WSMO-PA: Formal Specification of Public Administration Service Model on Semantic Web Service Ontology

Xia Wang, Tomas Vitvar, Vassilios Peristeras
Digital Enterprise Research Institute (DERI)
National University of Ireland, Galway, Ireland
$\{xia.wang, toams.vitvar, vassilios.peristeras\}@deri.org$
Adrian Mocan
Digital Enterprise Research Institute (DERI)
Leopold-Franzens Universität Innsbruck, Austria
$adrian.mocan@deri.org$

Sotirios K. Goudos, Konstantinos Tarabanis
Informatics and Telematics Institute
Center for Research and Technology Hellas, Greece
$\{sgoudos, kat\}@uom.gr$

Abstract

In this paper we define a formal model for a Public Administration service on the basis of the Web Service Modeling Ontology (WSMO). For this purpose we employ the generic public service object model of the Governance Enterprise Architecture (GEA) providing a PA domain specific semantics. We investigate conceptual mappings between PA entities and WSMO elements, and on the real-world use case present the detailed formal PA service model based on the WSMO service model.

1 Introduction

E-Government is an attractive domain for both academia and industry. Recently, the emphasis is put on the modeling of Public Administration domain, and application of the semantic web technologies to integration of eGovernment systems. Some discussions on applying semantic and web service technology in the eGovernment domain are concluded as follows:

- Public administration (PA) is a huge, diverged and distributed environment layered in clearly defined organizational levels (e.g. local, regional, state, national). It causes difficulties when testing and applying semantic technologies and solutions in a large scale.
- Public administration is hierarchically organized, which means that there is a rather clear line of command, a central coordination and a rigidly defined corpus of rules that explicitly define the system behavior. These characteristics describe a domain that is supposed to be more easily standardized in comparison to the totally decentralized and competitive environments in the private sector.
- Public administration is currently considered as the heaviest service industry, with a service production distributed in hundreds (even thousands) of partially independent agencies. This means that Service Oriented Architecture (SOA) paradigms putting the "service" notion at the core of development (e.g. Semantic Web Services) are particularly suitable and fit well with these structural characteristics of the PA domain.
- Through the implementation of the European Union policies, there is an increasing need for collaboration and interoperation among the Member States Administrations (MSA). Semantic interoperability is perceived as a key aspect to be adequately addressed in this environment in order to make feasible direct interoperation and communication between MSAs (aka Pan-European eGovernment Services - PEGS).

Until now, the research streams on modeling of e-government and semantic web services have not been connected adequately. There are few efforts trying to combine

a generic service ontology with PA domain models in order to present rich PA service descriptions with executable features in web service technological environments. In other words, there is no domain specializations/instantiations of generic service models. This is the problem for example of [16, 20].

Especially in PA, detailed service description would be very useful, in order to solve some demanding semantic interoperability problems in this particularly complex and distributed environment. Therefore, Our Governance Enterprise Architecture (GEA) [9, 10] defines a generic domain model for PA. This model defines common aspects and generic features of the domain, with emphasis on service and process models. Also it constitutes the basis for a reference ontology of the e-government domain [17]. Such ontology is generic enough to cover the overall e-government domain while at the same time specific enough to sufficiently model PA specific semantics. The introduction of GEA is in Section 2.

On the other side, it is an obvious approach to implement e-government business by semantic web service technologies. Currently, there are several well known description models for semantic web services - WSMO, OWL-S [4] and WSDL-S [5], which have different mechanisms to describe service semantics.

WSMO conceptual model describes all relevant aspects related to general semantic services by four top-level elements: *goals*, *web services*, *ontologies* and *mediators*. It has a family of layered logical languages providing rich semantics together with reasoning capabilities (WSML). It also has a set of implementation mechanisms/environments for Web services (e.g. WSMX, IRS-III). Due to the limitation of paper space, for the detailed comparison between the other models we refer to [19] and [1]. This paper focuses on combining a conceptual service model for PA with WSMO/WSML.

In this paper we extend our earlier work [9]. We define a GEA ontology for the generic public services of the e-goverment domain, and present a formal way to model object entities of PA services in a WSMO ontology. The goal of this research is to propose a WSMO representation of a PA service model taking into account domain semantics as modeled in GEA. The WSMO representation of a PA service model will capture and manifest PA-specific representation features enabling intelligent and client-friendly PA service discovery, composition, invocation, execution and monitoring.

The rest of the paper is organized as follows. In section 2 and section 3 we introduce the GEA PA service model and WSMO respectively. In section 4 we analyze and define mappings between PA service entities and WSMO concepts. In section 5 we present a driving license example used throughout this paper. In section 6, we present details of the formal representation of WSMO-PA concepts, including definitions of PA domain ontology, web services, goals and mediators. These concepts are illustrated on the driving license example. In the section 7 we summarizes the paper and give an outlook on our future work.

