a detailed discussion on interoperability issues in cross-border eGovernment services and their resolution).

**Table 1.** Classification of interoperability issues according to Park and Ram [10]

Conflict Type		
Data-Level Conflicts	Data Value	
	Data Representation	
	Data Unit	
	Data Precision	Data Granularity
		Spatial Resolution
	Known Data Value Reliability	
Schema-Level Conflicts	Spatial Domain	
	Naming	Attribute Synonyms
		Attribute Homonyms
		Entity Synonyms
		Entity Homonyms
	Entity Identifier	
	Schema Isomorphism	
	Generalization	
	Aggregation	
	Schematic Discrepancies	Data Value-Attribute
Attribute-Entity		
Entity-Data Value		

Park et al. [10] argue that semantic conflicts in information systems can occur at two different levels: at the data level and at the schema level. *Data-level conflicts* are differences in data domains caused by the multiple representations and interpretations of similar data. *Schema-level conflicts* are semantic conflicts which are characterized by differences in logical structures or inconsistencies in metadata of the same application domain. The complete classification of Park et al. is reported in Table 1.

According to our research, the different types of conflicts reported in Table 1 can affect different elements in the cross-border eGovernment services domain. In particular the core GEA concepts from GEA that are usually involved in such conflicts are: evidences, evidence placeholders, service preconditions, service effects and clients.

**Table 2.** Interoperability conflicts in PEPS

	Conflict Type	Country A	Description	Country B
Data-Level Conflicts	Data-value	Evidence A	Different meaning	Evidence B
		Precondition A	Different meaning	Precondition B
		Effect A	Is not applicable or valid	Effect B
		Consequence A	Is not applicable or valid	Consequence B
	Data representation	Evidence value A	Different format	Evidence value B
	Interoperability Conflicts in Cross Border e-Government Services			
Data Units	Evidence value A	Expressed in different units	Evidence value B	
Data precision	Evidence value A	Expressed in different values scale/grade	Evidence value B	
Granularity	Object Property	Evidence A	Conflicts of any type	Evidence B
	Object	Evidence Placeholder A	Conflicts of any type	Evidence Placeholder B
Schema-Level Conflicts	Naming	Service Provider A	Similar names/different services Or different names/similar services	Service Provider B
		Evidence Placeholder A	Similar names/different usage Or different names/similar evidence	Evidence Placeholder B
	Entity Identifier	Client A	Identified differently	Client B
	Schema-isomorphism	Evidence Placeholder A	Contain different set of evidence	Evidence Placeholder B
	Generalization	Evidence Placeholder A	EP1 in one country = EP1+EP2 in another	Evidence Placeholder B1+B2
		Client A	Different categorizations/groups	Client B
	Aggregation	Service Provider A	Different administrative and organizational structures	Service Provider B
Evidence Placeholder A		Evidences aggregated differently	Evidence Placeholder B	
Schematic discrepancies	Evidence Placeholder A	Similar evidences different names	Evidence Placeholder B	

Table 2 summarizes a typology of semantic interoperability conflicts in cross-border eGovernment services using the GEA concepts. It is important to mention at this point that in the subsequent analysis we present the more prevailing and interesting case of these conflicts [21].

For each type of the interoperability conflicts we discuss the approach and the technical artifacts required in order to solve this conflict. We provide for most of the conflict types examples that illustrate these conflicts and the solutions adopted. We use WSML to model our examples; for brevity those parts of the source and the target models that are not directly affecting that particular conflict are modeled identically.

Since the *Granularity Conflict Type* in Table 2 represents rather a different classification of the previously listed conflicts at the data level than a new conflict type, in the next section we will not discuss this type of conflicts. The solutions to be applied are quite similar and it remains as an exercise for the reader to identify which of them are exactly applicable to this new classification.

Since we are acting in semantic environment all the notions and the concept we refer to in the PA domain have to be semantically described by the means of ontologies. In most of the cases the semantic interoperability conflicts are introduced by the differences in the ontologies modelling overlapping domains.

It is important to note that as far as mediation between different semantic models (i.e. different ontologies) is concerned in the PA domain, our focus is in providing techniques and solutions for seamless transformation of instances expressed in terms of on ontology (the source) in terms of instances expressed in terms of another ontology (the target). That is, different member states use different ontologies to model their information space and public administrations from these different member states are involved in interactions regarding PEPS consumption and implicitly message exchange. The Semantic Gateway needs to transparently provide mediation capabilities in such way that member states' public administrations can keep their mode of operation and still be able to communicate.

The interpretation and the execution of such mappings and mapping rules has to happen in a specialized component of the Semantic Gateway, component that will be further referred to as a *Data Mediator*. Further details on this component can be found in Section 4.

## 3.2 Data level conflicts

Data level conflicts occur when multiple representations and interpretations of similar data are used in the domain models that need to be mapped. The several types of possible data conflicts are described in this section (see Figure 1).

### 3.2.1 Data value conflicts

Such conflicts appear when identical notions or terms have different meaning in different ontologies.

**Listing 2.** Definition of *person*, *human* and *adult* in the source ontology  $O_1$  and in the target ontology  $O_2$

Ontology $O_1$	Ontology $O_2$
<pre> concept person   name ofType _string   age ofType _integer  concept adult subConceptOf person  axiom AdultDefinition   definedBy     ?x memberOf Adult :- ?x [age hasValue ?y]       memberOf person and ?y&gt;=17. </pre>	<pre> concept human   name ofType _string   age ofType _integer   ageCategory ofType AgeCategory  concept AgeCategory  instance adult memberOf AgeCategory  axiom AdultConstraint   definedBy     !- ?x[ageCategory hasValue adult] memberOf       human and ?x[age hasValue ?y] and ?y &lt; 18. </pre>

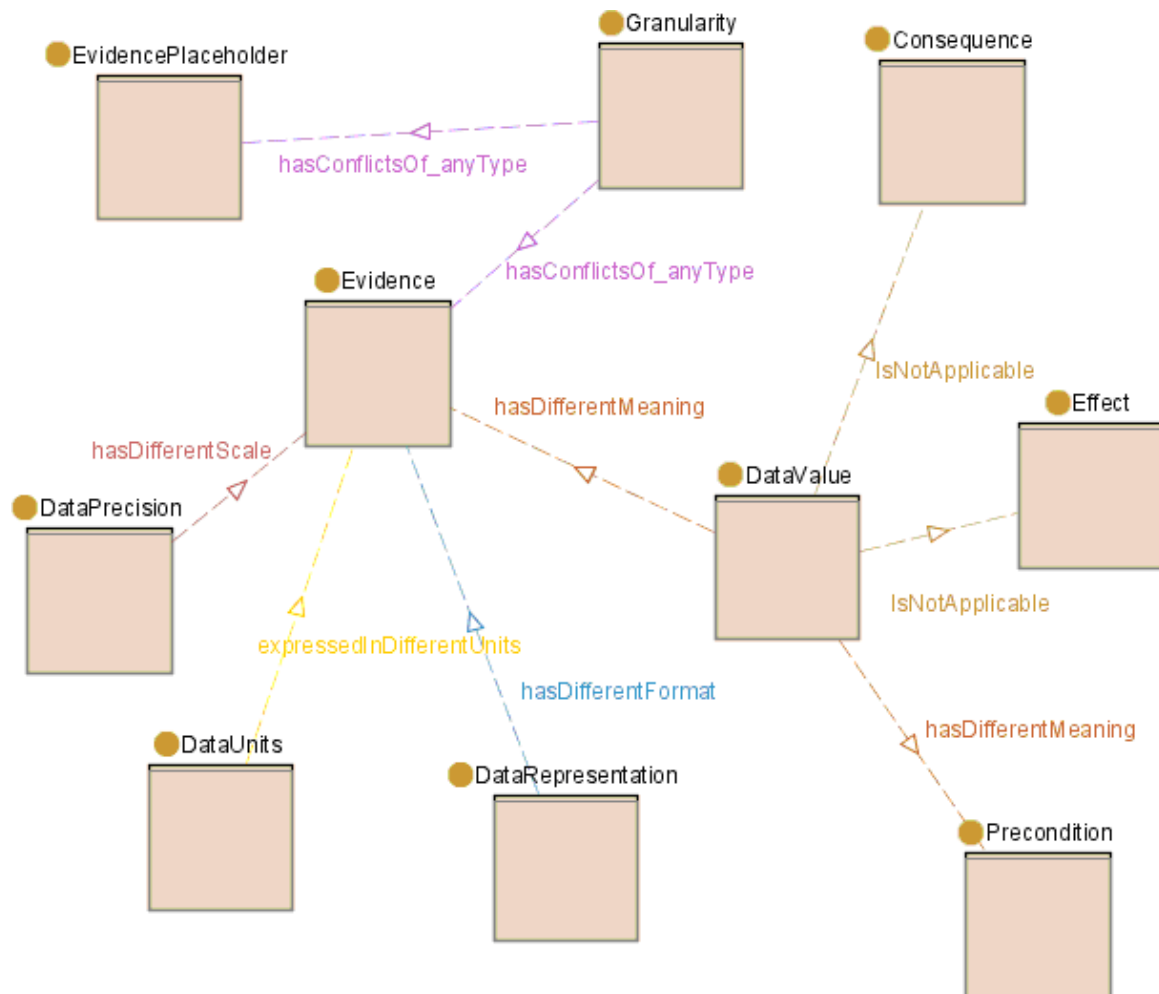


Figure 1. Data level conflict types

Listing 2 defines an “adult” in  $O_1$  by sub-classing the concept `person` and defining the new concept `adult` through an axiom. It also shows how “adult” could be modeled in the ontology  $O_2$  by adding a special attribute to the concept `human` and by constraining the values of these attribute. The axiom `AdultConstraint` triggers a constraint violation if an instance of `person` is created having the attribute `age` younger than 18 and the value of the attribute `ageCategory` populated with the instance `adult`. Both ontologies define the notion of `adult` but it has different meaning: in  $O_1$  an `adult` is a `person` older than 17 while in ontology  $O_2$  `adult` is an age category that identifies a human older than 18.

### 3.2.1.1 Differences in evidence domains caused by the multiple representations and interpretations of similar evidence

These conflicts arise when the same notion or term that is used as evidence is defined differently among different MSs, usually due to differences in legislation or to cultural differences.

The example in Listing 2 depicts the situation where similar evidences (i.e. a person or a human is an adult) are differently interpreted from one domain to another. If equivalences have to be established between evidence placeholders referring to the notion of `adult` mappings have to be created between the semantic models that capture these two definitions of `adult` (i.e. between concept of `person` in ontology  $O_1$  and concept of `human` in the ontology  $O_2$ ).

Listing 3 depicts a set of mappings between the ontologies  $O_1$  and  $O_2$  that are able to capture the differences in the `adult` definitions in the two ontologies.

Listing 3. Mappings between person and human resolving the adult conflict<sup>2</sup>

```

1  Mapping(o1o2o1#1
2    classMapping(two-way o1#person o2#human))
3
4  Mapping(o1o2o1#2
5    attributeMapping(two-way o1#person.name o2#human.name))
6
7  Mapping(o1o2o1#3
8    attributeMapping(two-way o1#person.age o2#human.age))
9
10 Mapping(o2o1#1
11  classMapping(one-way o2#human o1#adult)
12    attributeValueCondition(o2#human.age >=17))
13
14 Mapping(o1o2#1
15  attributeMapping(one-way o1#person.age o2#human.ageCategory)
16    attributeValueCondition(o1#person.age >= 18)
17    attributeValueCondition(o2#human.ageCategory = adult))

```

From lines 1 to 8 there are three bidirectional mappings that express the semantic relations between `person` and `human`: they both have a name and an age. At Lines 10 to 12 we have a unidirectional rule that specifies that a `human` is a special type of `person`, an `adult`, if the age of that `human` is higher than 17. Because `adult` is a sub-concept of `person`, based on the inheritance principles, the previous two mappings assures that the name and the age of an `adult` are correctly related with those of the `human`.

The data mediator part of the Semantic Gateway has to be able to apply these mappings in order to solve this type of conflicts whenever data is exchanged between different member states.

### 3.2.1.2 Differences in preconditions domains caused by the multiple representations and interpretations of similar preconditions

The conflicts in evidence data values may also cause problems in service pre-conditions due to the different legislation that governs the public service provision process in different MSs. Each Goal and Web service is semantically described in terms of the ontology that models that particular domain. Across domain there may appear data value conflicts as the one described in Section 3.2.1.1. Usually the preconditions are checked either by the discovery engine to match potential services compatible with the specified Goal or by the invocation engine to actually invoke the matched service.

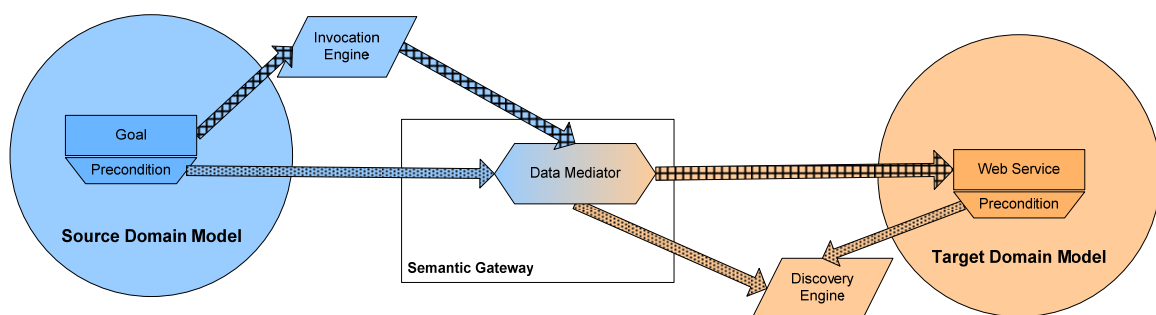


Figure 2. Mediated Discovery and Invocation

As depicted in Figure 2, if the discovery process takes place in a target domain those elements of the source domain used in describing the goal precondition need to be mapped to the corresponding ontological entities from the target domain. Because the ontological entities from the two domains are considered to be different unless stated otherwise (i.e. technically they are

<sup>2</sup> The statement `attributeValueCondition(o2#human.age >=17)` is in fact a syntactical sugaring (used for better readability of the examples) of `attributeValueCondition(o2#human.age, greater-than-or-equal, 17)`.

defined in ontologies with their own different namespaces), the data mediator has the role to make available the necessary mappings and mapping rules to the discovery engine (which otherwise would fail in finding the right matching services).

