On Managing Changes in the ontology-based E-Government

Ljiljana Stojanovic[1], Andreas Abecker, Nenad Stojanovic[2], Rudi Studer[1,2]

[1] FZI - Research Center for Information Technologies at the University of Karlsruhe, Haid-und-Neu-Str. 10-14, 76131 Karlsruhe, Germany
{Stojanovic, Abecker, Studer}@fzi.de
[2] Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
{Stojanovic, Studer}@aifb.uni-karlsruhe.de

Abstract: The increasing complexity of E-Government services demands a correspondingly larger effort for management. Today, many system management tasks are often performed manually. This can be time consuming and error-prone. Moreover, it requires a growing number of highly skilled personnel, making E-Government systems costly. In this paper, we show how the usage of semantic technologies for describing E-Government services can improve the management of changes. We have extended our previous work in ontology evolution, in order to take into account the specificities of ontologies that are used for the description of E-Government services. Even though we use the E-Government domain as an example, the approach is general enough to be applied in other domains.

1. Introduction

E-Government is a way for governments to use the new technologies to provide people with more convenient access to government information and services, to improve the quality of the services and to provide greater opportunities to participate in the democratic institutions and processes [14]. In addition to providing new ways of working with citizens, enterprises, or other administrations, E-Government is also concerned with creating an integrated environment for the development, deployment and **maintenance** of online services. In a fast changing world, this last requirement is especially important. Moreover, in the current economical situation, budgets are reduced and opportunities for gaining efficiency seem to be inevitable: the costs of control and maintenance have become the prime concern of public management. The emphasis in E-Government is thus shifting from implementation to cost efficient operations of service or data centres [9]. This effort includes the development of shared services centres that provide common services to local government organizations without affecting the autonomy of organizations and providing the flexibility to enhance and include additional functionality [5]. In such a distributed environment, the problem of efficient management of changes in E-Government has become even more critical.

The main focus of the current change management activities is the resolution of the so-called *dynamic* modification. It refers to the problem of managing running processes when unanticipated exceptions arise during a task execution, such as the appearance of some hazards in a system, or obtaining some unexpected results. These approaches ensure the consistent operation of a legacy system under unpredictable problems. However, in a dynamically changing political and economical environment, the regulations themselves have to be continually improved, in order to enable the efficient function of a modern society. Taking into account an enormous number of public services and dependencies between them [1], as well as the complexity of interpreting and implementing changes in government regulations, the process of reconfiguring the existing legacy systems (the so-called *static* modification) seems to be quite complex. Indeed, an efficient management system must provide primitives to allow the progressive refinement without rewriting it from scratch, and must guarantee that the new version of the service is syntactically and semantically correct [2]. However, an efficient management system for resolving *static* changes in an E-Government domain does not exist. In this paper, we present such an approach.

The approach is based on enriching current mechanisms for implementing E-Government processes, i.e. web services, with semantic technologies, in order to support a more efficient management of changes. Indeed, the current languages for describing web service[1] and their composition on the level of business processes[2] lack semantic expressivity that is crucial for capturing service capabilities at abstract levels. We argue that business process flow specifications should be defined at abstract task levels, leaving open the details of specific service bindings and execution flows. This abstract level enables the definition of domain specific constraints that have to be taken into account during the (re)configuration of a process flow. In order to model this abstract representation of web services, we introduce a set of ontologies for describing services.

Since the descriptions of semantic web services are ontologies themselves, we base the web services change management on our previous work in the distributed and dependent ontology evolution [11]. It enables us to develop a formal framework for coping with changes which includes the consistency of the service descriptions, possible changes, as well as their resolution. Consequently, we can reason about the change management process, making it very flexible and efficient. Due to our tasks in an ongoing project[3], we have realized our approach for the change management in the E-Government domain. However, the approach is general enough to be applied in an arbitrary application domain that uses (semantic) web services.

The paper is organized as follows: in section 2, we give the conceptual architecture of the change management system. This system is described in section 3. The problem is reduced to the evolution of the *Meta Ontology* (section 3.1). We define the set of changes and consistency constraints that this ontology introduces. Finally, we propose procedures for propagation of changes from business logic to description of services (section 3.2) and between services (section 3.3). Before we conclude, we present an overview of related work.

[1] WSDL - http://www.w3.org/TR/wsdl
[2] BPEL4WS - http://www-106.ibm.com/developerworks/library/ws-bpel/
[3] OntoGov-http://www.ontogov.org

2. Conceptual Architecture

In order to make the description of the approach more understandable, we define here the basic structure of an E-Government system. There are four basic roles played by actors in an E-Government system: (i) politicians who define a law; (ii) public administrators who define processes for realizing a law; (iii) programmers who implement these processes and (iv) end-users (applicants) who use E-Government services. Whereas politicians are suppliers of the E-Government system, end-users are its customers.

Public administrators have the key role. They possess a very good knowledge about the E-Government domain. This knowledge is needed for the design of a public service. It includes the legislation that a service is based on, the respective law, related decrees, directives, prerequisites etc. Based on the interpretation of a law, a public administrator describes a service as a sequence of activities that have to be done, which represents a business process. Due to the changes in the political goals of a government, changes in the environment, and changes in the needs of the people, or due to the possibility to organize regulations in a better way, the politicians might (i) make the revision of a law by accepting an amendment, (ii) enact a new law or (iii) even repeal a law. In the case of a new amendment, the public administrator must understand the changes in the law caused by the amendment; locate activities/services that implement this law, and translate changes into the corresponding reconfiguration of the business process. So far the changes are initiated and propagated manually that causes a lot of errors and redundant steps in the change management process.

