ABSTRACT

In this paper we investigate how the use of semantic Web Services can facilitate provision of e-government applications by employing the DAML framework. Using DAML+OIL for constructing domain ontologies at the content level and DAML-S ontology for describing services available on the Web, appear as an effective way to invoke, execute and monitor composite Web Services. Building on top of our recent work, which focused on configuring the “new company registration” as a Web Service, we investigate the applicability and the extra functionality gained by the employment of a semantic approach.
1. PROBLEM DEFINITION

Public administrations (PAs) produce and distribute administrative services in order to serve their customers. These services are sometimes simple but usually are fragments of a larger whole. We call the latter a composite service (or macro-service), the execution of which provides the requested output to the end customer. A typical composite service consists of a number of other constituent services that are typically executed by administration agencies other than the one responsible for the composite service.

The notion of a composite service gains particular importance in public administration as citizens usually have to deal with a large number of administrative actors and layers in order to receive the composite service they request. In addition, most such information exchanges currently require hard-copy documents sent through ordinary mail or even worse, by having the citizen serve as a messenger, providing the information needed to establish communications between administrations.

As an example which is studied in this paper, setting up a company demands a series of constituent services to be performed by numerous administrative agencies (e.g. tax and legal related certifications, registration in several bodies, land-planning permissions, etc). All these services are intermediate steps of no particular value to the citizen and thus should be dealt with as transparently as possible within the public administration system as a whole.

Achieving this goal is difficult since public administration is characterized by fragmentation of responsibilities, different organizational and IT infrastructure and architectures, the absence of cooperation among different administrations and most importantly the lack of a common language (definitions and vocabulary). The above result in the inefficient execution and provision of such composite services (or macro-services).

E-government initiatives that are underway need to deal with these issues. As first steps towards this direction PA agencies establish Web sites in order to (European Commission, 2002):

- provide information (stage 1)
- download forms (stage 2)
- allow submission of forms online (stage 3)
- complete transactions (stage 4)

However, the above stages focus on a constituent, simple transaction and the above characterization does not take into account the integration requirements for the provision of a complicated transaction such as a composite service. Thus, a new entrepreneur will still have a hard time starting-up a business, even if all related PA agencies have a fourth stage website deployed. Amongst other things the entrepreneur would most probably have to:

- discover all agencies and units related to the composite service of interest (this discovery phase is twofold: to discover the agencies/units/subunits that institutionally participate in the process and then discover their websites),
- gather the inputs and verify the prerequisites needed for each constituent service to be executed locally at each respective website of the participating PA agency,
- execute all constituent services,
- collect all the outputs from the constituent services to formulate the required input for the composite service (ideally electronically), and
- coordinate execution of the composite service providing additional information or even executing additional constituent services in exceptional cases, and
monitor the whole process in order to be aware of inconsistencies and failures. Thus, e-government efforts need not only to complete transactions corresponding to simple services, but also to achieve the interoperability and cooperation amongst public administration agencies that is required in order to offer to their customers composite services in a transparent manner.

2. RECENT TRENDS AND STATE-OF-THE-ART

During the last several years the importance of achieving interoperability among PA information systems has become apparent. Several initiatives in various countries focused on improving communication of public administration systems. The Federal Enterprise Architecture Framework (FEAF) by the Federal Chief Information Officers Council in the USA (CIO Council, 1999), the Command, Control, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR) Architecture Framework by the U.S. Department of Defense (C4ISR Integration Task Force, 1997), the Unitary Network of the Italian Public Administration by the Autorità per l’Informatica nella Pubblica Amministrazione (AIPA) (Batini and Mecella, 2001) and more recently the e-Government Interoperability Framework by the Office of e-Envoy in UK (Office of e-Envoy, 2002) are some of these initiatives.

While the framework of enterprise architectures constitutes the basis for the two first initiatives, the latter build heavily on the Web Services paradigm and the employment of XML schemas for creating standardized structural descriptions of administrative documents. The differences in the two approaches -Enterprise Architecture versus Web Services- have been studied in the literature. While the Enterprise Architecture approach urges administrators to create an architected blueprint for the whole of the enterprise (public administration in our case) thus supporting a centralized management of informational planning, the Web Services model advocates a decentralized development process backed up by commonly agreed standards (Peristeras, Tsekos and Tarabanis, 2002). More specifically, the differences are three-fold (The Stencil Group, 2002)

- Web Services are designed to allow for smaller, modular functionality that can be assembled and reassembled into dynamic processes, while Enterprise Architecture solutions tend to be employed in order to link existing, monolithic applications into a common infrastructure,
- Web Services enable open-ended, one-to-many connections, while most Enterprise Architecture technologies are designed to form discrete, pre-specified connections,
- Web Services can be deployed with incremental cost and effort while Enterprise Architecture solutions tend to be "all or nothing" models that require a significant commitment of strategy and resources.