As a consequence we can conclude that if the data value conflicts are handled in the manner presented in Section 3.2.1.1 and if the discovery approach involves data mediation in the matching process this type of conflict is resolved.

### 3.2.1.3 Differences in effect domains caused by the multiple representations and interpretations of similar effects.

The notion of effect is tightly related to the notion of a service and its execution. As the service is modeled in terms of an ontological model the effect of the execution of that service is consistent with that particular model. Listing 4 shows two different models of driving license: the Irish driving license is divided in two different categories `temporary` and `permanent` and the German driver license which is always permanent and does not distinguish between these two categories.

**Listing 4.** Definition of the German and Irish driving license in the ontology  $O_1$  and  $O_2$

Ontology $O_1$	Ontology $O_2$
<pre> <b>concept</b> IrishDrivingLicense   ownerName <b>ofType</b> _string   type <b>ofType</b> IDLType  <b>instance</b> permanent <b>memberOf</b> IDLType <b>instance</b> temporary <b>memberOf</b> IDLType  <b>concept</b> person   name <b>ofType</b> _string   age <b>ofType</b> _integer   hasDrivingLicense <b>ofType</b>     IrishDrivingLicense  <b>relation</b> canDrive(<b>ofType</b> person,                   <b>ofType</b> _boolean)  <b>axiom</b> whoCanDrive <b>definedBy</b>   canDrive(?x, _boolean("true")):-     ?x[hasDrivingLicense <b>hasValue</b> _#]     <b>memberOf</b> person. </pre>	<pre> <b>concept</b> human   name <b>ofType</b> _string   age <b>ofType</b> _integer   ageCategory <b>ofType</b> AgeCategory   drivingLicense <b>ofType</b> GermanDrivingLicense  <b>concept</b> GermanDrivingLicense   ownerName <b>ofType</b> _string  <b>relation</b> canDrive(<b>ofType</b> person,                   <b>ofType</b> _boolean)  <b>axiom</b> whoCanDrive <b>definedBy</b>   canDrive(?x, _boolean("true")):-     ?x[drivingLicense <b>hasValue</b> _#] <b>memberOf</b>     human. </pre>

A PA service that issues driving licenses would have in both cases as output the driving license itself and as effect the ability to drive a car. The ability to drive a car is marked in the knowledgebase by an instance of the relation `canDrive` which is either generated by the invocation of the driving license issuing service having such an effect or enforced by an axiom in the form of `whoCanDrive` axiom (in the cases when the driving license has not been issued by invoking such a service).

**Listing 5.** Mappings between `person` and `human` resolving the driving licenses conflicts

```

1  Mapping(o1o2o1#4
2    classMapping(two-way o1#person o2#human))
3
4  Mapping(o1o2o1#5
5    attributeMapping(two-way o1#person.hasDrivingLicense
6                      o2#human.drivingLicense))
7
8  Mapping(o1o2o1#6
9    classMapping(two-way o1#IrishDrivingLicense
10                   o2#GermanDrivingLicense)
11    attributeValueCondition(o1#IrishDrivingLicense.type
12                             = permanent))
13
14 Mapping(o1o2o1#7
15   attributeMapping(two-way o1#IrishDrivingLicense.ownerName
16                     o2#GermanDrivingLicense.ownerName))

```

In such a situation a citizen that obtains a temporary driving license in Ireland will have the right to drive in Ireland but he/she is not entitled to drive in Germany. Such a conflict can be resolved by creating mappings that state that a German driving license corresponds only to the permanent Irish driving license and other way around. Listing 5 depicts such mappings.

Lines 10-11 show a condition specifying that an `IrishDrivingLicense` is mapped to a `GermanDrivingLicense` only if its type is `permanent`. Hence if the above condition does not hold, for any human instance in  $O_2$  that has to be created (based on instance of `person` from  $O_1$ ) there is no driving license associated with it. As such, even if an individual obtains a temporary driving license in Ireland it will not be transferred to the German model and he/she will not be allowed to drive in Germany (according to the `whoCanDrive` axiom).

### 3.2.2 Data representation conflicts

Data representation conflicts occur when two semantically equivalent data are represented in a different way (that excludes unit conversion issues). For example a date in the extended format in the Italian PA domain can be represented by the following string: “4 Ottobre 1980”, while in the short format in any European PA domain it would be 1980-10-04 (according to the standard EN 28601). This kind of conversions requires either the use of specialized data manipulation functions within the reasoning environment or the access to external services that can apply the conversion after the reasoning occurred.

Since the data mediator relies on a generic abstract mapping language that is not bounded to any specific reasoning and mediation runtime, the second solution is preferable. In fact, even specific reasoners that could be used after the abstract mapping language is grounded to a concrete language normally support only a limited number of data manipulation functions. The use of external services for the data manipulation after the actual reasoning task offer a higher flexibility and it is not depending of the reasoning infrastructure.

The generic support for specifying the call to a service within the mapping language is obtained by associating to the target value to be computed a service call, passing the relative parameters and setting the output.

This is achieved by allowing the attribute for which the data value needs to be computed to take (besides the correct expected values) also an instance of the `serviceConfiguration` concept, a class that models calls to external web services that resolve the data representation conflicts (lines 1-4 in Listing 6).

**Listing 6.** Model of the `serviceConfiguration` concept

---

```

1  concept serviceConfiguration
2      uri ofType (1 1) _string
3      inputParameter ofType (0 *) {serviceParameter, serviceConfiguration}
4      outputParameter ofType (1 1) serviceParameter
5
6  concept serviceParameter
7      path ofType (1 1) _string
8      value ofType (0 1) _string
9      type ofType (0 1) DataType
10
11 concept DataType
12 instance date memberOf DataType
13 instance string memberOf DataType
14 ...
15 axiom outputParameterUsage
16     definedBy
17         !- ?x[outputParameter hasValue ?y] memberOf serviceConfiguration
18         and ?y[value hasValue ?z] memberOf serviceParameter.

```

---

Listing 6 presents the model of the `serviceConfiguration` concept (lines 1-4) and its related aspects (lines 6-14). These aspects contain the lower level technical details needed to actual

perform the invocation of the service. A `serviceConfiguration` specifies that the external service called for the data manipulation has a certain URI (that refers to its WSDL, where its endpoint is specified), a set of input parameters and an output parameter. An input parameter (line 3) can be either a `serviceParameter` or a `serviceConfiguration`. Each `serviceParameter` (lines 6-9) is identified by the `path` attribute that contains the Xpath [22] expression that specifies where the parameter is located within the SOAP [26] message created to invoke the service or returned by the service. The actual value to be used during the invocation is specified by the `value` attribute. This attribute is not used for `outputParameter` (it is forbidden by the axiom from lines 15-18 in Listing 6) since the output values are not known at this stage. The data type of the parameter is specified by the `type` attribute. This attribute is optional since if the target attribute is bound to only one data type we already have this information. Otherwise, if the target attribute may assume more than one data type, we need to specify the `type` attribute of the `serviceParameter` instance.

When the input parameter is an instance of `serviceConfiguration` then it has to be computed invoking another service and this leads to the creation of a chain of invocation that finally returns the final output parameter. Multiple references to the same `serviceConfiguration` instance are allowed within an invocation chain if its termination is guaranteed, i.e. there are no loops (this can be enforced with an axiom).

This solution assumes that we may have zero or more input parameters for the external service and one and only one output. Instances of the correct `serviceConfiguration` can be created at design time within the mapping environment by connecting the attribute of the source and target ontologies that presents data representation conflicts, selecting the proper service for the data conversion and configuring it (some details may be preconfigured e.g. the uri and xpaths).

### 3.2.2.1 Differences in evidence domain caused by multiple representation of similar evidence

This kind of differences can be easily resolved with the before mentioned technique. For example, we can consider the case where in the Italian Identity Card the birthday date for a person is specified in the extended format (e.g., “4 Ottobre 1980”) and this evidence is needed by another European PA where the standard short date format is used (e.g., “04/10/1980”). Fragment of the two concepts for the two different Member States are depicted in Listing 7.

**Listing 7.** Definition of the Italian and the Belgian Identity Card in the ontology  $O_1$  and  $O_2$

Ontology $O_1$	Ontology $O_2$
<pre>concept IdentityCard   name ofType (1 1) _string   ...   birthdate ofType (1 1) _string</pre>	<pre>concept IdentityCard   name ofType (1 1) _string   ...   birthdate ofType (1 1) _date</pre>

Listing 8 reports an example rule that specifies the mapping between the Italian (ontology  $O_1$ ) and Belgian domain (ontology  $O_2$ ).

**Listing 8.** Mapping between the source ontology  $O_1$  and in the target ontology  $O_2$  (lines 1-9) and the involved service configuration (lines 11-19)

```

1 Mapping(itbe#1
2   classMapping(one-way it#IdentityCard be#IdentityCard))
3 Mapping(itbe#2
4   attributeMapping(two-way it#IdentityCard.name be#IdentityCard.name))
5 Mapping(itbe#2
6   attributeMapping(one-way
7     it#IdentityCard.birthdate be#IdentityCard.birthdate)
8   attributeValueCondition(be#IdentityCard.birthdate =
9     serviceDataConversion1))
10
```

```

11 instance serviceDataConversion1 memberOf serviceConfiguration
12   uri hasValue "http://www.semanticgov.org/services/dateconversion.wsdl"
13   inputParameter hasValue {serviceInputParameter1}
14   outputParameter hasValue serviceOutputParameter1
15 instance serviceInputParameter1 memberOf serviceParameter
16   path hasValue "mm:conversionRequest/mm:inputDate"
17   value hasValue "4 Ottobre 1980"
18 instance serviceOutputParameter1 memberOf serviceParameter
19   path hasValue "mm:conversionResponse/mm:outputDate"

```

An example of instance from the original domain is reported in Listing 9.

**Listing 9.** Example of fragment from the source ontology  $O_1$

```

1 instance AnnaIdentityCard memberOf it#IdentityCard
2   name hasValue "Anna"
3   ...
4   birthdate hasValue "4 Ottobre 1980"

```

The resolution of this conflict takes place in two steps: first the data mediation run-time engine is used as usually to perform the transformation of the source instances in target instances. The only difference appears in the values of the attributes that have to be computed by external services calls (see line 6 in Listing 10).

**Listing 10.** Output of the first step of the data mediation process for the input in Listing 9

```

1 concept serviceConfiguration subConceptOf _string
2
3 instance AnnaIdentityCard memberOf be#IdentityCard
4   name hasValue "Anna"
5   ...
6   birthdate hasValue serviceDataConversion1

```

During the data mediation an instance of `serviceConfiguration` will be assigned as value to the attribute `birthdate`. According to the original model, the attribute `birthdate` expects a string as value, so in order to avoid constraints violation<sup>3</sup> one more definition has to be added to the ontology during the rules evaluation (Line 1 in Listing 10)<sup>4</sup>. From this point on any instance of the `serviceConfiguration` can play the role of a string as well without triggering any integrity constraint violations.

Finally, the data mediation runtime checks the reasoning output to see if any remote service invocation is needed to complete the mediation from the source ontology to the target ontology. In this case it will invoke the remote service as specified by the instances of the `serviceConfiguration` concept (in Listing 8, lines 11-19). The out parameter specified will be then used to populate the attribute that previously contained the `serviceConfiguration` instance (see Listing 10).

**Listing 11.** Final result of the data mediation process

```

1 instance AnnaIdentityCard memberOf be#IdentityCard
2   name hasValue "Anna"
3   ...
4   birthdate hasValue _date("1980-10-04")

```

<sup>3</sup> Such constraining behaviour can be implemented in the rule-based reasoners and it is similar with the constraints system in the data bases. The reasoner used by the Data Mediator in this work has such constraints mechanisms in place.

<sup>4</sup> Even if at the first look it might seem awkward to subclass a data type, WSMML supports this feature giving a greater flexibility of the overall logical framework.

### 3.2.2.2 Differences in preconditions domains caused by multiple representations of similar preconditions

In our approach the requester needs are formalized in the forms of a Goal. Each goal is defined by a set of elements such as the requested capability (which in its turn contains preconditions, postconditions, axiom and effects - for more details please see WSMO specifications) or the requested interfaces. This type of conflict appears when the goal's preconditions are expressed according to the domain model of the PA local to the requester but they have to be checked against the Web service descriptions defined in terms of a different PA domain model. In the SemanticGov architecture, this issue is solved by the discovery that converts an evidence from the representation of the original domain to the representation of the final domain so as to solve the data representation conflicts between the different preconditions by actually converting the input to the preconditions in the expected data format (of course this applies also to postconditions and effects).

### 3.2.3 Data unit conflicts

Data unit conflicts occur when two different member states adopt different unit of measures. From a technical point of view this conflict is solved exactly as the data representation conflicts (see Section 3.2.2). Suppose that we have to provide the current amount of an Greek bank account to a service of the UK PA (this may be required for example to ask for a residence permit, that in some country is issued according to the proof of economical capability to maintain yourself). While in Greece the currency adopted is Euro, in the UK the currency adopted is Pound, hence a data unit conflicts occurs and it has to be resolved by invoking the proper mediation. For example a possible representation of the `MoneyAmount` concept in the two domains can in the first case have the `currency` attribute constrained by an axiom to be `Euro` and in the second case to be `Pound`.

### 3.2.4 Data precision conflicts

Data precisions conflicts arise often among different Members States evidences because of the different legislations. For example the Belgian law identifies 13 different types of marital statuses, so its registry service provides information according to these types, while the Italian legislation specifies only 4 types for the marital status. Hence, these conflicts have to be resolved when an Italian PA needs to access information about the marital status of a person published by the Belgian registry.

From a technical point of view this issue is solved by creating at design time the mapping rules from the 13 Belgian types to the 4 Italian ones (*m:n mapping*). The created rules are then used by the runtime mediation component to convert any incoming instance of Belgian marital status to the correspondent Italian marital status.

