Παράρτημα ΙΙ:
Παραδείγματα Εφαρμογών

Τμήμα Διδακτορικής Διατριβής

Επιχειρησιακή Αρχιτεκτονική Διακυβέρνησης

– ΕΑΔ –

Για τον Ανασχεδιασμό της Δημόσιας Διοίκησης

Βασίλης Περιστέρας

Table of Content

1. INTRODUCTION

As presented in the previous chapters, both the re-engineered way of operation (chapter 2), and the GEA models (chapter 3) have been designed and introduced as Computational Independent Models, which means that their descriptions remain to a great extent technology independent and thus valid under different technology environments.

Having said that, our prior interest is clearly in implementing and using these models in a specific, advanced and promising technological environment based on Semantic Web technologies.

In this chapter, we present three cases that highlight and demonstrate several possible directions to exploit the models and the descriptions that were presented in this dissertation. The implementations presented here serve different purposes. Specifically:

- The first case presents the conceptual instantiation of the GEA Service Provision Process Model as applied to a specific business case in a cross-border setting, that of "New company registration in a multi-country setting".
- The second case presents a system that demonstrates on the fly composition of an ordered set of PA services based on input output relationships between them (i.e. the output of one serves as input to another). For building this system, concepts from the GEA PA Service Model (detailed model for service provision) have been employed.
- The third case deals with the problem of matching citizens' needs with available public services. This work is also based on the GEA PA Service Model and uses Semantic Web technologies for implementation purposes.

2. CONEPTUAL INSTANTIATION OF THE GEA SERVICE PROVISION PROCESS MODEL

In this part, we use the GEA Service Provision Process Model as a blueprint in order to design the service provision process in a specific setting: *electronic cross-border service provision*.

2.1 Motivation

Countries all over the world are facing global problems such as disease control, immigration and trafficking. The solutions to these complex problems usually require collaboration among government agencies in many countries (Su 2004).

Integrating and sharing information in multi-organizational government settings involves complex interactions within social and technological contexts (Pardo, M.Cresswell et al. 2004). This situation becomes even more complex when information sharing, coordination and collaboration among government agencies across national boundaries are needed.

The following problems have been identified to exist in a cross-border setting (Su 2004):

- Data heterogeneity
- Language heterogeneity
- Heterogeneity in people and working environments
- Heterogeneity in government policies, regulations, constraints, and security and privacy rules
- Difficulties in inter-agency and inter-government communication and coordination
- Heterogeneity in computing platforms

Thus, there is a need for developing and integrating technologies to enable government agencies to share information and work together across borders.

In European Union, this need for cross-border services amongst the EU Member States has attracted the interest of the European Commission and the term Pan-European eGovernment Services (PEGS) has been introduced and used for this type of services. PEGS attract the interest of the EU Interchange of Data between Administrations, Businesses and Citizens programme (IDABC) (European Commission 2004), and has fueled the work related to the European Interoperability Framework (IDABC 2004). Moreover, a set of reports has been drafted regarding PEGS requirements and architectures (IDA and CapGemini 2004). The work presented here could serve as a valuable input to this general line of work.

2.2 Model Presentation

The GEA Service Provision Process Model was presented in Section 3.2.6. This model analyses the phases a PA service goes through when executed. We used this model to design a specific business case of cross-border public service provision: the "*new company establishment*" service,

as provided by Greek public authorities to Italian citizens. Moreover, this service is executed as an electronic service, using an advanced information system environment based on Web Services and semantic technologies. The model presented here has been constructed as a pilot showcase during the EU-Publi.com project (IST-2001-35217) (Peristeras V. and Tarabanis K. 2005).

We present the abstraction of this case specific model, in order to propose a general description applicable in different business cases of cross-border public service provision. We deliberately avoided to present the case specific scenario addressed, i.e. new company establishment, but try to distinguish and focus on the case-independent, core aspects of the cross-border public service provision process. The general problem to be addressed in the above setting could be expressed as follows:

An entity E (person, company, organization) acting as a client and located in country A, needs to receive a service that is offered by a public administration agency PA_B located in country B…

Table 2-1: The problem to be addressed

It is obvious that this description is broad enough to cover different types of cross-border service delivery.

The generic GEA model proved useful during the following tasks:

- In the requirements analysis phase, in order to identify deficits in the current way of service provision.
- In the system design phase, as a roadmap to design the whole process instead of building it from scratch.
- In the technology implementation phase, by identifying specific areas where semantic technologies could add value to the process.

In the table below, we present the generic steps identified by the GEA model in the left column. These were discussed in detail in section 3.2.6. In the right column, we present the steps as executed in an electronic cross-border setting. There, we also highlight the main differences, extra facilitation and/or extra complexity introduced by the specific setting.

GEA Model	**Cross-border Model**
1. An atomic need eligible to be addressed in the public sphere emerges to a client.	1. An atomic need eligible to be addressed by a PA agency located in country B (PA_B) emerges to a Client located in a different country A. *This need may not be eligible to be addressed by all PAs.*
2. The Client identifies the type of Public Service (Service) available from the administrative system that addresses the emerged atomic need.	2. The Client identifies the type of Public Service (Service) available from the administrative system of country B that addresses the emerged atomic need. *This service may be different if compared to the respective service in country A.*
3. The Client finds the type of PA agency that provides the Service needed (e.g. the proper administrative level with the mandate to provide this type of service).	3. The Client finds the type of PA_B agency that provides the Service needed in country B. *This type of PA may be different than the relevant type in country A. Moreover, the Client during this step is supported to have access to a generic description of the service as provided in country B.*
4. The Client finds the specific instance of the	4. The Client finds the specific instance of the

PA agency that acts as the actual Service Provider for the specific case (WHO provides the Service).	PA_B agency that acts as the actual Service Provider for the specific case. *The logic used for this instantiation may be different if compared with country A.*
5. The Client finds the Location from where the Service is available by the specific Service Provider (WHERE the Service is provided).	5. The Client finds the electronic Location (url) from where the Service is available by the specific Service Provider PA_B *(the difference is that the service is electronically provided).*
6. The Client visits/contacts the Location where the Service Provider has the service available. A communication channel between the Client and the Service Provider is established.	6. The Client contacts the electronic Location where the PA_B has the service available. An electronic communication channel between Client and PA_B is established *(the difference is that the service is electronically provided).*
7. The Client gets information regarding various aspects of the service (HOW the Service is executed).	7. The Client gets information regarding various aspects of the service *(the difference is that the service is electronically provided).*
8. The Client initiates the Service.	8. The Client initiates the Service *(the only difference is that the service is electronically provided).*
9. The Service Provider receives and checks evidences (iterative). This processing may occur in stages.	9. PA_B receives and checks evidences (iterative). This processing may occur in stages. *In cases where evidences from country A should be forwarded to PA_B, semantic correspondence may be needed to find the equivalent evidences in a different administrative system.*
10. The Service Provider checks the overall service Procedural Preconditions. Evidence checks have been performed in the previous step. In this step, checks of the service as a whole take place.	10. PA_B checks the overall service Procedural Preconditions. Evidence checks have been performed in the previous step. In this step, checks of the service as a whole take place. *No difference in this step.*
11. The Service Provider comes up with the final decision regarding the case.	11. PA_B comes up with the final decision regarding the case. *No difference in this step.*
12. Output: The Service Provider prepares the Administrative Document, which is the output of the service. The Administrative Document includes the official decision on the case made by the Service Provider.	12. Output: PA_B prepares the Administrative Document, which is the output of the service. The Administrative Document includes the official decision on the case made by PA_B. *No difference in this step.*
13. Consequence: The Service Provider updates the archive kept internally (local effect) and informs other agencies (global effect).	13. Consequence: PA_B updates the archive kept internally (local effect) and informs other agencies externally (global effect). *There may be cases where external consequences should be communicated to country A. In these cases semantic mediation may be needed.*

Table 2-2: The Generic vs. the Use Case service provision process

The differences are summarized in the following points:

- In steps 10-12 there are no differences between the two processes.
- Steps 5-8 are performed similarly in the cross-border and national setting but there is an interesting customization of the generic steps due to the fact that the service is provided electronically.
- In steps 3-4, 9 and 13 there are differences due to the fact of the cross-border setting.
- Last, step 1 and partially step 2 refer to the needs-to-service mapping task (discussed in section 2.2.2.1) which is not supported or further discussed in this example.

In the table that follows, we have kept the major planning (informative) and execution (performative) parts from the generic description, and inside them we present a list of Use Cases (UC) that groups together all the above presented steps. Then, we describe each use case in details. We have chosen a use case based presentation to give a narrative, explanatory, and reader-friendly description of the model.

PLANNING PART
USE CASE 1: The Client identifies and locates the service URL as provided in country B (steps 2-3)
USE CASE 2: The Client explores the Service (step 3)
USE CASE 3: The Client instantiates or "tailors" the service (steps 4-5)
USE CASE 4: The Client explores the instantiated service(steps 6-7)
EXECUTION PART
USE CASE 5: The Client triggers the service and the Service Provider logs the new case (step 8)
USE CASE 6: The Service Provider finds, collects and processes all the service input (step 9-10)
USE CASE 7: The Service Provider solves semantic discrepancies (additional step applicable only in cross-border settings)
USE CASE 8: The Service Provider prepares and communicates the output of the service step (11-12)
USE CASE 9: The Service Provider handles the consequences of the service (step 13)

Table 2-3: Generic use cases for electronic cross-border PA service provision

2.2.1 Planning Part

In this phase, there is a set of use cases, which serves as a means for the client of the service to **identify, explore, tailor and orchestrate the service** (e.g. the electronic version of the service in the web). The planning phase ends with the service invocation.

Obviously, the planning phase in a electronic technological environment is executed differently than currently. Nowadays, the required identification, tailoring and orchestration of the service are performed through traditional means (e.g. contacting authorities by telephone/mail/fax, asking for lawyer's advice, checking whether a reply has been received in order to proceed to the next step of the process). The electronic provision setting entirely changes the way the Planning phase is performed. Briefly, this new way of execution is characterized by the disengagement of the client from the process, in the sense that important part of the communications, contacts and visits are substituted by specific information system functions. Thus, the client of the service is able,

- to remotely run the planning phase from his/her PC,
- to drastically cut down the overall time for getting all the relevant information
- to manage complicated services, without any external help.

More specifically in the informative phase, and as e-government wants to provide the client with an e-service fully available via the web, the first step is to support him/her in **finding** and

identifying the correct service (actually the correct URL where she/he can get the service), and to assist him/her to **explore**, and **tailor** it to his/her specific needs (specific circumstances of the actual execution e.g. geographically) without needing professional help.

The use cases (UC) that follow present generically these planning features for any type of public administration service operating in an e-government environment.

- UC 1: The Client identifies the service and locates its URL as provided in country B.
- UC 2: The Client explores the Service
- UC 3: The Client instantiates or "tailors" the service
- UC 4: The Client explores the instantiated service

2.2.1.1.1 UC 1: The Client identifies the service and locates its URL as provided in country B

The first step of the planning phase is to support the client in finding and identifying the correct service. In the given technological environment, this means finding the correct URL from where the service is accessible.

As the client needs a service from another country, there is a need to provide him/her with an infrastructure that will allow him/her to locate the service's URL in a different country.

Trying to identify a service (and the service provider) in a foreign administrative system increases the already existing difficulties at the national level. For example, the type of the service provider for a particular service may differ from one country to another. Geographical fragmentation and central versus regional service provision may also reveal great differences between administrative systems. The problem could be expressed as "*WHO* (public administration agency) provides *WHICH* (public administration) service electronically, and from *WHERE* (URL address)".