2 GEA Service Model for PA

The GEA object model for service provisioning is illustrated in Fig. 1. Such PA service definition and description presented is generic enough to cover all different application areas of public administration. This makes it highly reusable in different cases for the public service domain.

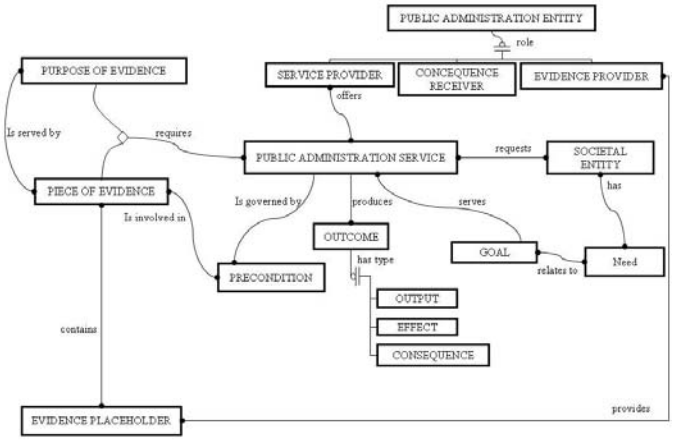

Figure 1. The GEA-PA Service Model

For the purpose of formulating PA applications, a brief description is represented here, details by referring to [10]. In Fig. 1, *Governance Entities* have two categories: *Political Entities* and *Public Administration* (PA) Entities. During execution time, PA entities have three kinds of roles: *Service Provider*, *Evidence Provider*, and *Consequence Receiver* (that is, PA services may have three kinds of capabilities, as providing service, providing evidence, and receiving consequence).

Political Entities define *PA Services*. *PA Entities* through their role as *Service Provider* offer real services. *PA Services* are governed by *Preconditions* usually specified by legal acts — *Laws*. And *PA service* defines three outcomes, which are *Output* (a kind of document decision yielded by service provider and returned to the client), *Effect* (the result of executing a service changing the state of the world, e.g., transferring money to an account), and *Consequence* (the information about an executed PA service that needs to be forwarded to interested parties, i.e., to an appointed consequence receiver).

Societal Entities (e.g., citizen, business) have *Needs* related to specific *Goals*. *Need* is an informal description of the *Goal*, as experienced from a client's perspective. Also a *Societal Entity* requests a *Public Administration* (PA) service to serve its *Goals*.

Evidence is primarily pure information; it is stored in *Evidence Placeholders*, which, in turn, contain *Pieces of Evidences*. *Preconditions* are validated by *Pieces of Evidence* and *Purposes of Evidence*. The direct relationship between *PA Service* and *Evidence Placeholder* depicts cases where *PA Services* preferably use specific types of *Evidence Placeholders*, e.g., when the law explicitly states that a birth certificate is needed for the execution of a particular service.

3 Introduction to WSMO

WSMO is a meta-model for Semantic Web services related aspects in order to facilitate the total or partial automation of discovery, composition, selection and invocation of electronic services over the web [1]. The Meta-Object Facility (MOF) [13] specification is used to specify this model. MOF defines an abstract language and framework for specifying, constructing, and managing technology neutral metamodels. Four top-level elements of WSMO are briefly described as follows:

- **Ontologies** provide terminologies used by other elements to describe the relevant aspects of the domains of discourse. The class *ontology* in WSMO specification explicitly defines a shared conceptualization, by specifying its attributes, including *hasNonFunctionalProperty*, *importsOntology*, *usesMediator*, *hasConcept*, *hasRelation*, *hasFunction*, *hasInstance*, and *hasAxioms* (for more information please see [1]).
- **Web services** represent computational entities able to provide access to services that, in turn, provide some value in a domain. In WSMO, web service is defined as a class *service* with attributes, such as *importsOntology*, *usesMediator*, *hasCapability*, *hasInterface*. All these aspects of a web service are described using the terminology defined by ontologies.
- **Goals** describe aspects related to user desires with respect to the requested functionality. The attributes of class *goal* consist of *importsOntology*, *usesMediator*, *requestsCapability*, and *requestsInterface*. Again, ontologies can be employed to define the domain terminology used to describe the relevant aspects of goals.
- **Mediators** describe elements that handle interoperability problems between WSMO elements, especially resolving mismatches between different terminologies used (data level), in communicating between web services (data and process level), and on the level of combining web services (data, process and functional level).

4 GEA-PA to WSMO Mapping

This section presents conceptual mappings between PA service model and WSMO model, aiming to provide a formal approach for defining and implementing PA services as WSMO services. We strictly follow the WSMO specification including its syntax and semantics to define a PA ontology, PA web services, PA goals and mediators in the WSMO-PA service model.