**Listing 12.** Description of the Belgian and Italian `MaritalStatus` in ontologies  $O_1$  and  $O_2$

Ontology $O_1$	Ontology $O_2$
<pre> <b>concept</b> MaritalStatus   hasMaritalStatusCode <b>ofType</b>     MaritalStatusCode   hasMaritalStatusDescription <b>ofType</b>     MaritalStatusDescription  <b>concept</b> MaritalStatusCode  <b>instance</b> _10 <b>memberOf</b> MaritalStatusCode <b>instance</b> _20 <b>memberOf</b> MaritalStatusCode <b>instance</b> _25 <b>memberOf</b> MaritalStatusCode <b>instance</b> _26 <b>memberOf</b> MaritalStatusCode <b>instance</b> _30 <b>memberOf</b> MaritalStatusCode ... </pre>	<pre> <b>concept</b> MaritalStatus   hasMaritalStatusCode <b>ofType</b>     MaritalStatusCode   hasMaritalStatusDescription <b>ofType</b>     MaritalStatusDescription  <b>concept</b> MaritalStatusCode  <b>instance</b> _1 <b>memberOf</b> MaritalStatusCode <b>instance</b> _2 <b>memberOf</b> MaritalStatusCode <b>instance</b> _3 <b>memberOf</b> MaritalStatusCode <b>instance</b> _4 <b>memberOf</b> MaritalStatusCode </pre>

<b>instance</b> _60 <b>memberOf</b> MaritalStatusCode	
<b>instance</b> _80 <b>memberOf</b> MaritalStatusCode	
<b>instance</b> _81 <b>memberOf</b> MaritalStatusCode	
<b>instance</b> _90 <b>memberOf</b> MaritalStatusCode	

The mapping between the two fragment of ontologies reported in Listing 12 is presented in Listing 13.

**Listing 13.** Mapping between the  $O_1$  and  $O_2$  ontology

```

1  Mapping(beit#11
2      classMapping(two-way be#MaritalStatus it#MaritalStatus))
3  Mapping(beit#12
4      attributeMapping(two-way be#MaritalStatus.hasMaritalStatusCode
5                          it#MaritalStatus.hasMaritalStatusCode)
6      attributeValueCondition(be#MaritalStatus.hasMaritalStatusCode = _10)
7      attributeValueCondition(it#MaritalStatus.hasMaritalStatusCode = _1))
8  Mapping(beit#13
9      attributeMapping(two-way be#MaritalStatus.hasMaritalStatusCode
10                         it#MaritalStatus.hasMaritalStatusCode)
11     attributeValueCondition(be#MaritalStatus.hasMaritalStatusCode = _20)
12     attributeValueCondition(it#MaritalStatus.hasMaritalStatusCode = _2))
13 Mapping(beit#14
14     attributeMapping(two-way be#MaritalStatus.hasMaritalStatusCode
15                         it#MaritalStatus.hasMaritalStatusCode)
16     attributeValueCondition(be#MaritalStatus.hasMaritalStatusCode = _30)
17     attributeValueCondition(it#MaritalStatus.hasMaritalStatusCode = _3))
18 Mapping(beit#15
19     attributeMapping(two-way be#MaritalStatus.hasMaritalStatusCode
20                         it#MaritalStatus.hasMaritalStatusCode)
21     attributeValueCondition(be#MaritalStatus.hasMaritalStatusCode = _40)
22     attributeValueCondition(it#MaritalStatus.hasMaritalStatusCode = _4))
23 Mapping(beit#16
24     attributeMapping(one-way be#MaritalStatus.hasMaritalStatusCode
25                          it#MaritalStatus.hasMaritalStatusCode)
26     attributeValueCondition(be#MaritalStatus.hasMaritalStatusCode = _41)
27     attributeValueCondition(it#MaritalStatus.hasMaritalStatusCode = _4))

```

Notice that the mapping between instance `_40` and `_4` is bidirectional while the mapping between `_41` and `_4` is only from the Belgian domain to the Italian one, this imply that going from the Italian domain to the Belgian on `_4` will be always transformed to `_40`. Listing 14 shows an example of a mediation result between the two domains.

**Listing 14.** Example of mediation between the two ontologies<sup>5</sup>

Ontology $O_1$	Ontology $O_2$
<b>instance</b> MaritalStatus <b>memberOf</b> be#MaritalStatus hasMaritalStatusCode <b>hasValue</b> _20 hasMaritalStatusDescription <b>hasValue</b> Marie	<b>instance</b> MaritalStatus <b>memberOf</b> it#MaritalStatus hasMaritalStatusCode <b>hasValue</b> _2 hasMaritalStatusDescription <b>hasValue</b> Coniugata

### 3.3 Schema-level conflicts

Schema-level conflicts in this case are semantic conflicts which are characterized by differences in logical structures or inconsistencies in metadata of the PA domain (see Figure 3).

<sup>5</sup> We are assuming that the person associated with the marital status is a woman, and hence in Italian her marital status description is *Coniugata*. The rule for mapping marital status description is not showed since it is nor relevant to the example.

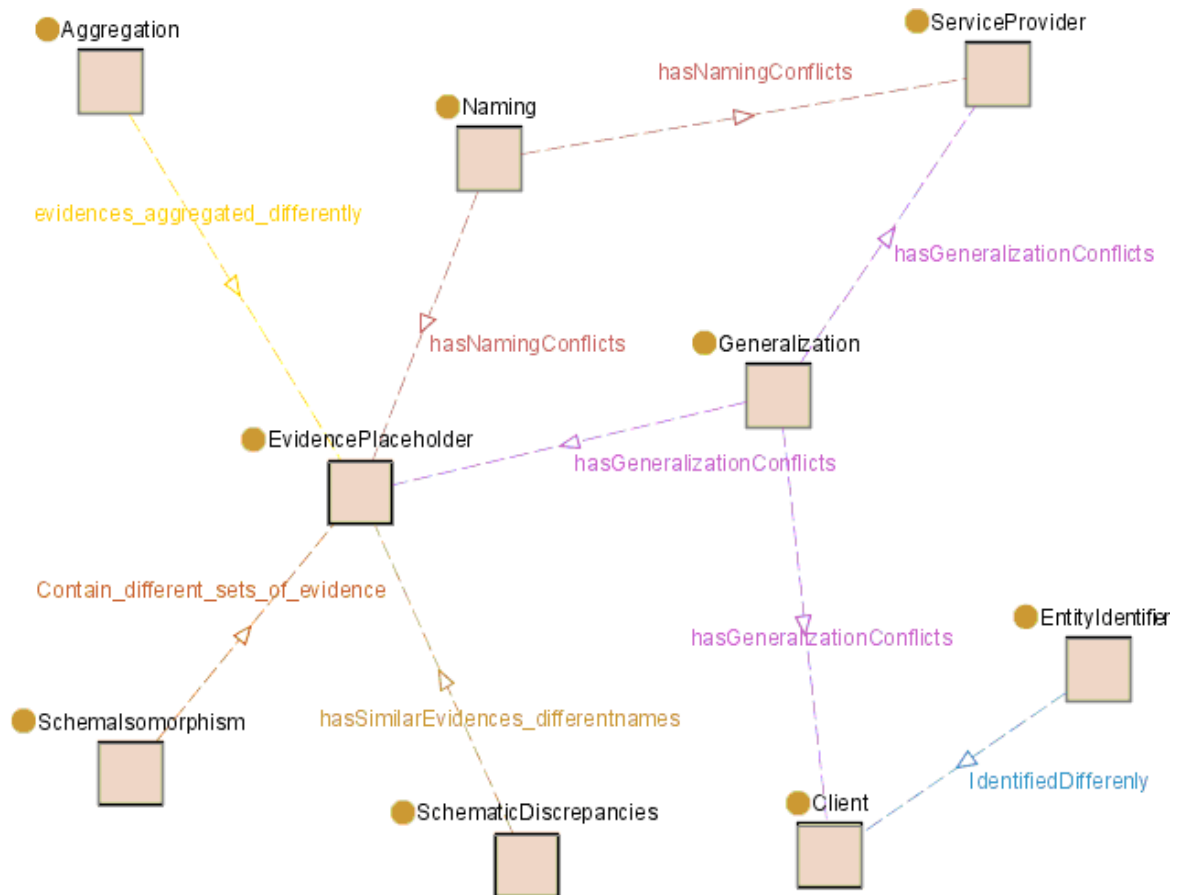


Figure 3. Schema Level Conflicts

### 3.3.1 Naming conflicts

Naming conflicts arise when similar concepts are labeled in a different way or different concepts are labeled in a similar way (i.e. terms that in different languages, and hence in different Member States have similar spelling but different meaning).

#### 3.3.1.1 Naming Conflicts in Service Providers

In two different MSs we could have two service providers that may be called differently but in fact are competent for the same services, or the opposite, which means that we could have two service providers that have different jurisdiction and are competent for the provision of different sets of services despite the fact that they have the same name.

This issue is hidden by the discovery component, which avoids that an incorrect naming for a service or its provider may lead to an incorrect discovery or service invocation by matching goal to services according to preconditions and postconditions. Naming conflicts at level of conditions on evidences are properly solved by mapping rules between the ontologies representing the different member state domains.

#### 3.3.1.2 Naming Conflicts in Evidence Placeholders

In different MSs, evidence placeholders with the same name but different purpose and usage may exist or evidence placeholders with different names may have similar usage and hold similar evidences.

This issue is solved by creating the correct mapping between concepts representing evidence placeholders in the source and in the target ontology. It is important to clarify that two concepts or instances having the same label and belonging to two different ontologies cannot raise any conflict unless incorrectly mapped. Actually, due to the fact that concepts are not only identified by their labels but also by the namespace of the ontology, they cannot be interpreted to be “similar” by reasoner unless differently stated by a mapping rule.

Listing 15 and Listing 16 better clarify the process of naming expansion that occurs within reasoners.

**Listing 15.** Description of the `Person` concept in two ontologies without the namespace expansion

Ontology O <sub>1</sub>	Ontology O <sub>2</sub>
<b>Namespace</b> { "http://semantic-gov.org/o1#" }	<b>namespace</b> { "http://semantic-gov.org /o2#" }
<b>ontology</b> Ontology1	<b>ontology</b> Ontology2
<b>concept</b> Person hasName <b>ofType</b> (1 1) _string hasSurname <b>ofType</b> (1 1) _string	<b>concept</b> Person hasName <b>ofType</b> (1 1) _string hasSurname <b>ofType</b> (1 1) _string

**Listing 16.** Description of the `Person` concept in two ontologies with the namespace expansion

Ontology O <sub>1</sub>	Ontology O <sub>2</sub>
<b>Namespace</b> { "http://semantic-gov.org/o1#" }	<b>Namespace</b> { "http://semantic-gov.org /o2#" }
<b>Ontology</b> http://semantic-gov.org/o1#Ontology1	<b>Ontology</b> http://semantic-gov.org/o2#Ontology2
<b>concept</b> http://semantic-gov.org/o1#Person http://semantic-gov.org/o1#hasName <b>ofType</b> _string http://semantic-gov.org/o1#hasSurname <b>ofType</b> _string	<b>concept</b> http://semantic-gov.org/o2#Person http://semantic-gov.org/o2#hasName <b>ofType</b> _string http://semantic-gov.org/o2#hasSurname <b>ofType</b> _string

Initially any two concepts (even having the same labels) are considered to be different concepts by the reasoning unless an appropriate mapping rule is created. The same discussion applies to concepts with different labels that describe similar concepts. For example consider the case of the Birth Certificate in UK and Italy (Listing 17).

**Listing 17.** Description of the `BirthCertificate` concept in ontology O<sub>1</sub> (English domain) and ontology O<sub>2</sub> (Italian domain) without the namespace expansion

Ontology O <sub>1</sub>	Ontology O <sub>2</sub>
<b>namespace</b> { "http://semantic-gov.org/o1#" }	<b>namespace</b> { "http://semantic-gov.org /o2#" }
<b>ontology</b> Ontology1	<b>ontology</b> Ontology2
<b>importsOntology</b> { "_http://www.semantic-gov.org/GEA#GEA" }	<b>importsOntology</b> { "_http://www.semantic-gov.org/GEA#GEA" }
<b>concept</b> BirthCertificate <b>subConceptOf</b> gea#Evidence_Placeholder hasName <b>ofType</b> (1 1) _string hasSurname <b>ofType</b> (1 1) _string	<b>concept</b> CertificatoDiNascita <b>subConceptOf</b> gea#Evidence_Placeholder hasName <b>ofType</b> (1 1) _string hasSurname <b>ofType</b> (1 1) _string

### 3.3.2 Entity identifier conflict

The entity identifier conflicts arise when in two different member states similar entities are identified by two different identifiers. This kind of conflicts cannot always be solved by simple data mediation or by generating random identifiers due to the fact that such identifier have to be generated by a legal authority (notice that in this context we are not talking about identifiers as for examples primary keys in a database table).

In many European countries people are identified with a different mechanism, and this can raise many entity identifier conflicts when a person is moving from is original country to a new one. Let us consider the Italian Fiscal Code, which is the univocal identifier of a person according to the Italian law. This identifier is created automatically at the birth of every Italian citizen and it is used for various purposes (e.g. paying taxes). Hence, to correctly access a tax paying service in Italy everyone (also foreigners working in Italy) has to provide as evidence her/his Fiscal Code. Foreign people that do not have a Fiscal Code have to apply for it before paying these taxes.

These conflicts maybe solved by using a service that actually creates a real and valid instance of the missing identifier (such a service would be most probably offered by the Italian Finance Ministry). In our context this issue can be solved by exploiting composition capabilities of the SemanticGov architecture and creating a composed service specific for foreigners that invoke a specific service to obtain a Fiscal Code and, once obtained, use it to invoke the tax payment service. The discovery matches the right service to the foreign citizen based on whether he/she has provided as input her/his Fiscal Code or not.

Simpler cases can be solved in creating rules that specifies that two entities are the same if certain conditions holds.

### 3.3.3 Schema-isomorphism conflicts

These conflicts are present when different attributes are used to describe the semantically equivalent concepts. Such an example can be seen in Listing 2 and in Listing 4, but in general this kind of conflict can take multiple forms. For example consider the ID card in Greece (ontology  $O_1$ ) and Belgium (ontology  $O_2$ ), presented in Listing 18: the information captured by the concept `IDCard2` in ontology  $O_2$  is richer than the corresponding concept `IDCard1` in ontology  $O_1$  where `hasAddress` and `hasNationality` are not present.

