The role of semantics in e-government service model verification and evolution

L. Stojanovic[(1)], A. Abecker[(1)], D. Apostolou[(3)], G. Mentzas[(2)], R. Studer[(1)]
[(1)]FZI, Research Center for Information Technologies at the University of Karlsruhe, Germany
[(2)] National Technical University of Athens, Greece
[(3)] University of Pireaus, Greece

Abstract

e-government systems are subject to a continual change. The importance of better change management is nowadays more important due to the evolution of Europe towards a multicultural, more open and international society with changing common values, increasing levels of education, demographic involvement and adoption of new technologies. In this paper we show how semantic technologies may improve change management. The novelty of the approach lies in the formal verification of the service description as well as in the using of formal methods for achieving consistency when a problem is discovered.

Introduction

An important characteristic of today's business systems is their ability to adapt themselves efficiently to the changes in their environment, as well as to the changes in their internal structures and processes. The continual reengineering of a business system, i.e. the need to be better and better, is becoming a prerequisite for surviving in the highly changing business world. Although changes encompass several dimensions of a business system (e.g. people, processes, technologies), most of them are reflected on its IT infrastructure. For example, the establishment of a new department in the organizational structure will require the corresponding changes in the business processes, enterprise portal, underlying groupware system, skill management system, etc. Therefore, the adaptability of the implemented IT solutions directly defines the efficiency of a business system.

However, building and maintaining long-living applications that will be "open for changes" is still a challenge. Change management in general refers to the task of managing change, which means making of changes in a planned and systematic fashion (Nickols, 2003). The aim is to more effectively resolve changes in an ongoing organization.

Change management is especially important for the applications that are distributed over different systems. Due to our tasks in an ongoing project[1], in this paper we treat change management problem in e-government systems. Indeed, e-government systems (e.g. portals) are typical examples of distributed applications. They enable integration of various, physically distributed services differing in the level of formality and the structure.

The changes to be managed lie within and are controlled by the public administrations. The most frequent changes are the changes of existing business processes based on the adaptation of the business goals, organisational structure or due to possibility to organise processes in a better way. For example, public administrations at the government level or at the federal level work on supporting unification of e-government services, on standards for data exchange as well as on providing examples of the process models of public services that are implemented by municipalities.

Moreover, the internal changes might have been triggered by events originating outside the public administration, i.e. by "the environment." Hence, the change management must take into account the response to changes over which the public administration exercises little or no control (e.g., legislation, social and political upheaval, the actions of competitors, shifting economic tides and currents, and so on). On the other hand, in a dynamically changing political and economical environment, the regulations themselves have to be continually improved, in order to enable the efficient functioning of a modern society. Taking into account an enormous number of public services and dependencies between them, as well as the complexity of interpreting and implementing changes in government regulations, the process of reconfiguring the existing public services seems to be quite complex. It is necessary to provide support for propagating changes to all dependent artefacts[2] by ensuring the consistency of the whole system. Otherwise, the reliability, accuracy and effectiveness of the e-government system decrease significantly.

Although the importance of change management is demonstrated in the practice (Hardless, et al. 2000), as known to the authors the corresponding methods and tools are still missing. However, since the demands for change-aware e-government are much higher (Stojanovic, et al. 2006), in this paper we propose an approach that enables agile response to frequent and huge changes in the environment or in the system itself.

The novelty of the approach lies in the formal verification of the service description as well as in the using formal

[1] OntoGov - http://www.ontogov.org

[2] For example, the e-government service for birthday certificate can be treated as a separate service or as a composite service in the context on other services such as passport issuance. The addition of a new input in the birthday certificate service requires the changes in the data-flow of many of services that include it (e.g. the passport issuance service) in order to achieve one-step e-government.

methods for achieving consistency when a problem is discovered. The verification is driven by a set of desirable properties including such as standard set of properties as well as domain-specific constraints (e.g. all activities are ground on some law or regulation). While performing the checks, the system generates specific suggestions on how to fix errors based on the type of errors and the situation at hand. Even though it is very desirable to identify errors and to resolve them in an early state of the service modelling, there are no tools that provide such means.

This paper is organized as follows: The set of ontologies used for describing e-government services is discussed in section 2. Change management process is introduced in section 3. The change preservation phase of this process that enables verification of a service description and generation of recommendations to fix problems founded in the description is elaborated in section 4. The implementation details are given in section 5. In section 6, an overview of related work is presented, while in section 7 we present the main conclusions of this work.

OntoGov Model

Before starting with the description of our approach for the change management, here we briefly describe the set of ontologies used for modelling e-government services. This set represents the Ontogov model. Dependencies between OntoGov ontologies are shown in Figure 1. They are called ***Meta ontologies***, since they define the schema i.e. the language for modelling the e-government services.

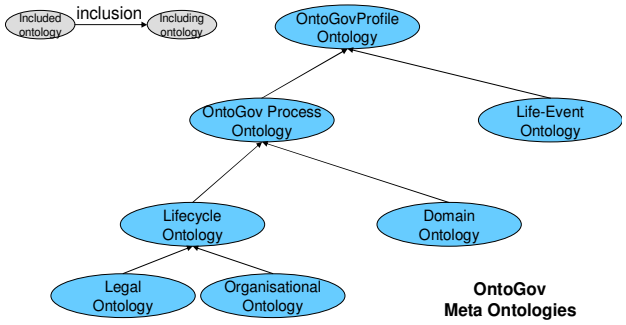


Figure 1. The OntoGov model

The OntoGov model consists of two major parts – the ***OntoGov Profile ontology*** and the ***OntoGov Process ontology***, which are developed based on the OWL-S[3] ontologies. However, both of them are extended / adapted in order to take into account unique characteristics of the e-government services as well as some aspects needed for the better management of changes.

For example, since a profile in generally is used for advertising and discovering of e-government services, a typical profile of an e-government service contains the information such as name, short description, version, status, date of creation, creator, etc. Additionally, the OntoGov Profile ontology includes the ***Life-Event ontology*** that is used for the classification of the e-government services. A part of this ontology is shown in Figure 2.