To address these issues, we introduce two infrastructures:

(a) The **National Web Service Repository (NWSR)**. This is an electronic repository, available on the web, which holds meta-data information for public administration services available as e-services at the national level.

NWSR is a technological implementation of the conceptual component called Central Public Administration Service Directory (CPASD), as introduced in section 2.2.2.1.

This information concerns: (a) the URL from where the service is available, (b) the service description, and (c) the logic for "tailoring" the service. Each country needs to implement such a repository, and make it available to the public. These repositories could be maintained e.g. by the Ministry responsible for public administration in each country.

We implemented it as a type of semantically enhanced UDDI registry (M. Paolucci et al. 2002) and built a UDDI PA-domain specific registry for storing information for PA e-services. This registry and the details of its technological implementation are discussed in more detail in section 4.2.

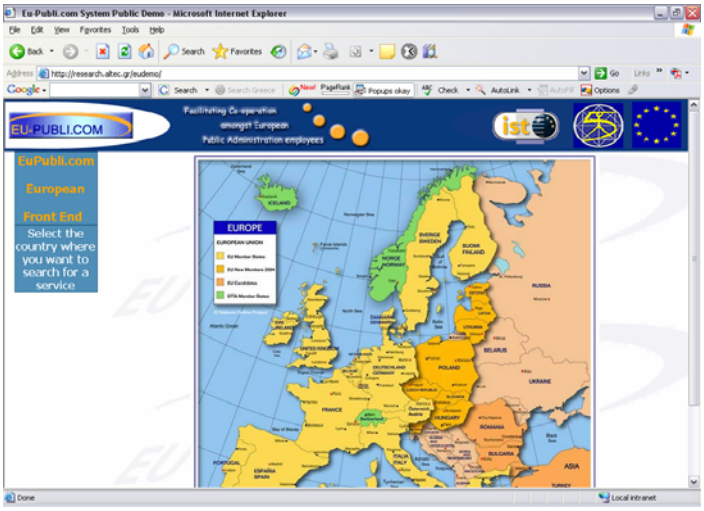


Figure 2-1: The Supra-national Web Service Broker (SWSB) infrastructure

(b) Then, at the supra-national level (e.g. European level), one single portal would provide access to all national repositories. This portal is part of the **Supra-national Web Service Broker (SWSB)** infrastructure. As a minimum the SWSB acts as a "pointer" to the various NWSRs. In our prototype, this was implemented as a clickable European map (fig. 4-1). Additionally to this "routing" functionality, the SWSB could provide access to "truly" supra-national e-services, which means services centrally provided by a set of countries (e.g. the EU) and not registered in any NWSR.

Returning to the use case, the client selects the country from where he/she asks for the service by clicking on the map and the SWSB returns the web page of the NWSR of the selected country. Now, the client has reached the NWSR that stores the links to all the public administration services available as e-services for the country of interest.

For the purpose of our scenario, the Greek Ministry of Interior and Public Administration maintains this central repository. The repository provides the user with a querying interface, in order to facilitate the identification of the service needed. The Greek NWSR and its querying interface are shown in fig. 4-2. The client enters key words and the NWSR returns services' names. In this example, the client searches for services related to "registration".

Figure 2-2: The Greek NWSR page

The client refines the query until he/she has found the service of interest. The use case ends with the NWSR providing to the client the URL of the service.

2.2.1.1.2 UC 2: The Client explores the service

The client needs information about the service, so he/she uses the URL provided in the previous steps and accesses the relevant information stored in the NWSR. This is the second functionality of the NWSR: it holds brief descriptions and information for all stored types of services. It is important to mention that in a conventional, non-electronic service provision environment, the exploration of the service at the *service type* is not feasible because as presented in section 2.2.3 *"… due to the lack of a "global" view of the administrative system as a whole. There is no actor to hold descriptions of services types. Each authority is responsible and just "knows" the "How" of its own services."*

The NWSR holds service descriptions that are related to several aspects of the service such as:

- The "type of" responsible agency.
- The scope of the service, when the service is applicable, what are the preconditions to apply for this service, important exceptions, etc.
- The inner structure of the service, including the macro-workflow.
- The "type of" participating actors (e.g. other PA agencies).
- The national average fees and costs for the service.
- National average time for service provision, with upper legal limitation if such exists.
- Relevant legislation.

There are two important limitations to the service descriptions as exist in the NWSR. Due to these limitations the service description may not be feasible to be depicted in detail, in the NWSR:

- The service execution may follow numerous alternative paths depending upon conditions that could not be known in advance, but decided during the service execution.
- Depending on the administrative system of each country, the execution of the same service may differ amongst the PA agencies. The more decentralized the administrative system

(e.g. federal government), the more likely for the service execution to diverge from a centrally provided description.

In the figure below, we can see part of this description with the steps of the process to be executed.

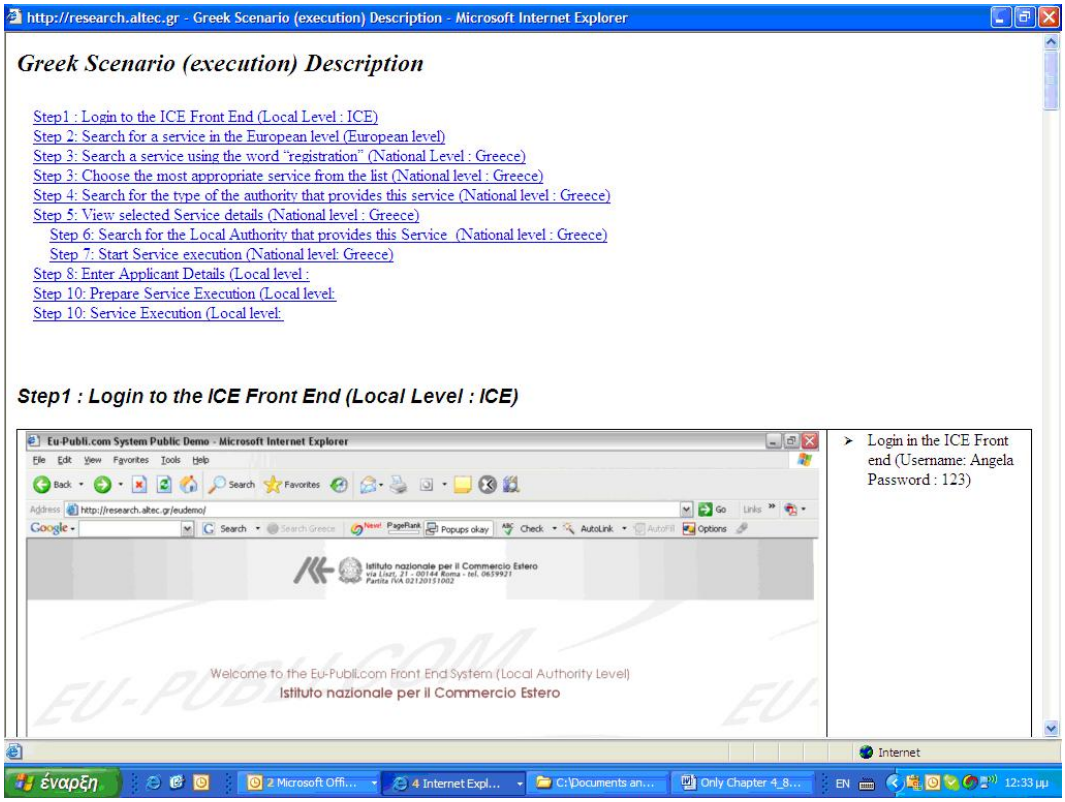


Figure 2-3: Service exploration at the NWSR

2.2.1.1.3 UC 3: The Client instantiates or "tailors" the service

Tailoring or finding the right version of a service means performing all the necessary preparatory actions in order to guide the client through a (usually) complicated set of possible alternatives in order to identify the exact and correct version of the service of interest, as provided by the appropriate service provider.

As the client expresses the interest to identify the specific provider for the service under request, the NWSR uses a set of questions that will be used by the system for the service provider identification. These questions depend on the criterion used for "versioning" the service (e.g. to geographically tailor a request for a birth certificate, the system could ask the user to enter the municipality where he/she was born, in countries where the municipality is the administrative level responsible for this service). The NWSR "knows" the internal logic of each service and uses this knowledge in order to address the proper queries to the client. This "versioning" of PA services is the third functionality of the NWSR. In our showcase, the client provides necessary information and the NWSR uses this information to identify the particular instance of the service provider. Succeeding in this, the NWSR returns the actual URL link of the service as provided by the specific service provider (first functionality of the NWSR). The use case ends with the NWSR providing the actual service URL to the client.

In the figure below, the system first identifies that the required type of service is provided in Greece by the Prefectural level. In order to identify the specific prefecture, the user enters the city

name (e.g. Thermi). The system returns in this case "*Prefecture of Thessaloniki*" and the link to the actual service.

Figure 2-4: The service tailoring interface

2.2.1.1.4 UC 4: The Client explores the instantiated service

This use case is quite similar to UC 2, having in the place of the NWSR, the actual Service Provider for the service. The Service Provider provides detailed information for the service execution. In this use case, the client's communication with the NWSR has ceased, as a direct communication link with the information system of the actual service provider has been established. It is for the first time that the client has access and communicates directly with the information system of the actual service provider. By the end of this use case, the service provider has communicated to the client the exact description of the service in every detail. As already discussed, this description may slightly differ from the one presented centrally by the NWSR.

2.2.2 Execution Part

In this phase, there is again a set of use cases, which serves as a means for the client of the service to **execute** the electronic version of the service on the web. The execution (or performative) part practically starts with the submission of the client's application and the subsequent service invocation. This is the first "official" act that takes place and recorded by the administrative system.

The performative phase in an advanced technological environment (e.g. SW) is drastically different than in the current modus operandi. As in the planning phase, nowadays, the client has to go through a frustrating process that usually requires visits at several PA authorities that may participate in the process of service provision. Again, these contacts are performed through traditional means (e.g. visiting the physical location). As in the planning phase, the use of ICT technologies is characterized by the disengagement of the client from the process through the automation of all parts that can be automated.

The use cases that follow present generically the execution part for any type of public administration service operating in an e-government environment according to the new modus operandi proposed earlier.

- UC 5: The Client triggers and the Service Provider logs the new case
- UC 6: The Service Provider finds, collects and processes all the necessary input for the service (evidences)
- UC 7: The Service Provider solves semantic discrepancies
- UC 8: The Service Provider prepares and communicates the output of the service
- UC 9: The Service Provider handles the consequences of the service

2.2.2.1.1 UC 5: The Client triggers and the Service Provider logs the new case

We assume that the client has decided to officially ask for the particular service execution (this may not always be the case). So he/she sends an official request to trigger the service registration. At this point, we pass from the informative part to the performative part of the service execution. From now on, the client interacts with the information system of the service provider.

This use case does not present any difference when executed in a cross-border setting. The interest lies in the electronic way of execution. The request for the service execution is processed by a specially designed software component that provides the necessary management of the overall service execution workflow.

This component is an implementation of the conceptual infrastructure called "Orchestrator" and introduced in 2.2.2.1.2 and 2.3.2.1.1 sections. In this case, the "Orchestrator" has been implemented as a ActiveBPEL engine[1] which runs BPEL4WS - Business Process Execution Language for Web Services[2]. Additional information on the implementation of this component is given in the next showcase (section 4.2). Thus, the Orchestrator becomes the first software component, inside the service provider's information system, that is called by the client during the service execution.

The Orchestrator "knows" the details of the workflow execution that is the conditions for invocation, management and termination of the workflow throughout the service execution. The Orchestrator "moves" the execution of the service according to a pre-defined model. The business logic and knowledge needed for this task already exists inside each service provider, as each public agency "knows" the process to be followed. We embed this knowledge into the Orchestrator to automatically execute the service without any external consultation.