**Listing 18.** Description of the `IDCard` in two different ontologies

Ontology $O_1$	Ontology $O_2$
<pre> <b>Namespace</b>   {"http://semantic-gov.org/o1#"}  <b>ontology</b> Ontology1  <b>importsOntology</b>   {"http://www.semantic-gov.org/GEA#GEA"}  <b>concept</b> IDcard1 <b>subConceptOf</b>   gea#Evidence_Placeholder   hasName <b>ofType</b> (1 1) _string   hasSurname <b>ofType</b> (1 1) _string   hasFatherName <b>ofType</b> (1 1) _string   hasMotherName <b>ofType</b> (1 1) _string   hasBirthDate <b>ofType</b> (1 1) date </pre>	<pre> <b>namespace</b>   {"http://semantic-gov.org /o2#"}  <b>ontology</b> Ontology2  <b>importsOntology</b>   {"http://www.semantic-gov.org/GEA#GEA"}  <b>concept</b> IDCard2 <b>subConceptOf</b>   gea#Evidence_Placeholder   hasName <b>ofType</b> (1 1) _string   hasSurname <b>ofType</b> (1 1) _string   hasFatherName <b>ofType</b> (1 1) _string   hasMotherName <b>ofType</b> (1 1) _string   hasBirthDate <b>ofType</b> (1 1) date   hasAddress <b>ofType</b> (1 1) _string   hasNationality <b>ofType</b> (1 1) _string </pre>

The most important aspect regarding this conflict is if the information that is captured by the source model is complete in respect with the target model and other way around. As a

consequence we can distinguish two cases when interoperability has to be achieved while sending information from the source to the target:

- a. **The source ontology element covers more information than the target ontology element.** This means that there is information in the source ontology that cannot be “understood” (and by this is useless) by the target party. As such interoperability can be achieved by simply discarding the extra information, simply by not linking this extra information by mappings with any elements from the target.
- b. **The source ontology element covers less information than the target ontology element.** This is one of the interoperability conflicts that hamper the automation degree of the mediation solutions. Because the source ontology does not capture the extra information required by the target (and by this we can say that the source does not “understand” this information), the source party does not have any means to produce that information. In order to resolve this type of conflict there are two main approaches:
  1. **Default values generation.** The middle layer, the mediator, assumes the responsibility to fill the empty slots with default values. In some situation this task can be performed by the target as well: rules are present in the target ontology and when for example values are not provided for some attributes they populate them with default values. The advantage of this technique is that it can be automatically applied. The disadvantage is that this method fails when the reason why this information is missing from the source is not because it is meaningless in the source domain but because the model is incomplete or faulty. In this situation a different value and not the default one should be used.
  2. **Additional interactions with the requester.** This method involves the participation of the requester (in the PA domain the client) in providing the missing information. Such interaction can be carried on at the portal level and the client can directly fill the required information as specified in the target model. The disadvantage is that the client needs to understand the information that is required from its side – the portal cannot offer such support since its (source) model is less expressive than the one of the target (this is where the conflict originates). The advantage is that the information provided in this way is guaranteed to be correct.

### 3.3.4 Generalization conflicts

Generalization conflicts occur when two similar concepts are defined using a different model in two different member states.

#### 3.3.4.1 Generalization conflicts in evidence placeholders

For example two different member states may model the information for a birth certificate placeholder with different aggregation: that is, in the first member state the birth certificate is only one document while in the second member state the same evidences are captured by more than one document. Listing 19 depicts how the information contained by the `BirthCertificate` in  $O_1$  is captured by the `BirthCertificate` and the `FamilyCertificate` in  $O_2$ .

Listing 20 shows another example, the definition of genders in  $O_1$  by using a special attribute of the `person` concept and populating it with one of the two allowed values, `male` and `female`, while in ontology  $O_2$ , `man` and `woman` are subclasses of the `human` concept.

**Listing 19.** Generalization conflict in defining the BirthCertificate in one MS and the FamilyCertificate in another MS

Ontology O <sub>1</sub>	Ontology O <sub>2</sub>
<pre> Namespace   {"http://semantic-gov.org/o1#"}  ontology Ontology1  importsOntology   {"http://www.semantic-   gov.org/GEA#GEA"}  concept BelgienBirthCertificate   subConceptOf gea#Evidence_Placeholder   hasName ofType (1 1) _string   hasSurname ofType (1 1) _string   hasFatherName ofType (1 1) _string   hasMotherName ofType (1 1) _string   hasBirthDate ofType (1 1) date   hasBirthPlace ofType (1 1) string </pre>	<pre> namespace   {"http://semantic-gov.org /o2#"}  ontology Ontology2  importsOntology   {"http://www.semantic-gov.org/GEA#GEA"}  concept BirthCertificate   subConceptOf gea#Evidence_Placeholder   hasName ofType (1 1) _string   hasSurname ofType (1 1) _string   hasBirthDate ofType (1 1) date   hasBirthPlace ofType (1 1) string  concept FamilyCertificate   subConceptOf gea#Evidence_Placeholder   hasName ofType (1 1) _string   hasSurname ofType (1 1) _string   hasFatherName ofType (1 1) _string   hasMotherName ofType (1 1) _string </pre>

**Listing 20.** Generalization conflict in defining the male and female

Ontology O <sub>1</sub>	Ontology O <sub>2</sub>
<pre> concept person   name ofType _string   age ofType _integer   hasGender ofType gender   hasChild ofType person  concept gender   value ofType _string  instance male memberOf gender   value hasValue "male"  instance female memberOf gender   value hasValue "female"  concept birthCertificate   owner ofType person   birthDate ofType _date   placeOfBirth ofType _string   hasMother ofType person   hasFather ofType person </pre>	<pre> concept human   name ofType _string   age ofType _integer  concept man subConceptOf human concept woman subConceptOf human concept birthCertificate   owner ofType person   birthDate ofType _date   placeOfBirth ofType _string  concept familyCertificate   owner ofType human   hasMother ofType woman   hasFather ofType man </pre>

Listing 21 shows the mappings that will assure the right correspondences between the O<sub>1</sub> and O<sub>2</sub> ontologies.

**Listing 21.** Mappings resolving the generalization conflicts in O<sub>1</sub> and O<sub>2</sub>

1	<b>Mapping</b> (o1o2o1#1
2	<b>classMapping</b> (two-way o1#person o2#human))
3	
4	<b>Mapping</b> (o1o2o1#2
5	<b>attributeMapping</b> (two-way o1#person.name o2#human.name))
6	
7	<b>Mapping</b> (o1o2o1#3
8	<b>attributeMapping</b> (two-way o1#person.age o2#human.age))
9	
10	<b>Mapping</b> (o1o2o1#7
11	<b>classMapping</b> (two-way o1#person o2#man)
12	<b>attributeValueCondition</b> (o1#person.hasGender = male))
13	
14	<b>Mapping</b> (o1o2o1#7
15	<b>classMapping</b> (two-way o1#person o2#woman)
16	<b>attributeValueCondition</b> (o1#person.hasGender = female))
17	

```

18 Mapping(o1o2o1#8
19   classMapping(two-way o1#birthCertificate o2#birthCertificate))
20
21 Mapping(o1o2o1#9
22   attributeMapping(two-way o1#birthCertificate.owner
23                     o2#birthCertificate.owner))
24
25 Mapping(o1o2o1#10
26   attributeMapping(two-way o1#birthCertificate.birthDate
27                     o2#birthCertificate.birthDate))
28
29 Mapping(o1o2o1#11
30   attributeMapping(two-way o1#birthCertificate.placeOfBirth
31                     o2#birthCertificate.placeOfBirth))
32
33 Mapping(o1o2o1#12
34   classMapping(two-way o1#birthCertificate o2#familyCertificate))
35
36 Mapping(o1o2o1#13
37   attributeMapping(two-way o1#birthCertificate.hasMother
38                     o2#familyCertificate.hasMother))
39
40 Mapping(o1o2o1#14
41   attributeMapping(two-way o1#birthCertificate.hasFather
42                     o2#familyCertificate.hasFather))

```

### 3.3.4.2 Generalization conflicts in service providers

The description details of service providers are generally considered and modeled as non-functional properties (i.e. exactly who provides the service does not directly influences its capability). The Discovery engine is responsible to analyze not only the non-functional properties of the services (e.g. provider) but also the functional descriptions of the services in order to decide which are the suitable services for a given goal.

### 3.3.4.3 Generalization conflicts in clients

These conflicts may rise when classifications for the client exist in on MS but not in another. For example, enterprises may be grouped in different categories and sub-categories in different MSs. Listing 20 depicts an example of this conflict: in  $O_1$  we have one single concept defining a person while in  $O_2$  the human has to sub-classes, man and woman. In Listing 21 the mappings from lines 1 to 16 (together with the inheritance principles) show how this conflict can be resolved.

### 3.3.5 Aggregation conflicts

Aggregation conflicts can be of two types: (i) the same semantic concepts are aggregated in different hierarchical structures or (ii) a single data value of the source entity corresponds to more data values of target entity. The first class of aggregation conflicts can be solved creating the proper mapping rules so as to convert the source hierarchical structure to the target one. Listing 22 depicts an example.

**Listing 22.** Description of the Person concept in two ontologies with a different aggregation level

Ontology $O_1$	Ontology $O_2$
<pre> concept Person   hasName ofType (1 1) _string   hasSurname ofType (1 1) _string   hasTelephone ofType (0 *) _string   hasEmail ofType (0 *) _string </pre>	<pre> concept Person   hasName ofType (1 1) _string   hasSurname ofType (1 1) _string   hasContacts     ofType (1 1) contactDetails concept contactDetails   telephone ofType (0 *) _string   email ofType (0 *) _string </pre>

The mapping definitions reported in Listing 23 exemplify how to solve the aforementioned problem.

**Listing 23.** Mapping between the ontology  $O_1$  and  $O_2$

```

1  Mapping(o1o2#1
2      classMapping(two-way o1#Person o2#Person))
3  Mapping(o1o2#2
4      attributeMapping(two-way
5                      o1#Person.hasName o2#Person.hasName))
6  Mapping(o1o2#3
7      attributeMapping(two-way
8                      o1#Person.hasSurname o2#Person.hasSurname))
9  Mapping(o1o2#4
10     classMapping(two-way
11                 o1#Person o2#contactDetails))
12 Mapping(o1o2#5
13     classAttributeMapping(one-way
14                          o1#Person o2#Person.hasContacts))
15 Mapping(o1o2#6
16     attributeMapping(two-way
17                     o1#Person.hasTelephone o2#contactDetails.hasTelephone))
18 Mapping(o1o2#7
19     attributeMapping(two-way
20                     o1#Person.hasTelephone o2#contactDetails.hasTelephone))

```

The semantics of the mapping from lines 9 to 11 is that there is information in the concept `Person` from  $O_1$  that is related with the information from the `contactDetails` concept from  $O_2$ . The mapping from lines 13 to 14 states that the information from `o1#Person` that is related with the information from `o2#contactDetails` is referred from the upper level (from `o2#Person`) by using the attribute `hasContacts` from `o2#Person`.

The second class of conflicts requires data manipulation like in the case of data representation conflicts and data unit conflicts. For example, if in a MS a evidence placeholder contains one field for the name and one for the surname, and in another MS the same evidence placeholders contains only the full name as a unique field comprising name and surname (see Listing 24), the mediation between MSs requires the use of service able to concatenate strings (in direction ontology  $O_2$  to ontology  $O_1$ ) and a service able to split strings (in direction ontology  $O_2$  to ontology  $O_1$ ).

**Listing 24.** Description of the `BirthCertificate` concept in two domains with data values aggregated in a different way

Ontology $O_1$	Ontology $O_2$
<pre> concept BirthCertificate   hasName ofType (1 1) _string   hasSurname ofType (1 1) _string </pre>	<pre> concept BirthCertificate   hasFullName ofType (1 1) _string </pre>

### 3.3.6 Schematic discrepancies

Schematic discrepancies can occur when a set of evidences and their values belonging to an evidence placeholder in one MS are organized to form a different evidence placeholder structure in another MS. This is quite similar to the situation presented above as “naming conflict in evidence placeholders”.

## 4 Mediation services and solutions

This section describes the actual technological components/modules/design decisions that play a role in solving the semantic interoperability conflicts described in the previous section. It provides a short overview of the architecture adopted in the SemanticGov project and more details on the Data Mediator, with emphasis both on the design-time support and on the run-time support.

### 4.1 The Semantic Gateway Architecture

In this section we give a short description of the global view on the SemanticGov architecture and with particular attention to the Semantic Gateway (for more details please refer to D3.1 [15]). The architecture is composed by different layers: the Service Requestor layer, the Front Office Layer, the Middleware Layer and the Service Provider Layer.

**Service Requestors Layer** forms the stakeholders of the SemanticGov architecture. The top level group of stakeholders is identified as (1) *citizens*, (2) *businesses*, and (3) *public servants*.

**Front-office Layer** provides the access to the system for SemanticGov stakeholders. We identify the access provided for citizens and businesses through member state portals (e.g. portal of the public administration of the Czech Republic) and access provided through management tools for public servants (domain experts and administrators) e.g. ontology management tools, monitoring tools etc.

**Middleware Layer** is the core of the SemanticGov architecture. Enhanced with semantic capabilities, it provides the main intelligence to integration and interoperation of services provided by the PAs. The Middleware Layer is composed of two major building blocks: (1) *Member State Middleware*, and (2) *Semantic Gateway*. Member State Middleware facilitates the integration of e-government services provided by various PA within the state as well as horizontal integration with the Semantic Gateway. Semantic Gateway facilitates the integration of member states at the EU level, i.e. integration and interoperation of cross-border PA services (Pan-European e-Government Services). The integration process within both building blocks is maintained through a number of middleware services including operation, discovery, composition, interoperability, registry/repository, etc.