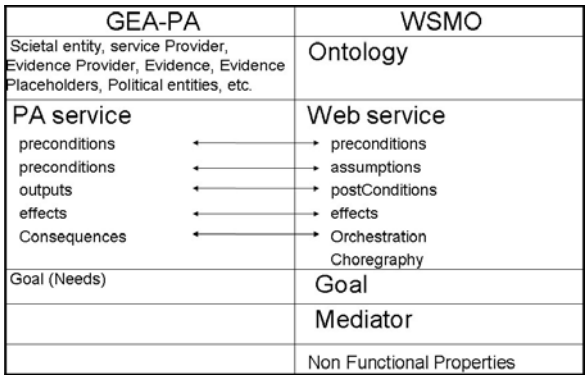

Figure 2. WSMO and GEA concepts

The mappings between GEA-PA and WSMO are illustrated in Fig. 2.

- We build a GEA domain-specific ontology on top of the other established domain-neutral specifications of WSMO. Such a generic GEA ontology can be shared by all elements of the WSMO-PA service model. Every object entity of PA service model is provided by a corresponding ontology class, their concepts, relations, functions, instances and axioms.

  In the GEA-PA service model, *Societal Entity*, *Service Provider*, *Evidence Provider*, *Evidence*, *Evidence Placeholder*, *Political Entity* and so on will be defined as subclasses of the general GEA Ontology. The definition is presented in Section V.B.

- *PA service* is modeled as a WSMO web service. It is described by capabilities and interfaces. For the capabilities, we map the PA preconditions into WSMO preconditions; PA preconditions (or *Rules* of a specific PA service) can be also mapped to WSMO assumptions. The separation between precondition and assumption is based on the fact whether required information is available in the information space (information used in assumptions are usually not included in the information space of service requester/provider). Finally, the

PA outputs are mapped into WSMO postconditions, the PA effects are mapped into WSMO effects.

For the interfaces, the choreography and orchestration are two special concepts only appearing in WSMO model. These definitions will be in detail presented in Section V.C. As described in Fig. 2, the PA consequences propagation mechanism can be covered by orchestration in WSMO model.

- *Societal Entities* act as clients (e.g., citizen, business), that present their goals by providing service *Needs*. Therefore, we map PA goal into WSMO goal.
- WSMO-PA includes the WSMO nonFunctionalProperties and WSMO mediators, as they play an important role in supporting a dynamic operations such as discovery, composition, selection, invocation of PA services.

5 Illustrating Example — Issuing Driving Licenses

In order to illustrate the WSMO-PA on a real-world case scenario, we describe a use case for issuing of driving licenses in Greece. In this use case we assume that the Ministry of Transportation Public Authority (MTA) is a service provider.

A citizen as an applicant for a driving license should submit all required certifications or documents, which can prove that he has the qualifications to obtain a driving license, e.g., a certification of having passed a driving test, having good eyesight, etc. Other authorities need to cooperate during the process, for example, driving schools having carried out driving tests. In this scenario, we assume that three auxiliary services are needed, i.e., by institutions providing driving tests, health tests, and by an authority registering residence. This use case will be the running example in this paper.

6 Formalizing of PA Services using WSMO

For the purpose of formalizing of the PA service model using a WSMO model, we define WSMO-PA ontology, WSMO-PA web service, WSMO-PA goal, and WSMO-PA mediators. Examples used for purposes of illustration are given in the WSML language [2]. The following formulation of WSMO-PA concept entities in WSML in this paper also consists of both conceptual syntax and logical expression syntax [2].

6.1 GEA Ontology Model

The GEA object model in Fig. 1 was shown in an E-R diagram. For simplicity, the model was not shown in its entirety. It is obvious that such a model can be implemented using a relational database. Such an approach would be complex since it would not exploit the advantages of declarative knowledge representation. The main requirement today is to be able to share information through the web for both humans and machines. Thus, we express the GEA model in the WSML ontology language and using WSMO model. WSML was the obvious choice since we decided to use the WSMO framework for modeling PA services. For the WSML implementation we have chosen the WSML-Flight language, which we has the expressiveness required for modeling of PA domain. The GEA ontology has been created using Web Services Modeling Toolkit (WSMT) [12].