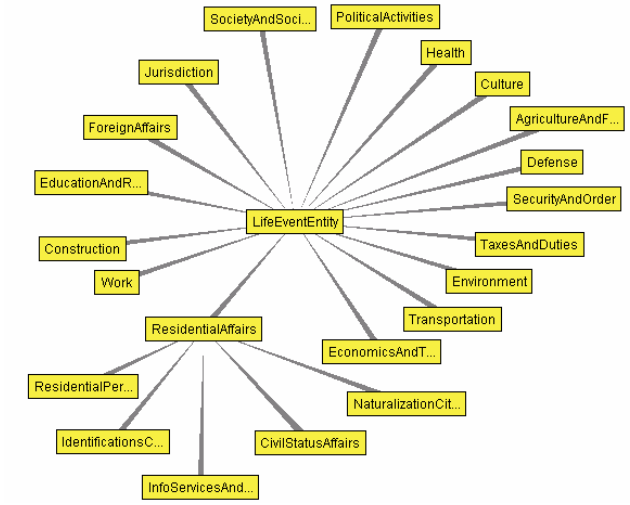


Figure 2. A part of the Life-Event ontology

The Life-Event ontology includes concepts such as residential affairs, residential permissions, identification certifications, naturalization citizenship, moving, education etc. It has been developed based on the existing standards for modelling lifeevents such as the Swiss Standard eCH-001[4] that aims to give an overview over all relevant e-government services in Switzerland and therefore to provide a consistent and standardized classification of the services. The inventory comprises 1.200 e-government services that are all services initialized by a citizen or internal administration processes. It is important to note that the Life-Event ontology is common for all the users even though they are often geographically distributed and experience significant problems in common communication language (e.g. English) and in the style of the communication. The lexical layer[5] of the Life-Event ontology enables to deal with different languages.

The OntoGov Process ontology models (i) process flow using activities (which can be either atomic or composite) and control constructs (e.g. sequence, split, join, switch, etc.) and (ii) data flow through inputs, outputs and equivalence relationship between them. Input and output of an activity are represented using entities defined in ***the Domain ontology***. The Domain ontology shown in Figure 3 encodes concepts of the public administration domain such as the "terminology" used in the e-government domain. Every public administration should keep its autonomy in describing its own domain. For example, the Domain ontology defines the type and structure of documents such as certificate.

Moreover, for each activity a set of metadata may be defined that includes name, description, preconditions, and

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

[4] Best Practice Structure Process Inventory - http://www.ech.ch

[5] The lexical layer of an ontology models various lexical properties of ontology entities, such as labels, synonyms, stems etc.

postconditions. This standard set of metadata is extended with the legal, organizational and lifecycle aspects defined in the corresponding ontologies. All these ontologies are used for the annotation of the e-government services in order to enable better and easier management of them.

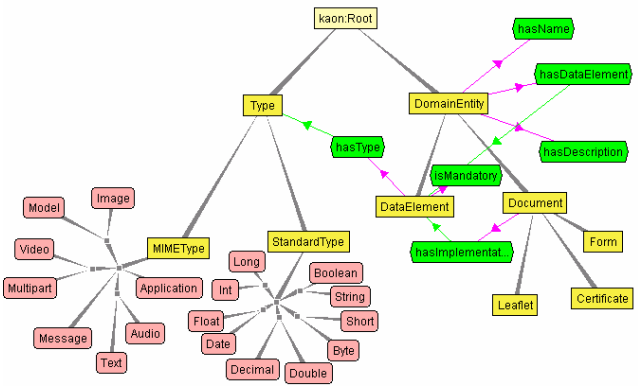


Figure 3. A part of the Domain ontology

For example, while in private organizations the decisions for process definitions are mainly based on time, cost and quality criteria, government processes must be in accordance with the existing law and regulations from different levels (state, region, and municipality). Therefore, we have developed the ***Legal ontology*** that models the structure of the legal documents, which includes paragraphs, sections, amendments, etc. It is very important to document the laws and regulations the process is based upon – not only for the whole process but also for specific activities, since the legislation regulates the accomplishments of the administrative services. By associating legislation to these services, it is possible to trace and propagate the effects that a change in the legislation (or administrative regulations) produces on the models of the administrative services.

To develop the Legal ontology that is shown in Figure 4 we have analyzed the structure of legal documents in Switzerland, Greece and Spain, since the goal of the OntoGov project is to pilot the system at three partners coming from these countries. We concluded that the legal documents have very similar structure independently of the country they are defined for. Even though different countries use different terminology to organize their legal documents, all of them use three levels of abstractions. Therefore, it was possible to extract the general structure of a law and to represent it in a form of the Legal ontology.

The ***Organisational ontology*** shown in Figure 5 describes the roles and areas of responsibility and capabilities within an organisation with respect to the activities of a process model. Moreover, it models the structure of an organisation, its resources, know-how, etc. For example, we distinguish two types of resources: (1) human resources who perform an activity and (2) equipment (i.e. hardware, software etc.) that is occupied by the activity. Note that equipment is needed to perform an activity. However, it is released after finishing this activity.

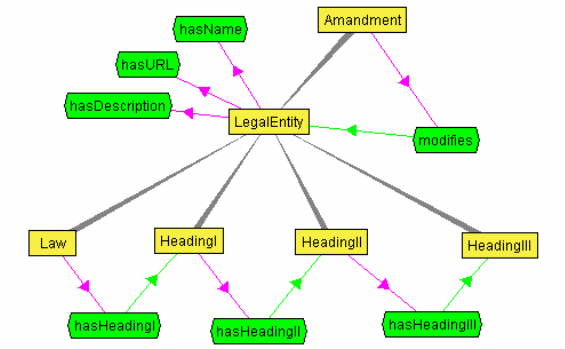


Figure 4. A part of the Legal ontology

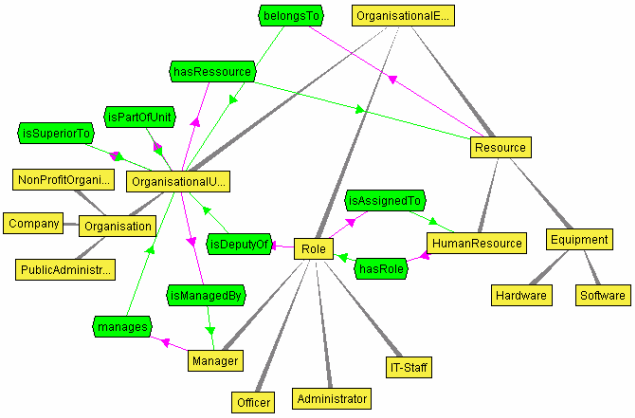


Figure 5. A part of the Organisational ontology

Finally, the OntoGov Process ontology includes the ***Lifecycle ontology*** that describes the decision-making process in the public administration. A part of this ontology is shown in Figure 6. It bridges the gap between decision making and realisation by providing means for describing these decisions and formally stating reasons that motivate the design decisions. Indeed, it is intended to support the transition from knowledge acquisition to implementation. It provides answers on the following questions: (i) "How have the process design (e.g. regarding atomic activities) and flow (e.g. regarding control constructs) been realized?" and (ii) "Why has a design decision been taken?".

Since it includes entities for documenting design decisions and the underlying rationale, it gives concrete clues on how the corresponding e-government service has to be modified. During ongoing development, it helps the public administrators 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 they were derived from and to the requirements that influenced design decisions. In this way we build model that supports not only the specification and design of e-government processes, but more important it provides an automated, transparent, and user centered

support to the entire process lifecycle, from analysis to execution, by suggesting solutions that can be adopted, refused, or refined by public administrators.