Returning to the description of the use case, the Orchestrator receives the request, checks its completeness and appropriateness (e.g. does the client request the right service?) and directs it to the service provider's service registration system. As the information system of the service provider logs the case (after performing again some internal checks), the first official administrative action (*performance*) takes place. All the previous stages have been informal and PA keeps no record of them. Now, the case is registered by the service provider.

After successfully registering the application, a confirmation message is sent to the Orchestrator. By receiving this confirmation, the Orchestrator triggers the service execution. At this point, use case 5 is terminated.

---

[1] http://www.activebpel.org/

[2] http://xml.coverpages.org/bpel4ws.html#articles

2.2.2.1.2 UC 6: The Service Provider finds, collects and processes the service input

As an introduction to this UC, we have to mention that this is perhaps the most demanding phase in cross-border public administration service delivery.

The client needs assistance in order to collect all the evidences needed by the service provider to be validated in order to successfully execute the service. So the client has to collect all the evidence placeholders (e.g. administrative documents) needed for the service execution.

The multi-country setting adds additional requirements to the service execution: all evidence placeholders (and the evidences they carry) have to be converted in order to become semantically compatible to the target administrative system. Moreover, the grouping of evidences into evidence placeholders may vary when passing from one country to another. So the system should be capable support these two aspects of evidence and evidence placeholder conversion.

With the help of the Orchestrator the client can execute important part of this UC remotely. With the exception of matching between evidence and evidence placeholder, the execution process is similar with that executed at the national level. Thus, we do not repeat here the whole process, which can be found in section 3.2.6.2.

2.2.2.1.3 UC 7: The Service Provider solves semantic discrepancies

This is a supportive use case, applicable and of critical importance in multi-country settings. This use case is not supposed to follow the previous one. It could be perceived as being internal to the previous one, aiming at solving semantic discrepancies during the evidence collection and processing. There are several types of discrepancies that can be found at three levels:

- At the evidence placeholder level. More specifically:
  - Mapping between equivalent evidence placeholders: e.g. how does certificate X is called in another country?
  - Finding equivalent, when no direct correspondence exists: e.g. what can be used from country X, instead of certificate Y that is normally used in country Z.
- At the evidence level. Sometimes the above questions cannot be addressed at the level of evidence placeholders; there may be no equivalent of certificate X in country Z. In these cases, the analysis should move to the evidences "packed" in the evidence placeholders (e.g. which evidence placeholder stores a valid date of birth for a citizen in country X?)
- There may be cases, when even the equivalence between evidences cannot work. Not only evidence placeholders, but also evidences may not exist in an administrative system, or may be out of context. For example, many European citizens do not possess an ID card number (evidence). What if an authority asks for an ID card? A third level should be activated: what is the purpose of this evidence, what administration wants to check? Finding the purpose could facilitate providing the appropriate substitute of the missing evidence.

In order to address the above, specific domain knowledge is needed. The Supra-national Web Service Broker could hold a repository for storing this knowledge. There are certain assumptions made at this point:

(a) All national PAs have developed PA domain ontologies.

(b) In the SWSB, a Supra-national meta-ontology of PA is stored. Mapping amongst the national PA ontologies would be possible through the SWSB meta-ontology, avoiding the exponential number of ontology mappings required to make each country's ontology compatible with all the other country ontologies. Of course, this is a difficult infrastructure to be built and maintained that requires the previous creation of PA domain ontologies at the national level. Although formal ontologies are very difficult to be specified and managed for such extended domains as national

public administrations, this is a core prerequisite for solving semantic incompatibility in multi-country settings. In this line, we have developed two national test-bed ontologies and provided mappings between them using SWRL.

The concept of the "*Evidence Purpose*" from the GEA PA service model is used for this purpose. As an example, we may have the case that one agency needs the client to provide her/his ID card number. The purpose of this evidence is "person identification". Let's assume that the Client comes from a country where there are no ID cards. The system receives as input the nationality of the Client and the purpose of the evidence needed by the service provider. Then, queries the other country's ontology to find the same purpose and the evidences that are commonly linked to it. In our example, the system returns the Social Security Number, or the passport number. These are different evidences, stored in different evidence placeholders that serve the same purpose, that of "*person identification*".

In this way, forms fields or terms used in administrative documents are translated semantically in an automated and appropriate way, as the system handles the differences between countries.

The SWRL syntax for matching the Greek ID card number (`Αρ_Ταυτοτητας`) as provided in the Greek ontology with the passport number (`Num_di_Passaporto`) as provided in the Italian ontology is shown in the figure below. This matching is provided by using reference to the `CitizenIdentificationID` concept, which documents the actual "*Evidence Purpose*".

```
<ruleml:imp>
          <ruleml:_rlab ruleml:href="#TranslateCitizenIdentificationID"/>
          <ruleml:_head>
                  <swrlx:datavaluedProperyAtom
swrlx:property="http://research.altec.gr/OnarServer/Καταστατικο.owl#Αρ_Ταυτοτητ
ας">
                          <ruleml:var>CitizenIdentificationID</ruleml:var>
                  </swrlx:datavaluedProperyAtom>
                  <swrlx:datavaluedProperyAtom
swrlx:property="http://research.altec.gr/OnarServer/Statuto.owl#Num_di_Passapor
to">
                          <ruleml:var>CitizenIdentificationID</ruleml:var>
                  </swrlx:datavaluedProperyAtom>
          </ruleml:_head>
   </ruleml:imp>
```

Figure 2-5: SWRL code for matching the Greek ID Card number to the Italian Passport Number

2.2.2.1.4 UC 8: The Service Provider prepares and communicates the output of the service

No matter what the administration has decided, the decision should be cast in a written official form: it should become an official administrative document with registration number, date of issuance and certifying stamps and signatures. If compared with the national PA service provision, there are no differences in this UC.

2.2.2.1.5 UC 9: The Service Provider handles the consequences of the service

As described, there may be two types of consequences.

Update the public administration registry: This does not acquire special characteristics in this case.

Inform consequence receivers: This means forwarding information related to the service to other public administration agencies. For this purpose, the activation of the Orchestrator and the NWSR is needed again. This activity in a multi-country setting is very demanding, as the service provider may have to communicate information (evidence) or administrative documents (evidence placeholders) to notify public entities in a different country for the service provision. In a cross-border environment the additional problems that may occur in this UC are similar to those

described in UC6, during the identification of evidence and evidence placeholder (administrative document) equivalence, conversion of terms, etc. Again the activation of UC 7 may be needed for solving semantic inconsistencies between terms, evidences and documents.

2.3 Conclusion – Future Work

The generic process model as proposed by GEA has been proven a valuable roadmap, in order to propose the service execution model in a cross-border setting. Moreover, the work on the showcase was used for validating the applicability and expressive power of the generic GEA model. The comparison between the two columns of table 4-2 (generic process execution versus process execution in a cross-border setting with electronic service provision) and the proposed Use Cases can easily reveal the close association of the descriptions. Certain differences exist, e.g. the multi-country setting creates additional requirements and use case 7 was introduced to address them, but the core structure of the execution process remains the same to a great extent.

We intend to further use the GEA process model to describe more PA services as implemented in different PA fields. This exercise is expected to prove valuable for the validation of the generic model as proposed by GEA.

3. AUTOMATIC COMPOSITION OF COMPLEX PUBLIC ADMINISTRATION SERVICES

3.1 Motivation

During the process of service execution, we identify two main sources for retrieving the needed evidence (input): the client and the public administration system. The first case is rather simple as the client is directly asked to supply the information that needs to be checked. Generally speaking, the more serious the implications of the service, the less likely it is for public administration to accept evidence directly from the client.

The motivation that drove the work presented in this showcase has been extensively presented in chapter 2. Here we present a brief overview of the business case.

We focus on the case, where evidences should be provided to the service provider by other PAs. This means that in order for the service provider SP1 to execute a service s1, which is requested by a client, information is needed that is actually the output of the execution of another service s2 provided by another service provider SP2, as presented in the next figure.

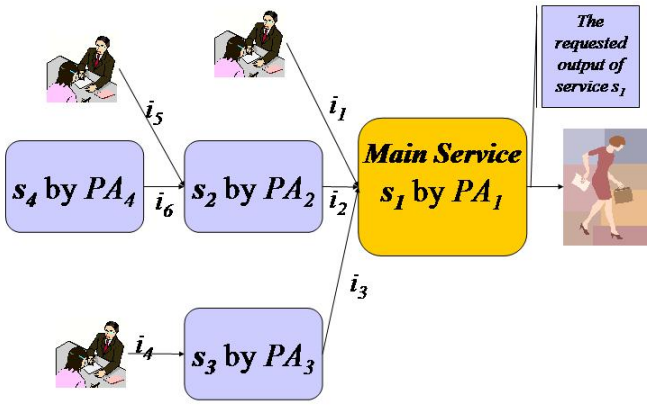


Figure 3-1: Execution of nested services

In this example, the client needs to receive the output of service s1, as provided by the PA agency PA1 (Main Service). S1 needs to check information (evidence) from three sources: an input directly provided by the client (i1) and the output of two other services (s2, s3) as executed by PA agencies PA2, PA3 to be used as input (i2, i3). In order to execute s3, PA3 needs only information from the citizen (i4). On the contrary, PA2 in order to provide s2 needs information not only from the citizen (i5), but also from another PA agency (PA4). So PA4 has to run s4, in order to provide the information needed as input (i6) for the execution of s2. In this example, services s2 and s3 are considered strictly nested in s1, while s4 is strictly nested in s2.

It is easy to understand that in complex PA services with numerous nested services provided by different SPs scattered geographically, a full tree of services should be executed in a specific

order before all the required input becomes available for the initial SP. In these cases, the nesting amongst services may be layered.

It is important to realize that this demanding task is currently carried out by the client with no help at all. This means that the client goes through the planning and execution parts of all s2, s3, s4, … sN nested services to s1 without any assistance or guidance from SP1 or anyone else. SP1 asks for specific documents as generated by specific SPs (actually for the evidence "packed" in these documents) and knows nothing about the underlying processes that govern these services. The client alone needs to:

- identify all the nested services that should be executed,
- find the SPs (names and locations) of all the nested services,
- understand the overall and each separate service,
- gather all the input needed separately by each nested service,
- trigger and monitor the service execution of all services,
- resolve possible problems during the execution of all these services,
- receive and gather all the outputs, and
- forward these outputs to the initial SP.

Obviously, the result is usually a process, which is difficult to handle, complex, time-consuming, frustrating and error-prone. It is worth mentioning that it is not the client's need that leads him/her to request the execution of the nested services, but rather the underlying logic of the main service.

3.2 Technical Implementation

In this showcase, we have tried to address the above-presented problems by providing an infrastructure to automatically construct the execution workflow of the nested services.

For this purpose, we employ an inference mechanism that consults a specific technological implementation of the Central Public Administration Service Directory (CPASD) that has been described in chapter 2. This is the NWSR infrastructure introduced and used in the previous showcase (section 4.1). The inference mechanism then builds on-the-fly the description of the service workflow. Last, we execute this workflow in an execution environment based on the ActiveBPEL engine introduced in the previous showcase (section 4.1).

For automatically constructing the service workflow, we take into account the relationship that exists between the *Evidence Placeholder* and *Output* concepts as documented in the GEA object model (for details on this relationship see section 3.2.4). The *Output* of a service is an *Evidence Placeholder* and *Evidence Placeholders* are used as service *input*. If one of the *Evidence Placeholders* needed as input in Service A is the *Evidence Placeholder* produced as output by another service B (in the showcase, this latter is called *Computed Input*), the workflow inference mechanism includes the nested service B in the service execution workflow of Service A. Service B can also contain *Computed Inputs* that require the execution of other nested services, and so on.