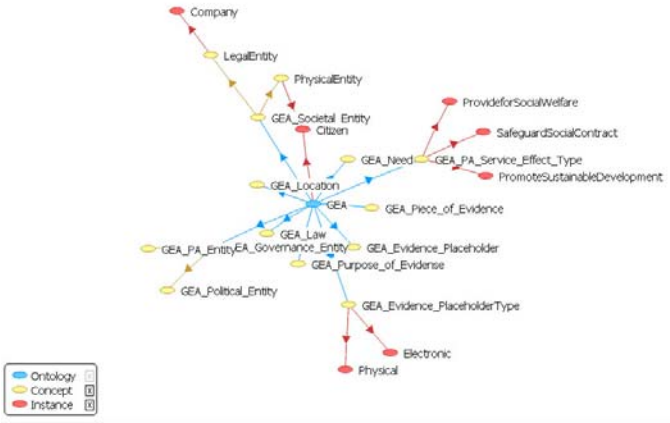

Figure 3. GEA model concept hierarchy

A fragment of the GEA model class hierarchy in WSML is shown in Fig. 3. Our work was based on a recent ontology implementation [15] of the GEA model. An important concept for GEA is EvidencePlaceholder. Every document required or produced by the PA is a $GEA\_EvidencePlaceholder$ and every EvidencePlaceholder can be Electronic or Physical (instances of $GEA\_EvidencePlaceholderType$ concept). For example the Passport photocopy is modeled as a subconcept of $GEA\_EvidencePlaceholder$. This subconcept has attributes of Passport photocopy (e.g. Name, Birthplace and PassportNo). The owner of the passport is modeled as the instance of this concept. Examples of $GEA\_EvidencePlaceholder$ subconcepts required for a driving licence are shown in Fig. 4.

6.2 GEA Ontology Definition

We follow the WSMO specification [1] to define the GEA Ontology. It generally provides a standard approach for defining any PA Ontology. The following description

Figure 4. $GEA\_EvidencePlaceholder$ **concept tree**

contains the definitions of *nonFunctionalProperties*, *importsOntology*, *concept*, *relation*, *instance* to *axiom* with the example in WSML[2].

- **NonFunctionalProperties** are mainly used to describe non-functional aspects such as creator, creation date, natural language descriptions etc. The non-functional properties used to describe the Driving License Service are shown bellow:

```
namespace _"http://www.semantic-gov.org/services/DrivingLicense"
  dc _"http://purl.org/dc/elements/1.1# "
  wsml _"http://www.wsmo.org/wsml/wsml-syntax#"
  gk _"http://www.semantic-gov.org/ontologies/GeneralKnowledge#"
  dlo _"http://www.semantic-gov.org/Ontologies/DrivingLicense_ Ontology#"
  gea _"http://www.semantic-gov.org/Ontologies/GEA#"
  foat _"http://www.semantic-gov.org/Ontologies/GEA#"
  webService _"http://www.examples.org/DrivingLicense_ WebService"
   nonFunctionalProperties
     dc#contributor hasValue "Sotiris Goudos"
     dc#publisher hasValue "CERTH"
     wsml#version hasValue "0.1"
     dc#date hasValue "19/04/06"
  endNonFunctionalProperties
```

- **Concepts** constitute the basic elements of the agreed terminology (e.g. *driver's license*) for the specific application domain (e.g. *Driving License*). To define a concept, one should specify its attributes (with attribute name and type), instances, and superconcept. For example, concept *DrivingLicense*, a type of EvidencePlaceholder, is defined as follows:

```
concept DrivingLicense subConceptOf GEA_Evidence_PlaceHolder
    ownerName ofType foaf#name
    ownerAddress ofType (1 1) gk#Address
    ownerBirthDate ofType (1 1) _date
    ownerBirthPlace ofType (1 1) gk#Location
    issuingPlace ofType (1 1) gk#Location
    issuingDate ofType (1 1) _date
    issuingAuthority ofType (1 1) LegalEntity
    expirationDate ofType (1 1) _date
    driverLicenseUniqueId ofType (1 1) _string
    category ofType (1 ) Category
    notes ofType (1 ) _string
    isValid ofType (1 1) _boolean
```

Note that for clarity we explicitly listed some inherited attributes from its superconcept, i.e. *ownerName*, *ownerAddress*.

- **Relations** are used in order to model interdependencies between several concepts (respectively instances of concepts). For example, if a driver's license is issued by an official government authority, named Ministry of Transportation and Communications, such relation can be described as follows:

```
relation drivingLicenseIssuance
   (ofType issunanceAuthority, ofType applicant,
   ofType applicant, ofType evidencePlaceholder)
   subRelationOf Issuance
```

- **Instance** is a specific concept with concrete values for each attribute. Instances are either defined explicitly or by a link to an instance store, i.e., an external database of instances and their values [1]. An example of the instance of concept *DrivingLicense* is defined as follows:

```
instance Mary memberOf DriverLicense
    driverLicenseUniqueId hasValue "B072RRE2I52"
```

- **Axiom** is considered to be a logical expression together with its non-functional properties, which are widely used in the definition part of other WSMO elements. For example, the expression of a driving license which is issued by MTA is as follows:

```
axiom DrivingLicenseIssuingAuthority
 definedBy
   ?x memberOf DrivingLicense
      implies ?x[issuingAuthority hasValue MTA].
```

Where x is a free variable being member of driving license, and its issuing authority has a value as MTA.
In addition, there are three common concepts used by other WSMO elements. In order to avoid repeated explanations in the rest of this paper, we give following summary:

- **NonFunctionalProperties** are allowed in the definition of all WSMO elements.
- **importOntology** In order to avoid an iterative definition of ontology or to make full use of reusable ontologies, importing an ontology is an optimal approach and widely used in WSMO specification.
- **usesMediator** is necessary for resolving mismatches between ontologies, web services and goals. When there is an interoperation problem between WSMO entities, mediator is used to resolve the problem.