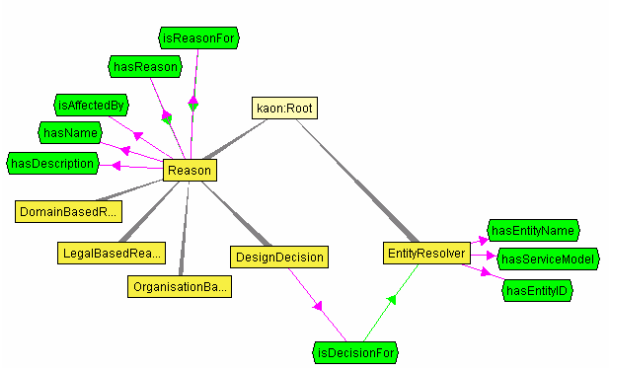


Figure 6. A part of the LifeCycle ontology

All the previously mentioned ontologies represent the OntoGov model. In order to model concrete e-government services and all data relevant for these services we have developed the ***Domain-oriented ontologies***. Therefore, this cluster includes a set of ontologies that are structured in accordance with the specific domain, e.g., pilot. These ontologies are "***specialization***"[6] of ontologies belonging to the cluster of Meta Ontologies or other domain-oriented ontologies.

For example, at the government level we may define the ***Legal-Federal ontology*** based on the Legal ontology that belongs to the OntoGov model. It contains the entities representing the laws that hold at federal level. Each federal state has its own laws. Therefore, the ***Legal-State ontology*** may be a specialization of the Legal-Federal ontology (since a state must satisfy all federal laws) by extending it with the knowledge related to the federal state laws. Further, each municipality may create its ***Legal-Municipality ontology*** that extends the Legal-State ontology with some regulations. This example is shown in Figure 7. We note that there is no constraints regarding the depth of the specialization. However, our approach is currently limited to including entire models rather than including subsets. Also, when a model is reused, information can only be added, and not retracted.

The main ontology in the cluster of domain-oriented ontologies is the so-called ***Service ontology***. Each e-government service is represented by one Service ontology. A Service ontology is an instantiation of the OntoGov Profile Ontology and it contains specializations of all ontologies included in the OntoGov Profile Ontology.

We note that the "specializations" of the Legal, Organizational and Domain ontologies may be shared between several Service ontologies. It means that for each particular governmental institution the domain experts have to define their own specialization of the Legal, Organizational and Domain ontologies by taking into account the specificities of this public administration. Moreover, they have to reuse as much as possible of already existing knowledge. For example, to define their own ontology for legal aspects they should reuse the ontology representing the law at the state level (or at least at the federal level) and not to start directly from the Legal ontology. This will speed up the ontology development process and will increase the interoperability.

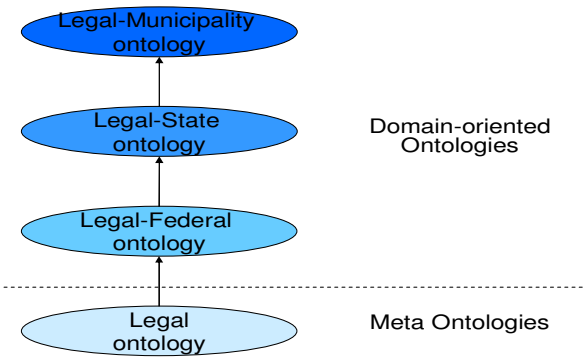


Figure 7. Specialisation of the legal ontologies

On the other hand, the specializations of the Lifecycle, Process and Profile ontologies are specific for each concrete service. This specialization is done by creating instances and property instances of corresponding concepts defined in some of the included ontologies. It means that for each e-government service we have exactly one specialization of these three ontologies, since each e-government service has its own profile, process model as well as life-cycle. We note that a Service ontology may include other Service ontologies.

An example of modelling e-government services

An example of the process part of the e-government service "Announcement of moving", modelled by using the OntoGov model, is shown in Figure 8. This service is classified as of high potential for European e-government improvement, as it is typically involving various public and private institutions. Today, the service is split into few separated tasks. In the case that a citizen invokes this service from a web portal, she is asked to provide all information needed to perform the complete service (cf. "*EnterApplicationForm*" in Figure 8). After submitting the requested information, the eligibility is checked (cf. "*CheckEligibility*"). Based on the result, the service can be either broken (cf. "*RejectApplication*") or continued. The next step depends on whether she is already registered (cf. "*Deregistration*") or not (cf. "*Registration*"). Deregistration has to be performed in one municipality. In addition, the person has to register himself/herself in the new municipality. In the meantime several private or semi-private entities, like telecommunication companies or the electricity company, have to be notified about the change

[6] An ontology ***OS*** is defined as a ***specialization*** of an ontology ***O*** if it includes the ontology ***O*** and extends its entities either at the conceptual level (e.g. by defining the specialization of a concept) or at the instance level (e.g. by instantiating a particular concept). We treat the term ontology specialization as a synonym for term ontology reuse or ontology modularization.

of the address (cf. "*GetThirdPatiesAddress*" and "*NotifyThirdParties*"). Finally, the citizen has to be informed about the result of the service. By describing the e-government service "Announcement of moving" in this way the quality of the service is improved, since it is performed by the citizen as *one* task regardless what and how much (technical) processes run behind.

Moreover, not only knowledge on *how* to execute the service is stored but also *why* it was designed as it is. Therefore for every entity in the process model of a service (i.e. either an activity or a control construct), information on the underlying design decisions is stored. An example of the design decision that is defined for the activity "*CheckEligibility*" is shown in Figure 8. This decision is legally grounded: the information that public administrators need to know regarding this activity is defined by law (cf. *SR 101* and *SR 201 Art. 22A-26A*).

Additionally, a decision may stem from technical reasons or organizational reasons. In the case that a reason changed, this information is used to propagate the change to the affected service(s).

Change Management Process

Change management is the timely adaptation of a system to the changes in business requirements, users' needs, etc. as well as the consistent propagation of these changes to dependent artefacts (Nickols, 2003). A modification in one part of the system may generate many inconsistencies in other parts of the same system (Stojanovic, 2004). This variety of causes and consequences of the changes makes the change management a very complex operation that should be considered as both an organizational and a technical process (Stojanovic, Stojanovic, 2004).

Existing approaches for change management in e-government focus mainly on manual managing of a particular, isolated service and on supporting only message-based communication between public administrators. It means that public administrators can exchange raw information, but not semantically more complex structures, like decisions, since e.g. they are missing a commonly agreed description of problems.