At the implementation level, the NWSR was deployed as an enhanced UDDI registry following the approach presented in (M. Paolucci et al. 2002; Paolucci M. et al. 2003). Generally, the UDDI meta-model provides rather limited information for the functional properties of the services that are published there (e.g. (Dimitrios Tektonidis, Albert Bokma et al. 2005)). For automatically constructing the nested services workflow, we had to document and include

information about two GEA concepts namely the service "*Output*" and the "*Evidence Placeholder*" (input) needed by the service for each published service in the UDDI. These two concepts (or relevant) are not present in the UDDI meta-model (OASIS - UDDI Spec TC 2002). The five core UDDI data structures are shown below.

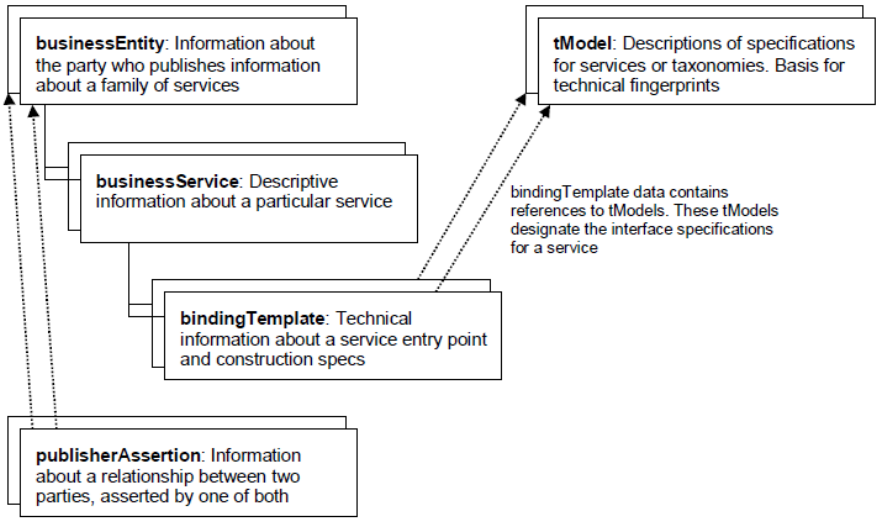


Figure 3-2: The UDDI specifications for the businessService element

The specifications of the businessService core structure are presented in fig. 4-8. The discovery mechanism that is provided in UDDI registries works only with the presented below pre-defined service attributes that are part of the UDDI service description (OASIS - UDDI Spec TC 2002).

```
<element name="businessService" type="uddi:businessService" />
<complexType name="businessService">
  <sequence>
    <element ref="uddi:name" minOccurs="0" maxOccurs="unbounded" />
    <element ref="uddi:description" minOccurs="0" maxOccurs="unbounded" />
    <element ref="uddi:bindingTemplates" minOccurs="0" />
    <element ref="uddi:categoryBag" minOccurs="0" />
  </sequence>
  <attribute name="serviceKey" type="uddi:serviceKey" use="required" />
  <attribute name="businessKey" type="uddi:businessKey" use="optional" />
</complexType>
```

Figure 3-3: The UDDI specifications for the businessService element

The two GEA concepts should be imported in this meta-model. We have to extend the existing UDDI service description in order to enable a richer description of the published PA services.

For this purpose, we have used the tModel UDDI mechanism to store this extra information that was needed for PA services.

We then used a control vocabulary for expressing these concepts. This vocabulary is actually a list containing all the official names of two types of documents:

- Evidence Placeholders: documents that are needed as input by the service provider (e.g. ID Card, passport)
- Service Output: documents that are provided as output to the client (Output) by PA agencies (e.g. birth certificate, driving license).

The inference mechanism receives as input the name of the service that the client wants to execute and provides as output – what we have called – the Semantic Service Description (this is explained below). This is achieved by following the steps below:

- First, the inference mechanism finds in the enhanced UDDI all the evidence placeholders needed for the service under request.
- Then, it queries the UDDI registry to find whether there is a (nested) service that produces as output one of the needed evidence placeholders.
- If this is the case, the (nested) service is attached to the workflow.
- The inference mechanism retrieves again the evidence placeholders needed for this nested service to be executed.
- Again, some of these evidence placeholders may be provided as output by other services … and so on.

By executing these steps, the inference mechanism starts drafting the Semantic Service Description (SSD). SSD is an XML structure. More specifically, in this structure every service is described by a process class that contains information regarding the Service Name, the Service Provider, the Access Point and the Operation. In addition it contains information about the Service Outputs (Output List), the Service Computed Inputs (Input Lists) and the Service Simple Inputs (Form List). The latter is evidence to be provided directly by the client. Every Process Class can contain as child classes other Process Classes (nested services). The structure of the SSD is shown in fig. 4-9.

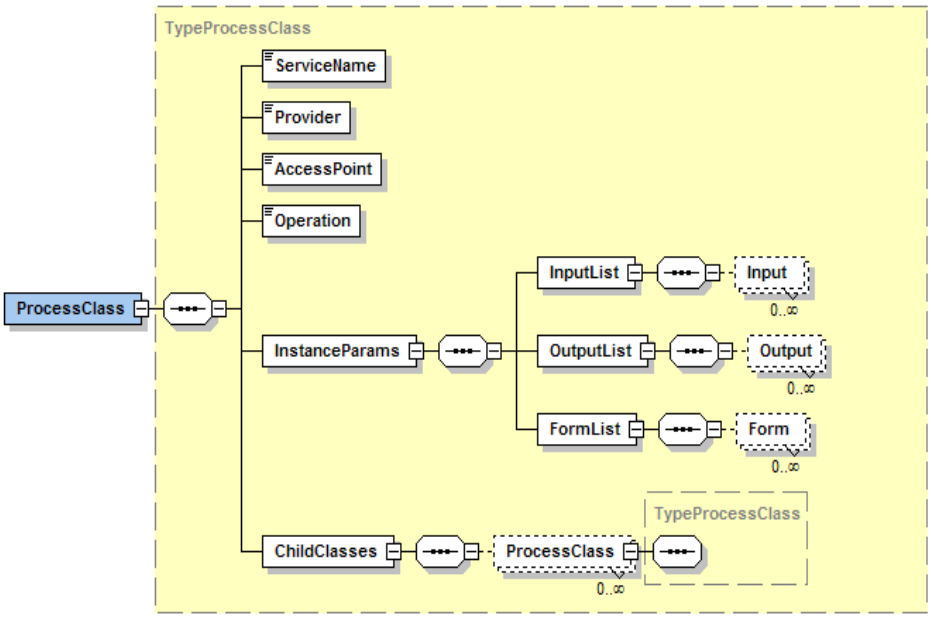


Figure 3-4: The Semantic Service description XML structure

In this way, the whole workflow of the service is composed in a backtracking and on-the-fly way, using the information that exists in the UDDI. If a service description published there changes e.g. due to a new legislation, the inference mechanism will "read" the new description and – if appropriate - will create a different workflow on-the-fly. The importance of this on-the-fly composition is related to the fact that in the complex PA environment, changes in services are quite usual. The system avoids the approach of hard-coding complex service workflows and allows the workflow to be generated each time by using the updated information directly from the UDDI following a clear SOA approach.

Moreover, the enriched UDDI provides advanced features for querying the registry based on the needed evidence placeholders or the service output e.g. it can answer to queries like, "which services use as input a client's birth certificate?". This is considered an important functionality for maintaining and managing the overall registry infrastructure.

In our application, the SSD is graphically presented to the PA employee. An example is presented in the next section (fig. 4-10). Through this graphical interface the employees has access to a description that:

- Summarizes the workflow of the services that need to be executed presenting the steps to be followed and the agencies that should be contacted.
- Presents in detail all the documents that need to be acquired by the citizen for the execution not only of the main service under request, but also for the nested services.
- Includes and analyzes all the nested services that are linked to the specific service workflow.

3.3 Showcase Example

The case that was implemented by using the prototype system has been that of registration of an affiliated Limited Company (AE), in Greece.

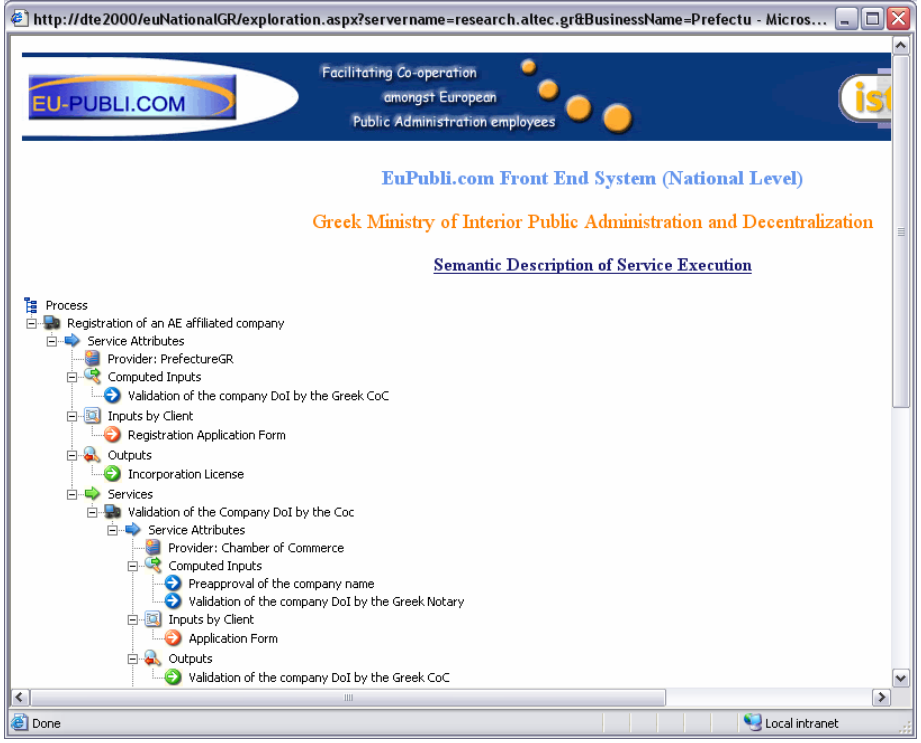
Briefly, the entrepreneur has to run several services in order to get all the necessary inputs needed before applying for the service in a Greek prefecture (service provider).

Amongst these services, there is a need for the Chamber of Commerce (CoC) to validate the Deed of Incorporation (DoI) of the new company. This validation is a *Computed Input* to the main service that is provided as *Output* by a service called "*Validation of a Company DoI by the CoC)*". The inference mechanism matches the input needed by the main service with the output provided by the CoC and includes this service in the SSD.

Figure 3-5: The Semantic Service Description of the showcase

In fig. 4-10, the clients of the service (the PA employee and the entrepreneur) are informed that in order to execute the service "*Registration of an affiliated company*", the "*Validation of the Company Deed of Incorporation by the Chamber of Commerce*" service should be executed in advance. Computed Input is marked with blue arrow in the SSD graphical representation.

The client also is informed that in total two forms should be completed (Inputs by Client), namely the "*Registration Application Form*" to be forwarded to the Prefecture and the "*Application Form*" to be forwarded to the CoC. Input by Clients is marked with red arrows.



Taking a closer look at the description in fig. 4-10, we find that in order for the nested service to run at the CoC, two more computed inputs (blue arrows) are needed: "*Pre-approval of the company name*" and "*Validation of the company DoI by a Greek Notary*". These two inputs are again outputs produced by other services and the SSD includes again their detail description. These two are actually services nested in the previously analyzed CoC nested service, and so on. The descriptions of these services are not presented in the figure for simplicity.

