WS-Governance

A language for SOA Governance Policies definition

Josè Antonio Parejo Maestre, Pablo Fernandez and Antonio Ruiz Cortés
Computer Science and Engineering School
Department of Computing Languages and Systems
University of Seville
{japarejo,pablofm,aruiz}@us.es

Applied Software Engineering Research Group
`http://www.isa.us.es`

University of Sevilla
`http://www.us.es`

This work is partially supported by:

Ministerio de Ciencia e Innovación

Consejería Innovación, Ciencia y Empresa, Junta de Andalucía

January 28, 2011

Contents

1 Introduction

SOA adoption brings an increase on the number of elements of the IT architecture, where proper management and control become capital issues. In this context, SOA Governance is defined as the management process aimed at delivering the SOA promise of reuse, business goals support and responsiveness [11, 15]. Both industry and academia have identified SOA Governance as a promising area [15, 7].

According to [21] SOA Governance Lifecycle can be divided into six stages, from more abstract business levels to more concrete operational levels: Create a SOA strategy, Align Organization, Manage Service Portfolio, Control Service Lifecycle, Policy Definition and Enforcement and Service Level Management. In this paper we focus on the policy definition as the key stage that requires a deeper analysis in order to support an agile governance.

Effective governance requires a formalization of the governance policy management, including : (i) the definition of policies that encode governance rules and (ii) the establishment of appropriate conformance testing and enforcement mechanisms for defined policies. Moreover, we have identified the need to incorporate the structure of the organization as an essential information to take into account when the governance policies are designed. Our case study shows an important number of elements where governance policies can be specified (such as services, applications or departments) This high variability represents an important drawback in terms of management since it boosts the possibility of specifying inconsistent policies; specifically, in our case study, the structure, size and departmental autonomy of the organization implies that multiple administrators could specify policies in a distributed and independent way. In this context, the scenario represents a Cooperative Information Systems reality with highly distributed inter-organizational interaction that should be coordinated. Therefore, the capability of automatic consistency checking of policies is highly valuable.

The current governance tools market is vendor-driven and turbulent, where tools are based on proprietary technology and its features are guided by the specific aspects where its vendors have expertise [7, 8]. Furthermore, most current governance platforms assume that policy definitions are error-free, but policies can be inconsistent [10].

The contribution of this paper is twofold: (i) First, a language for governance policies definition is presented. This language defines governance documents; which make policies unambiguous by providing a rich context for governance policies and their metadata, while maintaining their definition independently of the SOA elements to govern, their internal organization and the underlying infrastructure. (ii) Secondly, a formal definition of governance document is proposed, describing the elements to govern, their properties and the policies that govern them. This formal definition allows the automation of policies consistency checking.

To the best of our knowledge, this proposal provides a novel approach, paving the way for building more powerful and automated governance tools. This proposal has been developed and tested on a proof of concept prototype.

The rest of the paper is structured as follows: In the next section preliminary concepts are provided. In section 3, the case study that motivates the research presented on this paper is described, drawbacks of WS-Policy for governance policy specification are depicted and the need of a governance document is motivated. Section 4 presents

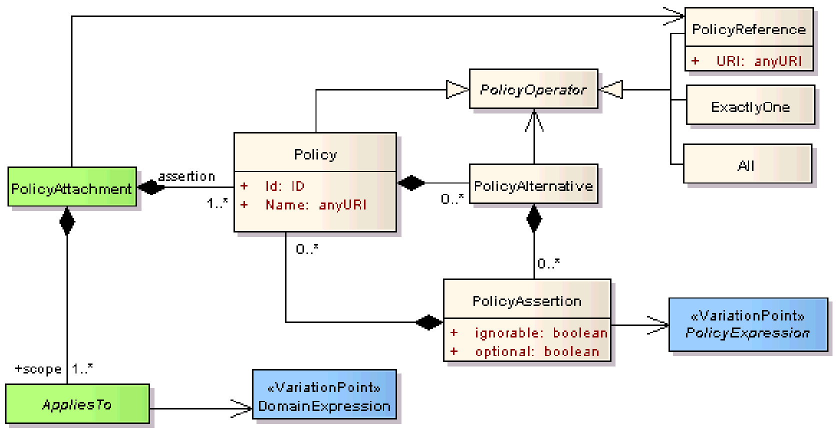


Figure 1: UML metamodel of WS-Policy

WS-Governance, our proposal of XML-based language for governance policies specification, and its plain text equivalent WS-Gov4People. Section 5, presents a mapping of governance documents to Constraint Satisfaction Problems that allow the automated checking of properties. Finally, in sections 6 and 7 related work is described and conclusions are drawn.