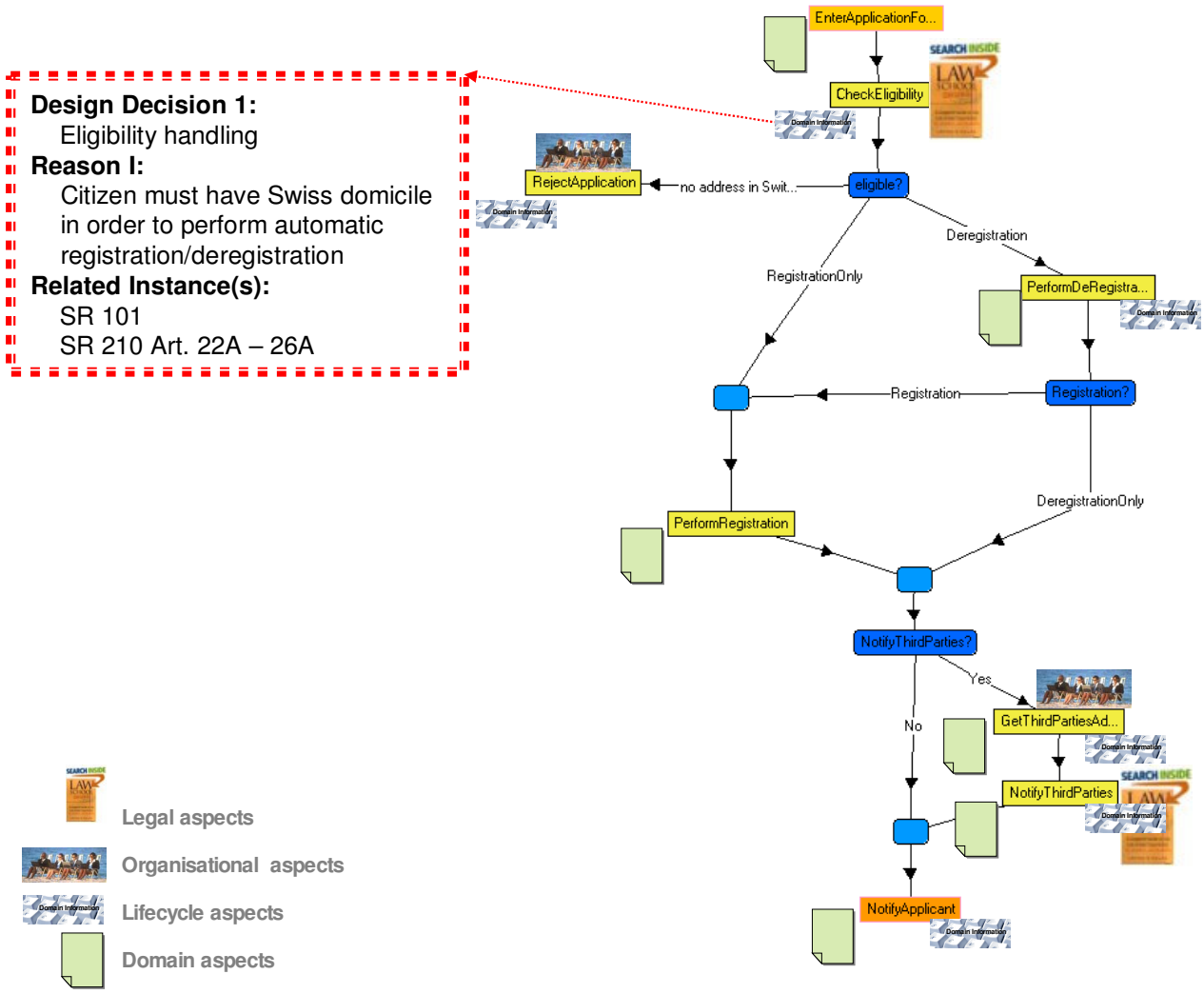


Figure 8. "Announcement of moving" e-government service modelled using the OntoGov model

Moreover, the existing approaches require a growing number of highly skilled personnel, making the maintenance costly. Finally, the changes that affect the system are resolved and propagated in an ad-hoc manner.
However, the ad hoc management of changes might work only for particular cases. It can scale neither in space nor in time. Therefore, in order to avoid unnecessary complexity and failures in the long run, change management must be treated in a more systematic way.
Our approach for Change Management enables consistent propagation of changes within a service and between the services in order to ensure the quality of the decision making process. Since the services are represented as a set of metadata (instances) related to an ontology, each step in resolving changes can be formalized and automatically performed. We define a four-phase change management process as shown in Figure 9. The process starts with representing a request for a change formally and explicitly. Then, the change preservation prevents inconsistencies by computing additional changes that guarantee the transition of the model into another consistent state. During the change implementation phase, required and derived changes are applied to the system in a transactional manner. In the change propagation phase all dependent knowledge items are found and updated.

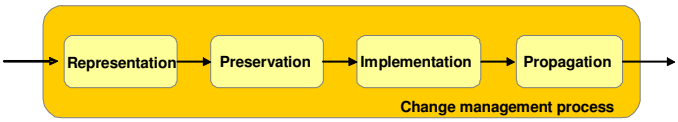


Figure 9. Four phases of the change management process

In the rest of this section we describe this four phases briefly. The detailed description of the change preservation phase is given in next section.

Change representation

The OWL ontology language represents the standard for representing ontologies on the Web and it is used as the representation language of OntoGov ontologies. Each entity of the OWL ontology model represents one logical axiom. Therefore, a complete set of changes determined by the OWL ontology language includes only two changes: "*AddAxiom*" and "*RemoveAxiom*". However, this granularity of the ontology changes is not always appropriate. For example, to make the service s1 a predecessor of the service s2, the public administrator needs to apply a *list* of ontology changes that includes creating a sequence between s1 and s2 and connecting outputs of s1 to the corresponding inputs of s2.
Therefore, public administrators require a method for expressing their needs for changes 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. The full set of changes for the OntoGov model is defined in (Stojanovic, at al., 2006). These changes correspond to the "conceptual" operation that someone wants to apply without understanding the details (i.e. a set of ontology changes) that the change management system has to perform.

Change preservation

Changes are forces that drive the evolution. They can be applied to a consistent description of an e-government service, and after all the changes are performed, the description must pass into another consistent state.
Therefore, when updating a service description, it is not enough just to consider the entities figuring in the request for a change, because the other entities in the same description may also be affected by the updates. Since it is not sufficient to change only a part of the description that is related to the request for a change while keeping all the other entities intact, we introduce the ***change preservation phase***. Its task is to enable the resolution of changes in a systematic manner by ensuring the consistency (Haase, Stojanovic, 2005) of the whole description of an e-government service. This is elaborated in next section.

Change implementation

The role of the change implementation phase is (i) to inform a public administrator about all consequences of a change request, (ii) to apply all the (required and derived) changes and (iii) to keep track of performed changes.

