Deploying Semantic Web Services-Based Applications in the e-Government Domain

Book Section

Deploying Semantic Web Services-based Applications in the e-Government Domain

Alessio Gugliotta[1], John Domingue[1], Liliana Cabral[1], Vlad Tanasescu[1], Stefania Galizia[1], Rob Davies[2], Leticia Gutierrez Villarias[2], Mary Rowlatt[2], Marc Richardson[3], Sandra Stincic[3]

[1] Knowledge Media Institute, The Open University,
Walton Hall, Milton Keynes, MK7 6AA, UK
{a.gugliotta, j.b.domingue, l.s.cabral, v.tanasescu, s.galizia}@open.ac.uk
[2]Essex County Council, County Hall,
Chelmsford, CM1 1LX, UK
{Leticia.gutierrez, maryr}@essexcc.gov.uk
rob.davies@mdrpartners.com
[3]BT Exact
Adastral Park Martlesham, Ipswich IP5 3RE, UK
{marc.richardson, sandra.stincic}@bt.com

Abstract. Joining up services in e-Government usually implies governmental agencies acting in concert without a central control regime. This requires to the sharing scattered and heterogeneous data. Semantic Web Service (SWS) technology can help to integrate, mediate and reason between these datasets. However, since a few real-world applications have been developed, it is still unclear which are the actual benefits and issues of adopting such a technology in the e-Government domain. In this paper, we contribute to raising awareness of the potential benefits in the e-Government community by analyzing motivations, requirements and expected results, before proposing a reusable SWS-based framework. We demonstrate the application of this framework by showing how integration and interoperability emerge from this model through a cooperative and multi-viewpoint methodology. Finally, we illustrate added values and lessons learned by two compelling case studies: a change of circumstances notification system and a GIS-based emergency planning system, and describe key challenges which remain to be addressed.

Keywords: e-Government, Semantic Web Services, Case Study, GIS, Change of Circumstances.

1 Introduction

To a large extent, tiers of government – such as national, county, and district – operate autonomously, without central control of service provision. Additionally, they each have distinct viewpoints which may differ from that of general citizens.

Therefore, integration and interoperability are significant requirements in the development of service-oriented applications in the e-Government domain.

Integration leads to “form a temporary or permanent larger unit of government entities for the purpose of merging processes and/or sharing information” [18]. In particular, this requires the assembly and transformation of processes needed to support specific user tasks into a single service with the corresponding back-office practices. As a result, interoperation among multiple government entities at different levels occurs “whenever independent or heterogeneous information systems - or their components - controlled by different jurisdictions/administrations or by external partners smoothly and effectively work together in a predefined and agreed upon fashion” [18].

Interoperability is a key issue in order to allow for data and information to be exchanged and processed seamlessly across governments. A working paper by the Commission of European Communities [6] emphasized its role in e-Government, not only as a technical issue concerned with linking up computer networks, but also as a fundamental requirement to share and re-use knowledge between networks, and re-organize administrative processes to better support the services themselves. Additionally in [7], the following three levels of interoperability were individuated:

I. *Technical*: concerning with the technical issues of linking up computer systems, the definition of open interfaces, data formats and protocols, including telecommunications;
II. *Semantic:* concerning with the exchange of information in an understandable way - whether within and between administrations, either locally or across countries and with the enterprise sector - by any other application not initially developed for this purpose.
III. *Organizational:* concerning with modelling business processes, aligning information architectures with organizational goals and enabling processes to co-operate, by rewriting rules for how governmental agencies work internally, interact with their customers, and use Information and Communication Technologies (ICT).

The semantic Web [3] can alleviate integration and interoperability issues by creating a universal medium for information exchange and by giving meaning (semantics) to contents on the Web, in a manner understandable by machines. The semantic Web moreover allows the development of easy to use applications and transparent access to services and data. In particular, Semantic Web Services (SWS) technology [31], [8] provides an infrastructure in which new services can be added, discovered and composed continually, and the organization processes automatically updated to reflect new forms of cooperation [16]. SWS combine the flexibility, reusability, and universal access that typically characterize Web services with the expressivity of semantic mark-up and reasoning, in order to make feasible the invocation, composition, mediation, and automatic execution of complex services with multiple paths of execution and levels of process nesting.

The adoption of SWS in e-Government therefore appears to be a natural development. However, demonstrating this to the e-Government community requires the achievement of several prerequisites: (a) creating compelling demonstrators and prototypes; (b) establishing visible standards; (c) developing stable and mature technology and products; (d) proving convincing business cases.

In our work, a close collaboration has been established with the *Essex County Council (ECC)* - a large local authority in South East England (UK) comprised of 13 boroughs and containing a population of 1.3M – to deploy real-world applications in the e-Government domain. During this collaboration, we developed, tested and refined a specific framework designed around an existing SWS broker: IRS-III [8]. In this paper, we report our experience by firstly introducing the devised approach and then focusing on the obtained results. The main contributions are to provide a proof of concept of the added values introduced by SWS in real-world application scenarios, propose a guide for the deployment of new e-Government applications, test the IRS-III approach with complex use cases and outline future research directions on the basis of the lessons learned.

The rest of the paper is structured as follows: Section 2 briefly introduces the technologies at the basis of our work: Web services, ontologies and SWS; in Section 3 we discuss the rationales that prompted our work by identifying motivations, requirements and expected results of matching two present-day research areas: SWS and e-Government; Section 4 and Section 5 provide an overview of IRS-III and our framework for creating SWS-based applications; Section 6 details and demonstrates our approach through two e-Government applications. On the basis of these two implementations, we summarize the lessons learned and point out the open challenges in Section 7. Finally, Section 8 describes the related work and Section 9 reports our conclusions.

2 Web Services, Ontologies, and Semantic Web Services

From an information technology viewpoint the two important features of *Web Services* are that: (a) they are accessible over the Internet using standard XML-based protocols and (b) the interface of a Web service encapsulates its actual implementation. The first feature gives Web services high availability whereas the second feature facilitates reusability and interoperability since interface descriptions are independent from software platforms.

From a business perspective one key feature is that Web services can be used to expose the business services – i.e. value-producing activities directly accessible by the customer - of an organization. For example, Google [13] has a Web service interface to its search engine and Amazon allows software developers to directly access their technology platform and product data [2]. The ability to couple business services to Web accessible software components will have profound effects on the nature of business and on the structure of participating organisations.

Three main technologies are currently used to implement Web services: SOAP [28], WSDL [36] and UDDI [33]. SOAP is an XML based, stateless, one-way message exchange paradigm for interacting with Web services. SOAP messages are transported over HTTP and are composed of two elements: a header and a body. WSDL is also an XML-based format and defines services as collections of network endpoints or ports. UDDI is a registry which allows clients to find Web services through descriptions of theirs entities, provided functionalities or via technically oriented aspects.

A key problem with the above technologies is that they are purely syntactic. They thus rely on human developers to understand the intended meaning of the descriptions and to carry out the activities related to Web service use.

The *semantic Web* [3] is an extension of the current Web where documents incorporate machine processable meaning. The overall semantic Web vision is that one day it will be possible to delegate non-trivial tasks, such as booking a holiday, to computer based agents able to locate and reason with relevant heterogeneous online resources. One of the key building blocks for the semantic Web is the notion of an ontology [14]. An *ontology* is an explicit formal shared conceptualization of a domain of discourse. More specifically, an ontology captures the main concepts and relations that a community shares over a particular domain. Within the context of the semantic Web, ontologies facilitate interoperability as the underlying meaning of terms within a Web document can be made explicit for computer based agents to support processing.

Semantic Web Services (SWS) research aims to automate the development of Web service based applications through the semantic Web technologies. By providing formal descriptions with well defined semantics, SWS facilitate the machine interpretation of Web service – functional and not functional - properties. The research agenda for SWS identifies a number of key areas of concern, namely:

- *Discovery:* finding the Web service which can fulfil a task. Discovery usually involves matching a formal task description against semantic descriptions of Web services.
- *Mediation*: we can not assume that the software components which we find are compatible. Mediation aims to overcome all incompatibilities involved. Typically this means mismatches at the level of data format, message protocol and underlying business processes.
- *Composition*: often no single service will be available to satisfy a request. In this case we need to be able to create a new service by composing existing components. Artificial Intelligence (AI) planning engines are typically used to compose Web service descriptions from high goals.

3 Motivations, Requirements, and Expected Results

In our work, we address the following two research questions: (a) how can semantic Web support interoperability and reuse of software components available on the Web? (b) How can SWS support e-Government? In the following, we detail these two perspectives by analyzing motivations, requirements, and expected results of moving from SWS to e-Government (Section 3.1) and from e-Government to SWS (Section 3.2), respectively.

3.1 From Semantic Web Services to e-Government

Currently, one of the main needs of SWS technology is the development of real-world applications that demonstrate its added (business) values. The next application-driven research challenge thus can be defined only through the feedback from practical prototypes and applications. The full potential application of SWS requires many more large-scale testing domains.

Since it is an enormous challenge to achieve interoperability and to address semantic differences related to the great variety of datasets and information technology solutions which should be networked, e-Government may be a very effective test-bed for evaluating SWS technology. E-Government moreover exhibits further significant characteristics which may indicate several research issues for SWS. For example, e-Government is characterized by top-down prescribed constraints in key areas (e.g. laws, legal requirements, policies in the use of services and access to data); limited central control; strong requirements to come to same decisions in similar situations; high requirements for non-functional properties such as security, privacy, and trust; wide information imbalances between stakeholders, as well as multiple and heterogeneous stakeholders involved in the same process.

3.2 From e-Government to Semantic Web Services

The ability to aggregate and reuse diverse information resources relevant to a given situation in a cost-effective way and to make this available as a basis for transparent interaction between community partner organizations and individual citizens is a key benefit that SWS technology can provide to e-Government. Specifically, SWS technology promises to:

a) *Provide added value joined up services:* allowing software agents to create interoperating services transparently to the users and hence automate integration, reasoning and mediation among heterogeneous data sources and processes available at distinct governmental levels.
b) *Enable formalization of government business processes in an unambiguous structure:* allowing the creation of a common understanding of processes and visualization of the knowledge involved. This could eventually lead to a reengineering of the governmental systems and simplification of processes.
c) *Reduce risk and cost*: (i) moving from “hard coding” services to reusable functionalities through, for example, utility computing of shared services (e.g. payment platforms, legal resources, etc.); (ii) keeping government organizations’ autonomy in the description/management of their domains; (iii) increasing flexibility; enabling discovery of new or previously unknown services; (iv) aggregating services on the basis of user preferences; (v) providing better service to third-parties and customers; (vi) easily addressing the evolution and change of existing services and scenario.
d) *Provide better support to front line:* allowing one-stop, customer focused, and multiple viewpoint access to services and shared information.

The e-Government community (stakeholders, administrations, end-users, but also researchers) needs to perceive these benefits more clearly before it will adopt the technology. At present, Web services are being introduced as infrastructure (often experimental) in some areas of government and the broad awareness of need for semantic enrichment is increasing. However, since SWS are completely new – and are mainly visible to the academic and industrial research ‘e-Government’ sector - a measurable benefit to service and achievable cost savings, or “cashable benefits” will need to be established.

In absence of golden standards, demonstrating real-world applications is the important first step to accomplish this goal. Perhaps more importantly, this may provide a way to address existing barriers and perceptions, such as:

e) *Trust in automated data sharing*: governmental organizations are concerned about: (i) ownership, control and quality among service providers; (ii) security, data protection, confidentiality, and privacy issues.

f) *Patchy awareness of Web services*: stakeholders are often unclear about the distinction between Web services and general services available via Web.

g) *Up-front Infrastructure costs* (e.g. investment in Web Services): governmental organizations are reluctant to be the pioneers which take the initial financial ‘hit’ in implementing SWS, as with almost any new technology.

h) *Market development:* in terms of raising the awareness of potential SWS benefits in e-Government, increasing pilot applications, and promoting the availability of working SWS platforms.