Our goal is to free public administrators from many of today's management tasks. We need a system that is not people-intensive anymore, which would result in decreasing of related management costs. Since autonomic computing systems [7] allow people to concentrate on what they want to accomplish rather than figuring out how to do that, we use the analogy with autonomic computing systems and try to apply their principles on the management of the semantic web services. Therefore, the change management system is realised according to the **MAPE** (**M**onitor **A**nalyse **P**lan **E**xecute) model [7], which abstracts the management architecture into four common functions: (i) **Monitor** – mechanism that collects, organises and filters the data about changes in the law or in the needs of end-users; (ii) **Analyse** – mechanism that aggregates, transforms, correlates, visualises the collected data, and makes proposals for changes in the ontologies; (iii) **Plan** – mechanism to structure actions needed to apply the changes by keeping the consistency of the system; (iv) **Execute** – mechanism to update the code of the underlying web services according to the changes applied in the ontology. This is shown in Figure 1.

The change management system continually **monitors** (i) its suppliers (i.e. politicians who define the law) to ensure that it works with up-to date information and (ii) its customers (i.e. the end-users) to ensure that the services, which it offers, meet the customers' requirements. First, each change in the law is stored in the evolution log. This log is based on the formal model of ontology changes (i.e. on the *Evolution Ontology* [18]). Second, to cover bottom-up changes all end-users' interactions with the E-Government portal are tracked in the usage log file. The usage log is structured according to the *Usage Ontology* [16], and contains meta-information about the content of visited pages. Based on the **analysis** of the evolution log and the usage log the recommendations for the continual improvement of the E-Government services may be generated.

The task of the change detection phase of the change management system (cf. Change Detection in Figure 1) is (i) to locate services that are out-of-date and (ii) to determine how to change them. Our goal is to develop the change management system that allows for the change propagation and traceability, contributing in this way to the bridging of decision making with technical realisation. To achieve this goal, it is not sufficient to use ontologies for modelling services and to focus public administrators only on the relevant services. Rather it is required to model the dependencies between different stakeholders that define this business logic in a collaborative way. In order to help public administrators find out the right changes needed to synchronise the service with the law, we develop the so-called *Lifecycle Ontology*. It describes the information flow and the decision making process in the public administration. It is intended to support the transition from knowledge acquisition to implementation, i.e. the design phase [10]. Therefore, it includes entities for documenting design decisions and the underlying rationale. In this way it gives concrete clues (i.e. **plans**) on how a service has to be modified.

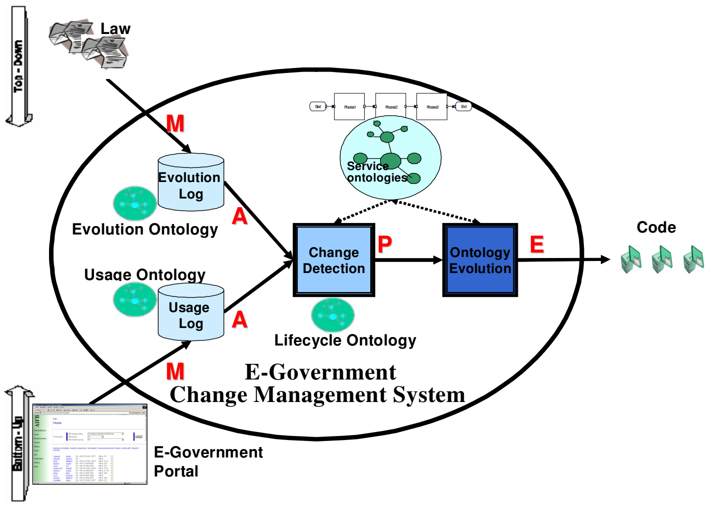

Figure 1. The conceptual architecture of the management system.

Since the application of a single ontology change can cause the inconsistency in the other part of this ontology and all the artefacts that depend on it, the ontology evolution process (cf. Ontology Evolution in Figure 1) has to be applied. It guaranties the transfer of the ontology and dependent artefacts into another consistent state. In this phase the specificities of the E-Government domain must be taken into account. Namely, a new amendment of some law does not cause the ontology inconsistency. However, it causes the so-called semantic web service inconsistency, since the existing E-Government services do not reflect the law.

Finally, the change management system has to notify programmers about necessity to update a code, i.e. the modification of the code has to be **executed**. However, this phase cannot be automated, since a programmer has to modify a code manually.

3. Approach

We have developed an approach for the change management of semantic web services. Note that even though we use the E-Government domain as an example, the approach is general enough to be applied in other domains. In order to emphasise this generality, in this section, we substitute the E-Government vocabulary used in the previous section with the commonly-used business process management terminology. Therefore, instead of the term *law* we use a *business rule*, a *public E-Government service* is treated as a *business process* and a *manager* plays the role of a *public administrator*.

Since we assume that services are described using ontologies, the management of changes requires the management of these semantic descriptions. Therefore, our approach can be based on our previous work in ontology evolution. Moreover, we have extended the work (see section 3.1), in order to take into account the specificity of semantic web services. Then, in section 3.2 we discuss the way of bridging the gap between business rules[4] and semantic web services implementing these rules. Finally, in section 3.3 we define procedures for the change propagation within the description of the semantic web services by defining the semantics of the required changes.

3.1 Evolution of the Semantic Web Service Ontology

Ontology evolution can be defined as the timely adaptation of an ontology and a consistent propagation of changes to the dependent artefacts [18]. In this paper, we extend our approach for ontology evolution toward handling the evolution of semantic web service ontologies. Since the evolution is driven by the set of changes that have to preserve the consistency, the approach requires (i) the explicit specification of changes that can be applied and (ii) the consistency definition. Both of them heavily depend on the underlying model and thus they vary from application to application. Therefore, we firstly introduce an ontology for describing semantic web services (section 3.1.1). Secondly, we define more complex changes that can be applied to these descriptions (section 3.1.2). Finally, we specify the consistency constraints that are derived from the semantics of this ontology (section 3.1.3).