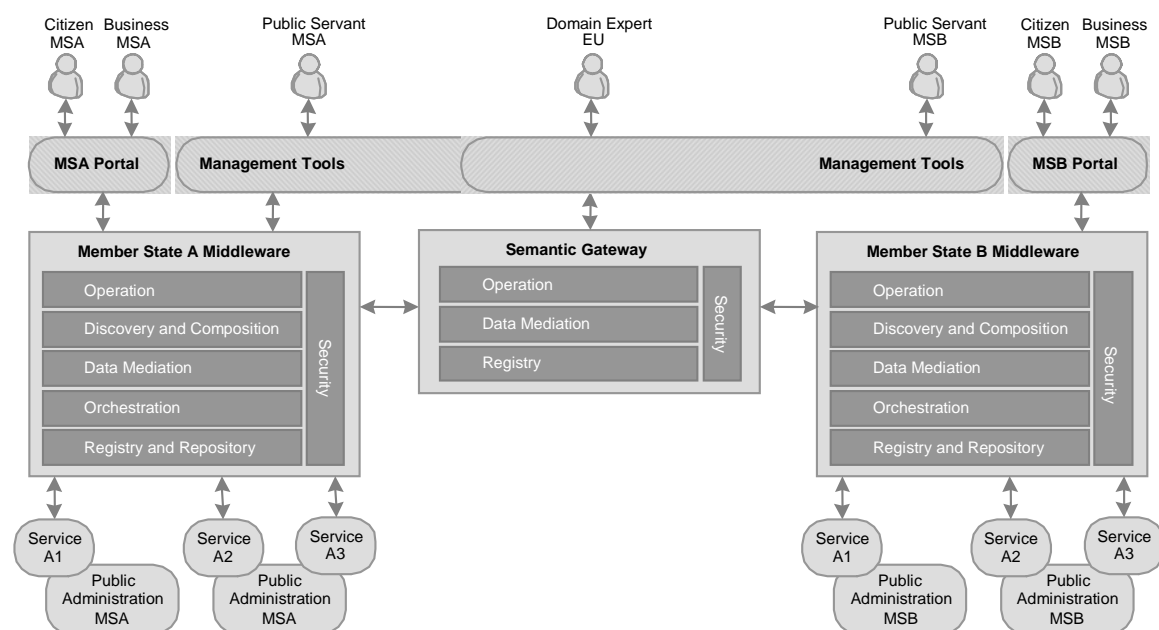


Figure 4. The Semantic Gov architecture

**Service Providers Layer** contains various PAs in a member state which exposes various PA services through their existing back-end applications.

The middleware layer represents the Semantic Gateway which allows for interoperability conflicts resolution by means of interaction of its different components. The middleware adopted is based on WSMX [16]. The core element for the resolution of interoperability conflicts is the data mediator, which is described in the detail in the next paragraph.

## 4.2 Data Mediator

In SemanticGov project, the solution for data mediation is based on ontologies that semantically describe the data to be mediated and on mapping rules that describe how to mediate between different ontologies. This implies a distinction between the design-time and the run-time phase, where the first represents the support (a prerequisite) for the second. At design-time the mappings between ontologies have to be established (i.e. between the source and the target ontologies) while during run-time phase these mappings are applied in concrete scenarios on data instances.

In the design-time phase semi-automatic techniques are used, that is the human domain expert is required to perform a set of validations and choices based on machine findings, in order to assure that the mappings between the source and the target ontologies are correct. It is important to note that since these mappings are created on the ontology level (i.e. schema level), they are created only once and they can be used multiple times with no other updates (as long as the ontologies do not evolve). The mappings are applied during run-time on concrete data multiple times without any human intervention, in an automatic manner.

The Ontology Mapping Tool used by the SemanticGov project is an Eclipse plug-in which allows the domain expert to create mappings between two ontologies (source and target) and to store them in a persistent mapping storage. It also supports the specification of external transformation web services as described in Section 3.2.1. The Ontology Mapping Tool will be integrated in the SemanticGov User Tools suite. The ontology mapping tool (as well as the counterpart run-time component) developed in SemanticGov is based on well established previous research conducted in other EU founded reaserch project as DIP<sup>6</sup> or Knowledge Web<sup>7</sup>.

The following subsection gives a short overview of the main general features of the Ontology Mapping Tool and of the Run Time Data Mediator. These features include a new and innovative way of browsing the ontologies by using different view-point on the source and target ontology, a powerful decomposition algorithm that allows both a contextual navigation throughout the ontologies and two different mapping strategies (top-down and bottom-up).

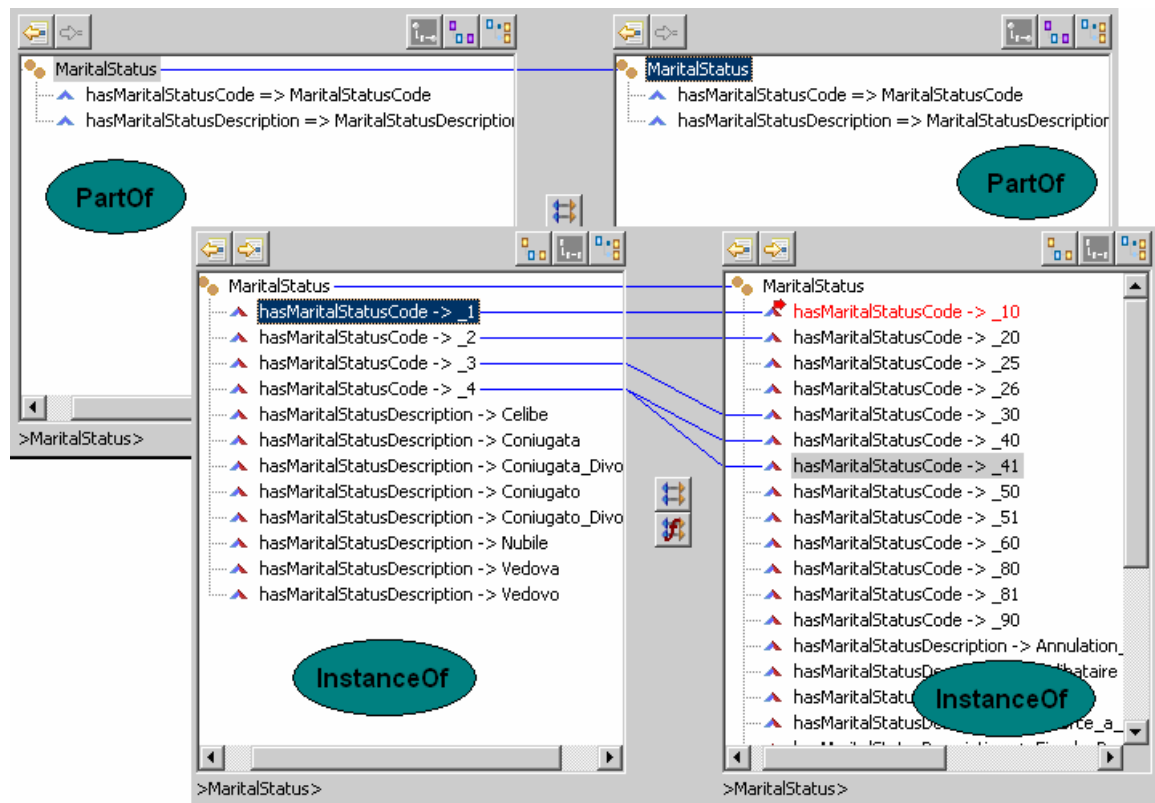
### 4.2.1 Ontology Mapping Tool

**Perspectives.** The graphical viewpoint adopted to visualize the source and the target ontologies is important to simplify the design of mappings according to their type. These viewpoints are called perspectives and by switching between combinations of these perspectives on the source and the target ontologies, certain types of mappings can be created using only one simple operation, map, combined with mechanisms for ontology traversal and contextualized visualization strategies. Figure 5 shows how the same fragment of ontology is represented by using different perspectives (more details about this can be found in Section 4.3.3).

---

<sup>6</sup> Please see <http://dip.semanticweb.org/> for more details.

<sup>7</sup> Please see <http://knowledgeweb.semanticweb.org/> for more details.



**Figure 5. The same ontology fragment visualized by using different perspectives**

A perspective contains a subset of the ontology entities (e.g. concepts, attributes, relations, and instances) and the relationships existing between them. Usually the perspective used for browsing the source ontology (source perspective) and the perspective used for browsing the target ontology (target perspective) are of the same type, but there could be cases when different perspectives types are used for source and target. In each of the perspectives there are a predefined number of roles the ontology entities can have. In general, particular roles are fulfilled by different ontology entities in different perspectives and the algorithms (such as the decomposition or the suggestions making algorithms) refer to roles rather than to ontology entities.

The roles that can be identified in a perspective are: *Compound Item*, *Primitive Item*, and *Description Item*. A *Compound Item* has at least one description associated with it while a *Primitive Item* does not have any description associated. A *Description Item* offers more information about the Compound Item it describes and usually links it with other Compound or Primitive Items. By this the *Successor* of a Description Item is defined as the Compound or Primitive Item it links to. More details about perspectives as well as a formal model that governs their main principles can be found in [20].

Not all of the information modeled in the ontology is useful in all stages of the mapping process. As such, a *context* is a subset of a perspective that contains only those ontological entities, out of that perspective, relevant to a concrete mapping stage.

A notion tightly related with contexts is the *decomposition*. A context can be created from another context (this operation is called context update) by applying decomposition on an item from a perspective or a context. Decomposition allows navigating between contexts and links consecutive nested levels; the way the contexts are navigated when creating mappings influences the creation types of mappings that are created (see Figure 6).

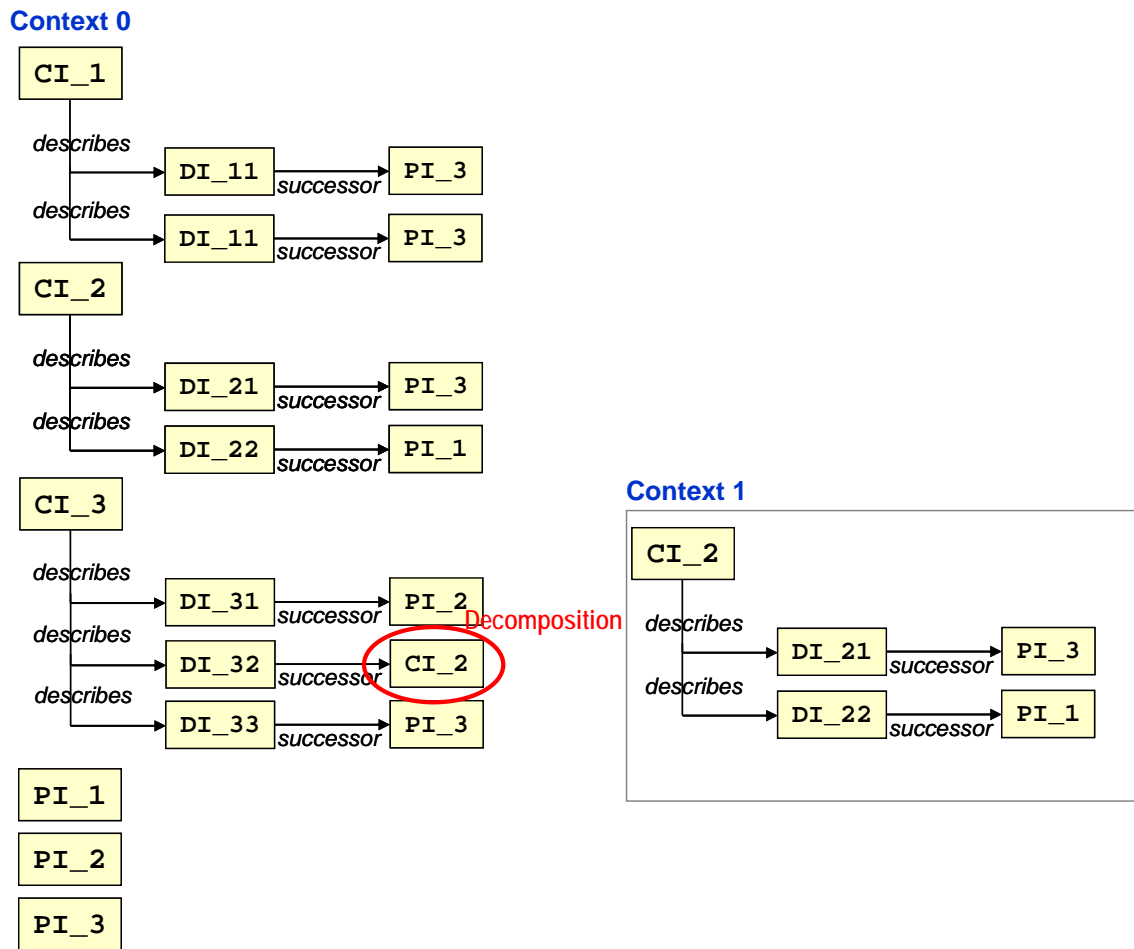


Figure 6. Decomposition and context update – abstract view

The abstract view presented in Figure 6 could correspond to the situation depicted in Figure 7.

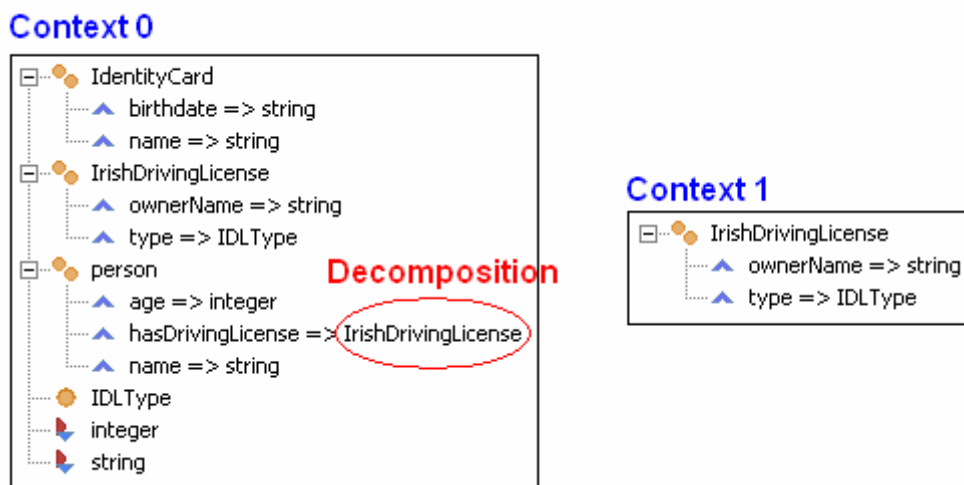


Figure 7. Concrete examples of Decomposition and Context update

It is easy to note that in the above example IdentityCard, IrishDrivingLicense and person correspond to compound items while IDLType, integer and string correspond to primitive items. The concepts' attributes correspond to description items.

**Decomposition algorithm.** The decomposition algorithm is used to offer guidance to the domain expert in the mapping process and to compute the structural factor as part of the suggestions

algorithms (described later in this section). By decomposing the descriptions of a compound item are exposed and made available to the mapping process. The decomposition algorithm can be applied to description items and returns the description items (if any) for the successors of those particular description items. An overview of this algorithm is presented in Listing 25.