4 IRS-III: A broker-based approach for SWS

IRS-III [8] is a platform and broker for developing and executing SWS. By definition, a broker is an entity which mediates between two parties and IRS-III mediates between a service requester and one or more service providers. To achieve this, IRS-III adopts a semantic Web based approach and is thus founded on ontological descriptions. At the heart of IRS-III there is the SWS Library*,* where semantic descriptions of various aspects of Web services, reference Domain Ontologies and Knowledge bases (instances) are stored using OCML representation language [23]. Specific IRS-III components interpret such descriptions to discover and select the appropriate Web service, choreograph and ground to the Web service operations [9], orchestrate multiple Web services, and mediate semantic descriptions by running mediation rules or invoking mediation services [5]. Note that IRS-III supports grounding to standard Web services with a WSDL description, as well as stand-alone Java and Lisp code. Similarly, Web applications accessible as HTTP GET requests are handled internally by IRS-III.

4.1 The IRS-III service ontology

The IRS-III service ontology forms the epistemological basis for IRS-III and provides semantic links between the knowledge level components describing the capabilities of a service and the restrictions applied to its use. The IRS-III service ontology is based on the WSMO standard [37] which specifies the following main aspects:

- *Non-functional properties*: these properties are associated with every main component model and can range from information about the provider such as organisation, to information about the service such as category, cost and quality of service, to execution requirements such as scalability, security or robustness.

- *Ontologies*: provide the foundation for describing domains semantically. They are used by the three other WSMO components.
- *Goal-related information*: a goal description represents the user perspective of the required functional capabilities. It includes a description of the requested Web service capability.
- *Web service-related information:* a Web service interface represents the functional behavior of an existing deployed Web service. It includes a description of: (a) *Functional capabilities* which represent the provider perspective of what the service does in terms of assumptions, effects, pre-conditions and post-conditions. Capabilities are expressed by logical expressions that constrain the state or the type of inputs and outputs. (b) *Choreography* which specifies how to communicate with a Web service. (c) *Grounding* which is part of the Web service choreography and describes how the semantic declarations are associated with a syntactic specification such as WSDL. (d) *Orchestration* which specifies the decomposition of Web service capability in terms of the functionality of other Web services.
- *Mediators*: in WSMO, a mediator specifies which WSMO top elements are connected and which type of mismatches can be resolved between them. WSMO defined four kinds of mediators: *GG-mediator* which links different goals; *WG-mediator* which connects Web services with goals; *OO-mediator* which enables components to import heterogeneous ontologies; and *WW-mediator* which links Web services to Web services.

The WSMO conceptual model has been represented using OCML representation language [23] and extended in the following ways:

- *Explicit input and output role declaration*: IRS-III requires that goals and Web services have input and output roles, which include a name and a semantic type. The declared types are imported from domain ontologies. This makes the definition of goal and Web services easier when complex choreographies are not required.
- *Web services are linked to goals via WG-mediators:* if a WG-mediator associated with a Web service has a goal as a source, then this Web service is considered to solve that goal. An assumption expression can be introduced for further refining the applicability of the Web service.
- *GG-mediators provide data-flow between sub-goals* – in IRS-III, GG-mediators are used to link sub-goals within an orchestration and so they also provide dataflow between the sub-goals.
- *Web services can inherit from goals* - Web services which are linked to goals 'inherit' the goal's input and output roles. This means that input role declarations within a Web service are not mandatory and can be used to either add extra input roles or to change an input role type.
- *Client choreography* – the provider of a Web service must describe the choreography from the viewpoint of the client. This means IRS-III can interpret the choreography in order to communicate with the deployed Web service.
- *Mediation services are goals* – a mediator declares a goal as the mediation service which can simply be invoked. The required transformation is performed by the associated Web service.

- *IRS-III component goals* – the main components of IRS-III (e.g. the orchestration and choreography interpreters and the handlers for the different WSMO mediators) are implemented using internal goal, Web service and mediator descriptions. Additionally, a number of utility goals, for example a number of arithmetic and list primitives are incorporated.

4.2 The IRS-III Core Functionalities

A core design principle for IRS-III is to support capability-based selection and invocation of Web services. A client sends a request which captures a desired outcome or goal and, using the set of semantic Web service descriptions introduced in the previous section, IRS-III will:

F1. Discover potentially relevant Web services.
F2. Select the set of Web services which best fit the incoming request.
F3. Invoke the selected Web services whilst adhering to any data, control flow and Web service invocation constraints.
F4. Mediate any mismatches at the data, goal or process level.

In the following sub-sections, we highlight the main aspects associated with the aforementioned functionalities.

4.2.1 Discovery

As introduced in Section 4.1, IRS-III makes use of WG-mediators to link a goal to all Web services that can solve it. Figure 1 depicts the specific ontology concepts and relations involved in the IRS-III discovery and selection.

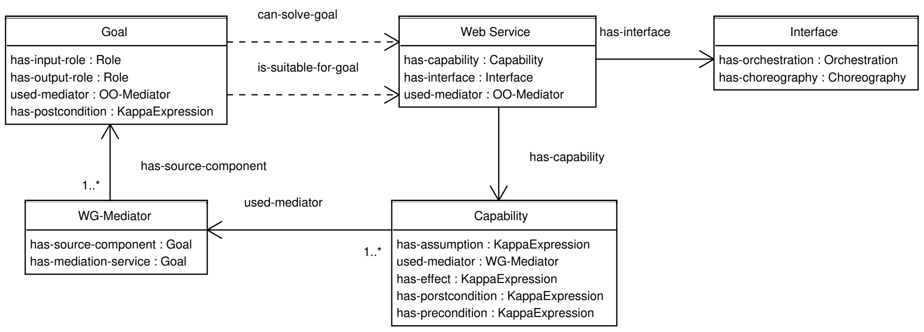

Goal
has-input-role : Role
has-output-role : Role
used-mediator : OO-Mediator
has-postcondition : KappaExpression
can-solve-goal
is-suitable-for-goal
Web Service
has-capability : Capability
has-interface : Interface
used-mediator : OO-Mediator
has-interface
Interface
has-orchestration : Orchestration
has-choreography : Choreography
has-capability
has-source-component
1..*
WG-Mediator
has-source-component : Goal
has-mediation-service : Goal
used-mediator
1..*
Capability
has-assumption : KappaExpression
used-mediator : WG-Mediator
has-effect : KappaExpression
has-porstcondition : KappaExpression
has-precondition : KappaExpression

Figure 1. The ontology concepts and relations involved in the IRS-III discovery.

Given a goal, multiple WG-mediators can define such a goal as their source component. In turn, distinct capability descriptions can refer to a specific WG-mediator and thus link to a goal. Finally, each capability description is part of a unique Web service description.

On the basis the semantic descriptions above, a pool of Web services that potentially satisfy a given goal is identified by a backward chaining rule-based reasoning. In particular, the can-solve-goal relation is inferred at runtime during the goal achievement process. The listing below shows the OCML [23] definition of the can-solve-goal relation. The sufficient conditions of the definition (:sufficient) specify the clauses to be proved when inferring such a relation. The IRS-III interpreter will fire the clauses in the order in which these are listed, by finding any instance which makes true the specific clause. As a result, starting from a goal instance given in input (?goal), it is possible to identify: (i) all of the WG-mediators (?mediator) which use such a goal as source component[1]; (ii) the capabilities (?capabilities) which use the identified WG-mediators; (iii) the Web services (?thing) which define the identified capabilities.

```
(def-relation can-solve-goal (?goal ?thing)
  "Returns the web services which solve a goal.
   Uses the mediator to find the link"
  :sufficient (and (instance-of ?goal goal)
                   (= ?goal (the-slot-value ?mediator has-source-component))
                   (instance-of ?mediator WG-mediator)
                   (= ?mediator (the-slot-value ?capability used-mediator))
                   (instance-of ?capability capability)
                   (= ?capability (the-slot-value ?thing has-capability))
                   (instance-of ?thing web-service)))))
```

4.2.2 Selection

The selection process aims to identify the most appropriate Web services which satisfy a goal, starting from the results of the previous phase (can-solve-goal relation). On the basis of the current goal inputs, the IRS-III interpreter will test the applicability conditions of each discovered Web service.

The listing below shows the suitable-web-service-goal which is invoked to check if a Web service is satisfactory for a specific goal invocation[2].

```
Suitable-web-service-goal
Input Role
      has-goal goal "sexpr"
      has-actual-role-pairs list "sexpr"
      has-web-service web-service "sexpr"
      has-combined-oo-mediator-ontology ontology "sexpr"
Output Role
      is-suitable-web-service boolean "sexpr"
Post Condition
      (kappa (goal-inst)
      (== (has-role-value goal-inst is-suitable-web-service)
          (is-suitable-for-goal
               (instantiate (has-role-value goal-inst has-goal)
                            (has-role-value goal-inst has-actual-role-pairs))
                web-service)))
```

Suitable-web-service-goal has four input roles which respectively represent: (a) the current goal; (b) the values for the current goal's input roles; (c) the Web service under consideration; and, (d) an ontology created specifically for the goal

[1]The the-slot-value function returns the value of a specific slot (e.g. has-source-component) of an instance (e.g. ?mediator).

[2]As mentioned earlier, IRS-III components themselves are modelled using WSMO descriptions.

invocation. Note that for each input role, we specify the type of values permissible and a SOAP grounding (`sexpr` in the listing above) which is 'inherited' by Web services linked through a WG-mediator. Moreover, the ontology created in step (d) combines the goal and Web service ontologies, making use of OO-mediators – both goal and Web service descriptions refer to OO-mediators (Figure 1) - to resolve any data mismatches (Section 4.2.4).

The output role (`is-suitable-web-service`) is a boolean value which is true if the Web service is suitable for the goal instance, false otherwise.

The post condition expresses the expected result as an OCML anonymous relation, called *kappa expression*. The latter takes as argument the `suitable-web-service-goal` itself and is satisfied if its clauses hold for the given argument. In the given example, the `is-suitable-for-goal` relation is used to state the relationship between the considered goal and the selected Web services.

To accomplish the `suitable-web-service-goal` introduced above and thus infer the `is-suitable-for-goal` relation, an internal IRS-III Web service is invoked. The latter exposes an OCML function which performs the following tasks: (i) retrieving the applicability conditions – currently the assumptions defined in the WSMO capability description – of a given Web service and (ii) testing the applicability conditions according to the input roles defined in the given goal instance. Checking the following OCML relation is the core of such a function.

```
(def-relation applicable-to-goal (?web-service ?goal)
  :iff-def (or (not (and (= ?capability
                             (the-slot-value ?web-service has-capability))
                          (instance-of ?capability capability)
                          (= ?exp (the-slot-value ?capability has-assumption))
                          (not (= ?exp :nothing))))
               (and (= ?capability
                        (the-slot-value ?web-service has-capability))
                    (instance-of ?capability capability)
                    (= ?exp (the-slot-value ?capability has-assumption))
                    (not (= ?exp :nothing))
                    (holds ?exp  ?goal))))
```

Sufficient and necessary conditions of the definition above (`:iff-def`) specify the clauses to be proved. Similar to the `can-solve-goal` relation introduced in Section 4.2.1, the IRS-III interpreter will fire the clauses. The `or` expression of the definition introduces two main cases[3].

The first case manages the situation of Web services that do not define any assumption. We assume that Web services which do not define assumptions are applicable to the goal. In this way, for example, we can deal with general purpose Web services.

The second case manages the situation of Web services that define assumptions. The `?exp` variable captures the stated assumption which is expressed as a *kappa expression* (e.g. the goal post condition defined above). The `holds` function invokes the IRS-III interpreter to test the retrieved kappa expression, using the current goal instance `?goal` as given parameter. If the kappa expression is satisfied, the Web service is applicable to the goal.

[3] As in Prolog, depth-first search with chronological backtracking is used in OCML to control the proof process.

Note that several Web services can be selected. The current IRS-III policy is invoking the first Web service of the list, since a ranking mechanism is not defined. However, future work concerns improving current IRS-III selection with trust-based mechanisms [12].

4.2.3 Invocation, Choreography and Orchestration

According to the WSMO model, the IRS-III interface provides information on how the functionality of the deployed Web services is achieved, and, as stated in Section 4.1, the main interface components are orchestration and choreography. The semantic descriptions of the interface model are interpreted by IRS-III when the latter identified the Web service to satisfy a goal. According to such descriptions, specific actions are performed.