Notification

In order to avoid performing undesired changes, before applying a change to a service description, a list of all implications to the service description should be generated and presented to a public administrator who modifies the service description. Only if the public administrator is informed about all the changes that are going to be performed on a request, she can make strategy decisions posed by the system. The public administrator should however have possibilities to make such choices or even to abort the entire modification when she realizes that it would have undesired consequences for other parts of the service description. Consequently, she should be able to comprehend a list of all the changes and approve or cancel them. When the changes are approved, they are performed by successively resolving changes from the list. If changes are cancelled, the service description remains intact.

Change application

In order to give a public administrator a chance to cancel a change after it has been completely analyzed, it is necessary to separate the analysis[7] of the user's request for the change from the final execution of this request within the change management system. Therefore, the main task of *the* ***change implementation phase*** is the application of changes. During this phase all changes (i.e. required and derived changes) are applied to a consistent service

[7] The analysis of a change covers the change preservation and the change propagation phases, where the change is extended with the additional changes that ensure the consistency.

description and result into a new consistent service description.

Indeed, one of the main advantages of our change management process is the separation of the phases where requests are analyzed from the final execution of the changes. This separation was naturally driven by the need for the transaction[8]. Only after a successful commitment of the hypothetical "reasoning" performed by the change preservation, the changes in effect took place on the service description itself. Once acknowledged by the public administrator for the implementation, all the changes are considered as an atomic "transaction" (i.e. they act like a transaction), although they are executed step by step.

Change Logging

The next task of the change implementation is to keep track about the performed changes. Information about changes can be represented in many different ways. To communicate about changes, we need a common understanding of a change model and of a log model. Therefore, we introduce the service evolution ontology and the service evolution log (Stojanovic, 2004). The service evolution ontology is a model of ontology changes enabling better management of these changes. The service evolution log tracks the history of applied changes as an ordered sequence of information (defined through the service evolution ontology) about a particular change.

Moreover, the *change reversibility* is also supported. It enables undoing and redoing changes made in an ontology-based description of the e-government service. Consequently, changes can be executed in reverse order thus forcing the service description to return to the conditions prior to the change execution. It is important to note that reversibility means undoing all effects of some change, which may not be the same as simply requesting an inverse change manually. For example, if an atomic service is deleted from a process model, its metadata and links will need to be deleted as well. Reversing such a change is not equal to recreating the deleted atomic service – one needs, also, to revert the metadata and the links into the original state.

Change propagation

The basic requirement for a management system is that it has to be simple, correct and usable for public administrators. Note that they are responsible for keeping semantic description of 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 semantic description of e-government services and ranking them according to the importance. When such problems arise, a management system must assist the public administrators 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.

The role of a change management system is much more than finding inconsistencies in a description and alerting a domain expert about them. This is pretty much the kind of support provided by conventional compilers. However, helping public administrators notice the inconsistencies only partially addresses this issue. Ideally, a change management system should be able to support domain experts in resolving the problems at least by making suggestions how to do that. The following procedure has been realized:

- ***Checking actuality of the associated Ontologies*** – Since each ontology has a version number associated with it that is incremented each time the ontology is changed, checking the equivalence of the original of the included ontology and the replica can be done by a simple comparison of the version numbers.
- ***Extracting Deltas*** – After determining that the included 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 synchronization with the replica, as determined by the version numbers.
- ***Analysis of changes*** – Each performed change is analysed, in order to find e-government services that have to be updated. We distinguish between the addition and the deletion of an entity from the included ontology. Removals can be resolved directly by applying the consistency preservation mechanism (see next section), since it ensures the consistency by generating additional changes. For example, the removal of a role from the Organisation ontology causes the removal of all annotations of activities made using this role or all instantiations of this role.

On the other hand, the addition requires an additional effort that depends on the structure of the included ontologies. 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 activity is referred to a law/chapter/paragraph/article, the corresponding activities 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.

- ***Making recommendation***: In order to make recommendations how to adapt the service description we use the Lifecycle ontology. Since it 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.

Let's consider an example. A change in the Organizational ontology could be the split of the organizational unit into two sub-units. This "organizational reason" might cause the design decision "Executing an activity in two steps", i.e.

[8] A transaction represents a sequence of actions that is treated as a unit for the purposes of satisfying a request. For a transaction to be completed, it has to be accomplished in its entirety.

two (atomic) activities. For example, the decision to split the activity "CheckEligibility" shown in Figure 8 into two activities can be caused by the fact that two different public authorities are responsible for this action: the residents' registration verifies personal information and the "immigration office" verifies the validity of the visa, in case the citizen is foreigner.

Change preservation

In this section, we present a novel approach to the consistency preservation that supports the public administrators in managing and optimizing the service descriptions according to their needs. The underlying system is able to find the "weak places" in the description of the e-government services (e.g. unreachable entities, non-expected data, etc.) by considering the semantics of the underlying OntoGov model.

The proposed approach incorporates mechanisms for verifying the service description with respect to different consistency criteria as well as mechanisms enabling us to take actions to optimize it. It has been realized through two sub-tasks:

- ***Verification:*** It is responsible for checking the consistency of a service description. Its goal is to find "parts" in the description that do not meet consistency conditions;
- ***Evolution:*** It is responsible for ensuring the consistency of the service description by generating additional changes that resolve detected inconsistencies.

In the rest of this section we describe our approach for inconsistency detection. Thereafter, we present our approach for "moving" the inconsistent ontology back into a consistent state, i.e. change generation.

Verification

In this section we explore the verification of the OntoGov model, which means checking of the correctness of the service description with the respect to the service consistency definition. Moreover, it provides enough information to analyze the sources of conflicts. Its role will be to inform a public administrator about the necessity for updating the description of an e-government service, and to allow the application of the service changes, enabling an easy spotting of potential problems.

The description of the e-government services (or more generally the description of the semantic web services) can be arbitrary complex, containing multiple concurrent threads that may interact in unexpected way (Ankolekar, et al., 2005). We propose an approach that is able to verify numerous properties. The set of properties is not predefined, which means that it does not include only the standard properties such as safety, liveness, etc. (Naumovich, G., Clarke, L., 2000), but more important it can be easily extended by the application specific properties.

Verification of description of e-government services is realized using formal methods. These methods seek to establish a logical proof that a system works correctly. A formal approach provides:

- a modeling language to describe the system;
- a specification language to describe the correctness requirements; and
- an analysis technique to verify that the system meets its specification.

The model describes the possible behaviors of the system, and the specification describes the desired behaviors of the system. The statement the model P satisfies the specification α is now a logical statement, to be proved or disproved using the analysis technique.

Since the goal of the inconsistency detection is to check whether a service description satisfies the required specification, it can be treated as a formal verification problem in which: (1) a modelling language used to describe a system is defined through the OntoGov model, (2) a specification language corresponds to the consistency constraints that must be preserved and an analysis technique can be treated as inference process. Whereas the model of the e-government services is described in section OntoGov Model, the consistency is defined in (Stojanovic, et al., 2006). In the rest of this section we focus on (3).