3.1.1. Ontologies used for modelling semantic web services

The first step that has to be clarified is the description of web services. We distinguish among the following ontologies:

- *Meta Ontology* - it contains entities needed to describe services;
- *Domain Ontology* - it contains domain specific knowledge;
- *Service Ontologies* - they describe concrete services.

For each service, a *Service Ontology* that includes the *Meta Ontology* and the *Domain Ontology* is defined, and it might include (reuse) other *Service Ontologies*. For example, the service ontology for the driving licence issuance E-Government service describes that it is a composite service that is realized through the application,

4 Note that in the E-Government domain business rules represent the laws, since the laws define how to realize the E-Government services.

verification/qualification etc., which can be considered as atomic services (i.e. an activity). Therefore, it includes the *Meta Ontology*, since the *Meta Ontology* defines the building blocks for the service description. Each of these services (application, verification/qualification etc.) is related to the *Domain Ontology*. For example, the application service requires the birth certificate that is the domain knowledge.

We do not consider dynamic web services whose process flow can be composed on the fly. However, we allow the dynamic binding of web services during the execution. Therefore, we focus on the static web services, whose composition is explicitly predefined by the business rules (i.e. a law). In order to model the dependency between a business rule and the service implementing it and to take into account the other specificities of the E-Government services we introduce the *Meta Ontology*. We note that it is not possible to reuse OWL-S[5] or WSMO[6] that are the most salient initiatives to describe semantic web services. Whereas the WSMO ontology does not contain the process model, the OWL-S ontology does not allow[7] using the domain ontology entities as inputs/outputs of an activity in the process model. Moreover, the formalism for expressing conditions is not defined.

Similarly to the OWL-S ontology, the *Meta Ontology* consists of two parts: the profile that is used for the service discovery and the process model that is used to describe the process flow. To define the *profile* we extend the *OWL-S service profile ontology* in several ways. First, we define the property "*hasReferencedBusinessRule*" that establishes a reference between the service description and the business knowledge that is represented in the form of an ontology. This ontology is called *Business Rule* ontology and depends on the application domain. In the E-Government domain, this ontology contains the knowledge about laws, and is called the *Legal Ontology*. It is important mentioning that this ontology may be used as a well-defined vocabulary (semantics) for describing (annotating) both the content and the structure of legal documents [3]. However, for the problem we are aiming to resolve, it is necessary to model only the structure of legal documents, not their content. More information about the *Legal Ontology* is given in [15].

The second extension of the service profile ontology comes from the business process modelling point of view. Indeed, in order to model the resources involved in a business process, we introduce additional entities such as the property "*requires*" and the concept "*Resource*" which can be either a person who is involved in the executing a service or an equipment (i.e. hardware or software) that performs a service automatically. In that way, we establish a bridge between the common language used by business people – in order to describe the business processes (i.e. web services) - and the ontology language used for describing web services.

Finally, the last extension of the *OWL-S service profile ontology* is achieved by taking into the consideration the standard metadata defined for the particular domain, since ontologies may advance metadata solutions. Our goal was to model all information that exists in the standard including the implicit knowledge. Even though we use the *CEN Application Profile v.1.0 metadata standard*, which is used as a standard in the E-Government domain, we note that similar strategies can be applied

[5] http://www.daml.org/services/owl-s/1.0/

[6] http://www.wsmo.org/

[7] In OWL Lite and OWL DL classes and individuals form disjoint domains. OWL Full is not decidable.

for other standards as well. The approach can be summarized as follows: (i) the metadata standard is transformed into a set of the ontology properties that are explicitly included in the *Meta Ontology*; (ii) the *Meta Ontology* is extended with several concepts (e.g. the concept "*Topic*") representing ranges of these properties with the goal to improve service discovery; (iii) "hidden" (hard-coded) knowledge embedded in the standard is translated into a set of rules in the corresponding ontologies and is used in typical inferencing tasks.

To describe the process flow we combine the results of the *OWL-S process ontology* with the experiences from the business process modelling by taking into the consideration the specificities of the E-Government domain. Similarly to the *OWL-S process ontology*, we distinguish between the services and the control constructs. Services can be either atomic or composite services. For each service we define the standard set of attributes such a name, a description, etc. However, there are specific requirements concerning retraceability, realisation, security, cost etc. Therefore, we introduce the E-Government specific properties:

- each service is associated to the law it is based upon. We note that it is very important to document the laws and regulations not only for the whole process but also for specific activities;
- each service is associated to the software component that implements it. However, it is possible that the same description of the service is related to the different implementation. For example, a service about performing deregistration of a citizen is related to the several implementations depending on the target municipality. To inform the workflow engine about the software component that has to be invoked, it is necessary to model the decision attribute;
- it is necessary to assign security levels to each service;
- information about cost and time restrictions can be also specified.

Similarly to the *OWL-S process ontology*, services have the inputs and output. The concepts "*Input*" and "*Output*" are defined as subconcepts of the concept "*Parameter*". Since some inputs have to be provided by the end-user the concept "*User-defined Input*" is defined as a specialisation of the concept "*Input*". To establish the equality between two parameters we introduce the symmetric property "*isEqualTo*".

Since it is required that inputs/outputs are defined in the domain ontology, we introduce the additional concept "*Reference*" due to two reasons: (i) a property may be attached to several domain concepts; (ii) a concept defined in the domain ontology may have many properties and only a subset of them is used as an input. In order to specify the context of the usage of a property and to select a subset of them, we introduce the properties "*hasConcept*" and "*hasProperty*" respectively. The range of these properties is the root "*KAON-Root*" concept that is included in each KAON[8] ontology. By using the KAON meta-modelling facilities, it is possible to reference any entity (i.e. a concept, a property or an instance) defined in the domain ontology. Furthermore, to name a parameter we define the attribute "*hasName*".