The overall view is that Web service execution consists of a number of discrete steps, and, at any given point, the next action performed within an interface execution will depend upon the current state. IRS-III performs its interface abstract model through the tuple $\langle E, S, C, T \rangle$, where: E is a finite set of events; S is the (possibly infinite) set of states; C is the (possibly infinite) set of conditions; T represents the (possibly infinite) set of transitions rules.

The ***events*** represent actions performed during the interface execution. The subset of events from E which can occur in choreography and orchestration differs. Specifically, $E = Ec \cup Eo$: where Ec is the set of choreography events; and Eo is the set of orchestration events. In more detail, *Ec = {obtain, present, provide, receive, obtain-initiative, present-initiative}* [9]. Every choreography event maps to an operation during the conversation viewed from the IRS-III perspective. Similarly, the set of possible orchestration events are *Eo = {invoke-goal, invoke-mediator, find-mediator, evaluate-logical-expression, return-output}*.

Given a transition step Ti, a ***state*** $s_i \in S$ is a non-empty set of ontologies that define a state signature over which transition rules are executed. Optional mediators are used to solve ontology or data mismatches (Section 4.2.4). The parameterized *choreography state* is a set of instances, concerning message exchange patterns and the choreography execution. Every state includes a constant subset, which identifies the Web service host, port, and location, which is invariant whenever the same Web service is invoked, and the event instantiation $e \in Ec$, dependent on the event which occurred at step Ti. The *orchestration states* characterize the phases of the workflow process during goal decomposition. Given a transition step Ti, an orchestration state contains a description of the triggering-event, the control flow step identifier, and the result - the output of the achieved sub-goal.

A ***condition*** $c \in C$ (also called guard) depicts a situation occurring during interface execution. Every constraint within the condition has to be verified before the next event is triggered.

The ***transition rules*** express changes of state by modifying a set of instances within the signature ontology. In particular, a transition rule, $t \in T$, updates the state after the occurrence of an event, $e \in E$, and consists of a function, $t : \left(S, 2^C\right) \xrightarrow[E]{} S$, that associates a couple $(s, \{c_1, .., c_n\})$ to s', where s and $s' \in S$, and every $c_i \in C$ *(1*

$\leq\ i \leq\ n)$. *Choreography transition rules* are defined with the following two specific restrictions: (a) '***If*** rules do not chain and are of the form "***If*** condition ***then Fire*** Event"; and (b) conditions are mutually exclusive so only one rule can fire at a time. These rules represent the interaction between IRS-III and the Web service and are applied when executing the choreography. *Orchestration transition rules* provide a workflow model based on the following set of control flow constructs: *sequence, conditional, loop, fork, join.* These rules describe the model of a composed Web service. The distinguishing characteristic of this model is that the basic unit within composition is a goal. Further, dataflow and the resolution of mismatches between goals are supported by mediators.

4.2.4 Mediation

The overall design goal for IRS-III is to act as a semantic broker between a client application and deployed Web services available at large on the internet. This brokering activity can be seen as mediation itself, which in IRS-III is further broken down into goal, process and data mediation [5]. *Goal Mediation* takes places during F2, and the types of mismatches that can occur are: the input types of a goal are different from the input types of the target Web service; and Web services have more inputs than the goal. A WG-mediator is mainly involved in this mediation. *Process Mediation* takes places during F3 – specifically, during orchestration - and the types of mismatches which can occur are: output types of a sub-goal are different from the input types of the target sub-goal; output values of a sub-goal are in a different order from the inputs of the target sub-goal; and, the output of a sub-goal has to be split or concatenated into the inputs of the target sub-goals. A GG-mediator is mainly involved in this mediation. *Data Mediations* is used by both goal and process mediation to map data across domain ontologies. An OO-mediator is mainly involved in this mediation.

In IRS-III, a mediator declares a source component, a target component and either a mediation service or mapping rules to solve mismatches between the two.

The *mediation service* is just another goal that can be accomplished by published Web services. For example (Figure 2), a mediation service of a WG-mediator (or GG-mediator) transforms input values coming from the source goal into an input value used by the target Web service (or Goal). The mediation goal is invoked and then accomplished when the respective mediator is considered by the IRS-III interpreter.

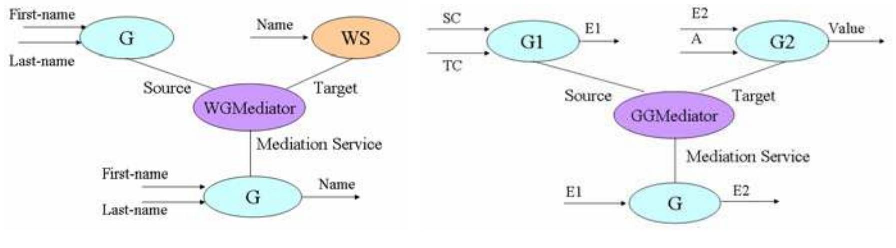


Figure 2. Use of mediation services for WG and GG mediators.

Mapping rules are used between two ontologies, source and target components (Figure 3). They represent backward chaining rules, based on three OCML main mapping primitives: `Maps-to`, a relation created internally for every mapped instance; `Def-concept-mapping`, generates the mappings specified with the maps-to relation between two ontological concepts; `Def-relation-mapping`, generates a mapping between two relations using a rule definition within an ontology. Since OCML represents concept attributes as relations, this primitive can be used to map between input and output descriptions.

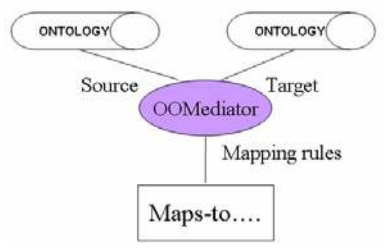


Figure 3. Use of mapping rules for OO-mediator.

5 Creating Semantic Web Services based Applications

In this section, we describe the general infrastructure and the methodology adopted to deploy our e-Government applications. Since government legacy systems are often isolated - i.e. not interconnected and/or use distinct technological solutions - our approach firstly enables the data and functionalities provided by existing legacy systems of the involved governmental partners to be exposed as Web services. The latter are then semantically annotated and published following the IRS-III approach (Section 4). The generic application architecture presented in Section 5.1 reflects and explains this double stage process. The setting up of a domain-specific application is driven by a cooperative and multi-viewpoint methodology refined during our work, and here described in Section 5.2.

5.1 Generic Application Architecture

The proposed generic application architecture is depicted in Figure 4. From the bottom up the four application layers are:

- *Legacy System layer:* consists of the existing data sources and IT systems available from each of the organizations involved in the integrated application.
- *Service Abstraction layer:* exposes (micro-) functionalities of the legacy systems as Web Services, abstracting from the hardware and software platforms. At this level we address thus the level I of interoperability defined by [7] and introduced in Section 1. Web Services are distributed and stored within the multiple organizational infrastructures that expose the functionality. Existing Enterprise Application Integration (EAI) software can be used to facilitate the creation of required Web Services. For example, for standard databases the necessary functionalities of Web Services can simply be implemented as SQL query

functions. Further services available on the Web - and not related to the involved legacy systems - can be integrated to perform supporting functionalities (e.g. mediation services).

- *Semantic Web Service layer*: this layer is implemented by IRS-III which provides the functionalities F1 – F4 described in Section 4.2. At this level we address thus the levels II and III of interoperability defined by [7] and introduced in Section 1. To set up an application, a set of application-specific SWS descriptions has to be provided: goals, mediators, and Web services, all supported by the relevant ontologies (see Section 5.2). These descriptions are centrally stored within the SWS Library of IRS-III (Section 4.1). Note that we distinguish two main sets of SWS descriptions: *basic SWS* (bottom of the layer) that simply wrap the Web Services to achieve simple goals; and *complex SWS* (top of the layer) that require a composition of basic or complex SWS to achieve complex goals.
- *Presentation layer:* consist of a Web application accessible through a standard Web browser. The goals defined within the SWS layer are reflected in the structure of the interface and can be invoked either through the IRS-III API or as an HTTP GET request. The goal requests are filled with data provided by the user and sent to the Semantic Web Service layer. We should emphasise that the presentation layer may be comprised of a set of Web applications to support distinct user communities. In this case, each community would be represented by a set of goals supported by community related ontologies.

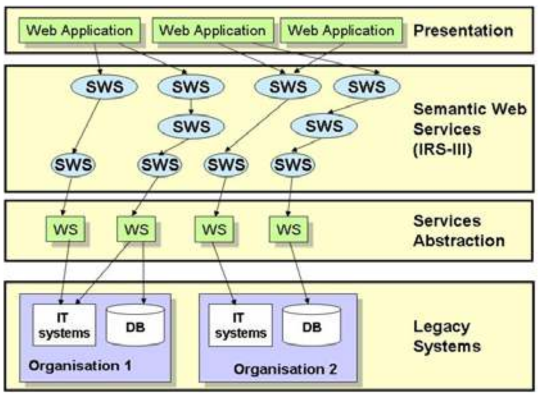


Figure 4. The generic architecture used to create IRS-III-based e-Government applications.

5.2 Development Methodology

In order to successfully create applications from SWS as depicted in Figure 4 four key activities need to be carried out as follows:

1. *Requirements capture:* the requirements for the overall application are captured using standard software engineering methodologies and tools. We do not advocate any particular requirements capture method but envisage that the

resulting documents describe the stakeholders, the main users, roles, and goals, any potential providers for Web services, and any requirements on the deployed infrastructure and interfaces.

2. *Goal description*: using the requirements documents above relevant goals are identified and semantically described in IRS-III. During this process any required supporting domain ontologies will either be created from scratch or existing ontologies will be re-used.
3. *Web service description*: descriptions of relevant Web services are created within the IRS. Again, any domain ontologies required to support the Web service descriptions are either defined or re-used as necessary.
4. *Mediator description*: mismatches between the ontologies used, and mismatches within and between the formal goal and Web service descriptions are identified and appropriate mediators created.

All of the above steps are carried out by the SWS application developer. The first two steps are user/client centric and therefore involve discussions with the relevant client stakeholders and domain experts, whereas step 3 will require dialogue with the Web service providers and domain experts. Steps 2 and 3 are mostly independent and in the future we expect libraries of goals and Web services to become generally available to support reuse. Steps 2, 3 and 4 are supported by means of IRS-III clients that provide a set of tools for defining, editing and managing a library of semantic descriptions, as well as for grounding the descriptions to services. As a result, we obtain a semi-automatic knowledge acquisition process for the development of our applications.

6 e-Government Applications

In this section, we demonstrate the feasibility and applicability of our approach by describing two compelling use cases in the e-Government domain: *Change of Circumstances* (Section 6.1) and *Emergency Management System* (Section 6.2). In the first one, the developed application integrates multiple datasets in order to automatically notify the change of a citizen situation. In the second one, the developed application supports emergency planning and management personnel by retrieving, filtering, and presenting data from a variety of legacy systems to deal with a specified hazardous situation. Both use case descriptions follow the generic application architecture introduced in Section 5.1, although the technical emphasis varies: the first one details the development of SWS descriptions for setting up a specific application; the second one highlights the use of SWS descriptions within a specific application.

6.1 Change of Circumstances

The application has been developed to solve a specific use case problem at Essex County Council (ECC). Whenever the circumstances in which a given citizen lives change, he/she might be eligible for a set of services and benefits provided by ECC and other governmental agencies together with public service providers. An example of such a change of circumstances is, if an elderly, partly disabled woman moves in together her daughter. This changes the circumstances of both, the mother and the daughter. For instance, the mother might no longer receive a “meals on wheels” service, whereas the daughter might get financial supporting for caring her mother. Starting from existing legacy systems, the aim is to provide integrated functionalities, such as: change patient details within multiple legacy systems, change patient pending equipment orders, list of all services for a patient, stop providing service to patient and assess equipment to patient.