Formal verification methods can be roughly classified as:

- Proof-theoretic: a suitable deductive system is used, and correctness proofs are built using a theorem prover, and
- Model-theoretic: a model of the run-time behaviour of the system is built, and this model is checked for the required properties.

In this section we explore the verification of the OntoGov model using proof-theoretic method. Once we have a service description plus the formally defined consistency constraints that correspond to the users' requirements, we can automatically check whether these constrains are satisfied in the service description with the help of the reasoning. The KAON2[9] inference engine is used, since it implements the proof-theory for DL and DL-safe rules. By performing an efficient exploration of the possible inconsistencies that can be built in the service description, the system is able to verify all the consistency constraints[10] defined for the OntoGov model. One example of these constraints is given in Figure 10.

The set of the consistency constraints as well as a description of the concrete service are inputs to the KAON2 inference engine that is used to automatically verify whether the service description satisfies the consistency. Practically, a query is sent, since possible problems are hierarchically organized. A trace of the answer to a query is considered as a model that reflects how different pieces of a service description are put together to generate the answer. If the KAON2 verifies that the consistency constrains are fulfilled (i.e. there is no

[9] http://kaon2.semanticweb.org

[10] Since in this work we use the KAON2 inference engine, the consistency constraints must be specified as DL-safe rules.

answer), then the service description is consistent. Otherwise, the KAON2 provides explanation about causes of problems, since it can identify the conditions under which the problem occurs.

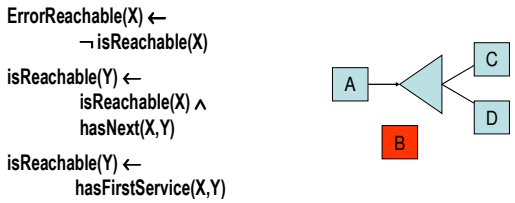


Figure 10. Verification based on the constraints. A part of consistency rules is depicted in the left part. The right part shows the process model that does not satisfies the rules

The realised verification procedure is depicted in Figure 11. The set of the consistency constraints selected/defined[11] by the public administrator are transformed into a set of DL-safe rules and these rules are included in the temporary version of the OntoGov Profile and OntoGov Process ontology, respectively. Since the description of a concrete service includes both of these ontologies, it will include the rules to be checked. The service description is given to the KAON2 reasoner and the query "about all possible errors" is initiated. The result produced by KAON2 reasoner is then presented to the public administrator in the form that he/she can understand. Even though logic provides an unambiguous formal specification, it is hard to imagine that a public administrator will comprehend it. Therefore, "wrapping" into a more friendly formalisms, i.e. natural language explanation[12] has been proposed. It means that in the case of any violations of consistency constraints, the reasoner will output a counterexample, which demonstrates the courses of wrong behaviour. An analysis of this counterexample provides information that helps to correct and refine the service description.

For example, a precondition of an activity is not achieved because there are some previous activities that undo the precondition. Let's consider the driving licence service for foreigners in Germany. The preconditions of the *Application* activity includes that foreigners come from non-EU countries. Since the special verification is required for the countries emerged from the break-up of Yugoslavia, there is an activity in the process model that has a precondition that the foreigners must be from Slovenia. However, the *Application* activity undoes this precondition, since Slovenia is a member of EU. It is very difficult for a user to notice that some of the paths in the model are not be possible due to at least two reasons: (i) this service description is very complicated with many disjunctive branches, and (ii) the background knowledge (i.e. the fact that Slovenia is in EU) is needed. Our system is able to detect this problem by applying reasoning methods (i.e. there is a corresponding consistency constraint[13]) and to help the user fix problem. It can find activities in the process model that should be executed before the failed activity that have effects that undid the unachieved preconditions. Moreover, it suggests modifying the activity whose precondition can never be achieved. For the above mentioned type of failure, our system suggests (i) changing or adding constraints for the *Application* activity and (ii) deleting or modifying the *Verification* activity.

Moreover, the system is also able to propose changing ordering constraints among the activities. For example, the user may either forget to specify connections between the activities or may specify wrong connections. These problems may be detected by checking a particular consistency constraint[14] that defines the certain ordering constraints already specified for the type of these activities. During the verification, the system checks (among others) the dependencies between activities using the ordering consistency constraint. In the case that some activity does not satisfy the ordering constraint, the system produces the error message containing the fixes such as adding or modifying dependency between activities.

We note that the same problem can be a consequence of different inconsistencies in the model, since one abnormality can lead to another. For example, missing the first activity in a model causes unreachable activities. To help avoid confusion, our system can selectively present suggestions for improvement by focusing the user on the actual cause of a problem. For the previous example, the system suggests starting with the resolution of the first activity problem. However, the user can check other problems as well, in the case that he/she what to do that.

For the description of e-government services the proposed solution seems to be an ideal technique, since only consistency constraints defined by the public administrators need to be considered. The probability of running into the undecidable solution is quite low, since the restriction to the DL-subset of SWRL rules has been chosen to make reasoning decidable. Moreover, reasoning in KAON2 is implemented by novel algorithms that allow applying well-known deductive database techniques, such as magic sets, to DL reasoning. According to the performance evaluation (Motik, Sattler, 2005), such algorithms make answering queries in KAON2 one or more orders of magnitude faster than in existing systems.

[11] A user can select consistency criteria from the list of available consistency constraints and/or can define a new consistency criterion.

[12] We do not use logical notations since public administrators do not have logic background knowledge. For each possible problem, an explanation in natural language is generated.

[13] If an activity precedes another activity, then its preconditions have to subsume the preconditions of the next one.

[14] Any specialisation of the activity A1 must always be a predecessor of any specialisation of the activity A2, where A1 and A2 are two activities defined in the OntoGov model and their order is given in advance.