As our intention is not only to compose this detailed description on the fly, but also to execute the actual services, we need a service execution environment, that is, an implementation of the conceptual infrastructure that has been called *Orchestrator* in chapter 2.

It is important to mention that the derived description in the SSD constitutes an automatically constructed conceptual workflow model, expressed in XML. This description is valuable per se and is loosely coupled to the implementation technologies. This means that the SSD could alternatively "feed" Orchestrators based on different technologies.

For the purpose of this specific showcase, we have implemented the Orchestrator as a BPEL4WS-based machine, based on ActiveBPEL open source engine, a technology and an implementation solution mature enough to give a stable running example. This machine reads the SSD, translates it into BPEL and directs the invocation and execution of the identified set of services (EU-Publi.com Consortium 2005). In this way the full process is automatically executed. A prerequisite for this fully electronic execution is that all PA services involved are already available as Web Services, and are published in the NWSR.

The picture below shows a graphical representation of a part of the BPEL process dynamically generated from the above SSD. Fig. 4-11 is a screenshot of ActiveBPEL administration service. The picture allows distinguishing the patterns used to create the BPEL flow, as structural BPEL elements (flow, sequence, etc.) are depicted as boxes, while activities are depicted as colorful icons. An example of the automatically generated BPEL code is given in fig. 4-12.

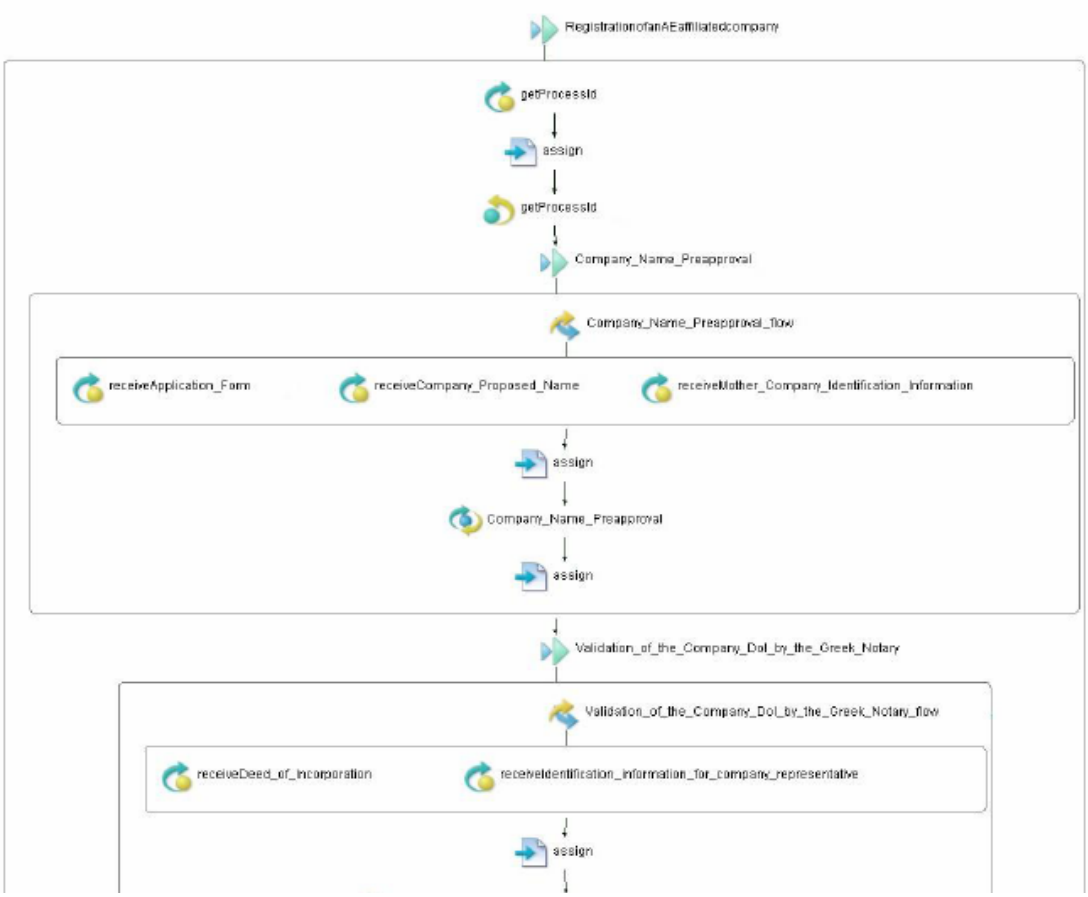


Figure 3-6: ActiveBPEL screenshot (EU-Publi.com Consortium 2005)

```
<sequence name="Company_Name_Preapproval">
    <flow name="Company_Name_Preapproval_flow">
            <receive name="receiveApplication_Form"
              partnerLink="RegistrationofanAEaffiliatedcompany_myRole_PL"
              portType="company:RegistrationofanAEaffiliatedcompany"
              operation="receiveApplication_Form" variable="Application_Form"
              createInstance="no">
                    <correlations>
                      <correlation set="TransactionIdSet" initiate="yes"/>
                    </correlations>
            </receive>
            <receive name="receiveCompany_Proposed_Name"
              partnerLink="RegistrationofanAEaffiliatedcompany_myRole_PL"
              portType="company:RegistrationofanAEaffiliatedcompany"
```

Figure 3-7: Part of the BPEL code created (EU-Publi.com Consortium 2005)

3.4 Conclusion – Future Work

Concluding, this showcase presents the automatic and on-the-fly composition of complex workflows that often need to be executed in PA service provision. The implementation was based on semantic technologies. More specifically, domain concepts from the GEA models were used

in order to provide the necessary semantics to a UDDI registry. This registry is consulted by an inference mechanism that matches a service output with a service input and constructs the workflow description in XML on the fly. This description is then executed by an ActiveBPEL engine.

The future work related to this showcase includes two interesting directions:

The first is related to the use of ontologies to present the service "Output" and not just controlled vocabularies as in the current showcase. This advancement, coupled with an extension of the inference mechanism to be able to handle not only simple string matches, but also rule-based language queries, will provide the prototype with an increased semantic capacity in the match-making process (e.g. to identify that the "certification of birth" evidence placeholder is equivalent to the "birth certificate").

The second direction is to use the SSD for "feeding" different process execution environments. For example, as we are currently working with the WSMO and WSMX execution environment, we intend to use the service description in the WSMX process execution machine.

4. MAPPING CLIENTS PROFILES TO PUBLIC ADMINISTRATION SERVICES

4.1 Motivation

As introduced in chapter 1, due to the dual PA communication/integration problem, contemporary PA systems strive, (a) to achieve internal integration at the administrative intra- and inter- agency level, as well as (b) external integration and user-centric communication channels with society.

The work presented here addresses the second type of the above-presented PA integration deficit. It does so, by implementing an application that facilitates the identification of relevant PA services based on the profile of a client. This profile-to-service resolution is related to and can be perceived as a type of the more general problem, that of mapping the client's needs to PA.

Briefly, the problem stems from the different perspectives PA and clients hold for PA services. As discussed in chapter 2, the client is *needs-aware*, but not services-aware (that is, aware of the actual services he/she really needs). On the contrary, PA is *services-aware*, but not needs-aware. The need may arise for several reasons, e.g. due to an external event (life-event or business episode) or due to a profile change of the customer (e.g. unemployed with five children). This latter case attracts our interest in this chapter.

The services that address this need may be mandatory (e.g. to register a new-born child) or simply beneficial to the client (e.g. receive a grant). To facilitate the communication between the two actors, there is a need for a consistent mapping between services and needs, and vice-versa.

Usually this matching is done on an ad-hoc basis by the client alone in an empirical, time and energy consuming, as well as frustrating way, without much support from public administration. Taking into account the fragmentation of the administrative space this task is currently performed in a very suboptimal manner.

To facilitate this task, we elaborate on a conceptual component presented in chapter 2, namely the Needs-to-Services Converter. This should be able to receive a client's need as input and to provide as output a set of public administration services that address this need.

Here, we try to implement part of this Converter - more specifically the capability of mapping profile-to-services - by employing semantic technologies and using the GEA object model (PA service model) (Goudos S., Peristeras V. et al. 2006). Our showcase shares many ideas with the famous wine ontology that is exhibited by the Stanford University as an example of an ontology-based application[3].

The showcase presentation is organized as follows: a brief overview of the part of the GEA model employed in the showcase is presented in section 4.3.3. The prototype system is presented in section 4.3.4. Specifically, in section 4.3.4.1, the business case is described. The ontology implementation of the GEA model in OWL is given in section 4.3.4.2. Sections 4.3.4.3 and

[3] http://www.daml.org/ontologies/76

4.3.4.4 present respectively the proposed system architecture and the running prototype system. Finally conclusions and plans for future work can be found in section 4.3.5.

4.2 Overview of the GEA model employed

For implementing this showcase, we have used the GEA object model for service provision (fig. 4-13). As the model has been presented in chapter 3, here we only introduce some categorizations we employ in four objects to facilitate the need-to-services mapping. These objects are the following:

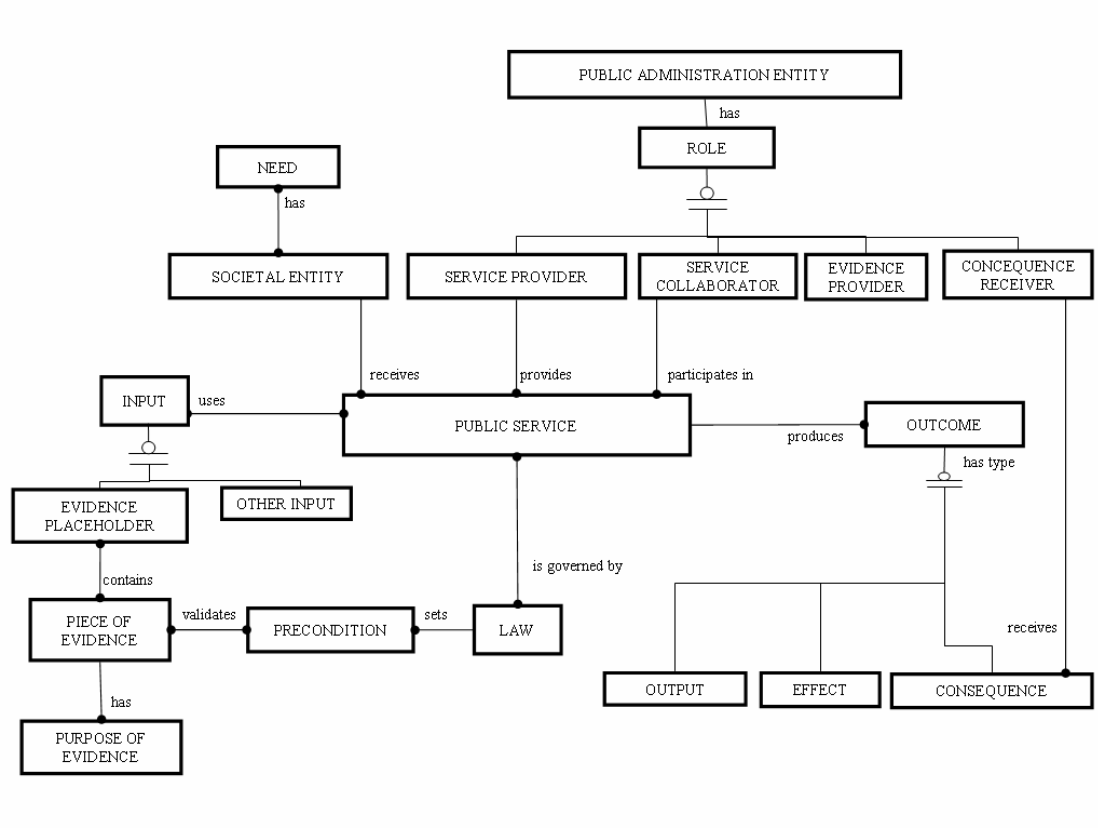


Figure 4-1: The GEA detailed object model for service provision

Societal Entity: As presented in section 3.2.1.1 and fig. 3-6, there are several types of Societal Entities (e.g. legal entity, physical person) requesting and receiving Public Services. Each Societal Entity has also a Profile (e.g. young businessman, disabled person).