6.3 WSMO-PA Web Service

The *web service* element of WSMO provides a conceptual model for describing a web service in a unified way, including its non-functional properties, capability, and interface.

Based on the three roles of the public administration entity (i.e., service provider, consequence receiver, and evidence provider in Fig. 1), we define three kinds of services, which are provided by PA services. To describe a web service, its capabilities and interfaces need to be specified. The following presents the details of WSMO-PA web service.

- **Capability** The functionalities offered by a given web service are described by its capabilities. In the WSMO specification [1], class *capability* has several attributes, which are *hasNonFunctionalProperty*, *importsOntology*, *usesMediator*, *hasSharedVariables*, *hasPrecondition*, *hasAssumption*, *hasPostcondition*, and *hasEffect*. We will present the detailed definitions of the last five attributes.
  - **Shared Variables** are the variables used across preconditions, postconditions, assumptions and effects.

    If $?v_1, ?v_2, ..., ?v_n$ are the shared variables defined in a capability, then it holds that [1]:

    forAll $?v_1, ?v_2, ..., ?v_n$
    (pre($?v_1, ?v_2, ..., ?v_n$) and ass($?v_1, ?v_2, ..., ?v_n$)
    implies post($?v_1, ?v_2, ..., ?v_n$) and eff($?v_1, ?v_2, ..., ?v_n$))

    In the example of the driving license web service, two variables are used:
    $shareVariables\{?applicantSignedDoc, ?license\}$
  - **Preconditions** specify the required states of the information space before web service execution. In the driving license use case, one of the preconditions is defined as follows:

    precondition checkPassDriveTests
    definedBy
    ?applicantSignedDoc[signedby **hasValue** ?applicant]
    memberOf dlo#DrivingLicenseLegallySignedDocument
    and ?applicant[hasPassedDriveTest **hasValue** _boolean("true")]
    and ?applicant[hasPassedDriveTheoryExam **hasValue** _boolean("true")]
    and ?applicant[hasGoodHealth **hasValue** _boolean("true")].

    The above condition says that If an applicant applies for a driving license, he/she should submit a legally signed application, and should prove that he/she has passed driving test, drive theory exam, and pass a health test with "good" result.
  - **Assumptions** describe the state of the world which is assumed before the execution of a web service. Otherwise, the successful provision of the web service is not guaranteed. The difference to preconditions is that assumptions are not necessarily checked by the web service.
    For example, we assume that a driving license applicant is alive, which does not need to be checked by the web service before its execution. The definition looks like:

    assumption DrivingLicenseApplication
    definedBy
    ?applicant **memberOf** dlo#Driver
    and ?applicant[isAlive **hasValue** _boolean("true")].
  - **PostConditions** describe the state of the information space that is guaranteed to be reached after successful execution of a web service. For example, the following postcondition assures that the applicant will own driving license, after the invocation of the service issued in Greece:

    postcondition ProduceDrivingLicense
    definedBy
    exists ?x(?applicant[hasDrivingLicense **hasValue** ?x]
    and ?x **memberOf** dlo#DrivingLicense
    and ?applicant[hasName **hasValue** ?name]
    and ?x[ownerSurname **hasValue** ?name]
    and ?applicant[hasBirthdate **hasValue** ?bd]
    and ?x[ownerBirthDate **hasValue** ?bd]
    and ?x[issuingPlace **hasValue** ?place]
    and ?place[country **hasValue** "Greece"]).

– **Effects** describe that the state of the world is guaranteed to be reached after successful execution of a web service. For example, after the service execution the applicant becomes a driver, that is, he/she has the right to drive a car.

effect BecomingADriver
definedBy
?applicant **memberOf** dlo#Driver

The **Intuitive Semantics** of a WSMO-PA web service is, given the state s_0 before executing the web service and $do(service, s_0)$ the state after executing it, that the following relation holds between these formulas (cited from [1]):

$$\forall ?x_1, ?x_2, ..., ?x_n \; holds(\mathcal{PRE}(?x_1, ?x_2, ..., ?x_n) \wedge \mathcal{ASS}(?x_1, ?x_2, ..., ?x_n), s_0) \rightarrow holds(\mathcal{POST}(?x_1, ?x_2, ..., ?x_n) \wedge \mathcal{EFF}(?x_1, ?x_2, ..., ?x_n), do(service, s_0))$$

- **Interface** A web service interface is primarily meant for purposes of behavioral description, and is presented in a way suitable for software agents to determine the behavior of the web service and to reason about it. It might also be useful for discovery and selection purposes, and in its description the connection to some existing web service interface descriptions, e.g., WSDL, could also be specified.

