An Approach for the Change Management in the E-Government Domain

L. Stojanovic[1], A. Abecker[1], N. Stojanovic[2], R. Studer[1,2]
[1] FZI - Research Center for Information Technologies at the University of Karlsruhe,
Haid-und-Neu-Str. 10-14, D-76131 Karlsruhe, Germany
{Ljiljana.Stojanovic, Andreas.Abecker}@fzi.de
[2] Institute AIFB, University of Karlsruhe,
76128 Karlsruhe, Germany
{nst, 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, such as service re-configuration due to changes in the law, 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 description of semantic web services. Even though we use the E-Government domain as an example, the approach is general enough to be applied in other domains.

Key words: Ontologies, E-Government, Change Management

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 [6]. 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, WSDL and their composition on the level of business processes (BPEL4WS) 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 in the E-Government domain.

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 [9]. 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, we realized our approach for the change management in the E-Government domain. However, it 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 a short example from E-Government domain explaining the requirements for a management system. This system is described in section 3. The problem is reduced to the evolution of the Meta Ontology that is used for the service description. We define the set of changes and consistency constraints that this ontology introduces. Before we conclude, we present an overview of related work.

2 Motivating Example

In order to make the description of the approach more understandable, we define here the basic structure of an E-Government system and give a motivating example that will be used through the paper. 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.

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. For example, the generic schema for the public service for issuing (renewal) a driving licence is realized through the following five activities: (i) application, (ii) verification/qualification, (iii) credential issuance, (iv) record management and (v) revenue collection.

In the application activity, all the necessary application data/documents are provided by an applicant. In the next activity, the provided information/documents are verified (e.g. validity and liquidity of a credit card) and are qualified by testing whether the applicant meets the qualification requirements. In the issuance activity either a permanent or temporary credential document (i.e. driving licence) is issued. The record management activity ensures the ongoing integrity of the driving licensing and control record. Finally, the required fee is charged from the applicant's bank account. Each activity requires some inputs, produces some outputs. It can be executed only when its preconditions are fulfilled and it has postconditions that define the next activity in a conditional manner. In the case of the application activity of the driving licence service, inputs include a birthday certificate, the output is an application form, the precondition is that the applicant is older than 16 and the postcondition is that all fields in the application form are filled. Further, each activity can also be decomposed into several subactivities or can be specialized.

The crucial activity is the verification/qualification, since it reflects the constraints contained in the law. For example, it implements a rule that a person younger than 16 cannot apply for issuing the driving licence, whereas for motor cars (category B) the minimal age is 18. From the business process management point of view, the law can be treated as the business rule required to achieve goals of an organisation (defined by its business policy).

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 in which this law has to be implemented, and translate changes into the corresponding reconfiguration of the business process. Let us continue the example with the driving licence. Recently, the German law that regulates issuing driving licences has been changed, so that foreigners from non-EU countries must have the German driving licence, although they have a domestic licence. Let us analyse which changes in the existing business process for issuing the driving licence will be caused by this change in the law. For each change, we discuss the role that an efficient change management system should play. First of all, the public administrator should locate a

business process and the corresponding activities that should be modified due to this change in a law. This is a time-consuming action if it is performed in a non-systematic way. Therefore, an efficient change management approach should inform the public administrator on these activities automatically. It means that each business activity must contain a reference to a chapter/paragraph/article/amendment of a law it implements. For example, the activity verification/qualification of the driving licence service is based on the Chapter 2, Paragraph "Mindestalter" in the Law "Bundesgesetz ueber den Fuehrerschein".

After finding the service that has to be modified, the public administrator has to decide how to do that. She can specialise this service in the new one or she can adapt it to include new requirements. Let's assume that the public administrator made a decision to generate a specific driving licence service for foreigners. This service should not be generated from scratch. Rather, it should be a specialization of an already existing driving licence service. The public administrator has to change the preconditions of this new service, since it is only for foreigners from non-EU countries. This automatically causes a change in the preconditions of the original service[6], since the preconditions of two different services that provide the same functionality must be disjoint. Only in this way will the run-time system know which service to execute. It is clear that when the preconditions are semantically defined, the judgment about the inclusion relation among them can be done automatically.