2 Preliminaries

A Governance Policy represents a capability, requirement or behavior that allows the SOA to achieve its goals, and whose meeting is quantifiable and monitorable through time [2, 11, 4, 16]. Governance policies are as heterogeneous as the said governed elements, addressing the distributed, flexible, and heterogeneous nature of current SOAs. Moreover, governance policies originate from disparate sources, from legal regulations and their derived compliance issues to strictly technical details.

WS-Policy is a W3C recommendation that provides a framework for defining policies [25] which takes into account this very complex situation. As can be seen in Fig. 2 where the UML metamodel of WS-Policy is shown[1], the building blocks of policies are assertions (`PolicyAssertion`) that are composed using operators: `All` equivalent to logical AND, `ExactlyOne` equivalent to logical XOR, and the top level compositor `PoliciAlternative` equivalent to logical OR. Policy nesting is supported by meaning a logical AND operation of the global policy and the nested one.

Assertions represent domain-specific capabilities, constraints or requirements, where their grammar is left open by WS-Policy, thus allowing the use of XML-based Domain Specific Languages (DSLs) for that purpose (see "VariationPoint" stereotyped `PolicyExpression` in figure 2) . This is a common strategy also followed by recommendations such as WS–Agreement to meet the open–closed design principle: recommendation should be open for extension but closed for modification. WS-Policy has been mainly focused on the definition of policies related to specific service capabilities such as security, reliability, etc. In fact, there are a number of DSLs for those purposes.

[1]The UML class diagram in 2 represents our interpretation of the metamodel described by the XML schema specified in [25] and [24].

Unfortunately, describing assertions for SOA governance policies is a bit more complicated (see later) and as far as we know there is currently no single DSL to describe this kind of policies.

Two mechanisms are available in WS–Policy to associate policies with the elements to which they apply, i.e., their scope. The first mechanism aims to associate one or more policy definitions as a part of the element definition. For example, if we want to apply a policy to a single web service described in WSDL, the policy specified in WS-Policy has to be inserted into the WSDL code. We called this mechanism *endogenous attachment*, since the definition of the policy is internal to the element one. Left column in table 1 shows an example of this kind of attachment. In this case, two polices on the web service "StockQuote" are defined: one to ensure a reliable message and another to specify security mechanisms on web service binding. Notice that with endogenous attachment: i) attaching a set of policies to a set of elements at the same time is not possible (*one–element specification*) and ii) changing a policy requires modifying the definition of an element, in other words, it is an *intrusive* mechanism.

The second mechanism aims to associate one or more policy definitions to one or more elements, and more specifically to the references of these elements. To this end, WS-PolicyAttachment recommendation [24] proposes the use of `PolicyAttachemt` and `AppliesTo` (shaded classes in figure 2). In this case, the WS-Policy is not encoded in the same file that the specification of the element. We call this mechanism *exogenous attachment*. Right column in table 1 shows an example of this kind of attachment. In this case, secure binding mechanisms are asserted on the "StockQuote" and "MortgageRisk" services using its endpoint reference address. As it can be seen in figure 2, WS-PolicyAttachment also leaves open the language to specify the scope, in fact, it is the domain expression variation point, *domain expression* to specify policy subjects. Domain expressions grammar is left open by WS-Policy, thus allowing the use of XML-based DSLs, but WS-Policy provides a basic language to specify it based on URIs.

Note that with exogenous attachment: i) it is possible to attach a policy to a set of elements at the same time (*multi–element specification*) and ii) changing the policy attachment does not require modifying the definition of an element, in other words, it is a *non–intrusive* mechanism.