**Listing 25.** Decomposition Algorithm.

```

1  void decompose(Collection collectionOfItems){
2      Collection result;
3      for each item in collectionOfItems do{
4          result = result + [item];
5          if isCompound(item){
6              result = result + getDescriptions(item);
7          }
8          if isDescriptionItem(item){
9              Item successorItem = getSuccessor(item);
10             if (not createsLoop(successorItem)){
11                 result = result + [successorItem];
12                 result = result + getDescriptions(item);
13             }
14         }
15     }
16     return result;
17 }

```

**Suggestion Algorithms.** In order to deliver a truly semi-automatic mapping tool, suggestion algorithms are a necessity. The suggestion algorithms are used to help the domain expert to make decisions during the mapping process, regarding the possible semantic relationships between the source and the target items in the current mapping context. A combination of two types of such algorithms is used in our Ontology Mapping Tool: the first being a lexical algorithm and the second being the structural algorithms that consider the description items in their computations. Brief descriptions of the functionality that could be provided by such algorithms is provided below.

For each pair of items the suggestion algorithms computes an *eligibility factor* (EF), indicating the degree of similarity between the two items: the smallest value (0) means that the two items are completely different, while the greatest value (1) indicates that the two items maybe similar. For dealing with the values between 0 and 1 a threshold value is used: the values lower than this value, indicate different items and values greater than this value indicate similar items. Setting a lower threshold assures a greater number of suggestions, while a higher value for the threshold restricts the number of suggestion to a smaller subset.

The EF is computed as a weighted average between a *structural factor* (SF), referring to the referring to the structural properties and a *lexical factor* (LF), referring to the lexical relationships determined for a given pair of items. The weights can be chosen based on the characteristics of the ontologies to be mapped. For example when mapping between ontologies developed in dissimilar spoken languages the weight of LF should be close to 0 in contrast with the case when mapping between ontology developed in the same working groups or institutions (the usage of similar names for related terms is more likely to happen). More details about the EF, LF and SF can be found in [20].

**Bottom-Up vs Top-Down Approach.** Considering the algorithms and methods described above, two possible approaches during the ontology mapping can be followed: bottom-up and top-down approaches.

The *bottom-up approach* is a mapping process that starts with the mappings of the primitive items (if we imagine the ontological model as a tree<sup>8</sup> where the nodes are concepts and the edges attributes, the primitive items are positioned at the leafs level of the model) and then continues

<sup>8</sup> This is actually an imprecise analogy since the ontology is a directed graph. More precisely, the primitive items (the “leafs”) are nodes with no edges leaving from them.

with items that reuse primitive items to describe themselves (upper level nodes). According to this, mappings of primitive items act like a minimal basic set of relationships between the two ontologies that can be easily used to gradually derive more complex relationships. This approach is recommended when a complete alignment of the two ontologies is desired.

The *top-down approach* is a mapping process that starts from compound items (non-leaf nodes) and then continues drilling down to primitive items. It is usually adopted when a concrete heterogeneity problem has to be resolved and the domain expert is interested only in resolving a particular item's mismatches and not in fully aligning the input ontologies.

In the same way as for the other algorithms, the applicability and advantages/disadvantages of each of these approaches depends on the type of perspective used.

## 4.2.2 Run-time Engine

The *Run-time Engine* plays the role of the Data Mediation component in the Communal Gateway. It uses the abstract mappings created at design time, grounds them to WSML, and uses a reasoner to evaluate them against the incoming source instances. The mapping rules, the source instances and if necessary, source and target schema information are loaded into the reasoning space in what could be seen as a "pseudo-merged" ontology (i.e. the entities from the source and the target and the rules are strictly separated by namespaces). By querying the reasoning space for instances of target concepts, if semantically related source instances exist, the rules fire and produce as results the target instances.

It is important to note that the mappings grounding module is integrated as part of the run-time component. In this way, the same set of mappings (i.e. abstract, ontology, language-independent mappings) can be grounded to different languages depending on the scenario in which the run-time mediator is used. By grounding, a formal semantics is associated with the mappings, such that only at this stage is it stated what exactly it means to have two items mapped to each other. These formal semantics differ from one mediation scenario to another, i.e. different grounding has to be applied when using the abstract mappings in instance transformation than when using them in query rewriting. An additional advantage is an easier management of mappings that form the ontology alignment.

If ontologies evolve, the mappings have to be updated accordingly and it becomes more efficient to perform these updates free of any language-related peculiarities. The main scope of the run-time engine is to be deployed as a component in the semantic environments such as the SemanticGov architecture. Additionally it can be made available as a Web service (i.e. a Semantic Web service) that can be invoked directly via SOAP from specialized tools or through a Web interface using a Web browser. And finally, the Run-time mediation can be offered as a standalone application that helps in testing and evaluating the mappings during the mapping process at design-time.

## 4.3 Resolving the Semantic Interoperability Conflicts

This section shows how the examples presented in Section 3 can be modeled in our Ontology Mapping Tool. We have selected only the most relevant interoperability conflicts and omitted those that can be resolved in a similar manner with one of the conflicts already addressed.

### 4.3.1 Data Level Conflicts

Using the Ontology Mapping Tool's graphical interface the domain expert can create mappings between various evidence placeholders that need to be aligned. Figure 8 shows how the mapping tool presents the two ontology fragments shown in Listing 2 to the user and how mappings can graphically be created.

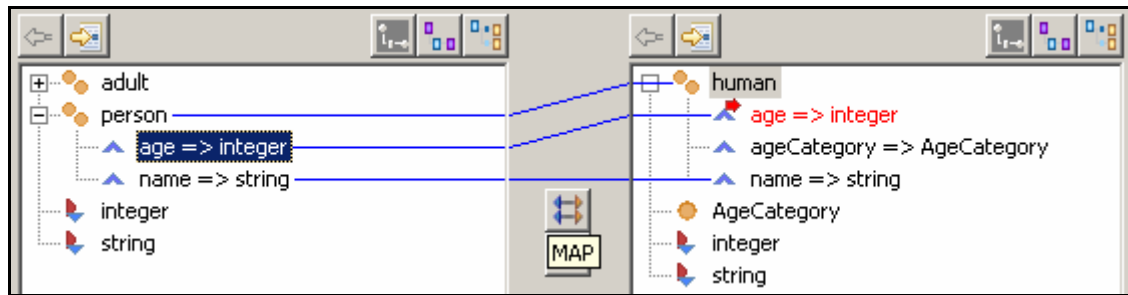


Figure 8. Simple Mappings

Some mappings, as shown in Listing 3, lines 14-18, have conditions associated, meaning that these mappings hold for all the cases for which the conditions hold. Such mappings and their associated conditions are transformed into complex logical expressions in the underlying formalism (WSML [1], [2]) but the ontology mapping tool offers the means to isolate the domain expert from the burdensome of the logical language used. Figure 9 illustrates how such conditions can be created: the domain expert can select one comparator for the attribute *age* of *person* and set the concrete age the attribute's value will be compared with.

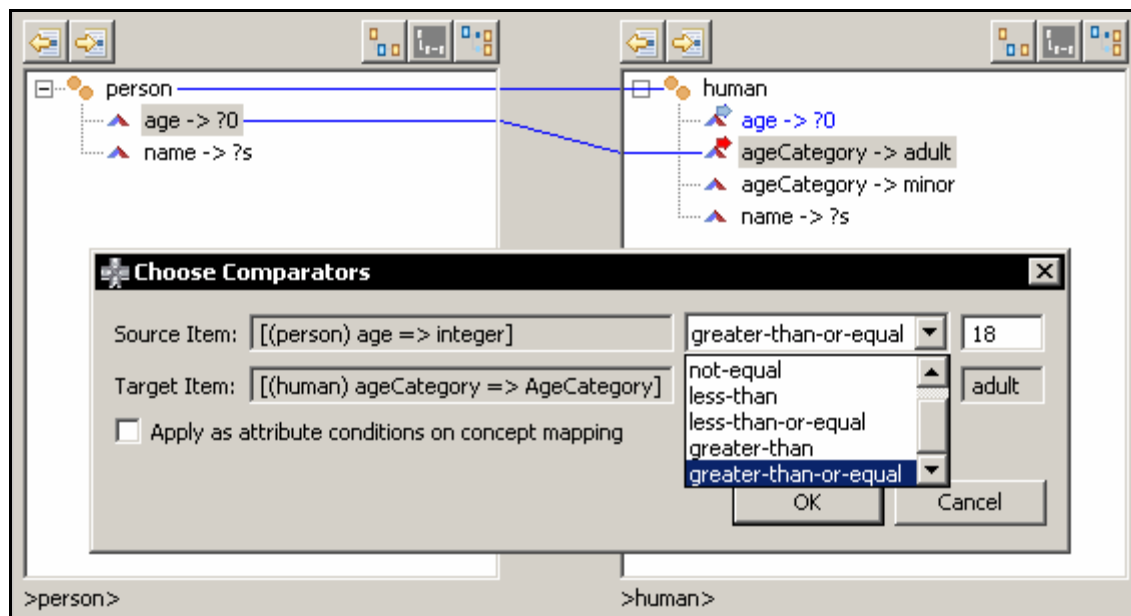


Figure 9. Conditions on Attributes Mapping

It is important to note that Figure 9 uses the *InstanceOf* perspectives to present the ontologies to the domain expert in contrast to Figure 8 which uses the *PartOf* perspectives. The *PartOf* perspective shows the concepts defined in the ontology together with their attributes and attributes' ranges while the *InstanceOf* displays the concepts and the possible attribute values their attributes can take. For the *ageCategory* attribute the two allowed values *adult* and *minor* defined in the ontology are presented as alternatives. In the case of infinite domain data types (e.g. string or integer) it is not possible to show all the values the attributes having them as range can take. Instead two placeholders are used, “?0” and “?s” to denote a value that can be later manually filled through the *Choose Comparators* dialog. The explicit instances (e.g. *adult*) appear in this dialog as fixed non-editable values and are directly dependent of selected attribute in their ontology.

Figure 10 shows that in a similar manner conditions can be associated with a given concept mapping (mapping between *human* and *adult*).

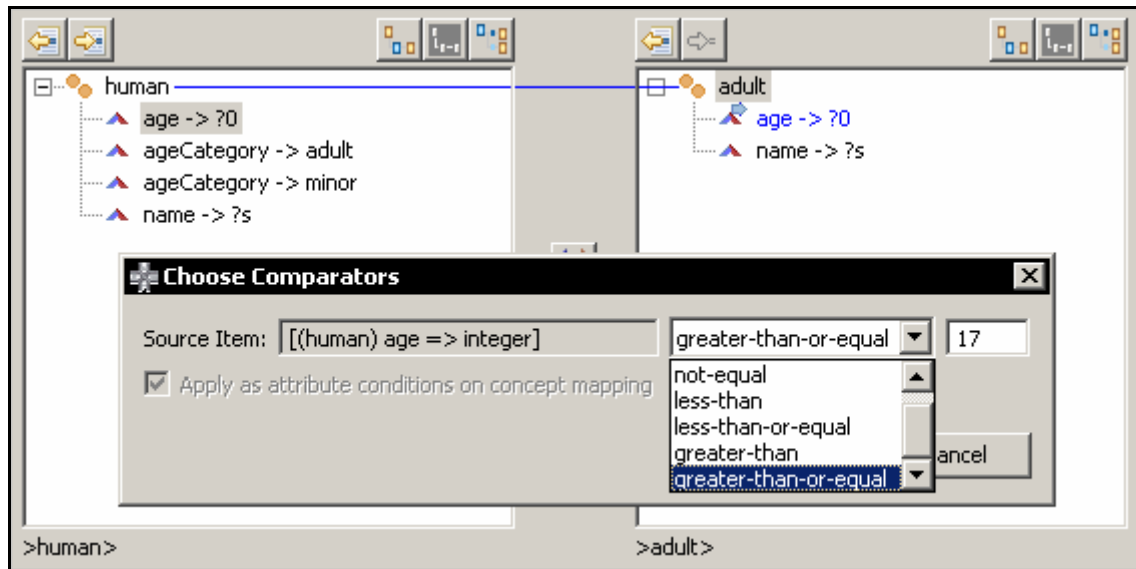


Figure 10. Conditions on Concepts Mapping

In the same fashion, the conflicts generated by differences in effect domains can be solved by having complete descriptions of the contextual knowledge (see Listing 4) complemented by the appropriate mappings between the domains (Listing 5). Such a mapping (whose creation is shown in Figure 11) is the associated with the concept mapping between the `IrishDrivingLicense` and `GermanDrivingLicense`: this mapping does not hold unless the type of the `IrishDrivingLicense` is permanent. As a consequence even if the attributes `hasDrivingLicense` and `drivingLicense` of `person` and `human` are mapped (lines 4-5 in Listing 5) no filler for attribute `drivingLicense` of concept `human` will be created.