  There are two kinds of interfaces provided for final users or other web services, respectively, which are:

  – **Choreography** defines how to communicate with a web service in order to consume its functionality, and concerns the interactions of services with their users. Basically, it includes the attributes *hasNonFunctionalProperty*, *hasStateSignature,* and *hasTransitionRules* [1].

    In WSMO, choreography description is based on the Abstract State Machine (ASM) mechanism[11]. The feature *stateSignature* is further defined by *hasNonFunctionalProperties*, *importsOntology*, *hasIn*, *hasOut*, *hasShared*, *hasStatic*, and *hasControlled*. A simplified example of the driving license web service choreography is presented as follows:

choreography WSDrivingLicenseChoreography
stateSignature WSDrivingLicenseStatesignature
importsOntology
"http://www.semantic-gov.org/Ontologies/GEA#"
in
concept gea#EyeHealthCertificate
concept gea#IDCardPhotoCopy
concept gea#PublicPaymentVoucher
concept gea#PassportPhotoCopy
out
concept gea#DrivingLicense

Please note, that transitions rules are omitted here both for brevity and space reasons.

– **Orchestration** defines how the overall functionality is achieved in terms of the cooperation of other service providers with the actual implementation [14]. For example, the driving license issuance service involves the use of at least three additional web services to validate the issued information by the applicant, checking if a driving test was passed, and checking the health status.
Generally, for a WSMO-PA web service in the e-government domain there are three cases of orchestrations with different functions:
  * **Verification** is an interface, which deals with the verification of the correctness of the preconditions and the guarantee of assumptions before the execution of a web service.
  * **Execution** relates to the choreography of other services, which defines goal information. The execution interface has the responsibility to implement a service by taking inputs and yielding outputs.
  * **Consequence** relates to other services, with which there are business connections. This interface uses the consequences of a PA service execution to update its information space.
    As an example, if the driving license application is approved, other public service records of the license applicant may be notified to update their databases by changing the status of *hasDrivingLicense* to 'yes'.

  More detailed definitions will be the subject of our future work.

6.4 WSMO-PA Goal

In WSMO, Goals are used to describe user's desires. Therefore, a GEA goal is mapped to a WSMO goal, which formally expresses the client's requirements.

The *Need* concept in PA services does not have a direct correspondence in WSMO. It is an informal description of a goal, as experienced from a client's perspective. WSMO assumes that informally expressed requirements will be transformed to formal goals prior to their processing in a WSMO environment. This assumption is, however, beyond the WSMO model's scope. It must be pointed out that both models support two views, a clients's and a service's view. Other service ontologies like OWL-S do not consider a client's perspective. This is one of the facts that have led to the selection of the WSMO framework for the implementation of the semantic PA web services.

Goals used in WSMO provide means to specify the requester side's objectives when consulting a web service, describing at a high level a concrete task to be achieved [1]. Similarly, its definition consists of *hasNonFunctionalProperty*, *importsOntology*, *usesMediator* (e.g., ooMediator, ggMediator), *requestsCapability*, and *requestsInterface*. The following example is a goal for obtaining a driving license, which requires an applicant to issue all of the required documents and yields a driving license for the applicant.

```
goal ObtainingDrivingLicenseGoal
 capability
 shareVariables {?applicant, ?license}
  postcondition NeedingADocument
   definedBy
    exists ?x(?applicant[hasDrivingLicense hasValue ?x]
    and ?x memberOf dlo#DrivingLicense
    and ?applicant[hasName hasValue ?name]
    and ?x[ownerName hasValue ?name]).
  effect NeedToBecomeADriver
   definedBy
    ?applicant memberOf dlo#Driver.
```

It is worth noting that the goal definition has the same structure as the service. This fact is prescribed by WSMO in order to enable intelligent, automatic discovery techniques based on goal and service descriptions. In our example, since no preconditions are specified in the goal we assume that the requester is willing to fulfill (adjust to) any preconditions required by the service.

6.5 WSMO-PA Mediator

Mediator is another concept which does not exist in the GEA model. This is due to the fact that mediators solve interoperability issues and do not have conceptual "meaning" from the PA and GEA perspectives. Depending on the resources to be mediated, [1] defines four types of mediators:

- *ooMediator*, resolving mismatches between ontologies and providing mediated domain knowledge specifications to the target component;
- *ggMediator*, allowing the creation of a new goal from existing goals and defining goal ontologies;
- *wgMediator*, linking a *Web service* to a *goal*, resolving terminology mismatches and states the functional difference between both;
- *wwMediator*, establishing interoperability between *Web services* that are not interoperable a priori.