While at first glance the two approaches appear quite contrary, when studied closer they can be found complementary. For example, the Web Services approach requires commonly agreed standards. Such standards would typically be defined as part of the models built in the Enterprise Architecture approach.

The Web Services paradigm has gained a great deal of advocates recently and is being considered as the state-of-the-art perspective when dealing with e-government issues. For example, in addition to the above-mentioned e-government initiatives, a number of e-government R&D IST funded projects at the European level are working in this direction (e-Gov, InfoCitizen, EU-Public.com, etc). Web Services can be employed in a variety of cases already familiar to human users: querying of databases, catalogs, digital libraries, and other types of information repositories- searches and classification services provided by portals- business-to-consumer (B2C) transactions and business-to-business (B2B) transactions, etc. It is worth noting that Web Services are not limited to the two-party, client-server approach most commonly used. Web Services can involve any number of parties, with complex patterns of interaction amongst them. Thus Web Services are suitable for the public administration macroprocess discussed in the previous section or the typical business-to-business transaction that may involve a buyer, a seller, a financer, and a shipper. The requirement of service transparency to the
user is another reason for the architectural choice of Web Services for public administration service provision.

However, passing from a Web containing static information to a Web that will enable all the services required, for example, by the new entrepreneur of the previous section wanting to start-up a new business in a transparent and integrated manner has proven to be a difficult task. Recent efforts around XML, XML schemas, RDF, RDF schemas, UDDI, SOAP and WSDL (W3C), have focused on realizing the afore-mentioned. Although these technologies support simple service discovery and configuration, they cannot effectively address highly complex services with extended workflows. However, the majority of the public administration composite services belong to this latter type of services. The technologies mentioned above are considered to lie at the syntactic layer. That is, this layer provides means to define vocabulary, structure and constraints for expressing metadata about Web resources, however, formal semantics for the participating objects, processes and other primitives are not provided. Such semantics can help in order to execute composite services on the Web. For instance, a case-based orchestration of a composite service can be facilitated by the use of semantics related to a complex process workflow model. For this purpose, an additional layer on top of the RDF Schema has been proposed. Tim Berners-Lee (Berners-Lee, Hendler and Lassila, 2001) has called this layered architecture the Semantic Web in order to express this new generation of the Web. The third layer adds semantics to Web resources, thus allowing the provision of more complex services (e.g. automatic service identification, composition and execution by machines).

![Figure 1: The languages stack in the Semantic Web (W3C).](image)

In order to provide Semantic Web Services, current approaches employ a new type of mark-up that is backed-up with a vocabulary for representing and communicating knowledge about some topic and the set of relationships that hold among the terms in that vocabulary. The latter constitutes an ontology and its main function is to explicitly represent specifications of a business domain, public administration in our case.

A proposed formalism in this respect is the Ontology Interchange Language (OIL) and the DARPA Agent Markup Language (DAML) (Figure 1). These two have been brought together to form DAML+OIL (W3C, 2001a), a language now being proposed as a W3C standard for ontological and metadata representation. The tasks that DAML is expected to facilitate are: automatic Web service discovery, automatic Web service invocation, automatic Web service composition and inter-operation, and automatic Web service execution monitoring. These tasks coincide to a large degree with the tasks required for public administration macroprocess execution as described in Section 1.

For the explicit description of Web Services and their semantic mark-up, a DAML-based formalism has been recently proposed: DAML-S (The DAML Services Coalition, 2002). DAML-S provides the ontology structuring mechanisms within which to have Web sites declare and describe their services. That is, DAML-S proposes an upper ontology for services. The model that is employed provides three essential types of knowledge about a service (Fig. 2):

- Who does provide the service? The answer to this question is given in the **Resource** class of DAML-S.
What does the service require of the user(s), or other agents, and provide for them? The answer to this question is given in the “profile”. DAML-S provides the class ServiceProfile which presents what a Service does.

How does it work? The answer to this question is given in the “model.” DAML-S provides the class ServiceModel, which describes how the class Service works.

How is it used? The answer to this question is given in the “grounding.” DAML-S provides the class ServiceGrounding, which supports access and invocation of the class Service.