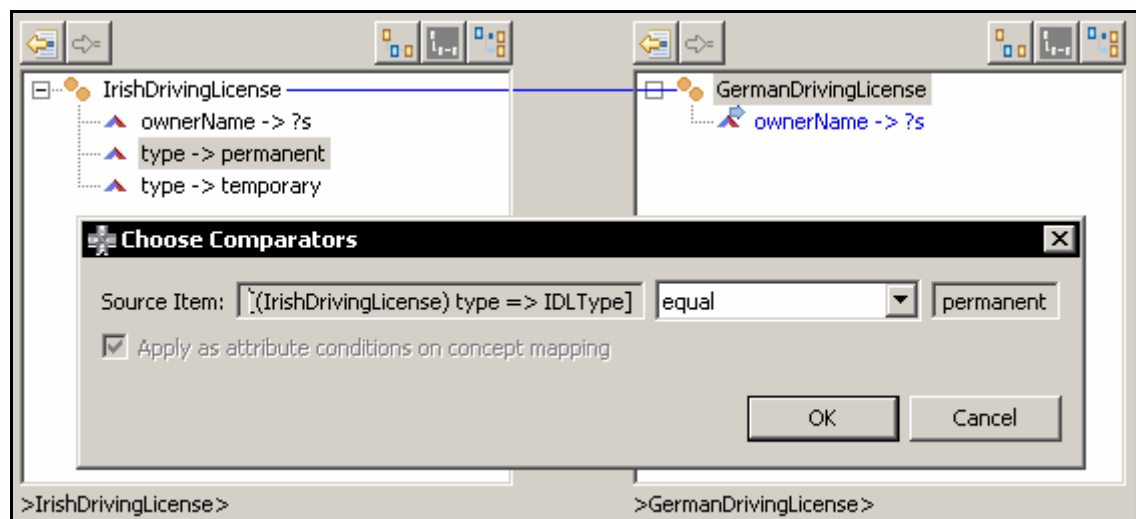
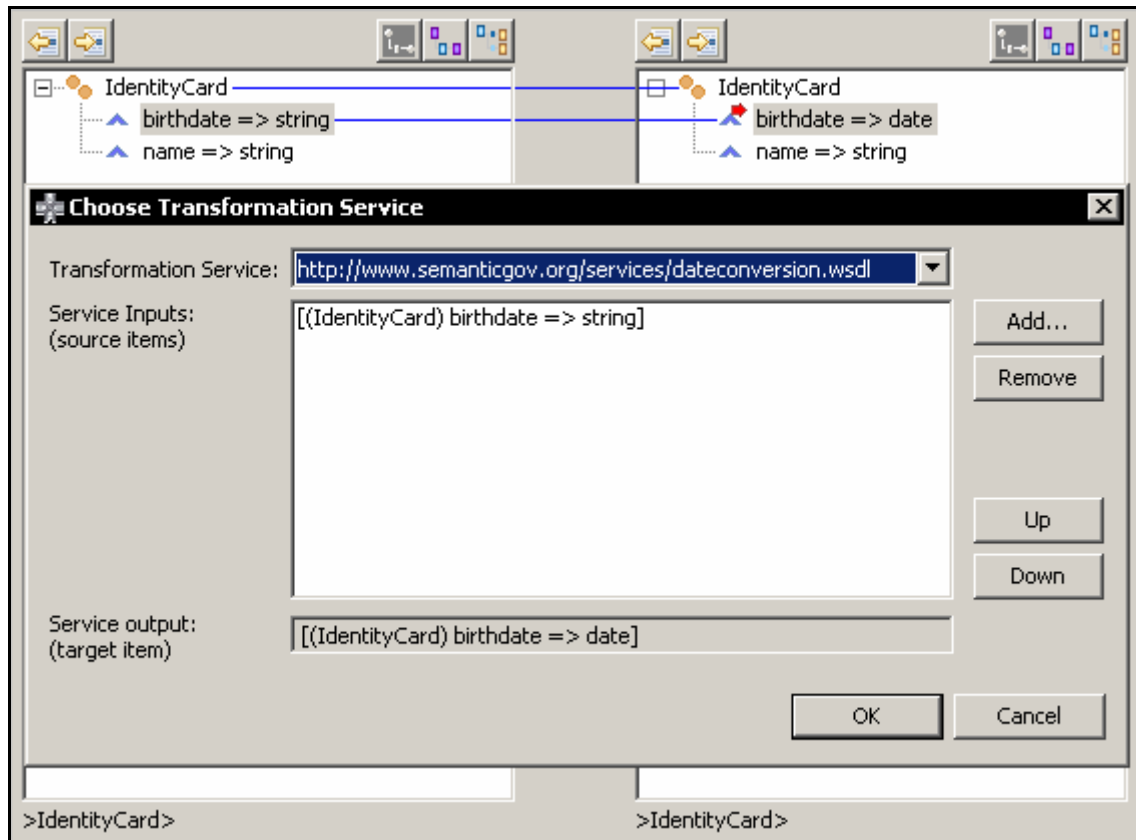


Figure 11. Conditions on Concept mappings (solving the differences in the effect domains conflict)

### 4.3.2 Data Representations Conflicts

In order to solve data representation conflicts external functions have to be applied on the values that need to be transformed from one representation form to another. Our approach uses Web services but it can be easily adjusted to accommodate calls to other (e.g. internal) implementations of these services. As Listing 8 shows the definition of all the details necessary to perform at run-time the service call is not trivial if it has to be done manually. In this way our Ontology Mapping

Tool provides graphical support that assists the domain expert in specifying the right service to be used for performing the transformation (see Figure 12).

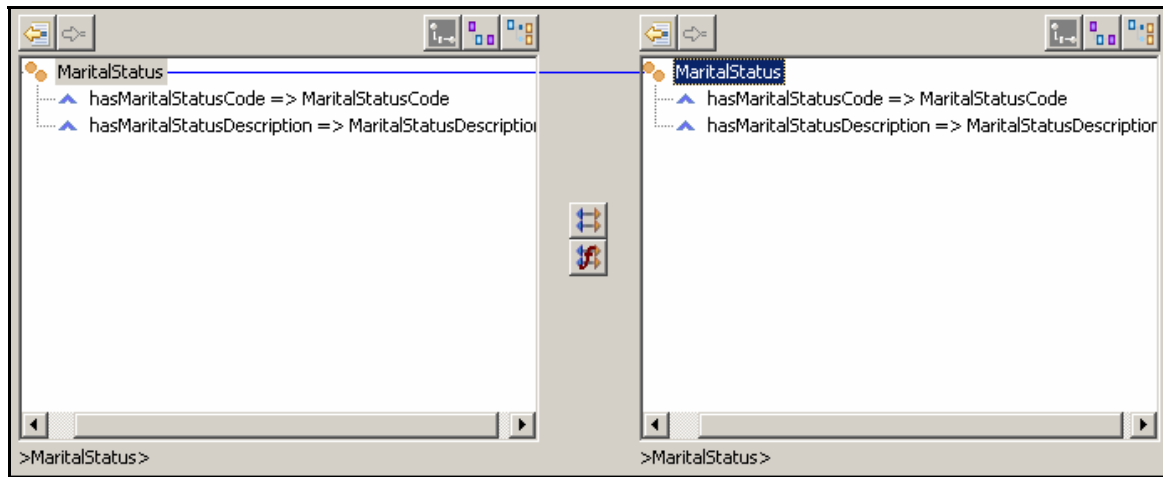


**Figure 12.** Applying Transformation Services to Solve Data Representation Conflicts

The domain expert simply selects the source and the target items whose values are to be used as input and output, respectively for the service. The domain expert may choose more attributes to be used as inputs for the service (he/she can change the order of this inputs as well); after this the service can be chosen from a list of pre-existing services. The set of predefined services can be easily extended using the configuration options of the tool. Of course if the details of transformation service are used in creating the mappings this service must exist in order to be invocable by the run-time engine.

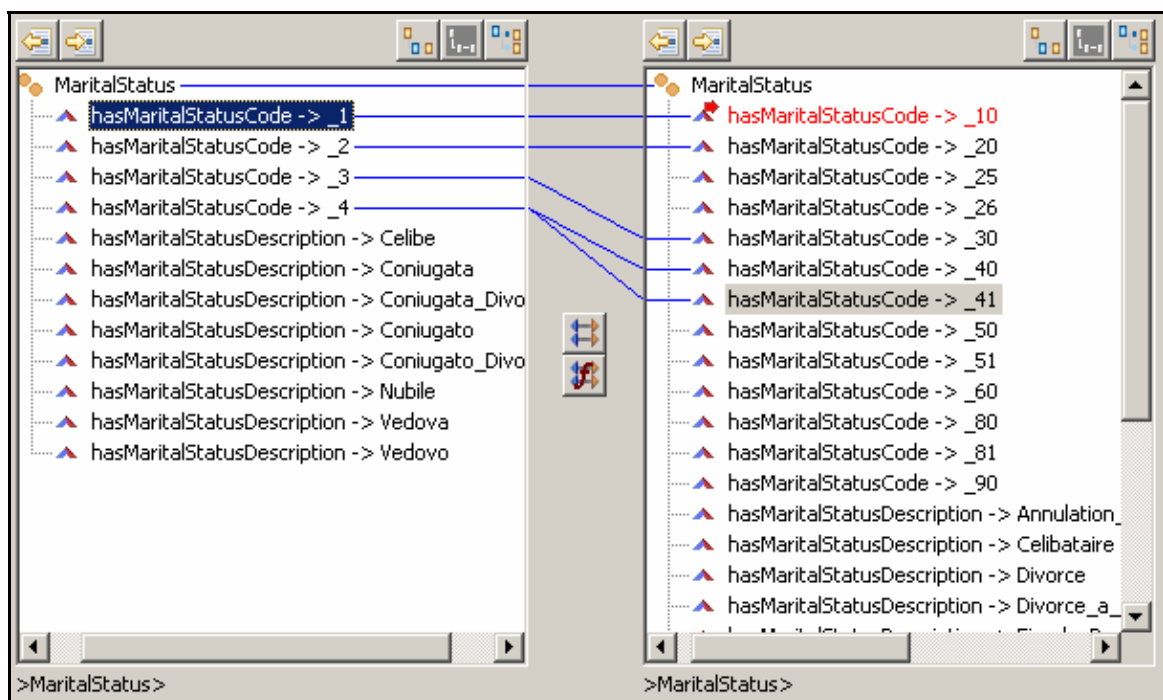
### 4.3.3 Data Precision Conflicts

The Data Precision Conflicts can be solved by using the *InstanceOf* perspective. As shown in Listing 12 the codes for the marital status are modelled as instances in both the Belgian and Italian ontology. If we would use the *PartOf* perspective (which is one of the common views on ontologies used in most of the ontology mapping tools) we could see only the concepts and their attributes (see Figure 13) and conditional mappings as the ones showed in Listing 13 would be impossible to create without manually editing the mappings or the mapping rules.



**Figure 13.** *MaritalStatus* Concepts Displayed in the *PartOf* view

By using the *InstanceOf* perspective (Figure 14) all the possible values that can be taken by the *hasMaritalStatusCode* attributes are available to be used in creating mappings (i.e. attribute mappings having associated conditions based on the corresponding values).



**Figure 14.** *MaritalStatus* Concepts Displayed in the *InstanceOf* view

#### 4.3.4 Entity Identifier Conflict

This kind of conflicts can be resolved by requesting an external service to compute a value for this identifier that is going to be used in all further communications. Such external service can be invoked either from a higher level (i.e. from the architectural level as part of the overall workflow) or from the mapping rules level in a similar manner as the transformation services are invoked in Section 4.3.2. (see Figure 15). In our example, since only a piece of data is missing, which is not relevant to the overall functionality that is being requested or offered, we consider that the second solution is preferable.

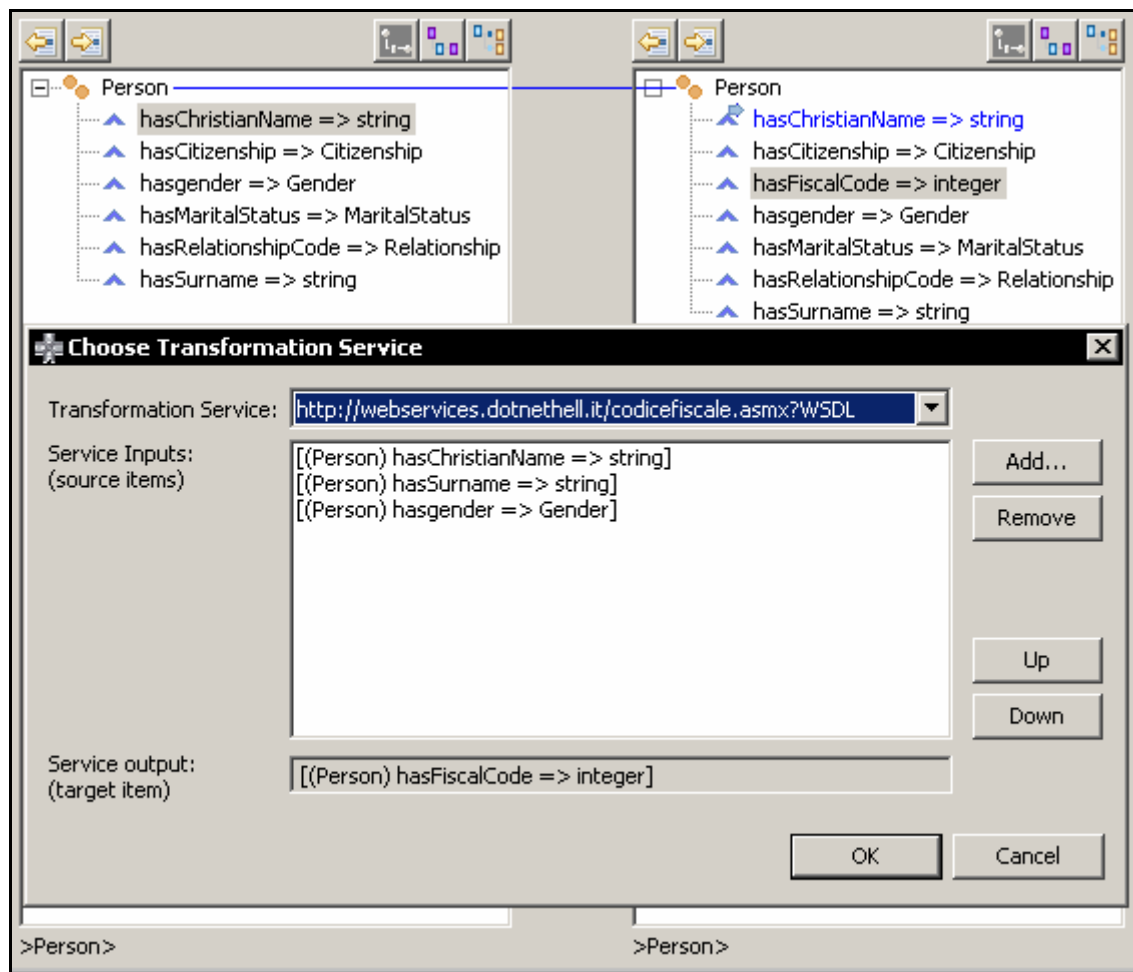


Figure 15. Using an External Service to Obtain an Identifier

#### 4.3.5 Schema-isomorphism Conflicts

The case when *the source ontology element covers more information than the target ontology element* is simply resolved by not creating mappings between the extra-items and anything else. For the when *the source ontology element covers less information than the target ontology element* the first situation (*Default values generation*) can be resolved either from the source side (by asking external services for the missing values, in a similar manner as in Section 4.3.4) or from the target side (by relying on the rules existing in the target ontology that will handle the incomplete instances when they arrive). For the later option the solving of the conflict is out of the scope of the data mediation infrastructure in the Semantic Gateway. For the second situation (*Additional interactions with the requester*) the extra interaction with the requester should be handled at a higher level, at the level of the overall workflow and it is as well out of the control of the mediation infrastructure in the Semantic Gateway.

For the current version of the Data Mediator implementation all the above cases are supported except the “*Additional interactions with the requester*” case.