2.1 Compatibility amongst policies

WS-Policy defines a mechanism to test the compatibility of two policies, called *policy intersection*. According to the WS-Policy specification [25] "policy intersection is optional but a useful tool when two or more parties express policies and want to limit the policy alternatives to those that are mutually compatible". The intersection consists of two parts: a domain-independent policy intersection and domain-specific processing. The former takes into account the assertion type equality, *i.e.* the XML element type equality and its nested elements, but not its parameters. The latter is not defined in WS-Policy, thus assertions authors have to define specific mechanism for incorporating the intended semantics of the assertions (and the specification does not provide a standard mechanism to integrate it). For instance, the first policy specified in the left column and the policy in the right column are incompatible, since their element types

WS-Policy 1		WS-Policy 2	
<wsdl:definitions name='StockQuote'	Element	<wsp:PolicyAttachment>	
xmlns:wsp='...' ...>	attached	<wsp:AppliesTo>	\
<wsp:Policy wsu:Id='RmPolicy' >	\	<wsa:EndpointReference>	| Policy
<rmp:RMAssertion>	|	<wsa:Address>.../MortageRisk.wsdl</...>	| Scope
<wsp:Policy/>	|	<wsa:Address>.../StockQuote.wsld</...>	| Scope
</rmp:RMAssertion>	|	</wsa:EndpointReference>	| def.
</wsp:Policy>	|	</wsp:AppliesTo>	/
<wsp:Policy wsu:Id='X509Policy'	| Policy	<wsp:Policy wsu:Id='X509Policy'	\
<sp:AsymmetricBinding>	| assertions	<sp:AsymmetricBinding>	|
...	|	...	| Policy
</sp:AsymmetricBinding>	|	</sp:AsymmetricBinding>	|assertions
</wsp:Policy>	/	</wsp:Policy>	/
...		</wsp:PolicyAttachment>	
</wsdl:definitions>			

Table 1: Endogeous vs Exogeous Attachment

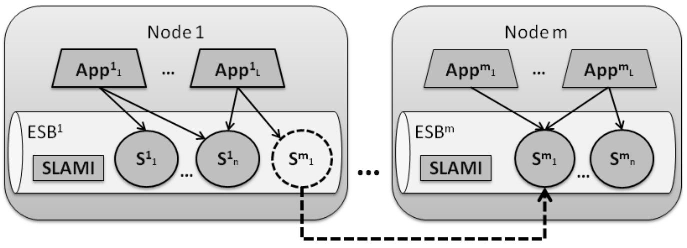


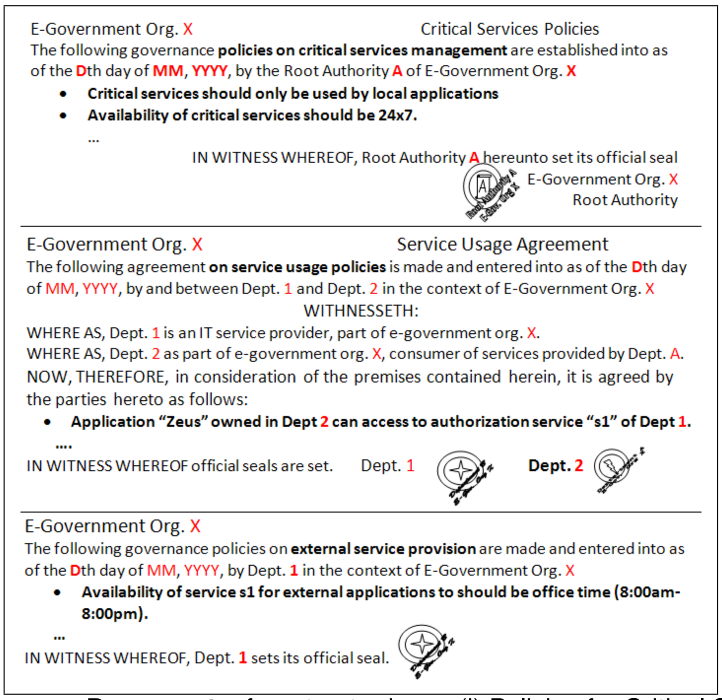
Figure 2: Case study architecture

$< rmp : RMAssertion >$ and $< sp : AsymmetricBinding >$ are different. The second policy in the left column could be compatible depending on the nested sub-elements, where each policy should contain a matching nested subelement, and those elements should be of the same type. The notion of policy intersection is thus basically syntactical and structural, and it is not valid for complex domain-specific processing or semantic reasoning about policies, as shown in [6] and [1].