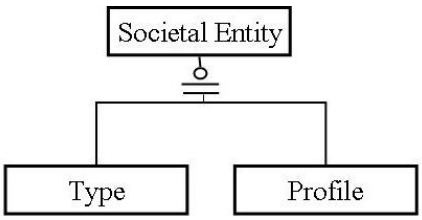


Figure 4-2: Societal Entity Type and Profile

Public Service: As discussed in section 3.2.1.5, services may be categorized based on the functional category they belong. For identifying such functional types, in this showcase we have used the categories used in the Business Reference Model of the USA Federal Enterprise Architecture (FEA) (TopQuadrant 2005) for categorizing services for citizens, as presented in section 1.3.4.6, fig. 1-9. There, several Domains are defined (e.g. Health, Transportation) that are further detailed in several Sub-Domains (e.g. Air Transportation and Water Transportation).

Service Provider: We categorize the Service Provider with regards to the administrative level they belong (e.g. municipal, regional, state, national).

Effect: The execution of a service may result in a change in the state of the world (e.g. transfer money to an account). At the top level, and as presented in section 3.2.7.2, there are three types of Effects expected from the execution of public services:

- Safeguard the Social Contract
- Promote Sustainable Development
- Provide for Social Welfare

4.3 Prototype Description

As a first step, we created an ontology based on the GEA object model. This ontology is then used as the knowledge base for a semantic application. The input to the application consists of the user profile, while the output returned consists of the public services that match the specified profile.

4.3.1 System Architecture

The proposed system architecture is given in fig. 4-15. This architecture consists of a web server, a reasoner and an OWL file which is used as the knowledge base. The users access the application through a common Internet browser. The advantage of every web-based front-end is that it requires only an Internet browser in order to execute and it can be accessed from anywhere on the Internet. The system architecture employed is server-side; therefore the client shows only the form and the results page.

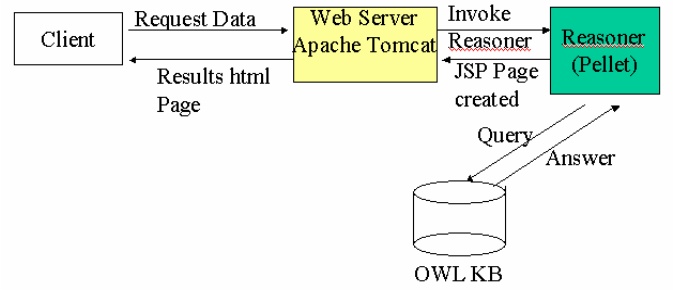


Figure 4-3: The application architecture

The server uses the data given to invoke the reasoner. The reasoner sends various queries to the knowledge base. SPARQL (Seaborne A. and Prud'hommeaux E. 2005) was selected as the query language. The answers returned are parsed by the web server that creates the results web page. The extracted results contain a list of the public services that match the selected profile. Specifically, the web server used was Apache Tomcat, the knowledge base was an OWL file with

the GEA ontology. The reasoner selected is Pellet[4]. Pellet is an open source OWL DL reasoner that can be used in conjunction with Jena either through a DIG interface or with native APIs. Pellet provides support for SPARQL.

4.3.2 The GEA Ontology

The GEA object model has been shown above as an E-R diagram. It is obvious that such a model can be implemented using a relational database. But as we wanted (a) to be able to share information through the web for both humans and machines and (b) to exploit the advantages of declarative knowledge representation, we decided to express the GEA model in an ontology language.

OWL DL (Dean M., Schreiber G. et al. 2004) was the obvious choice for several reasons; since 2004, OWL DL is an active recommendation of W3C group and various examples of models expressed in OWL exist. An important point that was taken into account for choosing this version of OWL has been the existence of OWL DL reasoners. OWL Full reasoners do not yet exist (Dean M., Schreiber G. et al. 2004).

The GEA ontology has been created using the Protégé tool[5] with the OWL plug-in (Holger K., Fergerson Ray W. et al. 2004).

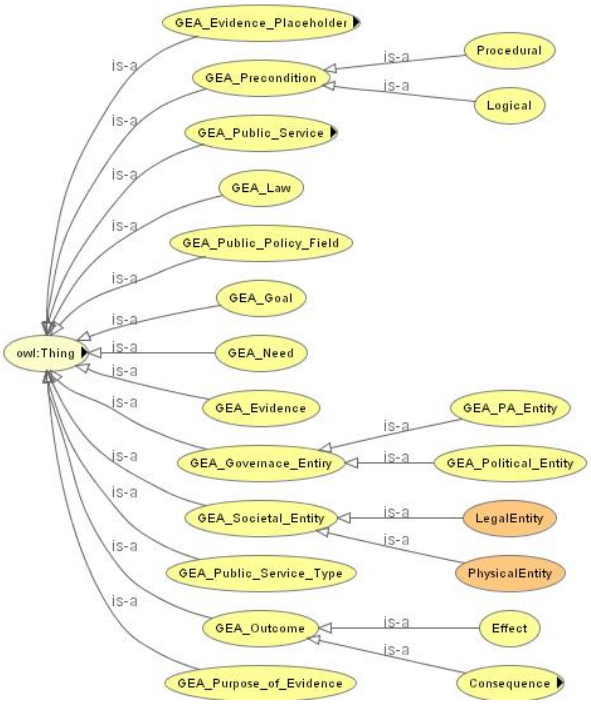


Figure 4-4: The GEA service ontology class hierarchy

[4] Pellet OWL Reasoner available at http://www.mindswap.org/2003/pellet

[5] http://protege.stanford.edu/

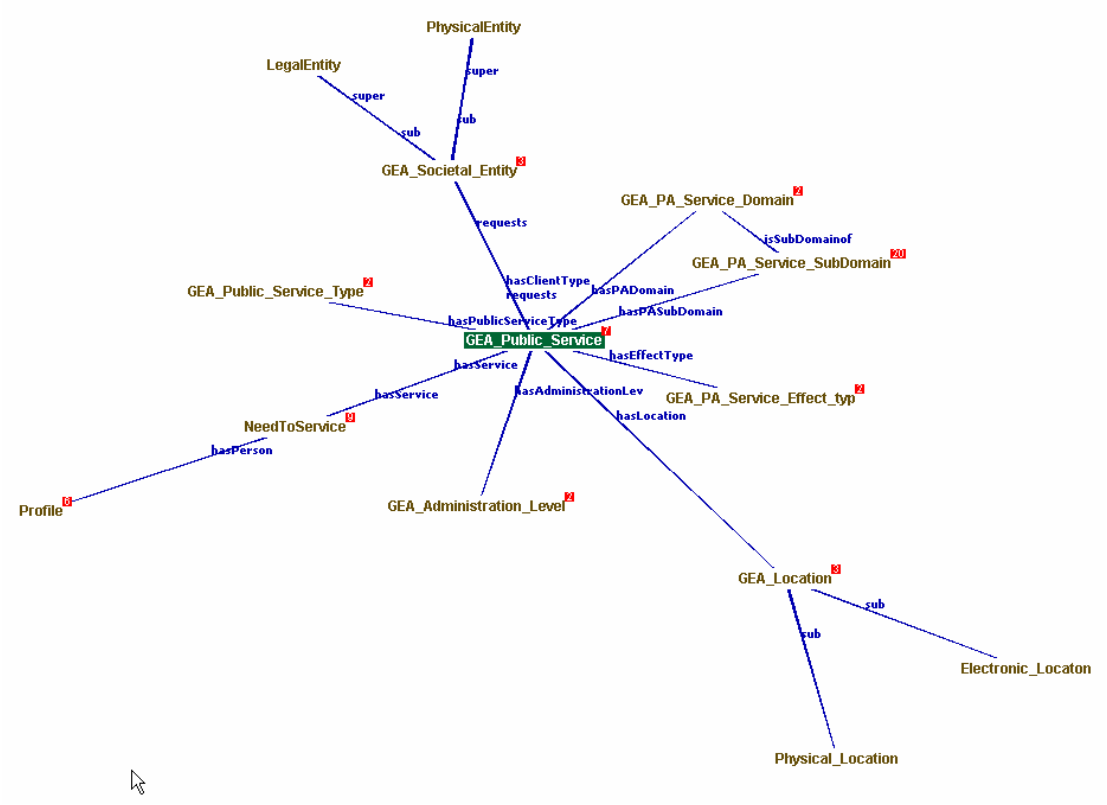


Figure 4-5: The GEA ontology main classes and object properties

A part of the GEA model class hierarchy in OWL DL is shown in fig. 4-16 and 4-17. The basic modeling principles followed were:

1) The GEA model entities were expressed in owl: Class elements
2) The relations between entities were expressed in `owl: ObjectProperty` Metaclasses. In cases were the relations were one-to-one, they were expressed in owl: `FuctionalProperty` metaclasses. For example a public service can have one administration level, therefore the property `hasAdministrationlevel` is defined as functional.
3) It is known that OWL DL uses the Open World Assumption so all classes at the same hierarchical level and all individuals that belong to the same class have to be explicitly declared as different. Therefore all the classes at the same hierarchy level were declared owl: `disjointWith`. The individuals that belonged to the same class were declared owl: `Alldifferent`.
4) In the GEA object model several has-type relations exist between entities. In some cases these were modeled in OWL as `rdfs: subClassOf`. In some other cases these relations were modeled using object properties. For example, the Output object was not modeled as a subclass of the Outcome object. The design decision that led to this was the fact that the Output object of the GEA model represents a documented decision therefore an `EvidencePlaceholder`. Two new object properties were created; `hasOutput` with domain `GEA_Public_Service` and range `GEA_Evidence_Placeholder`, and its inverse property `isOutputOf`. The `Effect` and `Consequence` objects were modeled as subclasses of the `Outcome` class.

5) The PA entity object in the GEA model has three distinct roles, `ServiceProvider`, `EvidenceProvider` and `ConsequenceReceiver`. These roles are depicted in OWL using three object properties. For example the `ServiceProvider` role is modeled using the `owl: ObjectProperty` providesServices with domain `GEA_PA_Entity` and range the `GEA_Public_Service` class (fig. 4-18).

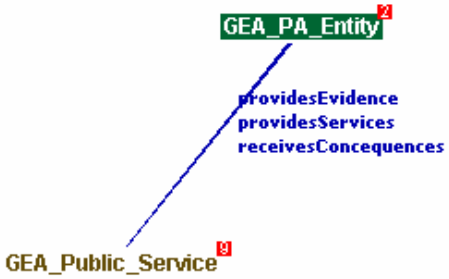


Figure 4-6: GEA_Public_Service and GEA_PA_entity class object properties

The classes that were not shown in fig. 4-16 are given in fig. 4-19. These classes represent the property descriptors. A property descriptor is a class which represents a certain attribute of a major class and is linked with an `owl:object` property.