The following is the definition of WSMO mediator [8].

```
Class mediator
   importsOntology type ontology
   hasSource type {ontology,goal,service,mediator}
   hasTarget type {ontology,goal,service,mediator}
   hasMediatorService type {goal, service,wwMediator}
```

6.5.1 Example of using mediators in WSMO-PA

Considering the driver's license use case the following WSMO-PA resources are defined:

- Five domain ontologies, namely, a citizen ontology for the driving license applicants $\mathcal{O}_{ci}$, a driving license ontology $\mathcal{O}_{dl}$, a health certification ontology $\mathcal{O}_{hc}$, a driving test ontology $\mathcal{O}_{dt}$, and a registered permanent residence ontology $\mathcal{O}_{rp}$.
- Goal $\mathcal{G}_0$ specifying "apply for a driving license" is used as a template to define a concrete goal $\mathcal{G}_M$ that states "Mary would like to apply for a driving license".
- Web service $\mathcal{WS}_{MTA}$ offered by the issuing authority MTA, which provides the services of examining and issuing driving license to a citizen applicant. This service also uses other services to validate the information of application. The needed services are web service $\mathcal{WS}_{CT}$ providing driving test, web service $\mathcal{WS}_{HT}$ providing health test, and web service $\mathcal{WS}_{RP}$ providing registered residence.

Fig. 5 illustrates the connections between the resources of WSMO-PA. In this figure, full arrows to a mediator denote the sources (whereas full arrows to a goal or a web service denote ontology import) and dashed arrows the targets of the mediators used.

There are five domain ontologies; $\mathcal{O}_{ci}$ and $\mathcal{O}_{dl}$ are imported by goal $\mathcal{G}_0$ and web service $\mathcal{WS}_{MTA}$; $\mathcal{O}_{rp}$, $\mathcal{O}_{ht}$, and $\mathcal{O}_{dt}$ are used by web services $\mathcal{WS}_{RP}$, $\mathcal{WS}_{HT}$ and $\mathcal{WS}_{DT}$, respectively. The *ooMediator* is needed between $\mathcal{O}_{ci}$, $\mathcal{O}_{rp}$, $\mathcal{O}_{ht}$ and $\mathcal{O}_{dt}$ to resolve the possible mismatches between them. It is also used within the *wwMediator* $\mathcal{WWM}$ that

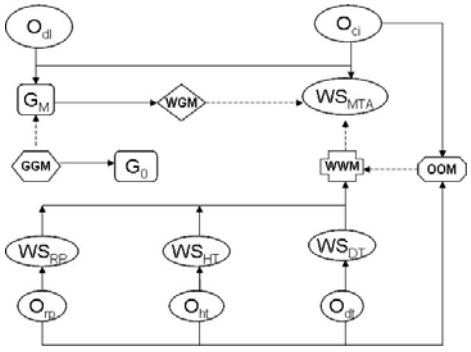

Figure 5. WSMO-PA mediators — usage example

connects web services and, in addition, resolves possibly occurring mismatches on the protocol or process levels.

In this use case, the MTA authority may provide several issuance services besides issuing driving licenses, while $\mathcal{G}_M$ just requests a Driving license. Therefore, there is a *wgMediator* $\mathcal{WGM}$ connecting $\mathcal{G}_M$ and $\mathcal{WS}_{MTA}$ for terminological mismatches and stating the functional difference between them. Also, a *ggMediator* $\mathcal{GGM}$ defines that $\mathcal{G}_M$ is an refinement of $\mathcal{G}_0$ by inheriting all description notions and refining the object specification to a concrete driving license application required by Mary. Also, the *wgMediator* $\mathcal{WGM}$ holds between $\mathcal{WS}_{MTA}$ and $\mathcal{G}_M$, as it is inherited from $\mathcal{G}_0$.

7 Conclusion and Future Work

Modeling of PA domain and development of information and service models for PA is promising to enable interoperability and interoperation of PA systems. The PA domain has specific features including high degree of formality of areas such as law, high demands in security, trust and privacy, typical and sometimes extremely long-running processes (e.g. urban planning), many different stakeholders in the same process (e.g. citizen vs. municipality, county council, federal government), etc. These features are of interest to the SW research since the PA domain can provide an ideal test bed for the SW research, and SW technologies can be an ideal platform to achieve the vision of a knowledge-based, citizen-centric, and citizen-empowering, distributed and integrated e-Government. The first step towards this vision is to develop formal PA service models as well as PA information models to be used as underlying formalisms in SW and SWS environments. In this paper we have shown how a generic PA service model can be described by means of WSMO - a formal model describing Semantic Web Services.