6.1.1 Legacy System Layer

Generally, even very simple process in a change of circumstances requires the interaction of many different government agencies. Each agency has different legacy systems in place to keep track of citizen records, provided services, third-party service providers, etc. In our prototype, the following two data sources provided by two different departments (at two distinct governmental levels) were considered:

- *Citizen Assessment (Community Care Department of the ECC)*: this relates to information about citizens registered in ECC for assessment of services and benefits (e.g. meals on wheels; someone goes and cleans the house; someone goes and stays with the patient, etc). This information is stored in the SWIFT database.
- *Order Equipment (Housing department of the Chelmsford District Council)*: this relates to information about equipment (e.g. stair lift, wheel chair, crutch, etc) which is provided to citizens registered in Essex. This information is stored in the ELMS database.

Both SWIFT and ELMS are relational databases that are independently developed and use different data formats to store the same information - e.g. they both hold personal details of the patients. Our prototype accesses two testing databases that exactly replicate the schemata of the two real systems and contain dummy data of the same quality – i.e. both databases contain records with errors, duplicates, inconsistent records. As a result, the two databases used in the prototype mimic the behavior and properties of the real systems.
Figure **5** depicts the database schema of the ELMS system.

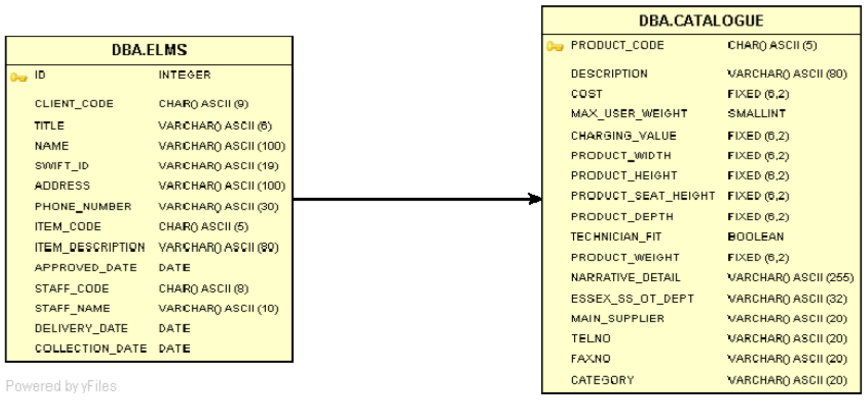

DBA.ELMS
ID INTEGER
CLIENT_CODE CHAR() ASCII (9)
TITLE VARCHAR() ASCII (6)
NAME VARCHAR() ASCII (100)
SWIFT_ID VARCHAR() ASCII (19)
ADDRESS VARCHAR() ASCII (100)
PHONE_NUMBER VARCHAR() ASCII (30)
ITEM_CODE CHAR() ASCII (5)
ITEM_DESCRIPTION VARCHAR() ASCII (80)
APPROVED_DATE DATE
STAFF_CODE CHAR() ASCII (8)
STAFF_NAME VARCHAR() ASCII (10)
DELIVERY_DATE DATE
COLLECTION_DATE DATE
DBA.CATALOGUE
PRODUCT_CODE CHAR() ASCII (5)
DESCRIPTION VARCHAR() ASCII (80)
COST FIXED (6,2)
MAX_USER_WEIGHT SMALLINT
CHARGING_VALUE FIXED (6,2)
PRODUCT_WIDTH FIXED (6,2)
PRODUCT_HEIGHT FIXED (6,2)
PRODUCT_SEAT_HEIGHT FIXED (6,2)
PRODUCT_DEPTH FIXED (6,2)
TECHNICIAN_FIT BOOLEAN
PRODUCT_WEIGHT FIXED (6,2)
NARRATIVE_DETAIL VARCHAR() ASCII (255)
ESSEX_SS_OT_DEPT VARCHAR() ASCII (32)
MAIN_SUPPLIER VARCHAR() ASCII (20)
TELNO VARCHAR() ASCII (20)
FAXNO VARCHAR() ASCII (20)
CATEGORY VARCHAR() ASCII (20)
Powered by yFiles

Figure 5. The database schema of the ELMS system.

Specific SQL queries provide for each of the tables of the two databases CRUD style functionalities; for instance functionalities for creating, reading, updating and deleting records.

6.1.2 Service Abstraction Layer

On top of the two legacy systems, we developed a set of Web services that perform the SQL queries introduced in the previous section and the basic operations introduced above. We created 8 Web services from the SWIFT database and 19 Web services from the ELMS database. The Web services were deployed and stored into the SAP Exchange Infrastructure (SAP XI) [27]. Moreover, we developed some Web services - implemented in a mixture of Common Lisp and OCML [23] – to support application-specific operations (e.g. merging results of distinct database queries).

6.1.3 Semantic Web Service layer

To provide the SWS descriptions (Section 5.1) and the required supporting domain ontologies - steps 2, 3, and 4 of our development methodology (Section 5.2) - we devised two teams composed of SWS developers and domain experts. Each team worked on a distinct domain: Citizen Assessment and Order Equipment. The following tables summarise the resulting ontologies.

User Oriented Domain Ontologies	
e-Government-upper-level-ontology (Citizen Assessment Team, Order Equipment Team)	It is an upper ontology for representing commonly accepted concepts, such as organization, person, citizen, etc. It has been used as the starting point for developing domain-specific user-oriented ontologies
Change-of-circumstances-citizen-ontology (Citizen Assessment Team)	It extends the concepts introduced in the e-Government upper level ontology by introducing domain-specific concepts, such as address, assessment, health problem and benefit.
Change-of-circumstances-equipment-ontology (Order Equipment Team)	It extends the concepts introduced in the e-Government upper level ontology by introducing domain-specific concepts, such as order, care-item, equipment and supplier.

Service Oriented Domain Ontologies	
SWIFT-service-ontology (Citizen Assessment Team)	It mainly represents concepts which map entities of the SWIFT database schema.
ELMS-service-ontology (Order Equipment Team)	It mainly represents concepts which map entities of the ELMS database schema.

SWS Description Ontologies	
Change-of-circumstances-citizen-basic-SWS (Citizen Assessment Team)	It contains goal, Web service and mediator descriptions which define basic and complex SWS on top of SWIFT database. The respective domain ontologies are: *Change-of-circumstances-citizen-ontology* and *SWIFT-service-ontology*

Change-of-circumstances-equipment-basic-SWS (Order Equipment Team)	It contains goal, Web service and mediator descriptions which define basic and complex SWS on top of ELMS. The respective domain ontologies are: *Change-of-circumstances-equipment-ontology* and *ELMS-service-ontology*
Change-of-circumstances-citizen-complex-SWS (Citizen Assessment Team)	It contains goal, Web service and mediator descriptions which define complex SWS, integrating functionalities of both domains. These descriptions refer to the *Change-of-circumstances-citizen-ontology* as domain ontology and make use of *Citizen Assessment* and *Order Equipment* basic SWS.

Figure 6 shows the graphical representation of the dependencies (i.e. "inheritance") among ontologies: WSMO is the top ontology; *white boxes* represent the domain ontologies (user and service oriented); *gray boxes* represent the ontologies containing SWS descriptions. It is important to note the absence of dependencies that cross the two different domains. Only the bottom ontology (`Change-of-circumstances-citizen-complex-SWS`) crosses the two domains; this ontology defines appropriate mediators to deal with mismatching.

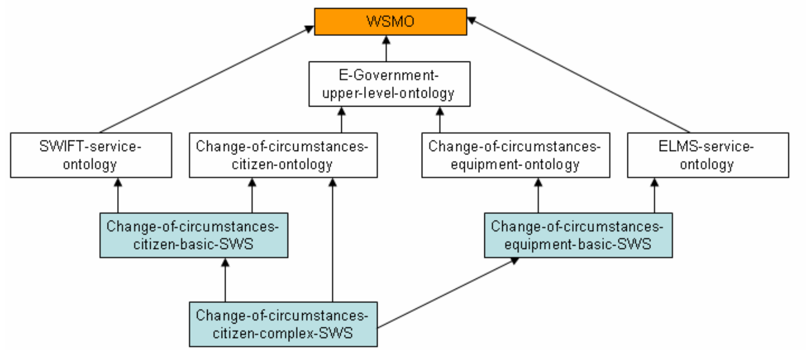

WSMO
E-Government-upper-level-ontology
SWIFT-service-ontology
Change-of-circumstances-citizen-ontology
Change-of-circumstances-equipment-ontology
ELMS-service-ontology
Change-of-circumstances-citizen-basic-SWS
Change-of-circumstances-equipment-basic-SWS
Change-of-circumstances-citizen-complex-SWS

Figure 6. The ontologies of the Change of Circumstances scenario.

To illustrate the development process, we first consider a SWS description of the Order Equipment domain: `Find Item ELMS by Impairment and Weight`. The latter is a complex operation which is decomposed into three basic operations: two queries of the ELMS database and intersecting the two obtained outputs (Figure 7).

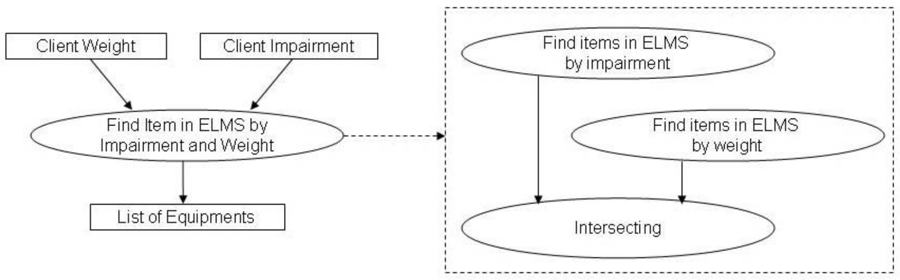
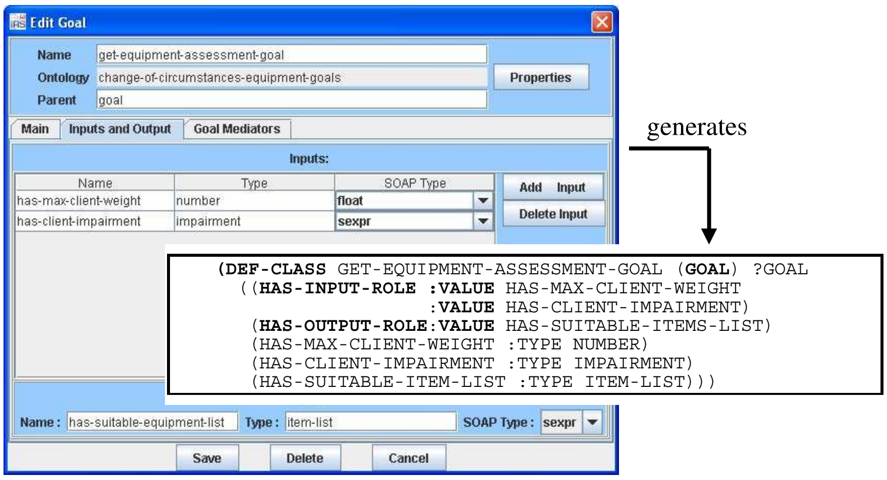

Client Weight
Client Impairment
Find Item in ELMS by Impairment and Weight
List of Equipments
Find items in ELMS by impairment
Find items in ELMS by weight
Intersecting

Figure 7. The Find Item ELMS by Impairment and Weight functionality

Each ellipse in Figure 7 represents a goal which has to be accomplished by simple or integrated functionalities. Specifically, the three goals on the right are accomplished by functionalities provided by Web services available at the service abstraction layer. Such goals have to be automatically orchestrated to accomplish the main goal on the left. Figure 8 depicts, as example, the IRS-III browser interface for describing the main goal above and the resulted OCML code [23]. The goal defines two inputs (`has-input-role`) and one output (`has-output-role`). The inputs (weight and impairment) are classes of the `Change-of-circumstances-equipment-ontology`. The output is a list of equipments (`item-list`). Every equipment description in the list is an instance of the `catalogue-data` class of the `ELMS-service-ontology`.