The next difference in comparison to the *OWL-S process ontology* is related to the conditions of a service. While *OWL-S* uses preconditions and effects to refer to the changes in the state of resources, we accept the *WSMO* interpretation. We use

[8] http://kaon.semanticweb.org

preconditions for defining what a service expects for enabling it to provide its service. Postconditions define what the service returns in response to its input. Indeed, they establish the relationship between inputs and outputs.

For a composite service we define the following additional properties: the property "*hasFirst*" indicating the first service in the process flow[9] and the transitive property "*consistsOf*" indicating all services that a service includes. Further, a set of rules is specified. For example, if a part of a service (either an atomic or a composite service) is related to the some part of the law, then the service itself is related to the same part of the law.

The process model provides the following control structures: sequence, split, join and if-then. We do not include while, repeat etc. which are defined in the *OWL-S process ontology*, since none of the E-Government use-cases we analysed require them. To connect the services and the control constructs we define the following properties: (i) "*hasNextControlConstruct*" and "*hasPreviousControlConstruct*" whose domain is the concept "*Service*" and range is the concept "*ControlConstruct*"; (ii) "*hasNextService*" and "*hasPreviousService*" whose domain is the concept "*ControlConstruct*" and range is the concept "*Service*". The properties are inverse of each other. For the concept "*if-then*" several additional properties are defined in order to determine the next services based on the fulfillment of the condition. A process part of the *Meta Ontology* is shown in Figure 2.

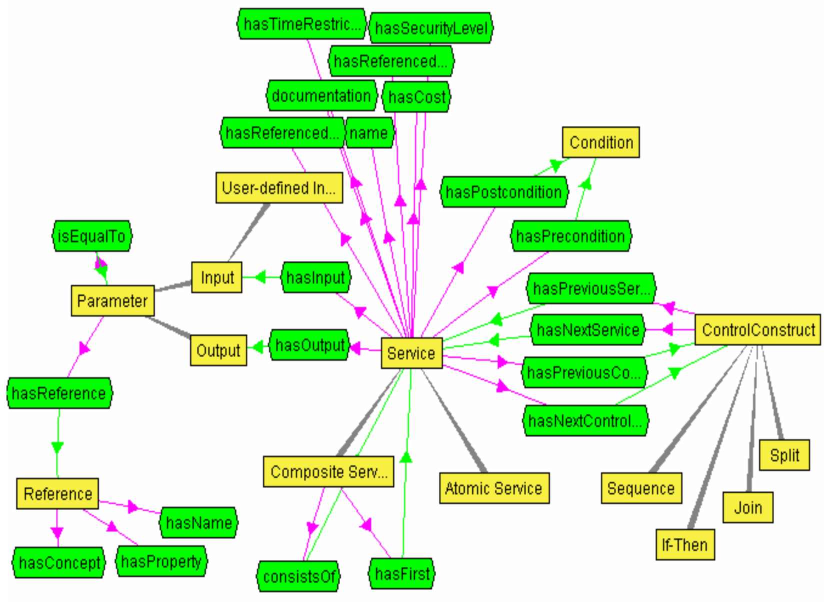

Figure 2. A process part of the Meta Ontology

[9] This information can be considered as redundant since it can be always derived from a service model based on a rule that each service that does not have a previous service/control construct is a first service. However, it should be specified in order to optimise the run-time performance.

3.1.2 Changes

The set of ontology changes[10] includes all elementary changes (e.g. "*AddConcept*") and some more complex changes, the so-called composite changes (e.g. "*MoveConcept*") [17]. However, this granularity level should be extended, in order to enable a better management of changes in a service description. For example, to make the service s1 a predecessor of the service s2, the manager needs to apply a *list* of ontology changes that connects outputs of s1 to the corresponding inputs of s2. We cannot expect that she spends time finding, grouping and ordering the ontology changes to perform the desired update. In order to do that, she should be aware of the way of resolving a change, she should find out the right changes, foresee and solve the intermediate conflicts that might appear, and order changes in a right way. This activity is time consuming and error prone, especially if an ontology is large.

Therefore, managers require a method for expressing their needs in an exacter, easier and more declarative manner. For them, it would be more useful to know that they can connect two services, rather than to know how it is realized. To resolve the above mentioned problem, the intent of the changes has to be expressed on a more coarse level, with the intent of the change directly visible. Only in this way can managers focus on what has to be done, and not on how to do that.

To identify this new level of changes, we start from the *Meta Ontology*. For each service, one can specify inputs, outputs, preconditions, postconditions, resources and business rules, other services that it either specializes or is connected with. Each of these entities can be updated by one of the meta-change transformations: add and remove. A full set of changes can thus be defined by the cross product of the set of entities of the *Meta Ontology* and the set of meta-changes. A part of them[11] is shown in Table 1.

Table 1 The taxonomy of changes of the semantic web ontology

	Additive Changes	**Subtractive Changes**
Service	AddService	RemoveService
Input	AddServiceInput	RemoveServiceInput
Output	AddServiceOutput	RemoveServiceOutput
Precondition	AddServicePrecondition	RemoveServicePrecondition
Postcondition	AddServicePostcondition	RemoveServicePostcondition
Service Specialisation	AddServiceSpecialisation	RemoveServiceSpecialisation
Next Connection	AddServiceNextService	RemoveServiceNextService
Previous Connection	AddServicePreviousService	RemoveServicePreviousService
Business Rule	AddServiceBusinessRule	RemoveServiceBusinessRule
Resource	AddServiceResource	RemoveServiceResource
Software	AddServiceSoftware	RemoveServiceSoftware

[10] Elementary ontology cannot be decomposed into simpler changes. Composite ontology changes represent a group of elementary or composite changes that are applied together.