Further, the verification/qualification activity of the new service requires checking whether a foreigner already has a domestic licence. Therefore, a new input for that activity is necessary. Since each input has to be supplied, this change is propagated to the previous activity, i.e. the application activity, which is responsible for the interaction with an applicant. It means that that activity has to deliver (as its output) the information about the domestic licence, the validity of which should be tested in the verification activity. Consequently, the application activity of the new service needs an additional input compared to the original service.

Obviously, different changes in a law have different consequences in the existing services. We briefly discuss one more example. Recently, the German law that regulates issuing driving licences has been changed, so that teenagers older than 17 can obtain a (temporary) licence for motor cars if they pass the exams and if they drive with a person that is older than 25, has the driving licence for more than five years, and has scored less than 20 negative points[7] in the last five years. In that case, the older person must have a licence for co-driving. This change in the law requires changes in the postconditions of the verification/qualification activity: instead of approval and non-approval of the licence, it can be temporarily approved. Further, the credential issuance activity has to generate an additional output, since the new co-driving licence should be printable, as well. An efficient change management system should enable the public administrator to perform all these changes efficiently (e.g. to make a minimal set of additional changes) and to ensure the overall consistency of the reconfigured service automatically (e.g. to prohibit that an activity has two contradictory preconditions). In the rest of the paper, we present a change management system that fulfils the above mentioned requirements.

3 Our approach

Given the requirements described in section 2, we have developed an approach for the change management for 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 (e.g. OWL-S or WSMO), the management of changes requires the management of these semantic descriptions. Therefore, our approach can be based on our previous work in ontology evolution ([10], [15], [16]). In this paper, we extend this work 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 that is used for describing semantic web services. Secondly, we define more complex changes that can be applied to these descriptions. Finally, we specify the consistency constraints that are derived from the semantics of this ontology.

3.1 Ontologies used for modeling semantic web services

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

following ontologies: (i) Meta Ontology that contains entities needed to describe services; (ii) Domain Ontology that contains domain specific knowledge; (iii) Service Ontologies that describe concrete services. The dependencies between these ontologies are shown in Figure 1.

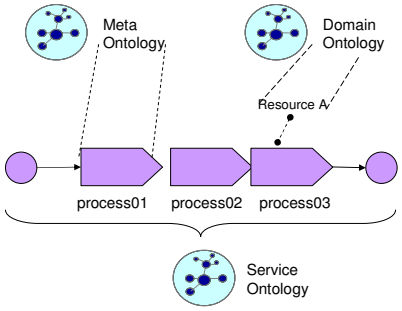

Figure 1 Different ontologies used for describing semantic web services

In defining the Meta Ontology, we based our work on the OWL-S ontology. 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, as we showed in section 2. In order to model the dependency between a business rule and the service implementing it, we extend OWL-S into our Meta Ontology.

More precisely, we extend the service profile ontology with 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. Legal documents have the structure that is typical for law, i.e. they contain explicit regulations (chapters, paragraphs, articles, etc.). This is modelled through the concepts "*Law*", "*Chapter*", "*Paragraph*", "*Article*", "*Amendment*" etc. and the relationships between them. For example, the property "*includes*" indicates that a law is specified through chapters or the property "*modifies*" stands for the fact that an amendment modifies a paragraph or an article.

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.

The third difference in comparison to OWL-S 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 [5]. We use preconditions for defining what a service expects for enabling it to provide its service and postconditions for defining the next service in a conditional manner, depending on the value of an output.

The Domain Ontology contains the domain specific knowledge. Returning to the example described in section 2, the domain ontology would consist of entities such as the concept "*driving licence*", properties "*hasName*", "*dateOfIssuance*", etc.

Finally, for each service, a Service Ontology that includes all previously mentioned ontologies is defined, and it might include other Service Ontologies. The Meta Ontology is reused, since it contains entities needed to describe a web service. The Domain Ontology is included, since it defines the domain-specific vocabulary. Therefore, the Service Ontology specialises concepts from the Meta Ontology (e.g. atomic service), establishes relationships between these concepts (e.g. what is the next or previous atomic service) and relates them to the domain entities from the Domain Ontology. 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, verification/qualification etc. (see section 2), which can be considered as atomic services (i.e. an activity). Therefore, this ontology includes 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.

3.2 Changes

For the ontology evolution, we defined the set of ontology changes that includes all elementary changes (e.g. "AddConcept") and some more complex changes, the so-called composite changes (e.g. "MoveConcept") [16]. 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 (e.g. thousand concepts) or complex (e.g. multiple concept hierarchy).