For example in a wine ontology[6] Class Color is a property descriptor of the Wine class. Class Wine is linked with Color class with `hasColor` property of the `GEA_Public_Service` class.

Due to visualization reasons only the classes that are used in the application appear. Below follows a brief description of these classes.

[6] http://www.daml.org/ontologies/76

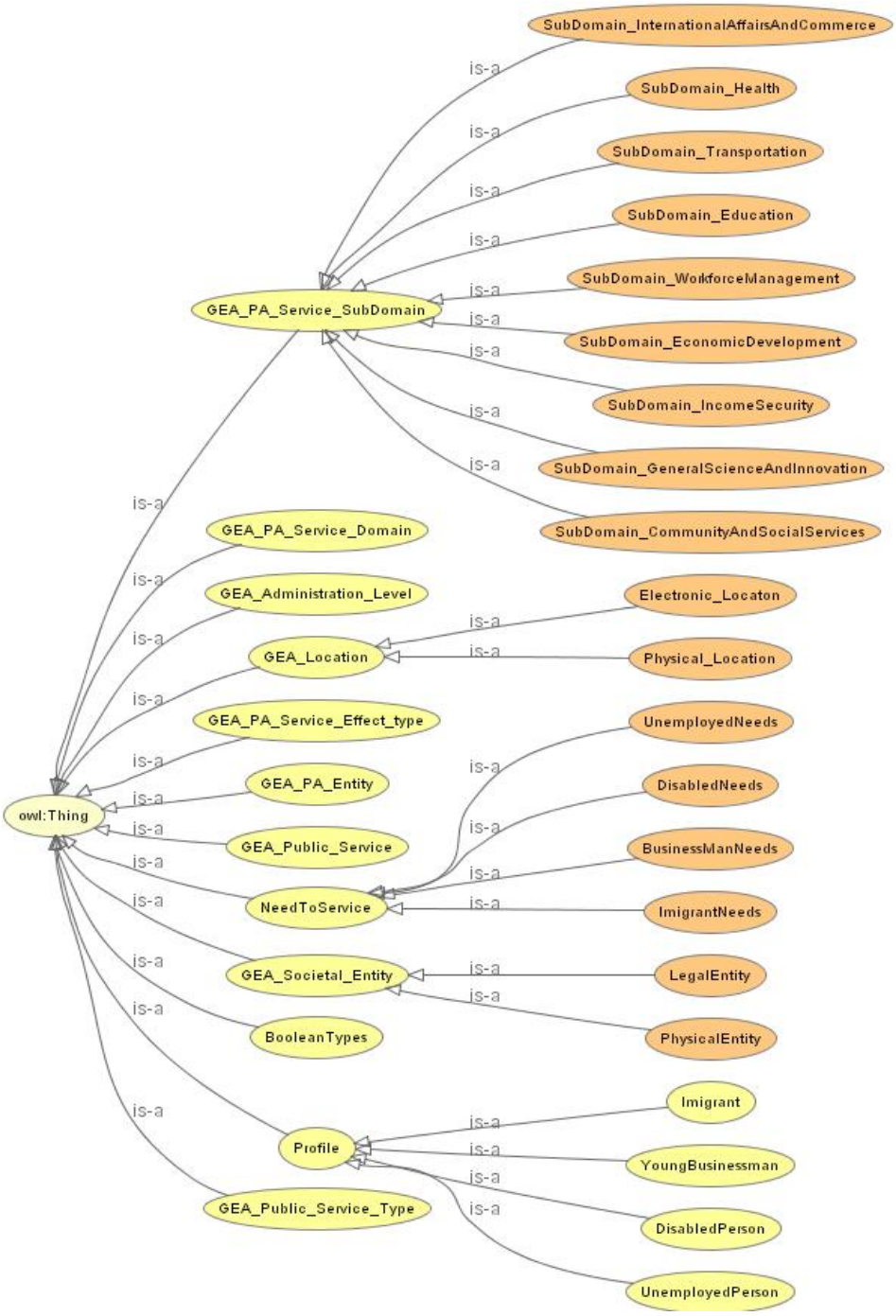


Figure 4-7: The GEA ontology class hierarchy

The `GEA_PA_Service_Domain` class represents the different PA domains. As already mentioned, this class has been populated with the following individuals taken from FEA;

- `CommunityAndSocialServices`
- `EconomicDevelopment`
- `Education`
- `Health`
- `Transportation`

- GeneralScienceAndInnovation
- IncomeSecurity
- InternationalAffairsAndCommerce
- WorkforceManagement

These individuals are declared different using the owl:all Different element. Each of these main domains has a number of sub-domains. These are represented by the GEA_PA_Service_SubDomain Class (equivalent to SubFunction concept in FEA).

Subclasses of this class have been created and they are modeled as an owl: EquivalentClass. Each subclass represents a PA domain and the individuals who belong to the class represent the PA sub-domains. For example, the definition of the SubDomain_Transportation Class is given in fig. 4-20 using OWL abstract syntax (Dean M., Schreiber G. et al. 2004).

```
Class(SubDomain_Transportation complete restriction(isSubDomainof value(Transportation)))

SubClassOf(SubDomain_Transportation GEA_PA_Service_SubDomain)
```

Figure 4-8: The SubDomain_Transportation class in OWL abstract syntax

The GEA_Administration_Level class is populated by four individuals Ministry_Level, Prefecture_Level, Municipality_Level and Region_Level. This class represents the unique administration level at which each public service is offered.

The GEA_Location represents the physical or electronic location where the public service is offered. For the physical location, a top-level location ontology can be imported (e.g. CIA World Fact Book in OWL http://www.daml.org/2003/09/factbook/us.owl)

The GEA_Public_Service_Effect_Type class represents at a high-level, the distinct effect types achieved by a public service. The individuals that belong to this class are

- ObtainSustainableDevelopment
- PromoteSocialWelfare
- SafeguardSocialContract

All the above public service descriptors are linked with individuals from the GEA_Public_Service class using owl:ObjectProperty elements with domain GEA_Public_Service and range the corresponding public service descriptor class.

These property descriptors correspond to Wine descriptors classes in the wine ontology e.g. Color, Flavor, Sugar… It is obvious that the GEA_Public_Service class is similar to the Wine class.

The GEA_Public_Service Class declaration is shown in fig. 4-21.

```
Class(GEA_Public_Service partial restriction(hasPASubDomain cardinality(1))
                                 restriction(hasEffectType cardinality(1))
                                 restriction(hasPublicServiceType allValuesFrom(oneOf(GEA_Authorization
                                                                                      GEA_Control
                                                                                      GEA_Production
                                                                                      GEA_Certification)))
                                 owl:Thing
                                 restriction(hasPADomain cardinality(1))
                                 restriction(hasAdministrationLevel cardinality(1))
                                 restriction(hasLocation minCardinality(1))
                                 restriction(hasClientType minCardinality(1)))
```

Figure 4-9: The GEA_Public_Service in OWL abstract syntax.

Fig. 4-22 shows GEA_Public_Service class example individuals from the Protégé individual editor. The individuals that belong to this class are public services. For example the public service individual that corresponds to the issuance of a parking license for a disabled person is shown in OWL abstract syntax in fig 4-23.

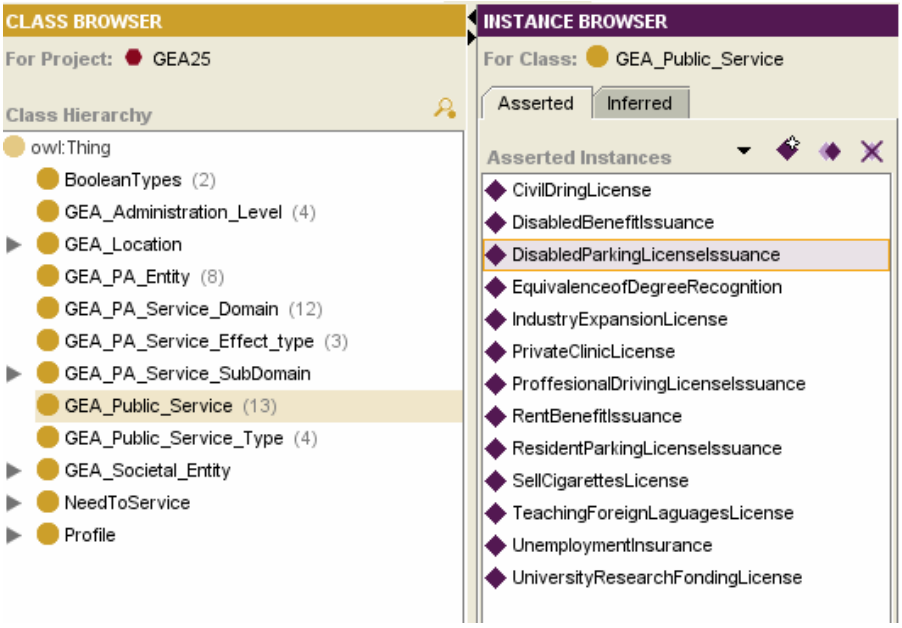


Figure 4-10: GEA_Public_Service class example individuals

```
Individual(DisabledParkingLicenseIssuance annotation(rdfs:comment "A service that issues  a parking license that is free for disabled
persons"^^<http://www.w3.org/2001/XMLSchema#string>)
                                        type(GEA_Public_Service)
                                        value(hasPADomain Transportation)
                                        value(hasClientType Citizen)
                                        value(hasPublicServiceType GEA_Certification)
                                        value(hasAdministrationLevel Municipality_Level)
                                        value(hasEffectType PromoteSocialWelfare)
                                        value(hasPASubDomain GroundTransportation))
```

Figure 4-11: The DisableParkingLicenceIssuance individual in OWL abstract syntax.

The Profile class represents the different user profiles (e.g. young businessman, immigrant, unemployed, disabled). This class is similar to the meal class in the wine ontology.

The three main classes that play a central role in the application are: the Profile class, the GEA_PublicService class and a new class which we call NeedToService class.

The NeedToService class links profiles to services. This is accomplished by using two owl:ObjectProperty elements, which are hasPerson and hasService. It is obvious that these properties have domain the NeedToService class and range the Profile and the GEA_PublicService class respectively. Additionally, both properties have owl:mincardinality restriction set at least to one, which means that each NeedToService class is linked to at least to one service and one profile. Again the similarity with the wine ontology is given by the fact that the NeedToService class corresponds to the MealCourse class.

The mapping of the GEA ontology classes to the wine example is given in the table below.

Wine Ontology Class	GEA ontology class
Wine	GEA_Public_Service

Meal	Profile
MealCourse	NeedToService

Table 4-1: Mapping of wine ontology classes to GEA needs to service ontology.

4.3.3 Use Case

At this point, we pass from the description of the application infrastructure to the specific example of our showcase, that is, we want to match the disabled person profile with public services that are available by PA agencies.

At the front-end, when the user enters the first page of the application, a list box containing a list of client profiles appears. The user selects a profile that he/she believes is best suited for him/her. Let us assume that the user selects the disabled person profile and wants to identify services that are suitable for her/him.

We assume that the municipal authority has – among others – a public service for the issuance of a free parking license for a disabled person (`DisableParkingLicenceIssuance`) and another service that issues a monthly benefit to disabled persons (`DisabledBenefitIssuance`). The first service has been described in fig. 4-23.

The main properties of the `GEA_Public_Service` Class that the application uses are shown in a graphical manner in fig. 4-24 below.