[11] We focus here only on the entities that are important for management. Other aspects, such as properties defined in the service profile, are ignored. Due to this abstraction of the *Meta Ontology*, only the most typical and most frequently occurring changes are shown.

The changes shown in Table 1 build the backbone of a semantic web service management system. They make the evolution of the semantic description of web services much easier, faster, more efficient, since they correspond to the "conceptual" operation that someone wants to apply without understanding the details (i.e. a set of ontology changes) that the management system has to perform. These changes can be further combined into more complex changes, such as grouping of services. Further, each of these changes is internally realized as a set of elementary or composite ontology changes.

3.1.3 Consistency

To define the consistency of the semantic web service ontologies, we start from the ontology consistency definition [17]: *An ontology is consistent with the respect to its model if and only if it preserves the constraints defined for the underlying ontology model*.

This set of constraints includes invariants, which are consistency rules that must hold for every ontology. For example, a concept hierarchy in the KAON ontology model must be a directed acyclic graph. Since ontologies that are used to describe semantic web services include other ontologies, we define the dependent ontology consistency in the following way [11]: *A dependent ontology is consistent if the ontology itself and all its included ontologies, observed alone and independently of the ontologies in which they are reused, are ontology consistent*.

The *Meta Ontology* can be considered as the meta-level for the semantic web service description. Since the set of consistency constraints heavily depends on the underlying model, the semantics of the *Meta Ontology* defines a set of constraints that all service ontologies have to fulfil. In this section, we discuss how the existing dependent ontology consistency definition has to be enriched, in order to take into account the specificities of *the Meta Ontology*. We introduce the following additional constraints[12]:

- ***Service profile specific constraints***:
  - *Business knowledge specific constraints*
    C1: Each service has to have a reference to at least one business rule.
  - *Tracebility*
    C2: Each service has to have at least one resource that controls its execution.
  - *Applicability*
    C3: Each service has to have at least one software component attached to it that implements it.
- ***Service process specific constraints***:
  - *Completeness*
    C4: Each service has to have at least one input.
    C5: Each service has to have at least one output.
    C6: Each service input has to be either output of some other service or is specified by the end-user.
  - *Satisfyability*
    C7: If the input of a service is the output of another service, then it has to be

[12] Note that each of these constraints is formally defined and is automatically verified against service descriptions.

subsumed by this output.

C8: If the input of a service subsumes the input of the next service, then its preconditions have to subsume the preconditions of the next one.

C9: If two services are subsumed by the same service, then their preconditions have to be disjoint.

- *Uniqueness*

  C10: If a service specialises another service, one of its parameters (i.e. inputs, outputs, preconditions or postconditions) has to be different. The difference can be achieved either through the subsumption relation with the corresponding counterpart or by introducing a new one.

- *Well-formedness*

  C11: Inputs, outputs, preconditions and postconditions have to be from the domain ontology.

- ***Domain specific constraints***:
  - *Structural dependency*

    C12: Any specialisation of the service S1 must always be a predecessor of any specialisation of the service S2, where S1 and S2 are two services defined in the *Meta Ontology* and their order is given in advance (i.e. S1 precedes S2).

It is worth mentioning that only consistency constraints *C1* and *C12* are domain-dependent. Whereas *C1* has a reference to the *Business Rules Ontology*, *C12* is related to the generic schema for the services and specifies the obligatory sequence among activities. In the E-Government domain, *C1* requires that each service is related to a law. *C12* states that the structure of *Service Ontologies* must follow predefined rules, so that a service specialising an application service has to precede a specialisation of a verification service.

We give short interpretations of some constraints from the change management point of view:

- *C1* enables to find the corresponding service if a law is changed;
- *C6* ensures that a change in an output of an activity is propagated to the inputs of successor activities and vice versa;
- *C8* prohibits the changes which lead to non-optimal service reconfiguration. For example, if the preconditions for an activity include a constraint that a person has to be older than 18, the preconditions of the next activity cannot be that a person has to be older than 16.

Finally, we define the consistency of the semantic web services in the following way: *A semantic web service is a **consistent service** iff its description is dependent ontology consistent and the additional constraints (C1-C12) are fulfilled.*

Note that a change in the business logic does not cause any ontology inconsistency. Regarding the E-Government domain, after the removal of a single input of an activity, the ontology consistency is still fulfilled. However, this change provokes the semantic web service inconsistency, since the consistency constraint *C4* is not satisfied. Therefore, the extension of the consistency definition is a prerequisite for the management of the semantic web services.

Since semantic web services must be compliant with the set of semantic web service consistency constraints, in the rest of this section, we discuss how to preserve the consistency. In section 3.2 we define a procedure that informs managers about changes in the business rules that provoke some inconsistencies. Thereafter, in section 3.3 we introduce the procedures for ensuring the semantic web service consistency.

3.2 Propagating changes from business rules to services

The basic requirement for a management system is that it has to be simple, correct and usable for managers. Note that they are responsible for keeping semantic web services up-to-date and don't need to be experienced ontology engineers. Thus, a management system must provide capabilities for the automatic identification of problems in the (description of the) semantic web services and ranking them according to the importance. When such problems arise, a management system must assist the managers in identifying the sources of the problem, in analysing and defining solutions for resolving them. Finally, the system should help in determining the ways for applying the proposed solutions.