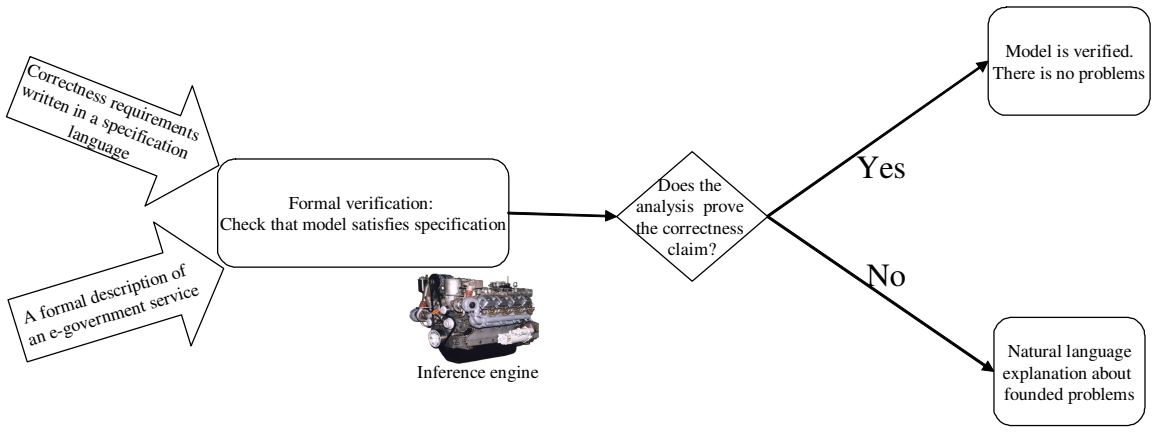


Figure 11. Formal verification: Based on the possible behaviours (i.e. a ontology-based service description) and on the desirable behaviours (i.e. formally defined consistency constraints) the system constructs a proof that either proves or disproves the correctness claim

In this work we applied the formal techniques based on the sound and complete set of consistency rules (provided with an inference mechanism) to verify the models. The informal approaches, such as the procedural approach, whose semantics is given by a procedural mechanism that is capable of providing answer to wide class of consistency problems, can be applied as well. However, the extensibility of the procedural approach is time-consuming and error-prone, since knowledge about consistency is represented by means of an ad hoc structure. Alternative approaches cover testing, which executes the actual model on selected inputs or simulation, which simulates a model on selected inputs (not exhaustive), cannot be applied, since they perform the so-called dynamic checks. On the contrary, our approach performs static checks by posting questions about various features of the service description.

Evolution

Changes are applied to a consistent service description, and after all the changes are performed, the description must remain consistent. This is done by finding inconsistencies in the description and completing required changes with additional changes, which guarantee the transfer of the initial consistent description into another consistent state. Indeed, the updated service description is not defined directly by applying a requested change. Instead, it is indirectly characterized as a service description that satisfies the user's requirement for a change and it is at the same time a consistent e-government service description.

Therefore, there are two major issues involved in the change generation. The first issue is the understanding how an ontology-based service description can be changed since the change management is realized by means of applying changes. To resolve the first issue a possible set of changes is defined in (Stojanovic, at al., 2006). The second issue involves deciding when and how to modify a service description to keep its consistency, which is elaborated in (Stojanovic, 2006). It is based on the formal approach for suggesting fixes that directly point to the source of the errors. Indeed, each possible change is formally modelled. The most critical part of a definition change is rules that specify the side effects of a change on the other related entities. To define the rules for each change, we started by finding out the cause-effect relationship between the changes. This kind of dependency between the changes forms the so-called ***change dependency graph***.

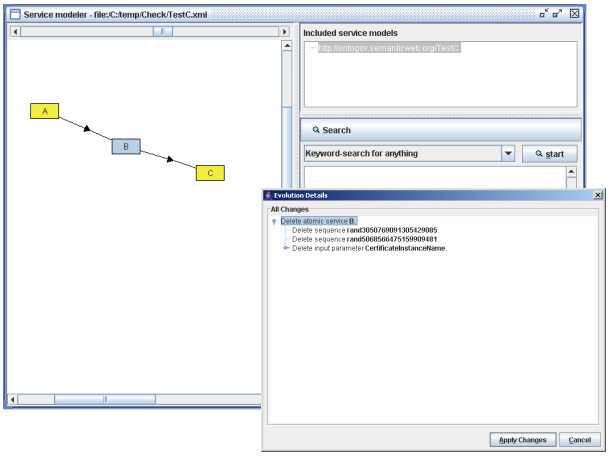
A sample screenshot of the OntoGov change management system illustrating triggered actions (i.e. generated changes) for the removal of the atomic activity is given in Figure 12. In this scenario, the user requested to remove the atomic activity *B*. According to the change dependency graph, this change may cause:

- Remove all input links[15] of AtomicActivity B;
- Remove all output links of AtomicActivity B;
- Remove all metadata defined for AtomicActivity B that includes:
  - the attributes such as name, description, fist and last service;
  - the relations to the associated ontologies (i.e. Legal, Organizational and Lifecycle ontology);
  - the relations to the inputs and outputs defined through the Domain ontology;
  - the pre- and post-conditions.

Before changes are performed, their impact is reported to the user (the right part of Figure 12). Presentation of changes follows the progressive disclosure principle: related changes are grouped together and organized in a tree-like form. The user initially sees only the general description of changes (cf. "*Delete atomic service B*" in Figure 12). By opening a node in the tree, the user can see what changes will actually be performed (cf. "*Delete input parameter CertificateInstanceName*" in Figure 12). Hence, the change information can be viewed at different levels of granularity. If the user is interested in details, she can

[15] A link can be a sequence or a relation to the split, join or switch control construct.

expand the tree and view complete information. She may cancel the operation before it is actually performed.



a) The initial model; b) the generated changes

Figure 12. Change generation for the request RemoveAtomicActivity(B)

Implementation

OMS is the ontology management system that has been developed within the OntoGov project. It is a management system for the ontology-based description of the e-government services. It is much more than a standard framework for creating, modifying, querying, and storing ontology-based description of e-government services. It provides support for the service lifecycle management, which includes service modelling; service reconfiguration, service reuse, service discovery and service analysis.

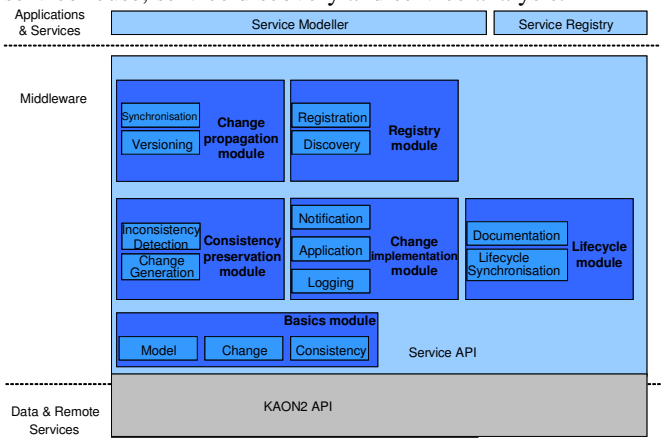


Figure 13. Conceptual architecture of the OMS

The simplified conceptual architecture of the OMS system is presented in Figure 13. Roughly, the OMS components can be divided into three layers:

- *Applications and Services Layer* realizes UI applications and provides interfaces to non-human agents. It includes: (i) *Service Modeller* – it is an editor for the semantic description of the e-government services[16]; and (ii) *Service Registry* – it is a registry of the e-government services.
- The Service API as part of the *Middleware Layer* is the focal point of the OMS architecture. The bulk of requirements related to the management of e-government service description is realized in this layer;
- *Data and Remote Services Layer* provides data storage facilities. It is based on KAON2 API, which is an API for OWL ontologies.