Figure 2: Top level of the service ontology (DARPA, 2002)

3. SEMANTICALLY ENHANCED E-GOVERNMENT WEB SERVICES BASED ON DAML

In order to realize semantically enhanced e-government services we have performed initial work in employing the DAML framework to the domain of public administration. While DAML-S provides an upper level service ontology, it requires an ontology, expressed in OIL, for the domain at hand—public administration in this case.

There generally appears to be a lack of content standards in the domain of public administration, not only at the international level from one administrative system to another, but even within a single country. Consequently, there is a need for a widely accepted public administration content ontology. Such a content ontology can be constructed in a top-down manner, for instance, based on an upper level model description of the domain. We are currently conducting some initial research work in this direction.

3.1. Adopting a content ontology

However, in order to address the need for an ontology of public administration concepts for the work described in this paper, we have employed the recent work conducted by e-Envoy in the UK, in particular the Government Common Information Model (GCIM) (Fig. 3). The part of the GCIM model highlighted in Figure 3 refers to the elements used in the showcase presented in this paper. For example Evidence is used to describe input and supporting information for a Service Interaction, which exists in advance of the time it is used. Outcome is information concerning the result of a Service Interaction. The GCIM model was extended in certain cases in order to address the needs of the particular scenario at hand. For example, the Outcome class of the GCIM model was specialized as that of Output(s) and Consequence(s) of a Service (Figure 4). Outputs are defined as the final “products” produced by the Service Interaction and received by the client who initiated the Service Interaction. Consequences are defined as all the by-products of the Service Interaction (e.g. resulting
in information that interests other Service Organization Subjects) (The InfoCitizen Project, 2002). These elements are defined in DAML+OIL syntax as depicted in Figure 5.

![Government Common Information Model (GCIM)](image)

**Figure 3: Government Common Information Model (GCIM)**

![Part of our PAconcepts Ontology](image)