#### 4.3.6 Generalization Conflicts

The generalization conflicts are solved by multiple mappings (n:m) created between different conceptualization levels in the source and the target ontologies. The example in Listing 19 can be solved as shown in Figure 16 and Figure 17. It is important to note that for this type of mappings

the decomposition and context (described in Section 4.2.1) play an important role in guiding the domain expert.

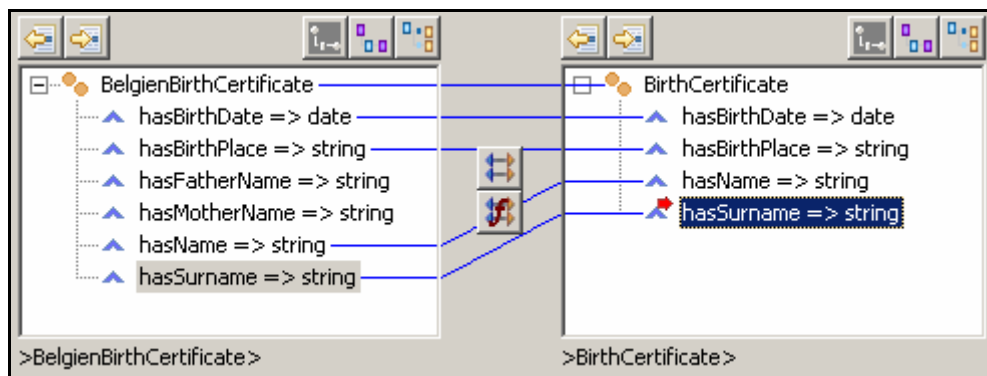


Figure 16. Mapping the two Types of Birth Certificates

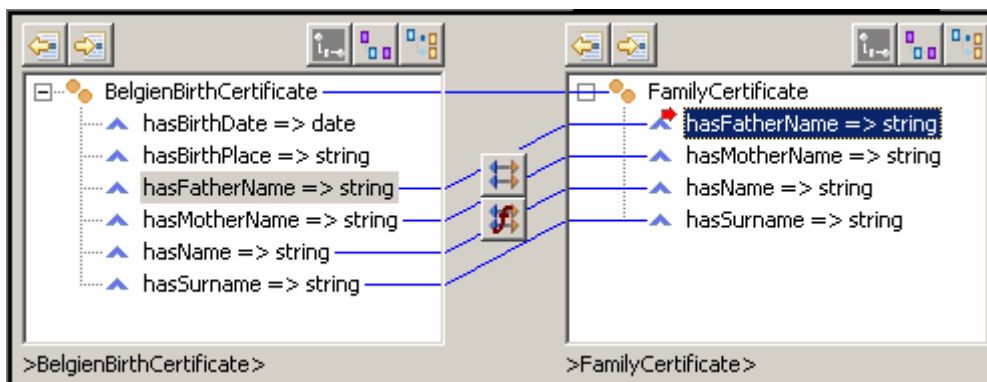


Figure 17. Mapping the BelgianBirthCertificate with the FamilyCertificate

Another aspect worth mentioning is that when mapping the `BirthCertificate` and `FamilyCertificate` to the `BelgienBirthCertificate` the values of the `BirthCertificate` and `FamilyCertificate`'s common attributes need to be identical (e.g. `hasName` attribute). Otherwise, no instance of the `BelgienBirthCertificate` is created by the reasoner. This feature can be used implicitly to make sure that from an arbitrary instances set of both `BirthCertificate` and `FamilyCertificate` the proper pairs are used in aggregating the information to be encapsulated in the instances of `BelgienBirthCertificate`.

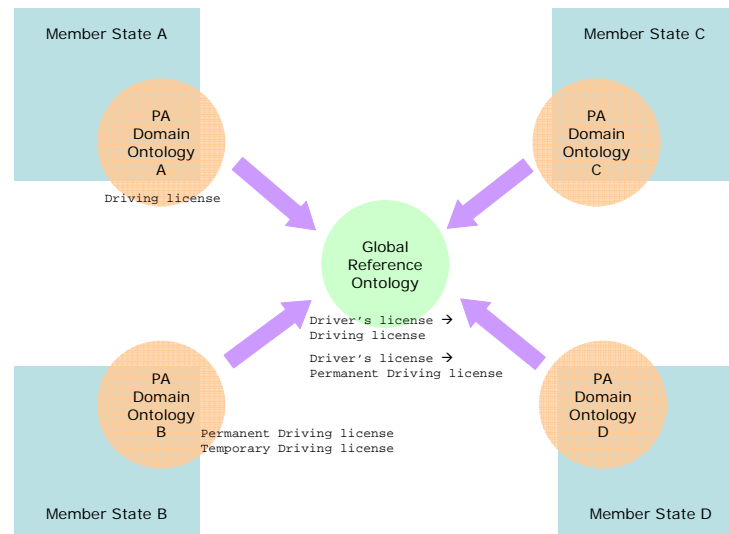
#### 4.3.7 Aggregation Conflicts

The aggregation conflicts created by using different levels of encapsulation at the schema level, (Listing 22), can be solved by creating mappings as presented in Section 4.3.6). The aggregation conflict created by different encapsulation used at the data level (Listing 24) can be solved by using transformation services as shown in Section 4.3.2).

### 4.4 Applying the Data Mediator

The mediation solutions encompass by the Data Mediator in the Semantic Gateway of the SemanticGov architecture, are to be applied in solving data heterogeneity problems between service consumers (i.e. citizens) and service providers from and across various member states. We did not make so far any assumption regarding the way these mediation facilities are going to be used. The solution we describe in this paper supports both a P2P fashion strategy and reference ontology strategy. In case of P2P approach a set of direct mapping rules between every pair of

member states has to be provided (there are at the time of writing this paper 27 member states part of the European Union, so this means  $27 \times 26$ , over 700 unidirectional sets of mappings); while using a global approach based on a reference ontology the number of required mappings will be reduced to  $2 \times 27$  (see Figure 18).



**Figure 18.** Semantic mediation using a reference ontology

The solutions we described in this paper could be used in either ways. If the data heterogeneity problems between  $MS_1$  and  $MS_2$  are to be solved via a reference ontology  $O_G$  then we need two sets of mappings, between  $O_{MS_1}$  and  $O_G$  and between  $O_G$  and  $O_{MS_2}$  (and one more pair for the other direction). When these mappings are required for a data mediation task, the reasoning and the derivation of the final mediated instances can be performed in a single invocation of the Data Mediator: based on the  $O_{MS_1}$  instances the first set of rules will trigger generating the first mediated instances in terms of the  $O_G$  ontology which in their turn will trigger the second set of rules that, finally, will the corresponding instances in terms of  $O_{MS_2}$ .

A third strategy can be adopted as well, when mappings are created between member states in such a way that a full connected directed graph is maintained at all time, where the nodes are the member states and edges symbolize mappings between the two member states and where there is always a path between to nodes. The data mediation between two arbitrary member  $MS_i$  and  $MS_j$  states can be done by chaining and executing the mappings between  $O_{MS_i}$  and  $O_{MS_{k1}}$ ,  $O_{MS_{k1}}$  and  $O_{MS_{k2}}$  and ... and  $O_{MS_{kn}}$  and  $O_{MS_j}$ . This case is a generalization of the case when only an intermediary ontology (the global ontology) is used.

## 5 Conclusions

Solving the semantic interoperability conflicts between public e-services at a pan-European level is a crucial aspect when aiming to build a distributed infrastructure able to support cross-member states interactions. European Union citizens must be able to benefit when appropriate from the services offered by other countries than their own and public administration agencies should be able to seamless cooperate and interact in solving requests that involve trans-national legal processes.

In this document we provided an analysis of the technical requirements for solving some of the most relevant types of interoperability conflicts. We used our implemented system to show how such requirements can be fulfilled and how exactly these conflicts can be solved. The conceptual analysis is based on the GEA object model and the well-known interoperability classification framework of information systems proposed in [10]. The solution we have adopted is based on ontology mapping and involves the creation of alignments between the domain ontologies at design-time and their application at run-time to solve concrete heterogeneity problems. The examples we chose to illustrate the interoperability conflicts and the solutions we apply in our approach are based on real case-studies developed in the SemanticGov project.

As future work, the current approach can be extended by providing dynamic support for discovery of value transformation services. That is, instead of having a fixed number of such services we can have an intelligent mechanism in place that would allow the dynamic discovery (e.g. during the mappings creation) of available services registered with given repositories. Also we intend to make several performance evolutions of the run-time mappings execution process and to compare the results when different mediation strategies are used (e.g. global ontology, P2P or chaining).

## References

- [1] J. de Bruijn, H. Lausen, R. Krummenacher, A. Polleres, L. Predoiu, M. Kifer, and D. Fensel. The Web Service Modeling Language (WSML). Technical report, WSML Working Draft, <http://www.wsmo.org/TR/d16/d16.1/v0.2/>, October 2005.
- [2] H. Lausen, J. de Bruijn, A. Polleres, and D. Fensel. WSML – A Language Framework for Semantic Web Services. In *W3C Workshop on Rule Languages for Interoperability*, April 2005.
- [3] A. Mocan, E. Cimpian, and M. Kerrigan. Formal Model for Ontology Mapping Creation. In *Proceedings of the 5th International Semantic Web Conference (ISWC 2006)*, Athens, Georgia, USA, November 2006.
- [4] F. Scharffe and J. de Bruijn. A language to specify mappings between ontologies. In *IEEE Conference on Internet-Based Systems SITIS6*, Yaounde, Cameroon, December 2005.
- [5] J. Euzenat, F. Scharffe, and L. Serafini. D2.2.6: Specification of the delivery alignment format. Knowledge Web Deliverable, February, 2006, <http://www.inrialpes.fr/exmo/cooperation/kweb/heterogeneity/deli/kweb-226.pdf>.
- [6] M. Billiet, S. Bonomi, E. van der Graaf, E. Kirgiannakis, N. Loutas, A. van Overeem, V. Peristeras, K. Tarabanis, and J. Witters. D2.1: Business, semantic and technical requirements analysis for semantic e-Government services at the national and pan-European level. Technical report, SemanticGov Deliverable, <http://www.semantic-gov.org/index.php?name=UpDownload&req=getit&lid=311>, December, 2006.
- [7] T. Vitvar, M. Kerrigan, A. van Overeem, V. Peristeras, and K. Tarabanis. Infrastructure for the semantic pan-european e-government services. In *AAAI Spring Symposium on Semantic Web Meets E-Government*, Stanford, CA, USA, March, 2006.
- [8] E. Tambouris and K. Tarabanis. Egovernment and interoperability. In *European Conference in Electronic Government (ECEG 2005)*, pp 399-407, Belgium, June, 2005.
- [9] J. Witters and A. van Overeem. Pegs infrastructure architecture v 1.0. IDABC, 2004.
- [10] J. Park and S. Ram. Information systems interoperability: What lies beneath? *ACM Transactions on Information Systems*, 22(4):595–632, 2004.
- [11] V. Peristeras. *The Governance Enterprise Architecture - GEA - for reengineering public administration*. PhD Dissertation, University of Macedonia, Thessaloniki
- [12] M. Kerrigan, A. Mocan, M. Tanler and D. Fensel. The Web Service Modeling Toolkit - An Integrated Development Environment for Semantic Web Services. In *Proceedings of the 4th European Semantic Web Conference (ESWC), System Description Track*, June 2007, Innsbruck, Austria
- [13] A. Mocan and E. Cimpian. An ontology-based data mediation framework for semantic environments. In *International Journal on Semantic Web and Information Systems (IJSWIS)*, 3(2), 66-95, April-June, 2007
- [14] D. Fensel, H. Lausen, A. Polleres, J. de Bruijn, M. Stollberg, D. Roman, and J. Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. 2006.
- [15] T. Vitvar, A. Mocan, E. Cimpian, S. Nazir, X. Wang, N. Loutas, V. Peristeras, K. Tarabanis, E. Kirgiannakis, K. Winkler, M. Dimitrov, V. Momtchev, and M. Mecella. D3.1: SemanticGov Architecture v.1. Technical report, SemanticGov Deliverable, December, 2006.
- [16] T. Vitvar, A. Mocan, M. Kerrigan, Michal Zaremba, Maciej Zaremba, M. Moran, E. Cimpian, T. Haselwanter, and D. Fensel. Semantically-enabled Service Oriented

- Architecture: Concepts, Technology and Application. In *Journal of Service Oriented Computing and Applications*, Springer London, 2007.
- [17] M. Dean and G. Schreiber. OWL web ontology language reference. Technical report, World Wide Web Consortium (W3C), available at <http://www.w3.org/TR/owl-ref/>, February 2004.
- [18] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceeding of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR99)*, pp. 161–180, 1999.
- [19] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. In *Journal of the ACM*, (42), July 1995, pp. 741–843.
- [20] A. Mocan, E. Cimpian, and M. Kerrigan. Formal Model for Ontology Mapping Creation. In *Proceedings of the 5<sup>th</sup> International Semantic Web Conference (ISWC 2006)*, Athens, Georgia, USA, November 2006.
- [21] V. Peristeras, N. Loutas, S. Goudos, and K. Tarabanis. Semantic Interoperability Conflicts in Pan-European Public Services. In *Proceedings of 15th European Conference on Information Systems*, St. Gallen, Switzerland , 2007.
- [22] A. Malhotra, J. Melton, and N. Walsh. XQuery 1.0 and XPath 2.0 functions and operators. Technical report, World Wide Web Consortium (W3C), 2005.
- [23] J. Yanosy: Semantic interoperability and semantic congruence. In *Collaborative Expedition Workshop*, August, 2005.
- [24] R. Baldoni, E. Cimpian, M. Dimitrov, Th. Fragis, G. De Giacomo, A. Kiryakov, N. Loutas, M. Meccela, A. Mocan, S. Nazir, A. v. Overeem, V. Peristeras, R. Russo, K. Tarabanis, E. Triantafyllou, T. Vitvar, W. Waterfeld, K. Winkler, and J. Witters. D1.2. State of the Art Report. Technical report, SemanticGov Deliverable, 2006
- [25] D. Peterson, P. V. Biron, K. Permanente, A. Malhotra, and C. M. Sperberg-McQueen. XML Schema 1.1 Part 2: Datatypes, Technical report, World Wide Web Consortium (W3C), 2005, <http://www.w3.org/TR/xmlschema11-2/>, 17 February 2006
- [26] N. Mitra and Y. Lafon. SOAP Version 1.2 Part 0: Primer, Technical report, World Wide Web Consortium (W3C), <http://www.w3.org/TR/soap12-part0/>, April 2007