In this section we define the procedure for finding the "weak places" in the description of the semantic web services by considering the changes in the business rules and their impact on the consistency. The procedure is focused on discovering inconsistencies in a semantic web service description, whose repairing improves the agreement of this ontology with the business rules. When we designed this support, we assumed that the update would be only a partially automated process rather than a fully automated process. For example, we do not want to update web services automatically, but rather to notify the managers about problems. It is up to the manager to decide how to resolve those problems. Our experience shows that this assumption is reasonable. In the E-Government domain, certain tasks could be automated, while other tasks could be supported, but not fully automated. For example, the manager should be informed about a new amendment. However, the realization of this amendment must not be automated, since it requires a lot of domain knowledge that cannot be formally represented in the *Legal Ontology*, and is a result of experiences. Therefore, our system only makes recommendations about a potential resolution of a problem. For example, a new amendment might be realized through the specialisation of a web service that implements the law for which this amendment is defined.

Obviously, the information about the business rule that is implemented by a service is very important for the change management. It means that the consistency can be achieved only by referring to this knowledge. This was one of the reasons for defining the *Meta Ontology* (see section 3.1.1).

The procedure for propagating changes from business rules to web services is based on our previous work on the evolution between dependent and distributed ontologies, since we assume that the *Business Rule Ontology* is reused in the *Meta Ontology* through the replication [11]. In the E-Government domain, the physical distribution is very important, since E-Government services must follow federal, state and local laws that are defined externally. Note that a *Service Ontology* might reuse the *Meta Ontology* either through inclusion or replication, which depends whether they are within the same system or not.

The procedure consists of four steps:

1. ***Checking actuality of the Business Rules Ontology*** – Since each ontology has a version number associated with it that is incremented each time when the ontology is changed, checking the equivalence of the original of the *Business Rules Ontology* and

the replica can be done by a simple comparison of the version numbers.

2. ***Extracting Deltas*** – After determining that the included *Business Rules Ontology* needs to be updated, the evolution log for this ontology is accessed. The extracted deltas contain all changes that have been applied to the original after the last synchronisation with the replica, as determined by the version numbers. For example, after the addition of the new amendment A7 in the *Legal Ontology* as the adaptation of the paragraph P2, the delta will contain changes shown in Figure 3.

```
<a:AddInstanceOf rdf:ID="i-1079962974979-1202219624"
   a:has_referenceConcept="Legal#Amendment"
   a:has_referenceInstance="Legal#A7"
   a:inOIModel="file:/C:/ontoGov/Legal"
   a:version="10">
...
</a:AddInstanceOf>
<a:AddPropertyInstance rdf:ID="i-1079962991620-1904187797"
   a:has_referenceProperty="Legal#modifies"
   a:has_referenceSourceInstance="Legal#A7"
   a:has_referenceTargetInstance="Legal#P2"
...
<a:has_previousChange rdf:resource="#i-1079962974979-1202219624"/>
</a:AddPropertyInstance>
```

Figure 3. A part of the evolution log of the *Legal Ontology*

3. ***Analysis of changes*** – Each performed change is analysed, in order to find semantic web services that have to be updated. We distinguish between the addition and the deletion of an entity from the *Business Rule Ontology*. Removals can be resolved directly by applying the existing ontology evolution system, since it ensures the consistency by generating addition changes [18]. However, the addition requires an additional effort that depends on the structure of the *Business Rules Ontology*. Here we describe how this problem is resolved in the E-Government domain by considering the *Legal Ontology*. We analyse the addition of a new amendment. The goal is to find services that realize the law related to this amendment, and to order them in an appropriate way. Since each service is referred to a law/chapter/paragraph/article, the corresponding service can be easily found. In case there are several services referring to the given law (e.g. through a paragraph or an amendment), they are ranked according to the semantic similarity that is based on calculating the distance between two entities in the hierarchy we proposed in [19].

4. ***Making recommendation***: In order to make recommendations how to adapt the up-to-date semantic web services we use the *Lifecycle Ontology*. It describes design decisions and their relationship to affected parts of the service as well as to the requirements that motivate the decisions [10]. Since the *Lifecycle Ontology* is a description of the service design process, which clarifies which design decisions were taken for which reasons, it proves to be valuable for further development and maintenance. During ongoing development, it helps the managers to avoid pursuing unpromising design alternatives repeatedly, but it also facilitates maintenance by improving the understandability of the service design. A description of the design process also supports traceability, since it links parts of the service design to the portions of the specification (i.e. to the activities/services in the process description) they were derived from and to the requirements that influenced design decisions in that derivation. More information about this ontology in given in [15].

3.3 Propagating changes within services

The key process in the change management is the resolution of the changes triggered by the procedure described in the previous section. It has to guarantee that a change is correctly propagated and that no inconsistency is left in the system. If this was left to the managers, the change management process would be error-prone and time consuming – it is unrealistic to expect that humans will be able to comprehend all the existing services and interdependencies between them. For example, in the E-Government domain, an unforeseen and uncorrected inconsistency is one of the most common problems.

Therefore, the change management has to be supported by a tool that improves the efficiency and the quality of this process. In order to develop such a tool, the problem has to be formulated in terms of a formal model. Since our approach is based on the semantic description of services, the formal model requires the specification of the semantics of changes that can be applied to the semantic web services.

For each change introduced in section 3.1.2, it is required to specify: the necessary ***preconditions***; (ii) the sufficient ***postconditions*** and (iii) the possible **actions**.

The ***Preconditions*** of a change are a set of assertions that must be true, to be able to apply the change. For example, the preconditions for the change *AddServiceSpecialisation(S1,S2)*, which results in the specialisation of the service *S1* in the service *S2*, are: (i) *S1* and *S2* are different services; (ii) *S2* is not an indirect parent (through the inheritance hierarchy) of *S1*; (iii) *S2* is not already defined as a specialization of *S1*; (iv) for each input/output/preconditions/postconditions of *S1*, there is a corresponding element in *S2* that is subsumed by the original.

The ***Postconditions*** of a change are a set of assertions that must be true after applying the change, and it describes the result of the change. For example, the removal of a service results in the fact that this service is not in this service ontology anymore.