The middleware layer of the OMS shown in Figure 13 emphasizes points of interest related to the change management. The main modules are (i) *basics module*; (ii) *consistency preservation module*; (iii) *change implementation module*; (iv) *change propagation module*; (v) *lifecycle module* and (vi) *registry module*. The functionality as well as the implementation of these modules is described in (Stojanovic, Apostolou, 2005). Our initial evaluation shows that the OMS is able to find all inconsistencies in the service description and to suggest useful fixes including the fixes that directly point to the source of the inconsistencies.

Related work

OWL-S process models are typically verified using human inspection, simulation and testing. In this paper we proposed the formal verification, which as opposed to traditional techniques such as testing and simulation has two main advantages (i) formality - the intuitive correctness claim is made formally; and (ii) verification - the goal of the analysis is to prove or disprove the correctness claim. It is not adequate to check a representative sample of possible behaviours as in simulation; rather a guarantee that all behaviours satisfy the specification is required.

The verification of the OWL-S process model is described in (Narayanan, McIlraith, 2002) and (Ankolekar, et al., 2005). Whereas the first paper proposes a Petri Net-based operational semantics, which models the control-flow of a process model, the second paper additionally models the data flow and applies the SPIN model-checker as a the automatic verification tool. We extend these works in several dimensions. First, we model not only the control-flow and data flow consistency constraints. We allow to the public administrators to specify arbitrary domain-dependent consistency constraints. In this way we are able to cover all perspectives of the business models, i.e. control flow, data flow, operational issues (e.g. interactions between systems) and resources (e.g. humans, machines etc.). Second, we do not consider only the process model but also the profile of a service. Finally, we have realized

[16] *OIModeller* is used as an editor for the "standard" ontologies. It is a graphical tool for ontology creation and maintenance. Since it is based on the different ontology model, we have realized a translator of the KAON ontologies (http://kaon.semanticweb.org/) into the KAON2 ontologies (http://kaon2.semanticweb.org/). We note that each KAON ontology can be transformed into a KAON2 ontology without loss of information.

the verification of the e-government service descriptions using rule-based inference process.

Many AI researchers have investigated useful ways of verifying and validating knowledge bases for ontologies and rules. However, it is not easy to directly apply them to checking process models. In (Kim, Gil, 2001)] the authors discussed the KANAL system that relates pieces of information in process models among themselves and to the existing knowledge base, analyzing how different pieces of input are put together to achieve some effect. It builds interdependency models from this analysis and uses them to find errors and propose fixes. However, it does not allow the user to specify their specific conditions, event though the predefined set of constraints doesn't cover all the users' needs. Our approach allows the user to define the user-defined conditions. Moreover, it separates the specification of consistency from the realization of the change preservation procedure. Finally, the inconsistency detection and the change generation procedures are governed by well-defined formal models that are fully automated. Therefore, the approach is accessible by public administrators who are not experts in formal methods.

There are many graphical tools (ARIS, Adonis, to name just a few) that lay out a process model and draw connections among steps. These tools lack formal methods for verifying properties of processes. Indeed, they tools are limited to simple checks on process models, since there is no semantics associated to the individual steps. In contrast, we propose an approach that allows to the users to formally specify consistency constraints. Ontologies and rules are used to represent this kind of background knowledge or user's needs. With this context, our system is much more helpful in checking the process model. Moreover, our system can check the service profile as well and it proposes suggestions for resolving the problems.

Conclusion

e-government systems are subject to a continual change. The importance of better change management is nowadays more important due to the evolution of Europe towards a multicultural, more open and international society with changing common values, increasing levels of education, demographic involvement and adoption of new technologies. It is especially true for the new EU countries, since the European integration has paved the way for new legislation, regulations and corresponding changes that affect the way public administrations in the enlarge Europe are organized and operate.

It is clear that ad hoc management of changes in eGovernment might work only for particular cases. In order to avoid drawbacks in the long run, the change management must be treated in a more systematic way. In this paper we presented an approach for ontology-based change management. Our approach goes beyond a standard change management process; rather it is a continual improvement process. The novelty of the approach lies in the formal verification of the service description as well as in the using of formal methods for achieving consistency when a problem is discovered.

Acknowledgement

The research presented in this paper was partially funded by the EC in the project "IST PROJECT 507237 - OntoGov".

References

Ankolekar, A., et al., 2005, Towards a Formal Verification of OWL-S Process Model*s*, in Proc. of the ISWC 2005, Galway, Ireland, LNCS 3729, pp. 37-51

Hardless, C., et al., 2000, The evolution of knowledge management system need to be managed, Journal of Knowledge Management Practice, Volume 3

Haase, P., Stojanovic, L., 2005, Consistent Evolution of OWL Ontologies, in Proc. of ESWC 2005, LNCS 3532, Greece, pp. 182-197

Kim, J., Gil, Y., 2001, Knowledge Analysis on Process Models, in Proc. of the IJCAI 2001, USA, pp. 935-942

Motik, B., Sattler, U., 2005, Practical DL Reasoning over Large ABoxes with KAON2, http://www.fzi.de/KCMS/kcms_file.php?action=link&id=580

Narayanan, S., McIlraith, S., 2002, Simulation, Verification and automated composition of web services, In Proc. of the 11th WWW Conference, USA, pp. 77 – 88

Naumovich, G., Clarke, L., 2000, Classifying properties, an alternative to the safety-liveness classification, ACM SIGSOFT Software Engineering Notes, Vol. 25, Num. 6, pp. 159-168

Nickols, F., 2003, Change management 101: A primer, http://home.att.net/~nickols/change.htm

Stojanovic, L., et al., 2006, Change Management in e-government: OntoGov Case Study, in Electronic Government: International Journal, Special Issue on Exploiting Knowledge Management for Ubiquitous E-Government in the Semantic Web Era, Vol.3, No 1

Stojanovic, L., 2006, Ontology-based Change Management: e-government case study, to appear in a book about Semantic Web Services

Stojanovic, L., Apostolou, D., 2005, Ontology-based Change Management of e-government services, in Proc. of WI2005 Conference - "Semantics and Orchestration of eGovernment Processes" Workshop, France

Stojanovic, L., 2004., Methods and Tools for Ontology Evolution, PhD thesis, University of Karlsruhe

Stojanovic, N., Stojanovic, L., 2005, A Change-Aware Framework for the Knowledge Management in eGovernment, in K. V. Andersen, Å Grönlund, R. Traunmüller, M. Wimmer (Eds.): Electronic Government - Workshop and Poster Proceedings of the Fourth International EGOV Conference 2005, Denmark, ISBN 3-85487-830-3, pp. 3-10