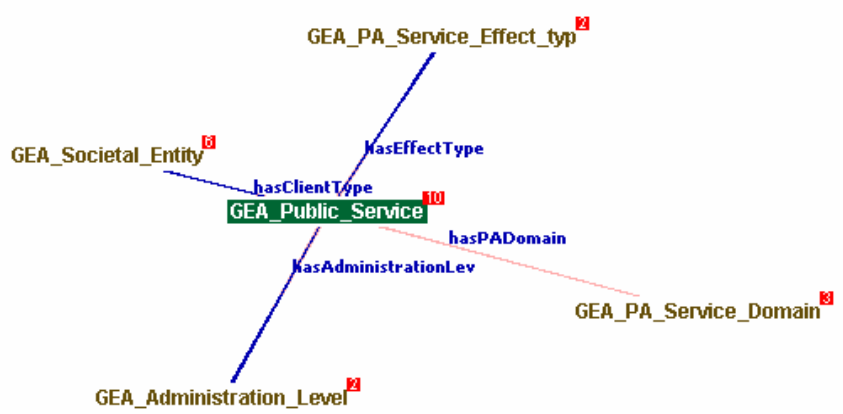


Figure 4-12: The main properties of GEA_Public_Service class

In order to match the specific profile to service(s), we have defined a new subclass of the `NeedToService` class, which we call `DisabledNeed`. Similarly in the wine ontology for every meal class a new `mealcourse` class has to be created to match certain wines. A new individual of this latter class is created on the fly (fig. 4-25). This is used to link the profile of the applicant (disabled person) to the relevant public services. This link is accomplished using `owl:allValuesFrom` restrictions to the values of the properties of the service class.

For example, as shown in the next figure, in order for a public service to be chosen all the restrictions should be satisfied, which means that the service should have "Citizen" as its client, "Promote Social Welfare" as an effect, "Municipality" as administrative level. Moreover, we have set two alternatives with regards to the `hasPADomain` property of the service. Specifically, we want the service to belong either to the `Community and Social Services` or to the `Transportation` domain. This is an assumption made for the showcase, as we assume that a

disabled person is interested to get information for services related to only in these two domains. Obviously the query could pose no restriction regarding the domain. We may also have added additional domains (e.g. `Education`) from which we could have additional services applicable and relevant to a disabled person.

```
Class(DisabledNeed complete restriction(hasPerson allValuesFrom(DisabledPerson))
                           NeedToService)

SubClassOf(DisabledNeed restriction(hasService allValuesFrom(restriction(hasClientType value(Citizen)))))
SubClassOf(DisabledNeed restriction(hasService allValuesFrom(restriction(hasEffectType value(PromoteSocialWelfare)))))
SubClassOf(DisabledNeed restriction(hasService allValuesFrom(restriction(hasAdministrationLevel value(Municipality_Level)))))
SubClassOf(DisabledNeed restriction(hasService someValuesFrom(restriction(hasPADomain value(CommunityAndSocialServices)))))
SubClassOf(DisabledNeed restriction(hasService someValuesFrom(restriction(hasPADomain value(Transportation)))))
```

Figure 4-13: The DisabledNeed class in OWL abstract syntax.

So, after the selection of the disabled profile, various queries are sent to the reasoner in two steps as described below. The system uses JSP (Java Server Page) and Java servlets in order to execute the user request and to dynamically create the results in HTML format respectively.

During the first step, the application asks for all the service property values. SPARQL queries are sent to the reasoner (Pellet) asking for all the properties of the public services (e.g. `gea:hasEffectType` or `gea:hasPADomain`) that are linked to the individual of the `DisabledNeed` class (fig. 4-25).

Such an example of a SPARQL query asking for the value of `hasPADomain` property is given in Table 4-5 below. In this case, the answer is `Transportation` and `CommunityAndSocialServices`, thus the reasoner returns two `PADomain` values: `gea:CommunityAndSocialServices` and `gea:Transportation`. This is due to the `DisabledNeed` class definition given earlier in fig. 4-25.

```
PREFIX gea:     < http://localhost/GEA.owl #>
PREFIX rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?Domain
WHERE { ? NeedToService rdf:type gea: NeedToService.
      ? NeedToService gea:hasService ?Service.
      ?Service gea:hasPADomain ?Domain.}
```

Table 4-2: SPARQL query for finding the PA Domain

In the same way, queries are sent for all the other properties of the `GEA_Public_Service` class, namely for the `hasEffectType`, `hasAdminstrationLevel` and `hasClientType` properties.

All the public services properties are returned by the reasoner for the specific `Profile` class individual, that is:

- `Has Domain`, `Transportation` and `Community And Social Services`.
- `Has EffectType`, `Promote Social Welfare`.
- `Has Administration Level`, `Municipality`
- `Has Client Type`, `Citizen`

The first step ends when all the needed property values of the relevant public service have been identified. Then a second querying step is triggered.

During the second step, the reasoner creates a new query that searches for public services that satisfy all the properties values found through the execution of the first querying step. In this way,

the reasoner returns the services that are found to match the selected profile. These services share the same property values with the profile selected.

There are cases where a property may be allowed to take more that one values. For example a profile may be connected to public services from different PA domains. In our example, this is true due to the fact that PA services from two PA domains are relevant as explained above (see fig. 4-18). These are the `Transportation` and `CommunityAndSocialServices` PA domains. To address these cases more than one query can be performed by the application. Each query is performed for every value of the PA domain property.

We present the example of the two queries performed for the `gea:hasPADomain` property in Tables 4-6 and 4-7. The first looks for services from the `Transportation` domain, while the second from the `CommunityAndSocialServices` domain.

```
PREFIX gea:  < http://localhost/GEA.owl #>
  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?Service
 WHERE {
 ?Service rdf:type gea:GEA_Public_Service.
 ?Service gea:hasPADomain gea:Transportation.
 ?Service gea:hasEffectType gea: PromoteSocialWelfare.
?Service gea:hasAdministrationLevel gea: Municipality_Level.
?Service gea:hasClientType gea: Citizen.}
```

Table 4-3: First SPARQL query for identifying Transportation Services

```
PREFIX gea:  < http://localhost/GEA.owl #>
  PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT  ?Service
   WHERE {
 ?Service rdf:type gea:GEA_Public_Service.
 ?Service gea:hasPADomain gea: CommunityAndSocialServices.
 ?Service gea:hasEffectType gea: PromoteSocialWelfare.
?Service gea:hasAdministrationLevel gea: Municipality_Level.
?Service gea:hasClientType gea: Citizen.}
```

Table 4-4: SPARQL query for identifying Social Services

After the query execution, the reasoner returns the public services `gea: DisabledParkingLicenseIssuance` and `gea: DisabledBenefitIssuance`. These are the two services that were found to be relevant to the profile entered into the application.

Interestingly, the first was found by the first SPARQL query (table 4-6) that searched in the transportation PA domain, while the second was found by the second query (table 4-7) that searched in the community and social services PA domain.

4.4 Conclusion and future work

The third case presented in this chapter deals with the problem of matching a client's need with public services that are available by PA agencies. This work is based on the GEA object model and uses semantic technologies for implementation purposes. The system concept is also similar to that used in the wine agent demo, as provided in http://www.daml.org/ontologies/76.

The GEA object model for public services (PA service model) has been used as the basis for a reference eGovernment domain ontology. An OWL DL ontology based on this model was created. This ontology serves as the knowledge base for the implementation of a Needs-to-Services semantic application. The user selects his/her profile and the application employs reasoning to find the public services that match the selected profile.

The GEA object model is proven to have the adequate expressiveness for supporting this match-making.

In this example several semantic technologies and tools have been used. In summary these are:

- OWL DL is the ontology language for the description of the GEA model
- Protégé Ontology Editor with OWL plug-in for the creation and editing of the ontology.
- Pellet for the required reasoning.
- Apache Tomcat for running the application.
- SPARQL as the query language for describing the queries to the reasoner.

Our experience with this example has shown that:

- OWL has the expressiveness to represent public administration domain specific knowledge and the GEA models.
- Protégé is yet the most efficient freeware ontology editor, but it is not bug-free. The ontology validation with an external DIG reasoner is a useful feature as it allows the reasoner to be accessed from any web location. Pellet provides support for DIG compliant reasoning and for several query languages (RDQL[7], SPARQL[8]). The main problem with Pellet is the lack of support for datatype reasoning. This results for example in limitations with regards to reasoning on numeric values.
- SPARQL (Seaborne A. and Prud'hommeaux E. 2005) is a relatively new W3C submission which provides an easy to use SQL-like RDF query language. The lack of update commands SQL update, delete) is still a disadvantage.
- The technologies used offered the capabilities to create a semantic application, but most of them are still on-going research projects.

This application is the first step towards semantic-based public service discovery. In a second stage the application should be able not only to discover, but also to execute some of the public services found in this way. For this, an approach based on a complete Semantic Web Services framework (e.g. WSMO) and an execution environment (e.g. WSMX) is needed.

Another issue that is not addressed here is the storage of the ontology in a repository. A large number of public services exist in public administration, therefore their ontological representation and storage requires an efficient and proven technology for storage. Performance issues are also of great importance for this type of reasoners and repositories. These issues will be part of our future work in the field.

---

[7] http://www.w3.org/Submission/RDQL/

[8] http://www.w3.org/TR/rdf-sparql-query/

5. REFERENCES

Dean M., Schreiber G., et al. (2004). OWL web ontology language reference, W3C Recommendation.

Dimitrios Tektonidis, Albert Bokma, et al. (2005). ONAR: An ontologies-based service oriented application integration framework. 1st International Conference on Interoperability of Enterprise Software and Applications, Geneva, Switzerland.

EU-Publi.com Consortium (2005). D4.1, Report on the prototype of the EU-Publi.com showcase. Thessaloniki, Greece.

European Commission (2004). DECISION OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 21 April 2004 on interoperable delivery of pan-European eGovernment services to public administrations, businesses and citizens (IDABC). **2004/387/EC**.

Goudos S., Peristeras V., et al. (2006). Mapping Citizen Profiles to Public Administration Services Using Ontology Implementations of the Governance Enterprise Architecture (GEA) models. Semantic Web for eGovernment workshop, European Semantic Web Conference (ESWC) 2006, Budva, Montenegro.

Holger K., Fergerson Ray W., et al. (2004). The Protégé OWL Plugin: An Open Development Environment for Semantic Web Applications. 3rd International Semantic Web Conference - ISWC 2004, Hiroshima, Japan.

IDA and CapGemini (2004). Architecture for delivering pan-European e-Government services, v.1.0.

IDABC (2004). European Interoperability Framework. Luxembourg, European Communities.

M. Paolucci et al. (2002). Semantic Matching of Web Services Capabilities. The SemanticWeb–ISWC 2002: 1st Int'l Semantic Web Conf.

OASIS - UDDI Spec TC (2002). UDDI Version 2.03 Data Structure Reference, UDDI Committee Specification.

Paolucci M. et al. (2003). Importing the Semantic Web in UDDI. E-Services and the Semantic Web Workshop.

Pardo, T. A., A. M.Cresswell, et al. (2004). Modeling the Social & Technical Processes of Interorganizational Information Integration. Proceedings of the 37th Hawaii International Conference on System Sciences.

Peristeras V. and Tarabanis K. (2005). Providing Pan-European eGovernment Services with the use of Semantic Web Service Technologies: A Generic Process Model. Electronic Government, DEXA, 4th International Conference EGOV 2005, Copenhagen.

Seaborne A. and Prud'hommeaux E. (2005). "SPARQL Query Language for RDF, W3C Working Draft (work in progress)." Retrieved 19 April, 2005, from http://www.w3.org/TR/rdf-sparql-query/.

Su, S. e. a. (2004). A Prototype System for Transnational Information Sharing and Process Coordination. dg.o 2004, Seattle, Washington, USA.

TopQuadrant (2005). FEA Refererence Model Ontologies (FEA RMO) v1.1.