This framework will be used as a test-bed for applying WSMO to a demanding domain with a lot of special features and characteristics. In our future work, we plan to apply and further develop existing Web Service Modeling eXecution environment(WSMX) as an environment for handling operations on semantic web services, WSMO Studio and Web Services Modeling Toolkit (WSMT) as modeling tools, for the e-government domain. In particular, in our SemanticGov research project, WSMX will serve as an execution environment for PA service provisioning in national and pan-European e-government contexts. The described PA service model using WSMO will be the underlying specification for PA service provisioning built on WSMX.

Acknowledgment

This material is based upon works supported by the Science Foundation Ireland under Grant No. SFI/02/CE1/I131 and the EU funded SemanticGov project (FP6-2004-IST-4-027517).

References

[1] D. Roman, H. Lausen, and U. Keller (eds.). Web Service Modeling Ontology (WSMO), *Applied Ontology*, 1(1):77-106, 2005.

[2] H. Lausen, J. de Bruijn, A. Polleres, and D. Fensel, WSML - a Language Framework for Semantic Web Services, In Proceedings of the W3C Workshop on Rule Languages for In-teroperability, Washington DC, USA, April 2005.

[3] A. Haller, E. Cimpian, A. Mocan, E. Oren, and C. Bussler. WSMX - A Semantic Service-Oriented Architecture. International Conference on Web Services (ICWS 2005), July 2005.

[4] D. Martin, M. Burstein, D. McDermott, S. McIlraith, M. Paolucci, K. Sycara, D. McGuinness, E. Sirin, and N. Srinivasan. Bringing semantics to web services with owl-s. World Wide Web Journal Special Issue on Web Services: Theory and Practice, 2006. To Appear.

[5] R. Akkiraju, J. Farrell, J.Miller, M. Nagarajan, M. Schmidt, A. Sheth and K. Verma, Web Service Semantics - WSDL-S, A joint UGA-IBM Technical Note, version 1.0, April 18, 2005.

[6] A. Patil, S. Oundhakar, A. Sheth, K. Verma, METEOR-S Web service Annotation Framework, The Proceedings of the Thirteenth International World Wide Web Conference, May, 2004 (WWW2004), pp. 553-562.

[7] E.Motta, J.Domingue, L.Cabral and M.Gaspari. IRS-II: A framework and infrastructure for semantic web services. 2nd International Semantic Web Conference (ISWC2003), Springer Verlag.

[8] A. Mocan (ed.), D29v0.2 WSMO Mediators. WSMO Working Draft 21 December, 2005, http://www.wsmo.org/TR/d29/v0.2/.

[9] V. Peristeras, A. Mocan, T. Vitvar, S. Nazir, S. Goudos and K.Tarabanis, Towards Semantic Web Services for Public Administration based on the Web Service Modeling Ontology (WSMO) and the Governance Enterprise Architecture (GEA). eGOV International Conference, DEXA 2006, Krakow, Poland.

[10] V. Peristeras and K. Tarabanis, Reengineering the public administration modus operandi through the use of reference domain models and Semantic Web Service technologies. in 2006 AAAI Spring Symposium, The Semantic Web meets eGovernment (SWEG), Stanford University, California, USA, 2006.

[11] Y.Curevich, Evolving Algebras 1993: lipari Guide, Specification and Validation Methods, ed.E.Borger, Oxford University Press, pp.9-36, 1995.

[12] M. Kerrigan, WSMOViz: An Ontology Visualization Approach for WSMO. 10th International Conference on Information Visualization, 2006.

[13] The Object Management Group: Meta-Object Facility, version 1.4, 2002.

[14] J.Scicluna, A. Polleres, D.Roman and D.Fensel: D14v0.2. Ontology-based Choreography and Orchetration of WSMO Service. WSMO Working Draft 13,January 2006.

[15] S. K. Goudos, V. Peristeras, K. Tarabanis, Mapping Citizen Profiles to Public Administration Services Using Ontology Implementations of the Governance Enterprise Architecture (GEA) models, Accepted for presentation in 3rd Annual European Semantic Web Conference, June 2006, Budva, Montenegro.

[16] D. Apostolou, et al., Towards a Semantically-Driven Software Engineering Environment for eGovernment, LNAI. p.157-168, 2005.

[17] A. Grunlund. What's In a Field - Exploring the eGoverment Domain. in 38th Hawaii International Conference on System Sciences, 2005.

[18] Brahim Medjahed, et al., Infrastructure for E-Government Web Services. IEEE Internet Computing, 7(1):58-65, 2003.

[19] A. Polleres, R. Lara (editors): A Conceptual Comparison between WSMO and OWL-S, WSMO Working Group working draft, 2005, available at http://www.wsmo.org/2004/d4/d4.1/v0.1/.

[20] B.Medjahed, A. Bouguettaya and M. Ouzzani. Semantic Web Enabled E-Government Services. ACM International Conference Proceeding Series, Vol. 130, pp.1-4, 2003.