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 (see section 3.1). 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 [16]. A full set of changes[1] can thus be defined by the cross product of the set of entities of the Meta Ontology and the set of meta-changes. They are shown in Table 1.

The previously defined set of changes is complete and minimal. Completeness refers to the possibility of transforming an arbitrary service description in any other. Minimality refers to the achievement of completeness with a minimal set of changes. 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.

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.

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
Service Connection	AddServiceNextService	RemoveServiceNextService
Business Rule	AddServiceBusinessRule	RemoveServiceBusinessRule
Resource	AddServiceResource	RemoveServiceResource

3.3 Consistency

To define the consistency of the semantic web service ontologies, we start from the ontology consistency definition [15]:

[1] Due to the abstraction of the Meta Ontology, only the most typical and most frequently occurring changes are shown, since they are relevant from the management point of view.

Definition 1: 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 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 [10]:

Definition 2: 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:

- ***Service profile specific constraints***:
  - *Business knowledge specific constraints*
    C1: Each service has to have a reference to at least one business rule.
  - *Traceability*
    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 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 specialization of the activity A1 must always be a predecessor of any specialization of the activity A2, where A1 and A2 are two activities defined in the *Meta Ontology* and their order is given in advance (i.e. A1 precedes A2).

It is worth mentioning that only consistency constraints *C1* and *C12* are domain-dependent. Whereas *C1* has a reference to the *Business Rules Ontology (*i.e. the *Legal Ontology* in the E-Government domain*)*, *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:

Definition 3: A semantic web service is a consistent service iff its description is dependent ontology consistent and the additional constraints (C1-C12) are fulfilled.

4 Related work

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 [7]. 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 paper, as we mentioned in the introduction.

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 (see section 3.1.3) 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 [18] 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 [13], [4] 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.

5 Conclusion

In this paper, we presented an approach for the management of changes in semantic web services. The approach extends our previous work on the ontology evolution for 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, logic-based framework for coping with changes. Consequently, we can reason about the change management process, making it very flexible and efficient.

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).

6 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).

7 References

1. N. Adam, et 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 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 and the Modeling of Complaint Regulations in OTM'03, pp. 607-620, Catania, Italy, 2003.
4. A. Gomez-Perez, et al., A Framework for Design and Composition of Semantic Web Services, First International Semantic Web Services Symposium, 2004 AAAI Spring Symposium Series, 2004.
5. D Fensel, et al., The Web Service Modelling Framework WSMF, In Electronic Commerce Research and Application 1(2), pp. 113-137, 2002.
6. 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.
7. G. Joeris, O. Herzog. Managing Evolving Work of Specifications with Schema Versioning and Migration Rules. TZI Technical Report 15, University of Bremen, 1999.
8. J. Kim, Y. Gil , Towards Interactive Composition of Semantic Web Services, First International Semantic Web Services Symposium, 2004 AAAI Spring Symposium Series, 2004.
9. G. Leganza, IT Trends 2003, Midyear Update: Enterprise Architecture, Report Giga Group, 2003.
10. 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.
11. S. McIlraith, et al., Semantic Web Services, IEEE Intelligent Systems, 16(2):46-53, 2001.
12. T. Menzies, Knowledge maintenance: The state of the art, The Knowledge Engineering Review, 14(1) 1-46, 1999.
13. S. Narayanan, S. McIlraith, Simulation, Verification and Automated Composition of Web Services, In Proc. of the Eleventh International World Wide Web Conference (WWW-2002), pp.77-88, 2002.
14. J.E. Stiglitz, et al., The Role of Government in a Digital Age, http://www.ccianet.org/digital_age/report.pdf
15. L. Stojanovic, et al., Ontology Evolution as Reconfiguration-design Problem Solving, In Proc. of the international conference on Knowledge capture – K-CAP'03, Sanibel Island, USA, pp. 162–171, 2003.
16. L. Stojanovic, et al., User-driven Ontology Evolution Management, In Proc. of the 13th European Conference on Knowledge Engineering and Knowledge Management, pp. 285-300, 2002.
17. N. Stojanovic, et al., SEAL — A Framework for Developing SEmantic PortALs, In Proc. of the international conference on Knowledge capture – K-CAP'01, pp. 155 – 162, 2001.
18. K. Verma, et al., On Accommodating Inter Service Dependencies in Web Process Flow Composition, First International Semantic Web Services Symposium, 2004 AAAI Spring Symposium Series, 2004.