**Figure 4: Part of our PAconcepts Ontology**

```xml
...<daml:Class rdf:ID="Outcome"></daml:Class>
<daml:Class rdf:ID="Evidence"></daml:Class>
<daml:Class rdf:ID="Consequence">  
    <daml:subClassOf rdf:resource="#Outcome"/>
</daml:Class>
<daml:Class rdf:ID="Output">  
    <daml:subClassOf rdf:resource="#Outcome"/>
</daml:Class>
...```

**Figure 5: Part of PAconcepts.daml ontology**

The service that we have worked on pertains to registration of a new company in Greece, denoted as CompanyRegistrationService (CRS). CRS takes as inputs The Company Name, Base-Address-Location, the Business-Type (see Company_Name_In in Figure 9 for an input example). Example outcomes (Fig. 3, 4, 5) of CRS are: a) a document confirming the Company’s Registration (this document contains the Tax Registration Identifier as an outcome TRS), and b) a Notification to the Greek Chamber of Commerce that a new company has been registered in Greece. These outcomes are respectively the output and effect of CRS in DAML-S terms, or the output and consequence of CRS in PAconcepts ontology terms (Fig.8).
CRS is a composite process according to DAML-S terminology, which comprises of a set of constituent services (atomic processes in DAML-S terms). An example atomic service is for instance that which uses Company_Name_In (Fig. 9) as an input and results in the output of the company being assigned a Tax Registration ID and entering the Greek Tax system, denoted as TaxRegistrationService (TRS).

The Company Registration scenario has been implemented in previous work (Tarabanis K. et al., 2002) as a plain Web service without semantic elements. Implementing a composite service in this manner called for the PA specific domain logic to be programmatically embedded and henceforth statically utilized. Trying to capture and maintain all the flow cases and patterns of a composite service in a software program calling simple Web Services has made our goals for e-Government seem unwieldy. In this paper the CRS has been enhanced with semantic elements from the DAML-S 0.7 release specification (DARPA, 2002). This solution offers the reconfigurability and adaptivity benefits stemming from documenting the services under consideration in a format easily understood by both humans and machines. This means that we can now describe a process model and orchestration in a declarative manner and capture the logic of a process (or service) execution in specific documents. These documents can be easily changed and stored by a human or even built dynamically by a machine thus supporting ever-changing and dynamic business processes (or macro-services or composite services).

3.2. Creating a DAML-S description for our service

At the first stage the Profile file provides the information needed for an agent to discover the service needed, namely the CRS in our case. After discovering the service, the Process and Grounding files provide adequate information for the agent on how to use the service, in order to perform its invocation, execution and monitoring. In DAML-S, both the Profile and the Process files are thought of as abstract representations; only the Grounding file deals with the concrete level of specification and can be considered as enabler for passing from simple Web Services to the DAML-S notions.

Following DAML-S, we built three files –parts of which are presented- for the Company Registration Service, namely the Grounding file (Fig. 7), the Profile file (Fig. 8), and the Process Model file (Fig. 9).

The Grounding file provides information on how an agent can access a service at a low communication level. Typically, the Grounding file specifies a communication protocol, message formats, and other service-specific details such as port numbers used in contacting the service. Here, we demonstrate how this is achieved through the use of WSDL files, showing how the (abstract) inputs and outputs of an atomic process of DAML-S are to be realized concretely as WSDL messages, which carry those inputs and outputs in some specific transmittable format (see Section 3.3). WSDL and UDDI registries can be considered as the key providing the necessary communication layer to realize abstract concepts and notions entailed in DAML-S files (particularly found in the Profile and Process files explained below).

The Profile file gives the types of information needed by a service-seeking agent (e.g. a public organization searching for a specific service) to determine whether a service meets the service requester’s needs. Among the various types of information this file offers (which are Provenance and Description, Functionality Description and Functional Attributes), we present those concerning ‘functionality description’, the later being the part of the overall Service Profile that has information about the service’s outputs and effects (see Section 3.4 Fig.8). We have also created but not depicted here other service parameters like MaxResponseTime denoting the maximum duration over which our CRS must be delivered.

The Process file describes what happens and what steps are followed when the service is carried out. For a composite service, like CRS, a service-seeking agent may use this description in various different ways (for instance, we showcase how a composite service (CRS in our case) can be executed as a composition of multiple atomic services (TRS)). TRS is an atomic service, part of CRS
composite service and follows a pattern of a sequential execution of CRS (see Section 3.5, Fig.9, where TRS is the second atomic service in this sequential execution pattern; the other atomic services in this sequence are CompanyBaseLocationVerificationService and InsuranceRegistrationService).

We start by investigating the Grounding file, due to our already mentioned previous work at the concrete level of service realization using WSDL (Fig.6).

### 3.3. Linking DAML-S with Web Services

DAML-S and its components at 0.7 version, have no means to express the binding information needed for executing a Web Service. For this purpose, we used WSDL. In general terms, a WSDL file residing in a UDDI registry, describes services in terms of where they are located and presents the inputs and outputs applicable for them.

In our previous work (Tarabanis K. et al., 2002), we had used the `CompanyRegistration_Service-interface.wsdl` file (part of which is shown in Figure 6), which simply provided the definition of the binding details (messages formats and access points) for the CRS without being able to express any semantics. To achieve the later, the classes of our PAconcepts ontology (Fig. 5) were linked with the WSDL file. For this, the DAML-S methodology uses the Grounding file (Fig. 7) as an interface between WSDL (Web Services) and the DAML descriptions.

![Figure 6: Part of TaxRegService-interface.wsdl](image)

As DAML-S is an XML-based language, and its atomic process declarations and input and output types already fit nicely with WSDL, it is easy to extend existing WSDL bindings for use with DAML-S, such as the SOAP binding. In Figure 6, we indicate briefly how the atomic TRS process, which is in turn a constituent part of the composite CRS, can be given a grounding using WSDL and SOAP in the CRS’s grounding file (Fig.7), with the assumption of HTTP as the chosen transport mechanism. The connection between the two documents is achieved by mapping DAML-S inputs (e.g. CompanyName_In) to the parts (e.g. `messageParameter_input_1`) of the input message (IngetTaxRegRequest) of our TRS WSDL operation (TaxRegistration_Service), and DAML-S outputs to the parts of the output message of our TRS WSDL operation (only inputs are shown in Figures 6 and 7).