Name	Type	SOAP Type
has-max-client-weight	number	float
has-client-impairment	impairment	sexpr

Name: get-equipment-assessment-goal
Ontology: change-of-circumstances-equipment-goals
Parent: goal

Name: has-suitable-equipment-list Type: item-list SOAP Type: sexpr

```
    (DEF-CLASS GET-EQUIPMENT-ASSESSMENT-GOAL (GOAL) ?GOAL
      ((HAS-INPUT-ROLE :VALUE HAS-MAX-CLIENT-WEIGHT
                       :VALUE HAS-CLIENT-IMPAIRMENT)
       (HAS-OUTPUT-ROLE:VALUE HAS-SUITABLE-ITEMS-LIST)
       (HAS-MAX-CLIENT-WEIGHT :TYPE NUMBER)
       (HAS-CLIENT-IMPAIRMENT :TYPE IMPAIRMENT)
       (HAS-SUITABLE-ITEM-LIST :TYPE ITEM-LIST))
```

Figure 8. Snapshot of the IRS-III editor and the generated OCML code.

Such a class maps the respective ELMS database schema (Figure 5). At runtime – when the goal is invoked to be accomplished - the instances of the input classes are selected through the user interface of the application, while the instances of the `catalogue-data` class are created on-the-fly - i.e. lifted from the syntactic to the semantic level - from the results of Web service invocations.

For each goal, the respective Web service and mediator descriptions have been created. Figure 9 below represents the `Find Item ELMS by Impairment and Weight` functionality introduced in Figure 7 in terms of goal, mediator and Web service descriptions. The Web service that accomplishes the main goal (`Get-equipment-assessment-goal`) defines the orchestration as the sequence of three sub-goals. In our approach the orchestration is defined at the semantic level as follows:

```
(DEF-CLASS GET-EQUIPMENT-ASSESSMENT-WEB-SERVICE-INTERFACE-ORCHESTRATION
    ((HAS-BODY
      :VALUE ((ORCH-SEQUENCE
                 FIND-ITEMS-MATCHING-WEIGHT-GOAL
                 FIND-ITEMS-MATHCING-IMPAIRMENT-GOAL
                 LIST-INTERSECTION-GOAL)
              (ORCH-RETURN (ORCH-GET-GOAL-VALUE LIST-INTERSECTION-GOAL)))))
```

Each sub-goal, invoked through the orchestration, is accomplished by the respective Web service. Conversely to the main Web service, these Web services ground to syntactic Web services - at the service abstraction layer - and they thus define choreography, as follows:

```
(DEF-CLASS FIND-ITEMS-MATCHING-WEIGHT-WEB-SERVICE-INTERFACE-CHOREOGRAPHY
    (CHOREOGRAPHY)
     ((HAS-GROUNDING
        :VALUE (GROUNDED-TO-WSDL ONLY-OPERATION
               ("c:/CatalogueEntryByWeightInterfaceOut.wsdl"
                 "CatalogueEntryByWeightInterfaceOut"
                 "CatalogueEntryByWeightInterfaceOut"
                 "http://sap.com/research/dip/wp9/elmdb"
                 "SAP"
                 ((has-client-weight "CatalogueEntryByWeightRequest-Type"))
                   "CatalogueEntryResponseType")))
      (HAS-GUARDED-TRANSITIONS :VALUE
                ((RULE1
                   (INIT-CHOREOGRAPHY)
                 THEN
                   (SEND-MESSAGE 'ONLY-OPERATION))))
```

Moreover, Figure 9 outlines the linking roles of WG and GG mediators in our approach: a goal to the Web services that may accomplish it; two sub-goals within an orchestration. More detailed descriptions about the use of WG and GG mediators, during discovery, selection and mediation phases, are presented in the next use case.

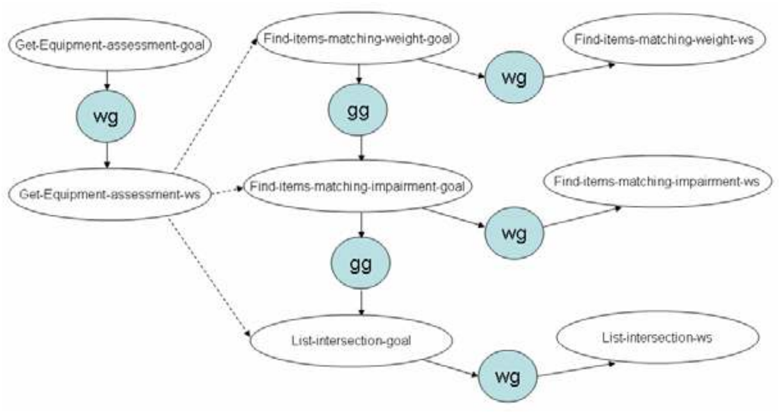

Get-Equipment-assessment-goal
wg
Get-Equipment-assessment-ws
Find-items-matching-weight-goal
gg
Find-items-matching-impairment-goal
gg
List-intersection-goal
wg
Find-items-matching-weight-ws
wg
Find-items-matching-impairment-ws
wg
List-intersection-ws

Figure 9. Structure of the SWS descriptions created for the Find Item ELMS by Impairment and Weight functionality

The resulting `Find Item ELMS by Impairment and Weight` SWS description accomplishes the requested functionality (goal) by integrating services of the same legacy system. Note that each legacy system is an autonomous entity within the given scenario and the provided Web services abstract from the underlying technology. Therefore, we would not have any central control on the involved parties and detailed information about the respective technologies. For example, we could not require a new SQL query of the ELMS database that can simply implement the `Find Item ELMS by Impairment and Weight` functionality.

The effectiveness of a SWS-base approach becomes clearer when integrating services from multiple distributed autonomous entities. In this case, we need to deal with the distinct viewpoints of each involved party. To prove this aspect in the current scenario, we consider a further complex SWS description: `Assess Equipment to Patient`. The latter is part of the `Change-of-circumstances-citizen-complex-SWS` ontology and integrates functionalities of both domains. It is decomposed into two complex operations (Figure 10).

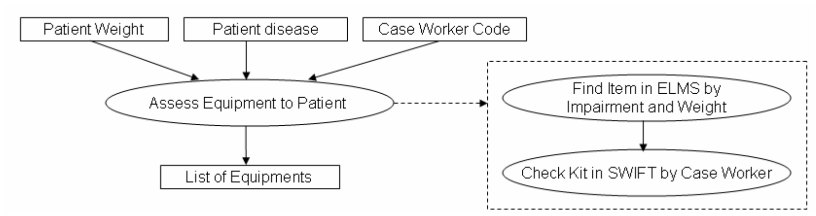

Patient Weight
Patient disease
Case Worker Code
Assess Equipment to Patient
List of Equipments
Find Item in ELMS by Impairment and Weight
Check Kit in SWIFT by Case Worker

Figure 10. The cross-domain Assess Equipment to Patient functionality.

The first operation is the aforementioned `Find Item ELMS by Impairment and Weight`. The second operation filters the equipments retrieved in the first operation by checking if the current case worker – an employee of the Community Care Department – is entitle to provide the equipments to the user. The following listing shows the goal and orchestration definitions of the `Assess Equipment to Patient` functionality.

```
   (DEF-CLASS ASSESS-EQUIPMENT-TO-PATIENT-GOAL (GOAL) ?GOAL
     ((HAS-INPUT-ROLE :VALUE HAS-CITIZEN-WEIGHT
                      :VALUE HAS-CITIZEN-DISEASE
                      :VALUE HAS-CASE-WORKER-CODE)
      (HAS-OUTPUT-ROLE:VALUE HAS-SUITABLE-ITEMS-LIST)
      (HAS-CITIZEN-WEIGHT :TYPE NUMBER)
      (HAS-CITIZEN-DISEASE :TYPE DISEASE)
      (HAS-CASE-WORKER-CODE :TYPE NUMBER)
      (HAS-SUITABLE-ITEM-LIST :TYPE ITEM-LIST)))

(DEF-CLASS ASSESS-EQUIPMENT-TO-PATIENT-WEB-SERVICE-INTERFACE-ORCHESTRATION
    ((HAS-BODY
      :VALUE ((ORCH-SEQUENCE
                  GET-EQUIPMENT-ASSESSMENT-GOAL
                  CHECK-EQUIPMENT-CASE-WORKER-GOAL)
          (ORCH-RETURN (ORCH-GET-GOAL-VALUE CHECK-EQUIPMENT-CASE-WORKER-GOAL))))))
```

As in the previous SWS description, the new functionality has been created by simply stating the sequence of goals to accomplish into an orchestration description. Note that the first goal of the orchestration is the goal depicted in Figure 8. Conversely to the previous SWS description, however, the first goal of the sequence refers to the Order Equipment domain ontologies, while the second one - as well as the main goal – refers to the Citizen Assessment domain ontologies. Particularly, the inputs of the main goal refer to `citizen` and `disease` classes, while the inputs of the first goal refer to `client` and `impairment` classes, respectively. Moreover, the first goal adopts the ELMS `catalogue-data` in the output list of equipments, while the second and main goals use the SWIFT `care-item` in the respective list of equipments. To map between the two domains and thus solve the mismatches, we make use of OO-mediators. As described in Section 4.2, OO-mediators are linked to the goal through the `used-mediator` relation and define mapping rules to solve data mismatching. The mapping rules are valuated when the goal is invoked. The listing below shows as excerpt of the mapping rules for the `catalogue-data` and `care-item` classes.

```
(def-concept-mapping catalogue-data care-item)

(def-relation-mapping catalogue-care-max-weight-mapping
      ((has-max-citizen-weight ?care-item ?value)
        If
      (maps-to ?care-item ?catalogue-data)
      (has-max-user-weight ?catalogue-data ?value)))
```

The example above makes use of the primitives introduced in Section 4.2.4. More specifically, the definitions above link the `has-max-user-weight` slot of class `catalogue-data` in the source ontology to the `has-max-citizen-weight` slot of class `care-item` in the target ontology. The `def-concept-mapping` construct associates each instance of the `catalogue-data` class to a newly created instance of the `care-item` class and link them by generating instances of the relation `maps-to` internally. The `def-relation-mapping` construct uses the generated `maps-to` relation within a rule which asserts the value of the mapped catalogue max user weight to the value of the care item max citizen weight.

As a result, we easily defined and reused SWS descriptions to implement an integrated functionality, abstracting from the underlying legacy systems, keeping the autonomy of involved parties and covering multiple heterogeneous domains. If new systems need to be integrated, we simply introduce the appropriate SWS descriptions and mediation facilities - when mismatches occur - likewise we have done in the second example of the present use case. Conversely, standard database techniques would necessitate that the different parties harmonise their database schemas or agree upon a unifying schema. The addition of a single new system would require a new consensus to be agreed upon.

Further benefits of our approach are highlighted in the next use case.

6.1.4 Presentation Layer

The application is a service oriented portal for the employees of the Community Care department at ECC. Employees assist citizens to notify their changes of circumstances, and the system delivers the change to the different agencies involved in the process. In this way, citizens only have to inform the public administration once about their changes. The user interface uses the Java API of IRS-III to invoke the defined goals. The user selects the action to perform from a list of available goals. After the user has entered the required data, he/she triggers the execution of a goal and IRS-III performs the appropriate Web service - in the case of get equipment assessment, the three basic Web services are performed.

6.2 Emergency Management System

In an emergency situation, multiple agencies need to collaborate, sharing data and information about actions to be performed. However, many emergency relevant resources are not available on the network and interactions among agencies or emergency corps usually occur on a personal/phone/fax basis. The resulting interaction is therefore limited in scope and slower in response time, contrary to the nature of the need for information access in an emergency situation.

Emergency relevant data is often spatial-related. *Spatial-Related Data (SRD)* is traditionally managed with the help of *Geographical Information Systems (GIS)*, which allow access to different layers of SRD such as highways, transportation, postal addresses index, land use, etc. GIS support decision making by facilitating the integration, storage, querying, analysis, modeling, reporting, and mapping of this data. Following several interviews with SRD holders in ECC, it was decided to focus the scenario on a real past emergency situation: a snowstorm which affected the M11 motorway on 31st January 2003.

6.2.1 Legacy Systems Layer

The Emergency Management System (EMS) aggregates data and functionalities from three different sources:

- *Meteorological Office*: is a national UK organization which provides environmental resources, such as weather forecast, snow and pollution data.
- *ViewEssex*: is a collaboration between ECC and British Telecommunications (BT) which has created a single corporate spatial data warehouse. As can be expected ViewEssex contains a wide range of data including data for roads, administrative boundaries, buildings and Ordnance survey maps, as well as environmental and social care data.
- *BuddySpace:* is an Instant Messaging client facilitating lightweight communication, collaboration, and presence management [10] built on top of the instant messaging protocol Jabber[4]. The BuddySpace client can be accessed on standard PCs, as well as on PDAs and mobile phones which in an emergency situation may be the only hardware device available.