The ***Actions*** are additional changes that have to be generated, in order to resolve the side effects of a change on other related entities. It means that each inconsistency problem is treated as a request for a new change, which can induce new problems that cause new changes, and so on. An inconsistency arises when one of the semantic web service consistency constraints (see section 3.1.3) is corrupted. For example, the addition of a service will trigger the addition of an input for this service (i.e. *AddServiceInput* change), since the consistency constraint *C4* requires that each service has to have at least one input.

To define the actions for changes introduced in section 3.1.2, we started by finding out the cause and effect relationship between them. The approach is based on a common technique for the maintenance of the knowledge-based systems [12], which states that dependencies between knowledge have to be represented explicitly. However, while in these systems the dependency graph consists of knowledge elements (e.g. rules in the expert systems), in our system the nodes of this graph are changes. For more details see [17].

Due to the lack of space, it is not possible to specify semantics for all changes introduced in section 3.1.2. Here we define the procedures for the *AddServiceInput(service, input)* change:

- **Preconditions** $- input \notin Inputs(service)$, where *Inputs* is a set of all inputs already defined for a service. This is in agreement with the single ontology consistency constraints that ensure the uniqueness of the definition.

- **Postconditions** – $input \in Inputs(\, service\,)$, which means that this input is defined for this service.
- **Actions** – *AddEqualsTo(input, x)*, where:

$\exists s1 \quad service \in hasNext(\, s1\,) \wedge x \in Outputs(\, s1\,) \wedge H_c^*(\, input, x\,)$, where *hasNext*[13] is a property defined in the *Meta Ontology* for connecting services, *Inputs/Outputs* represent a set of inputs/outputs defined for a service and H_c^* is the transitive closure of a concept hierarchy H_c.

A new input might corrupt the *C6* consistency constraint, since the inputs provided by the end-users are usually defined for the first service in the process flow. To resolve this problem, one has to specify that this input is provided by the output of the previous service. This can be realized as a request for a new change *AddEqualsTo*, which establishes the "*IsEqualsTo*" property between corresponding input/output parameters.

For example, according to the changes in a law, the driving licence verification activity requires fingerprint. This change causes the inconsistency, since the new input hangs. The problem can be resolved by generating the additional change *AddEqualsTo* between the verification activity and its predecessor. This further induces a new output of the predecessor, i.e. application activity, which can potentially trigger other changes, and so on.

Finally, it is important to note that any change in the domain ontology is resolved automatically by using the existing ontology evolution system [17]. For example, let's consider that the domain ontology contains the concept *Person* and two its specializations: *Child* and *Adult*. Since there is a special procedure for the passport issuance for the children, this service is a specialization of the standard service passport issuance service. The application service of the service for children requires an additional input (i.e. the parent authorization). The precondition for this application service is that it is required for a child. Let's now consider that the concept *Child* needs to be removed. The ontology evolution system will propagate this change to all ontologies that included the changing ontology. Therefore, the ontology describing the passport issuance procedure for children will also be informed about changes. Since, according to the ontology consistency definition, undefined entities are not allowed, the request for the removal of the corresponding application service will be generated.

5. Related Work

Although the research related to Web Services has drastically increased recently, there are very few approaches that cope with the changes in the process flow of a web service. The management approaches are mainly focused on the composition of a web service from scratch and neglect the problem of the continual improvement of the service. The change management approaches are mainly focused on re-implementing some software modules [5]. We found two reasons for such behaviour:

[13] $s2 \in hasNext(s1)$ means that *s2* is one of the services that *s1* precedes through one control construct.

1. Since the technology is rather new, the real challenges for the change management are still to come. Indeed, in the workflow community, from which web services are transferring a lot of experiences, the workflow maintenance is a well-researched topic;
2. The description of web services lacks a conceptual level on which the reasoning about a compositional model, including the reasons and the methods for its reconfiguration, will be possible. As we have already mentioned, the emerging semantic web services approaches introduce such a level and we give here a short overview of their achievement in the (re)composition.

Workflow
The workflow community has recently paid attention to configurable or extensible workflow systems, which present some overlaps with our ideas. For example, the work on flexible workflows has focused on the dynamic process modification [6]. In this publication, workflow changes are specified by transformation rules composed of a source schema, a destination schema and of conditions. The workflow system checks for parts of the process that are isomorphic with the source schema and replaces them with the destination schema for all instances for which the conditions are satisfied. However, the workflow schema contains fewer primitives than an ontology so that this approach is much less comprehensive then ours. Moreover, the change in the business policy is not treated at all.

The most similar to our approach is the work related to the workflow evolution [2]. This paper defines a minimal, complete and consistent set of modification primitives that allow modifications of workflow schemata. The authors introduce the taxonomy of policies to manage the evolution of running instances when the corresponding workflow schema is modified. However, the authors are focused on the dynamic workflow evolution, which is not the focus of this work.

Semantic Web Services
Recently, the approaches for the composition of semantic web services have emerged drastically. We discuss only the most relevant to our approach.

In [8] a framework for the interactive service composition is presented, where the system assists users in constructing a computational pathway by exploiting the semantic description of services. Given the computational pathway and the user's task description (i.e. a set of initial inputs and expected results), the system performs a set of checks (e.g. are all the expected results produced, are all the needed input data provided) in order to ensure the consistency of the resulted model. The checks used in this approach can be seen as a subset of the constraints we defined for ensuring the consistency. Moreover, since we derive the constraints from the ontology model behind the semantic web services, we can guarantee the completeness and the consistent propagation of the changes.