![Figure 7: Part of CompanyRegistrationServiceGrounding.daml](image)

We emphasize that only an atomic process can have low-level implementation links/maps to a `<message>` and `<part>` name found in the corresponding grounding file and in the WSDL implementation (DARPA, 2002) (Fig. 6).

### 3.4. Showcasing the semantic features of DAML-S
a. Using the semantic characteristics of the Profile file output during service execution.

A part of the ServiceProfile is given in Figure 8. This file includes only general information about the provider of a service, the inputs and outputs this service provides and so on, and relates them to the Process file for more information on how all these are used (see Fig. 8 <refersTo> element in <profile:output>). This is the location where an agent can find, for instance, that the CRS composite service will generate a Tax Registration ID for the company as an output of the TRS it entails as an atomic service. This will of course occur upon successful completion of TRS. Even if the composite process of a Company’s Registration is not successfully executed as a whole, the atomic process of getting a tax registration identifier has the effect of the automatic notification of the Greek Chamber of Commerce for the registration of a new company in Greece (see effect element ChamberCommerceID of the Profile DAML file in Figure 8). This DAML construct (effect) has been related and restricted to (see restrictedTo construct of <profile:effect> in Fig. 8) the notion of the ‘Consequence’ of our PA concepts ontology (see Fig. 4-5). Additionally, DAML-S effect refers to (see refersTo construct of <profile:effect> in Fig. 8) to a ChamberCommerceNotification, which is an instance class of the Consequence Class of our PAconcepts Ontology. This way we make the DAML-S notion of effect concrete and denote that the effect of CRS will be the notification to the Greek Chamber of Commerce for the registration of the new company in Greece. This notification occurs transparently for the company invoking CRS. ChamberCommerceID refers to a specific ChamberCommerceNotification, which is in turn an instance of a Consequence of our PAconcepts ontology. Thus, we cannot have as an effect of CRS a BankAccount, if BankAccount has not been declared as an instance of the Consequence class in the PAconcepts ontology.

Thus, we have demonstrated how we can relate DAML-S elements to elements of domain specific ontologies expressed in DAML+OIL syntax, in our case PA ontologies.

![Figure 8: Part of CompanyRegistrationServiceProfile.daml](image)

b. Linking WSDL inputs with Process file inputs

Now that the WSDL file has been linked with the DAML-S constructs via the Grounding file (section 3.3), we can proceed to utilize the knowledge in the Process file (Fig. 9). In this file, sequence of atomic processes –including TRS- of a composite process, inputs, outputs and their domain and range and other elements for the service under concern are denoted. As an example, the <daml-property> attribute (Fig. 6) embedded in the WSDL file facilitates the linkage between Process file and WSDL file.

Linking the Process file with the WSDL file as in our example makes possible for an agent to understand that the CRS service demands company name as an input. Therefore, a relationship is created between the <part> subElement of the WSDL’s Message Element (which describes the input or the output of the Service in hand) and the semantic meaning of it (which is located in the input
property <CompanyName_In> of the DAML-S process file for CRS) (Fig. 9). Examples of the semantics we can use are elaborated in the next section.

c. Using the semantic characteristics of the Process file input during service execution.

Using the <CompanyName_In> property and its sub-properties of the Process file we can address questions such as: how is the macro-service offering of company registration in Greece used and invoked (e.g. through the execution of the atomic processes it consists of). The <daml:range> sub-property of the <CompanyName_In> property, in the Process file, defines the allowable values for this input. These values have been defined in an DAML instance file (not presented in this paper), as instances of the Evidence class (Fig. 5) of our ontology for PA related concepts presented above. For example, if Evidence types are restricted only to “Passports” and this fact is declared in the PA related concepts ontology (evidence=passports), then allowable inputs for <CompanyName_In> can only be Passports. Moreover, using the other sub-property of the <CompanyName_In> property, the <daml:domain> element relates the <CompanyName_In> property with the defined class TaxRegistrationService (Figure 8, rdf:resource="#TaxRegistrationService"). Through this relationship the <CompanyName_In> is used as input to the TRS atomic service, which is in turn a constituent part of the composite CRS service (Fig .9). Thus, it is transparent to the requestor of the composite service CRS that the input CompanyName_In provided, is used by one (or could be more than one) atomic services. This fulfills the requirements of e-Government concerning a) minimization of input and re-input from the requestor of the service and b) provision of the service in an as transparent manner as possible.

Combining the two aforementioned sub-properties can provide the necessary knowledge to an agent to automatically “understand” that in order for a Company to be able to register in Greece, a TRS service has to be accomplished and a Passport is needed as an input for TRS execution.
4. Conclusion

In this paper we describe the use of semantic Web services for e-government applications by employing the DAML framework. More specifically, we have investigated how a Web service can be semantically enhanced with the aid of: a) a DAML+OIL concepts ontology encompassing extended models like GCIM, along with b) DAML-S ontology for describing services available on the Web. At an initial stage, we have found extended functionality and enhanced intelligence appearing to be an effective way to invoke, execute and monitor composite Web Services. DAML-S and Web Services offer technologies for realizing complex services provided over the Web and demanding specific semantics knowledge. The latter is provided by the ability of DAML-S files to reference particular domain ontologies (e.g. a PA ontology).

Although, DAML-S is not tested in real applications and it has currently some outstanding issues pending it is a promising way for making provision of complex services for e-Government feasible. We plan to continue our work, testing the DAML framework in real world situations.
References


The InfoCitizen Project (2002). Agent-based negotiation for inter- and intra-enterprise coordination employing a European Information Architecture for Public Administration, CERTH/ITI.