6.2.2 Service Abstraction Layer

We distinguish between two classes of services: *data* and *smart*. The former refer to the three data sources introduced above, and they are exposed by means of standard Web services:

- *Meteorological services:* provide weather information - e.g. snowfall level - over a given rectangular spatial area.

[4] Jabber. http://www.jabber.org/

- *ViewEssex services*: return detailed information on specific types of rest centre. For example, `getHospitals` is a Web service that returns a list of relevant hospitals within a given circular area.
- *BuddySpace services*: allow presence information on online users to be accessed.

Smart services represent specific emergency planning reasoning and operations on the data provided by the data services. They are implemented in a mixture of Common Lisp and OCML [23] and make use of the EMS ontologies. In particular, we created a number of services that filter the data retrieved from ViewEssex according to emergency-specific requirements: e.g. rest centres with heating system, hotels with at least 40 beds, easy accessible hospital, etc. The used criteria were gained from our discussions with emergency officers of ECC.

6.2.3 Semantic Web Service Layer

The following tables summarise the ontologies reflecting the client and provider domains to support SWS descriptions.

Service Oriented Domain Ontologies	
Meteorology Domain Ontology	It contains the concepts used to semantically describe the services attached to the data sources of the Met-Office domain, such as snow and rain.
Emergency Planning Domain Ontology	It contains the concepts used to semantically describe the services attached to the data sources of the ViewEssex domain, such as hospitals and supermarkets.
Jabber Domain Ontology	It contains the concepts used to semantically describe the services attached to the data sources of the Jabber domain, such as session and preferences.

As in the previous use case, we introduced *lifting operations* to get the information provided by Web services up to the semantic level. These lisp functions automatically extract data from SOAP/XML messages and create instances of the domain ontologies. The mapping information between syntactic data types and ontological classes is defined at design time by developers.

User Oriented Domain Ontologies	
GUI Ontology	It contains GUI and user-oriented concepts. It maps the ontology elements which will be displayed to the elements of the particular user interface which is used. Note that although the choice of the resulting syntactic format depends of the chosen lowering operation, concepts from the GUI ontology are used in order to achieve this transformation in a suitable way.
Archetypes Ontology	It is a minimal ontological commitment ontology aiming to provide a cognitively meaningful insight into the nature of a specialized object; for example, by conveying the cognitive (“naïve”) feeling that for example an hospital, as a “container” of people and provider of “shelter” can be assimilated to the more universal concept of “house”. The latter can be considered as an *archetypal* concept, i.e. based on image schemata and therefore supposed to convey meaning immediately. It is

	moreover assumed that any client, whilst maybe lacking the specific representation for a specific basic level concept, knows its archetypal representation.
Spatial Ontology	It describes geographical concepts of location, such as coordinates, points, polygonal areas and fields. It also allows describing spatial objects as entities with a location and a set of attributes.
Context Ontology	It allows describing *context n-uples* which represent a particular situation. In the emergency planning application, context n-uples have up to four components, the use case, the user role, the location, and the type of object. Contexts are linked with (WSMO-) goals; i.e. if this type of user accesses this type of object around this particular location, these particular goals will be presented. Contexts also help to inform goals, e.g. if a goal provides information about petrol stations in an area, the location part of the context is used to define this area, and input from the user is therefore not needed.

The purpose of the `GUI`, `Archetypes` and `Spatial ontologies` is the aggregation of different data sources on, respectively, a representation, a cognitive and a spatial level. Therefore we can group them under the appellation *aggregation* ontologies. They allow the different data sources to be handled and presented in a similar way. Inversely to the lifting operations, *lowering operations* transform instances of aggregation ontologies into syntactic documents to be used by the server and client applications. This step is usually fully automated since aggregation ontologies are, by definition, quite stable and unique.

SWS Description Ontologies	
Met-Office SWS	It contains goal, Web service and mediator descriptions which define SWS on top of the Met Office database.
Emergency Planning SWS	It contains goal, Web service and mediator descriptions which define SWS on top of the ViewEssex GIS system.
BuddySpace SWS	It contains goal, Web service and mediator descriptions which define SWS on top of the BuddySpace instant messaging system.

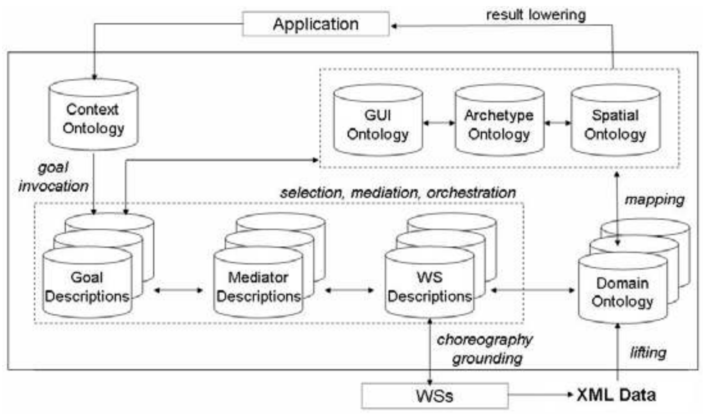


Figure 11. The use of semantics within the Semantic Web Service Layer

Figure 11 outlines how the ontologies and SWS descriptions stored within the SWS library of IRS-III link the user interface (Application) to the Met Office, ECC Emergency Planning, and BuddySpace Web services (WSs). Starting from the application, counterclockwise, the *italics* words in the picture represent the main operations performed within IRS-III. The Web service descriptions make use of domain ontologies - Meteorology, ViewEssex and Jabber – whilst the goal encodings rely on the GUI, archetypes and spatial ontologies. Mismatches are resolved by mediation services linked to WG and GG mediators.

Figure 12 shows an example of the created SWS descriptions: `Get-Polygon-GIS-data-with-Filter-Goal` represents a request for available shelters within a given area. The user specifies a polygon area and the shelter type (e.g. hospitals, inns, hotels). The results obtained by querying ViewEssex need to be filtered in order to return shelters correlated to emergency-specific requirements only. The problems to be solved in this example include: (i) *discovering* and *selecting* the appropriate ViewEssex Web service; (ii) *meditating* the difference in area representations (polygon vs. circular) between the user goal and available Web services; (iii) *composing* the retrieve and filter data operations.

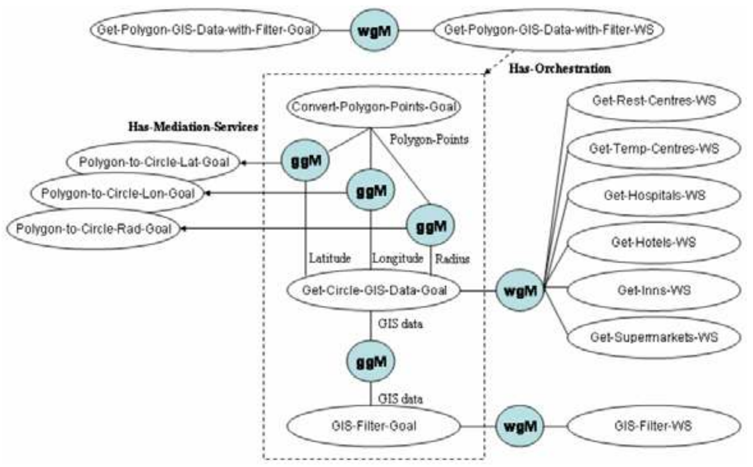


Figure 12. A portion of WSMO descriptions for the EMS prototype.

We outline below how the SWS representations in Figure 12 address these problems.

- *Web service discovery and selection:* when the `Get-Circle-GIS-Data-Goal` is invoked, IRS-III discovers all of Web services that can solve it by means of the WG-mediator (Section 4.2.1). Each semantic description of ViewEssex Web service defines the Web service capability - i.e. the class of shelter provided by the Web service. The listing below reports an example of kappa expression defining a capability assumption:

```
(DEF-CLASS GET-ECC-HOSPITALS-WEB-SERVICE-CAPABILITY (CAPABILITY) ?CAPABILITY
  ((USED-MEDIATOR :VALUE GET-GIS-DATA-MEDIATOR)
   (HAS-ASSUMPTION:VALUE
     (KAPPA(?WEB-SERVICE)
           (= (WSMO-ROLE-VALUE ?WEB-SERVICE'HAS-SPATIAL-OBJECT-QUERY)
              'HOSPITALSQUERY))))
```

If the Web service provides the class of shelters defined in one of the inputs of the goal, IRS-III selects it (Section 4.2.2). In the example above, the Web service is selected if the request class of shelters is hospital (`'hospitalquery`).

- *Area mediation and orchestration*: the `Get-Polygon-GIS-data-with-Filter-Goal` is associated with a unique Web service that orchestrates three sub-goals in sequence. The first one gets the list of polygon points from the input; the second one is the `Get-Circle-GIS-Data-Goal` described above; the third one invokes the smart service which filters the list of shelter data. The first and second sub-goals are linked by three GG-mediators which return the centre, in the form of latitude and longitude, and the radius of the smallest circle that circumscribes the given polygon. To accomplish this, we created three mediation services represented by three distinct goals: `Polygon-to-Circle-Lat-Goal`, `Polygon-to-Circle-Lon-Goal`, and `Polygon-to-Circle-Rad-Goal`. Each mediation service is performed by a specific Web service, exposing a Lisp function (the respective WG-mediator and Web service ovals were omitted to avoid cluttering the diagram). The results of the mediation services and the class of shelter required are the inputs to the second sub-goal. A unique GG-mediator connects the output of the second to the input of the third sub-goal, without introducing any mediation service.

Additionally to the benefits of our approach introduced in Section 6.1.3, this use case highlighted the following aspects:

- We created complex SWS descriptions on top of three distinct kinds of legacy system: database, GIS and instance messaging. The use of Web services allows us to abstract from the underlying technologies and ease thus their integration.
- A given goal – e.g. `Get-Circle-GIS-Data-Goal` – might be achieved by several Web services. The most appropriate one is selected on the basis of the specific situation. The effective workflow – i.e. the actual sequence of service invocations – is known at run-time only. In existing Web service-based approaches the functionalities are mapped at design-time, when the actual context is not known.
- The use of WG and GG mediators allows goal and process mediation and thus a smoothly crossing among services of distinct domains in the same workflow. The most appropriate mediation service is selected at run-time, according to the specific situation.
- If new Web services will be available – for instance providing data from further GIS - new Web Service descriptions can be simply introduced and linked to the `Get-Circle-GIS-Goal` by the proper mediators - or reusing the existing one, if semantic mismatches do not exist - without affecting the current structure. In the same way, new filter services - e.g. more efficient ones - may be introduced.

6.2.4 Presentation Layer

The Emergency Management System (EMS) prototype is in effect a decision support system, which assists the end-user – currently the Emergency Planning Officer (EPO) – in assembling information related to a certain type of event, more quickly and accurately. The application's user interface is based on Web standards. XHTML and CSS are used for presentation, while JavaScript (i.e. EcmaScript) is used to handle user interaction together with AJAX techniques to communicate with IRS-III. One of the main components of the interface is a map, which uses the Google Maps API [13] to display polygons and objects (custom images) at specific coordinates and zoom level. Each time an object is displayed by a user at a particular location, a function of the context ontology provides the goals which need to be displayed and what inputs are implicit. A screencast with an example of end-user interactions as well as a live version are available online[5], to be used preferably with the Firefox Web browser.

7 Lessons learned

On the basis of challenges encountered - and the ways in which they were overcome - we now summarize the lessons learned in terms of: identifying the suitable scenario, following the adequate development process, verifying the advantages of SWS over other technologies and outlining the open challenges.