In [20] the authors present a prototype for dynamic binding of Web Services for the abstract specification of business integration flows using a constraint-based semantic-discovery mechanism. They provide a way of modelling and accommodating scoped constraints and inter-service dependencies within a process flow while dynamically binding services. The result is a system that allows people to focus on creating appropriate high-level flows, while providing a robust and adaptable runtime. Similarly to our approach they contend that the selection of Web services for a step in a process flow is, often, not a stand-alone operation, as there may be dependencies on the previously chosen services for the process. They introduce two types of dependencies: description-based and domain constraints whereas both of them can be easily mapped into our business-knowledge specific constraints that ensure the meaningful order between services in a flow. Additionally we provide process specific constraints that ensure the consistency of the process flow.

Next, there are several approaches for the automatic composition of semantic web services [4], [13] that drive the design at a conceptual level in order to guarantee its correctness and to avoid inconsistencies among its internal components. In that context, our approach can be seen as an automatic re-composition of a service driven by the constraint derived from the business environment, domain knowledge and internal structure of a service.

Finally, the main difference between our approach and all the related researches is that we base our management framework on the systematic evolution of the model that underlines semantic web services (i.e. several dependent and distributed ontologies). It enables us to be predictive in the management (i.e. we can reason about the consequences of changes in the system) and to expand the framework whereas the consistency of the managed system is ensured, easily.

6. Conclusion

In this paper, we presented an approach for the management of changes of semantic web services. The approach extends our previous work on evolution of multiple and distributed ontologies. As a case study, we considered the E-Government domain, since E-Government services are under the continual adaptation to the political goals of a government and to the needs of the people. Up to now, the changes have been initiated and propagated manually, which causes a lot of errors and redundant steps in the change management process. Our approach enables the automation of the change propagation process, and ensures its consistent execution, since it is based on a formal framework for coping with changes. Consequently, we can reason about the change management process, making it very flexible and efficient. The approach has been implemented in the KAON framework.

In the future, we want to extend this approach by suggesting changes that can improve services. This can be done (i) by monitoring the execution of E-Government services (e.g. the activity that causes the delay is a candidate for optimization) and/or (ii) by taking into account the end-users' complaints (e.g. end-users might not be satisfied with the quality of services, since they have to supply the same information several times).

Acknowledgement

The research presented in this was partially financed by EU in the project "IST PROJECT 507237 - OntoGov" and by BMBF in the project "SemiPort" (08C5939).

References

[1] N. Adam, t al., *E-Government: Human centered systems for business services*, In Proc. of the First National Conference on Digital Government, pp. 48–55, 2001.

[2] F. Casati, et al., *Workflow Evolution*, In Proc. of the 15th Int. Conf. on

Conceptual Modelling (ER'96), Cottbus, Germany, pp. 438-455, 1996.

[3] A. Gangemi, et al., Some *Ontological Tools to Support Legal Regulatory Compliance - with a Case Study*, Workshop on Regulatory Ontologies, OTM'03, pp. 607-620, 2003.

[4] A. Gomez-Perez, et al., *A Framework for Design and Composition of Semantic Web Services*, First International Semantic Web Services Symposium, pp. 113-120, 2004.

[5] M. Janssen, R. Wagenaar, *An analysis of a shared services centre in E-government, System Sciences*, In Proc. of the 37th Annual Hawaii International Conference, pp.124- 133, 2004.

[6] G. Joeris, O. Herzog, *Managing Evolving Work of Specifications with Schema Versioning and Migration Rules*, TZI Technical Report 15, University of Bremen, 1999.

[7] J. Kephart, D. Chess, *The Vision of Autonomic Computing*, IEEE Computer, pp. 41-50, 2003.

[8] J. Kim, Y. Gil, *Towards Interactive Composition of Semantic Web Services*, First International Semantic Web Services Symposium, pp. 100-107, 2004.

[9] G. Leganza, IT Trends 2003, Midyear Update: Enterprise Architecture, Report Giga Group, 2003.

[10] D. Landes, *Design KARL – A language for the design of knowledge-based systems*, In Proc. of the 6th International conference on Software Engineering and Knowledge Engineering (SEKE'94), Jurmala, Lettland, pp. 78-85, 1994.

[11] A. Maedche, et al., *Managing multiple and distributed ontologies on the Semantic Web*, the VLDB Journal (2003) - Special Issue on Semantic Web, 12:286-302, 2003.

[12] T. Menzies, *Knowledge maintenance: The state of the art,* The Knowledge Engineering Review, pp. 1-46, volume 14, number 1, 1999.

[13] S. Narayanan, S. McIlraith, S*imulation, Verification and Automated Composition of Web Services*, In Proc. of the WWW-2002, pp. 77-88. Hawaii, USA, 2002.

[14] J.E. Stiglitz, et al., *The Role of Government in a Digital Age*, http://www.ccianet.org/digital_age/report.pdf, 2000.

[15] L. Stojanovic, et al., available as Deliverable D2, http://www.ontogov.org, 2004

[16] L. Stojanovic, et al*., OntoManager - a system for the usage-based ontology managemen*t, In Proc. the ODBASE 2003, pp. 858-875, 2003.

[17] L. Stojanovic, et al., *Ontology Evolution as Reconfiguration-design Problem Solving*, In Proc. of the int. conference on Knowledge capture – K-CAP'03, pp. 162 – 171, 2003.

[18] L. Stojanovic, et al., *User-driven Ontology Evolution Management*, In Proc. of the EKAW'02, Siguenza, Spain, pp. 285-300, 2002.

[19] N. Stojanovic, et al., *SEAL — A Framework for Developing SEmantic PortALs*, In Proc. of the K-CAP'01, Victoria, British Columbia, Canada , pp. 155 – 162, 2001.

[20] K. Verma, et al., *On Accommodating Inter Service Dependencies in Web Process Flow Composition*, First International Semantic Web Services Symposium, pp. 37-43, 2004.