7.1 The scenario

The first challenge is the identification of the proper scenario; i.e. a scenario where SWS technology can provide substantial benefits. On the basis of our experience, we can outline the following main features:

- The scenario is a distributed and heterogeneous environment with a lack of centralized control, which provides a large amount of alternative – i.e. providing different functionalities in distinct situations - and competitive - i.e. providing the same functionalities in the same situation – services.
- The services used in the scenario are connected to external environments and access to common data/resources already available on the Web.
- The scenario involves multiple stakeholders - clients and service providers - that need to collaborate. They represent the heterogeneous viewpoints/domains to describe.
- The scenario is not static, but subject to changes and evolutions. The dynamism may involve the viewpoint descriptions – e.g. government policies, citizen needs, agencies' participation – or the service descriptions - e.g. changes in the service business process, or new services provided by existing or new partners.

In our work, we preliminary identified a lot of promising service-oriented application fields, such as e-Procurement, school admissions, libraries, health, GIS applications (e.g. emergency planning), change of circumstance, child care/children's services, youth services, adult social care, benefits and revenues, and criminal justice initiatives. On the basis of existing legacy systems, services and datasets, resources, stakeholders' requirements and needs within ECC, we refined the list reported above and chose the use cases described in Section 6.

[5] http://irs-test.open.ac.uk/sgis-dev/

7.2 The development process

According to the features of the suitable scenario outlined in the previous section, we expect that, during the development process, new requirements may arise or some domain aspects may be better comprehended, new services need to be developed or integrated in order to cover existing lacks, and new datasets may be available in order to improve the existing information space. These aspects are common in almost every scenario, but they are particularly true when dealing with distributed and heterogeneous sources. Therefore, we aimed to design a *pragmatic* - in order to quickly lead to a working outcome – as well as *flexible* - in order to easily respond to possible changes or improvements and meet the multiple actors' viewpoints – development process. The prototyping approach is a commonly used methodology to mach such requirements. Moreover, the semantic approach generally helps to address flexibility, since the changes mainly concern the semantic descriptions only - e.g. ontologies and SWS descriptions of the Semantic Web Service layer - and not the overall architecture of the system. The challenge was to identify an appropriate prototyping methodology which takes advantage of the decoupled nature of SWS descriptions (WSMO approach). As a result, we tailored a *SWS-oriented prototyping development process* composed of the following three straightforward phases: requirements capture, SWS description, evaluation (Figure 13).

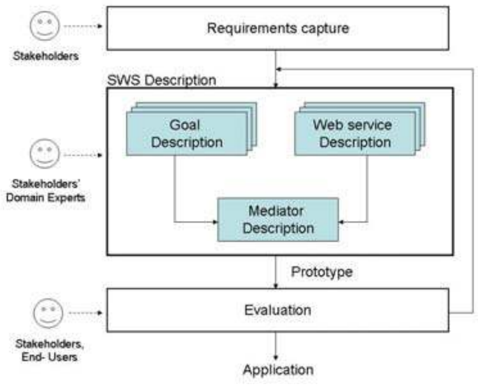


Figure 13. Main steps of the devised prototyping process.

The first phase represents the step 1 of the methodology presented in Section 5.2. The second phase focuses on the semantic descriptions, and encloses the required flexibility of the process. This phase is decomposed into several activities that deal with the knowledge acquisition and representation of the multiple domains and actors' viewpoints of the application context. Each activity can be independently iterated whenever an improvement or change only involves the respective domain/viewpoint. This phase represents the steps 2, 3 and 4 of the methodology presented in Section 5.2. The last phase introduces the prototyping iterations of the whole application development process. The prototype has been shown to stakeholders (clients and service providers) and end-users. Prototype improvements and changes have been mainly based on their feedback. Finally, it is important to note that:

- Along the whole development process, we keep a constant contact with the stakeholders and users. In the first phase, we mainly interview stakeholders' manager and technical people. Then, we cooperate with domain experts (i.e. organization employees). Finally, we consult again the stakeholders and involve the end-users. In this way, we can address the barriers e) and f) identified in Section 3.2.
- The structure of the second phase leads to a sound approach that separately focuses on each of the involved actors – i.e. their viewpoints and specific languages/terminology/skills - keeps organizations' autonomy in the description of their domain and allows the cooperative development of the application.
- The proposed methodology is not an e-Government specific formula.

7.3 The verified added values

The deployed applications highlighted advantages of adopting SWS over other technologies. In this section, we summarise the comparison with existing Web services-based and ontology-based approaches. Other technologies (e.g. standard database technologies), indeed, do not provide the adequate abstraction over heterogeneous and autonomous legacy systems (Section 6.1.3).

- *SWS vs. Web Services*. By using Web Services, data and functionalities can be shared with anyone through the Internet. As introduced in Section 2, the supplied services are autonomous and platform-independent computational elements. The syntactic definitions used in these specifications allow fast composition and good results in term of application performance. However, they do not completely describe the capability of a service and cannot be understood by software programs. A human developer is required to interpret the meaning of inputs, outputs and applicable constraints, as well as the context in which services can be used. Moreover, Web Services lack in flexibility: for instance, if a new Web Service is deployed, the application developers need to re-model several syntax descriptions – introducing a cost - in order to integrate it.
  On the other hand, the SWS approach is able to model the background knowledge of a context together to the requested and provided capabilities, and it hence addresses *automatic reasoning* and *reuse* (Section 6.1.3). As a result, service invocation, discovery, composition and mediation are *automated* by adopting the best available solutions for a specific request and increasing the *flexibility*, *scalability,* and *maintainability* of an application. Moreover, the execution sequence of a complex SWS (Sections 6.1.3 and 6.2.3) is not hard-coded, and it is dynamically created by using a goal-based invocation: several Web Services may be associated with a goal, and only the best one will be discovered and invoked at runtime only (late binding); if a new service will be available, the developers simply will describe and then link it to an existing goal; if a service will change, only the specific semantic description will be affected, and not the whole process (Section 6.2.3).
- *SWS vs. other ontology-based approaches*. Creating and managing ontologies is a bottleneck: understanding a domain, acquiring and representing knowledge, populating with instances and evolving ontologies are big tasks for the

application developers. In complex domain such as e-Government, centralized ontologies would require an unrealistic development effort with no guarantee of satisfactory results. Moreover, government agencies deal with huge datasets (e.g. demographic, GIS, etc.) that cannot easily transposed to ontology's instances. However, in the context of semantic-based applications, such a cost cannot be deleted, but it may be contained.

SWS technology makes knowledge capture and maintenance process simpler and more efficient (Section 6.1.3). (a) The only knowledge which must be modeled is related to the exposed functionality of the Web service. This means describing the concepts used by the Web service only, such as inputs and output. Moreover, the instances of a concept are not defined a priori, but they are created at runtime – i.e. lifted after the execution of the Web service. This minimalist approach makes easy the management of ontologies – i.e. evolution and maintenance. (b) The knowledge capturing process is distributed among all of the stakeholders: each partner describes – and it is responsible for – its particular domain. In this way, the several viewpoints can be independently and concurrently described by the proper knowledge holders. Partners can also reuse their own existing ontologies. As a result, we obtain a model that reflects the e-Government structure and addresses the required lack of central control.

Moreover, a WSMO based approach addresses *interoperability* among very heterogeneous knowledge sources and *mediation* among several viewpoints (users, multiple providers, etc.). WSMO mediators are mappings that solve existing mismatches and do not affect service descriptions. In our applications, we have gathered the following mediation requirements and solutions:

- *Data mediation*: organizations have their own databases and hence different data formats for the same concept. Lifting at the semantic level these distinct data formats, the resulting instances can be mapped by means of either mediation services (Section 6.2.3) or mapping rules (Section 6.1.3).
- *Goal mediation*: Multiple Web services can be linked to the same goal via mediators. In principle, goal and web service descriptions are provided by distinct organizations, and a mediation service is used to solve the existing mismatches (Section 6.2.3).
- *Process mediation*: organizational processes behave in different ways according to their own set of operational procedures, requirements and constraints. Added value functionalities can be provided by composing several goal descriptions. Mediation between two goals in sequence may be necessary to solve exiting mismatches (Section 6.2.3).

7.4 Open Challenges

Since we are adopting a young technology and e-Government is a very complex domain, we are aware that not all of the existing issues are completely addressed. The main remaining challenges identified are:

- *SWS infrastructure.* SWS technology is an ongoing research, and some of its main features - e.g. mediation, orchestration, non functional properties based discovery - are still under development. However, in order to respond to the needs of real-world applications, IRS-III already introduced some solutions. During the development of the presented applications, we continually improved and tested selection, choreography, orchestration and mediations of IRS-III. However, further use cases will highlight unconsidered aspects and allow us to improve IRS-III performances. Moreover, the choice of a specific SWS approach involves the adoption of its defined features; for instance, IRS-III uses client instead of service choreography, goal-based orchestration instead of goal and web service composition, etc. However, in a wide domain such as e-Government, some features may be adequate in a context but not in others, and several partners may adopt distinct approaches. The openness of IRS-III aims to address the interoperability of multiple SWS approaches.

- *Commercialization.* The transition of the currently available systems into a stable and robust infrastructure is one of the major challenges that need to be solved, before a SWS-based solution can be deployed into a productive environment. However, the prototyping development (Section 7.2) of carefully targeted applications, with clear objectives stated, can lead to real-world operational systems.
- *Organizational and social aspects*. The employees of governmental agencies usually perform their tasks well established procedures; the inappropriately-handled introduction of new processes or applications may lead to the reluctance of use them. Active participation of stakeholders and end-users in the design and development processes allows developers to deploy applications that respect current procedures and, at the same time, ease the work of staff, leading to improved acceptance. As described in Section 7.2, our approach follows this idea; however, more detailed investigations on the approach/methodology to follow and social implications could be performed.
- *Privacy, Security, and Trust.* As stated in Section 3.2, these are fundamental requirements in e-Government. At the syntax level, efficient solutions for addressing privacy and security issues already exist (e.g. SSL protocol and virtual private networks for protecting the communications, firewalls and digital certificates for avoiding malicious accesses and protecting data), or there is relevant ongoing research (e.g. enriching Web Services description with digital certificates and signatures). In the Change of Circumstances application, where citizen information had to be protected, we based on the security and privacy solutions provided by the adopted EAI system [27].
  The semantic level should extend the syntactic solutions by ontologically describing security and privacy policies of accessing data and processes. Moreover, trust-based discovery of SWS would be a crucial issue, in order to avoid invocation of malicious or unreliable services, for which there are no defined standards by which SWS may expose their policies and trust features. Most of existing approaches inherit methodologies from the peer-to-peer networks [21], [24]. Trust evaluation algorithms for SWS consider security issues, such as confidentiality, authorization, authentication, as rating statements [17],[19],[20],[21], or more generically Quality of Service performance properties [35], such as precision and accuracy of data, timeliness in executing a task, and security. The key to enabling a trust-based selection for SWS lies in a common ontological representation, where Web service and client perform their

trust guaranties and requirements. In [12], we propose our trust managing approach based on IRS-III. Essentially, all participants can expose their trust guaranties and requirements by specifying trust policies. Since this work is still in progress, we do not apply it to the presented use cases.

- *Ease of use of SWS technology in e-Government*. The full integration between e-Government and SWS is not an easy task. The following further requirements should be considered. (a) Government agencies usually do not directly use the SWS infrastructure to represent knowledge internally. For instance, organizations will likely adopt their own workflow paradigm to describe their processes [1]. (b) Organization processes involve interactions with non-software agents, such as citizens, employees, managers, and politicians; thereby, component services cannot in general be executed in a single-response step, and may require to following an interaction protocol with humans that involves multiple sequential, conditional and iterative steps. For instance, a service may require a negotiation between the citizen and the provider. In our approach, the management of such an interaction is embedded in the Presentation layer, because IRS-III supports one-shot goal invocation only.
  In order to address these issues, we argue that a more complex semantic layer – i.e. an *explicit e-Government framework* - needs to be modelled. In [15], we identify and model three knowledge levels: *Constraints*, describing the context that bounds the creation and evolution of services: legislations, policies, and strategies influencing the development and management of an e-Government service-supply scenario; *Configuration*, describing the context in which services are supplied: requirements, resources, actor's role, business processes, and transactions of an e-Government service-supply scenario; *Service delivery*, adopting SWS technology as the base for the description, discovery, composition, mediation, and execution of (Web) services.
- *Standardization*. Currently, there are not reference standards for (semantic) service oriented applications in e-Government. The e-Government community is still debating on the approach to follow between, as a broadly described option, standardization versus integration - i.e. focusing on interoperation among several existing approaches. We believe that our approach is open to both solutions and our results may contribute to the investigation of possible standards.

8 Related Work

In the last years, several projects applied SWS technology in the e-Government domain, but only a few of them show reusability and composability in real usage scenarios. The *OntoGov* project [25] develops a platform that will facilitate the consistent composition, reconfiguration and evolution of e-Government services. It focuses more on the service life cycle than the interoperability and integration issues. Services are described by means of a "meta ontology" that extends OWL-S [26] by introducing WSMO [36] features. *Terregov* [32] is a project at an early stage of development. It aims to address the issue of interoperability of e-Government services. Its architecture is composed by a framework and intelligent agents that will offer configuration and reconfiguration of service workflows by selecting competing Web Services on the basis of their performance, and composing dynamic workflows based on semantic descriptions. In order to represent e-Government processes, it adopts OWL-S for describing the behaviors of Web Services, and BPEL [4] workflow description language for their orchestration. *WebSenior* [22] uses ontologies to automatically generate Web Services customized to senior citizen's needs and government program laws and regulations. Differently to both OntoGov and Terregov, WebSenior proposes a solution to a specific real usage scenario. This highlights the practical applicability of its approach, but limits the reusability and flexibility.

No one of the mentioned approaches adopts mediation mechanisms to overcome data and process mismatches: they only propose centralized ontologies for representing the entire domain and thereby addressing interoperability.

Further efforts on investigating multiple aspects of the application of semantic technologies in the e-Government domain are under way: BRITE [11] aims to enable interoperations in a transnational scenario among institutions that concert the registration of businesses in the European Union; FIT [29] will develop, test, and validate a self-adaptive citizen-oriented e-government framework; SAKE [30] will develop a holistic framework - and the supporting tools – that will be sufficiently flexible to adapt changing, diverse environment, and needs; and SemanticGov [34] will provide a WSMO-based infrastructure for Pan-European e-Government services. Since all of these projects started in 2006, they are still in their initial phase.

9 Summary and Future Work

In our work, we successfully established a close collaboration with a large local authority in UK in order to define a reusable SWS-based framework for deploying real-world applications in the e-Government domain. The aim is to dealing with complex scenarios, by easily interconnecting heterogeneous domains and allowing governmental agencies to cooperate and consume shared data in an easy way and without a centralized control. SWS technology promises to address interoperability and integration issues, and automate the development of service-oriented applications through semantic Web technologies (Section 2).

The analysis of motivations, requirements, and expected results of matching SWS and e-Government research areas (Section 3) provided us the aspects to stress first in the design of our framework and then in the development of compelling use cases.

To provide semantics and step towards the creation of added value services, we adopted IRS-III, an existing SWS broker (Section 4). In our work, we enclosed IRS-III into a 4-layers generic application architecture (Section 5.1) and devised a development methodology (Section 5.2) to propose a reusable framework for deploying SWS-based applications. The layering of the architecture proved to be very useful: (a) the development of ontologies and SWS descriptions could be decoupled from the implementation of the user interface and the deployment of Web Services; (b) using Web Services on top legacy systems, we abstracted from the technical details of involved legacy systems. The proposed methodology allowed the easy development of agile and flexible applications based on the idea of reuse. For instance, the methodology involves the ontological decoupling of the client's context from the providers' one. This led us to lower the cost of application deployment by introducing cooperative development and creating small ontologies focused on the specific service functionalities.

Following our approach, we deployed two e-Government applications (Section 6): Change of Circumstances and Emergency Management System. In this way, we (a) tested the reusability and adaptability of our approach to different e-Government contexts, (b) proved how our framework addresses interoperability and integration issues, and (c) stressed all of the aspects associated with the development of SWS-based applications: e.g. knowledge acquisition, discovery, composition, and mediation. Note that the development of the second application got benefits from the lessons learned in the development of the first one. In particular, we reduced the time of capturing requirements and describing SWS and obtained more qualitative results.

On the basis of these considerations and the results obtained from the two applications introduced above, we reported the main lessons learned (Section 7). We outlined a general scenario where SWS technology can provide substantial benefits; detailed our prototyping development process and highlighted the active role of stakeholders and end-users; summarized the verified added values of SWS over other technologies; and pointed out the open challenges that will drive our future work.

The analysis of related work (Section 8) showed that the application of SWS in e-Government is a really interesting topic, but a few projects provide real-world applications yet. Since e-Government community claims for creating compelling prototypes, establishing visible standards, stable and mature technologies, and convincing business cases, we believe that our work may contribute to raising awareness of the potential benefits of SWS in e-Government. Perhaps more importantly, the lessons learned may be also used to (a) guide the efforts of new e-Government applications/projects; (b) influence the e-Government standards environment and the e-Government strategic environment so as to encourage take up of SWS technologies.

Acknowledgments. This work is supported by the DIP (Data, Information and Process Integration with Semantic Web Services) project. DIP (FP6 – 507483) is an Integrated Project funded under the European Union's IST programme.

References

[1] der Aalst, W. V., ter Hofstede, A., Weske, M.: Business process management: a survey. In proceedings of International Conference on Business Process Management (BPM). pp. 1-12, (2003).

[2] Amazon: Amazon web services, (2006). http://www.amazon.com/gp/browse.html/104-6906496-9857523?%5Fencoding=UTF8&node=3435361/

[3] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(4):34-43, (2001).

[4] BPEL4WS Consortium: Business process execution language for web services. (2004). http://www.ibm.com/developerworks/library/ws-bpel

[5] Cabral, L., Domingue, J.: Mediation of Semantic Web Services in IRS-III. In Proceeding of the Workshop on Mediation in Semantic Web Services in conjunction with the 3rd International Conference on Service Oriented Computing, Amsterdam, The Netherlands, (2005).

[6] Commission of the European Communities: The Role of e-Government for Europe's Future. Commission Staff Working Paper COM 567, 26.9, (2003).

[7] Commission of the European Communities: Linking up Europe: the Importance of Interoperability for e-Government Services. Commission Staff Working Paper SEC (2003) 801.

[8] Cabral, L., Domingue, J., Galizia, S., Gugliotta, A., Norton, B., Tanasescu, V., Pedrinaci, C.: IRS-III: A Broker for Semantic Web Services based Applications. In proceedings of the 5th International Semantic Web Conference (ISWC 2006), Athens, USA (2006).

[9] Domingue, J., Galizia, S., and Cabral, L.: Choreography in IRS-III- Coping with Heterogeneous Interaction Patterns in Web Services. In Proceedings of 4th International Semantic Web Conference, Galway, Ireland, (2005).

[10] Eisenstadt, M., Komzak, J., Dzbor, M.: Instant messaging + maps = powerful collaboration tools for distance learning. In Proceedings of TelEduc03, Havana, Cuba, (2003).

[11] van Elst, L., Klein, B., Maus, H., Schoning, H., Tommasi, A., Zavattari, C., Favaro, J., Giannella, V.: Business Register Interoperability throughout Europe: The BRITE Project. AAAI Spring Symposium “Semantic Web meets eGovernment”, AAAI Press SS-06-06, Stanford, California, (2006).

[12] Galizia, S.: WSTO: A Classification-Based Ontology for Managing Trust in Semantic Web Services. In proceedings of 3th International Semantic Web Conference, Budva, Montenegro, (2006).

[13] Google: Google Web APIs, (2005). http://www.google.com/apis/index.html

[14] Gruber, T. R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition, 5(2), (1993).

[15] Gugliotta, A., Cabral, L., Domingue, J.: Knowledge Modeling for Integrating e-Government Applications and Semantic Web Services. AAAI Spring Symposium “Semantic Web meets eGovernment”, AAAI Press SS-06-06, Stanford, California, (2006).

[16] Gugliotta, A., Cabral, L., Domingue, J., and Roberto, V.: A semantic web service-based architecture for the interoperability of e-Government services. In Proceeding of the International Workshop on Web Information Systems Modeling, Sydney, Australia, (2005).

[17] Kagal, L., Paoucci, M., Srinivasan, N., Denker G, Finin, T., Sycara K.: Authorization and privacy for Semantic Web Services. In Proceeding of AAAI 2004 Spring Symposium on Semantic Web Services, Stanford University, (2004).

[18] Klischewski, R., Scholl, H. J.: Information Quality as a Common Ground for Key Players in e-Government Integration and Interoperability. In Proceedings of the 39th Hawaii International Conference on System Sciences, Hyatt Regency Kauai, Hawaii, (2006).

[19] Kolovski, V., Parsia, B., Katz, Y., Hendler, J.: Representing Web Service Policies in OWL-DL. In Proceedings of 4th International Semantic Web Conference, Galway, Ireland, (2005).

[20] Mani, A., Nagarajan, A.: Understanding quality of service for Web Services: Improving the performance of your Web Services -IBM-report- (2002). http://www-128.ibm.com/developerworks/library/ws-quality.html

[21] Maximilien, E. M., Singh, M. P. : Toward Autonomic Web Services Trust and Selection. In Proceedings of 2nd International Conference on Service Oriented Computing (ICSOC 2004), New York, (2004).

[22] Medjahed, B., Bouguettaya, A.: Customized Delivery of E-Government Web Services. Web Services, IEEE Intelligent Systems, 20(6). (2005).

[23] Motta, E.: An Overview of the OCML Modelling Language, In Proceedings of the 8th Workshop on Knowledge Engineering Methods and Languages. (1998).

[24] Olmedilla, D., Lara, R., Polleres, A., Lausen, H.: Trust Negotiation for Semantic Web Services. In Proceedings of 1st International Workshop on Semantic Web Services and Web Process Composition in conjunction with the 2004 IEEE International Conference on Web Services, San Diego, California, USA, (2004).

[25] OntoGov: Ontogov project. (2004). http://www.ontogov.com

[26] OWL-S Coalition: OWL-S 1.1 release. (2004). http://www.daml.org/services/owl-s/1.1/

[27] SAP: SAP exchange infrastructure: The integration solution for process-centric collaboration. http://www.sap.com/xi

[28] SOAP: SOAP Version 1.2 Part 0: Primer, (2003). http://www.w3.org/TR/soap12-part0/

[29] Stojanovic, N., Stojanovic, L., Hinkelmann, K., Mentzas, G., Abecker, A.: Fostering self-adaptative e-Government services improvement using semantic technologies. AAAI Spring Symposium "Semantic Web meets eGovernment", AAAI Press SS-06-06, Stanford, California, (2006).

[30] Stojanovic, N., Mentzas, G., Apostolou, D.: Semantic-enbled Agile Knowledge-based e-government. AAAI Spring Symposium "Semantic Web meets eGovernment", AAAI Press SS-06-06, Stanford, California, (2006).

[31] Sycara, K., Paoulucci, M., Ankolekar, A., Srinivasan, N.: Automated discovery, interaction and composition of semantic web services. Journal of Web Semantic 1(1), (2003).

[32] TerreGov: Terregov project. (2004). http://www.terregov.eupm.net/my spip/index.php

[33] UDDI: UDDI Spec Technical Committee Specification v. 3.0, (2003). http://uddi.org/pubs/uddi-v3.0.1-20031014.htm

[34] Vitvar, T., Kerrigan, M., van Overeem, A., Peristeras, V., Tarabanis, K.: Infrastructure for the Semantic Pan-European E-Government Services. AAAI Spring Symposium "Semantic Web meets eGovernment", AAAI Press SS-06-06, Stanford, California, (2006).

[35] Vu L., H., Hauswirth, M. and Aberer, K.: QoS-based Service Selection and Ranking with Trust and Reputation Management. Technical Report IC2005029, Swiss Federal Institute of Technology at Lausanne (EPFL), Switzerland, (2005).

[36] WSDL: Web Services Description Language (WSDL) 1.1, (2001). http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[37] WSMO Working Group, D2v1.0: Web Service Modeling Ontology (WSMO). WSMO Working Draft, (2004). http://www.wsmo.org/2004/d2/v1.0/