3 A motivating use case

The motivation of our approach is derived from a case-study based on a real scenario involving a regional-wide governmental organization. This organization has a complex structure divided into 16 governmental departments with around 5,000 end users using a shared IT infrastructure. thousands of end users (civil servants using the IT infrastructure). This infrastructure is distributed in the different departments both logically and physically and is usually managed autonomously in each location. In recent years there has been a shift toward SOA and currently there is an important number of core services replicated in the infrastructure with different QoS capabilities.

From an architectural point of view, the infrastructure is designed as a federated bus of services; in this context, each department represents a node with two main elements: an Enterprise Service Bus and a Management System that provide different horizontal functionalities (such as monitoring, transactions or security). All the different nodes are integrated conforming the global infrastructure.


E-Government Org. X Critical Services Policies

The following governance **policies on critical services management** are established into as of the Dth day of MM, YYYY, by the Root Authority A of E-Government Org. X

- **Critical services should only be used by local applications**
- **Availability of critical services should be 24x7.**

...

IN WITNESS WHEREOF, Root Authority A hereunto set its official seal

E-Government Org. X

Root Authority

E-Government Org. X Service Usage Agreement

The following agreement **on service usage policies** is made and entered into as of the Dth day of MM, YYYY, by and between Dept. 1 and Dept. 2 in the context of E-Government Org. X

WITHNESSETH:

WHERE AS, Dept. 1 is an IT service provider, part of e-government org. X.

WHERE AS, Dept. 2 as part of e-government org. X, consumer of services provided by Dept. A.

NOW, THEREFORE, in consideration of the premises contained herein, it is agreed by the parties hereto as follows:

- **Application "Zeus" owned in Dept 2 can access to authorization service "s1" of Dept 1.**

....

IN WITNESS WHEREOF official seals are set. Dept. 1 **Dept. 2**

E-Government Org. X

The following governance policies on **external service provision** are made and entered into as of the Dth day of MM, YYYY, by Dept. 1 in the context of E-Government Org. X

- **Availability of service s1 for external applications to should be office time (8:00am-8:00pm).**

...

IN WITNESS WHEREOF, Dept. 1 sets its official seal.

Figure 3: Governance Documents, from top to down: (i) Policies for Critical Services enacted by root authority A; (ii) Policies on service usage between Dept. 1 and Dept. 2; (iii) Policies on service provision internal to Dept. 1

The different services are deployed in the bus and the consumer applications ask the bus for the appropriate provider. In figure 2 an architectural conceptualization is shown: each of the nodes correspond with a department; the nodes are composed of applications for end-users with the bus providing core common services. As depicted, the bus can provide access to external services from other nodes.

Due to the structure of the organization, each department has developed a high autonomy in its IT infrastructure management. Consequently, the integration of applications and services amongst different departments has raised an important issue: the need to specify a consistent normative framework on the whole organization for meeting business needs without breaching autonomy.

In this motivating scenario, a SLA Management Infrastructure (SLAMI) in charge of creating, storing, locating and enforcing SLAs for transactions has been deployed in each node. Based on the analysis of the common organization operations and the usage of the current SLAMI component, we have identified some examples in this scenario where SOA governance is applied. In the following cases, the infrastructure should *cual?* adapt to provide an appropriate behavior depending on different parameters:

- Depending on the application. Some applications have a higher priority than others or require different security restrictions and QoS levels.

- Depending on the application's department (i.e. node). Since applications can access services of external departments, the nodes could establish different rights for external (from other nodes) or local (the same node) applications.

Figure 3 shows three excerpts of different real governance documents found in our case study -conveniently modified in order to preserve privacy and meet confidentiality clauses-. Currently, these fragments correspond to human-oriented policies that should be enforced by administrators by means of configurations of the IT-infrastructure. Each document is enacted by a different organization: the first document by the main authority so it should be enforced by all sub-organizations (departments); the second document represents an integration agreement amongst two departments (1 and 2) and finally, the last document is an internal governance document of department 1.

There is a real and urgent need of a language to define governance policies unambiguously with precise semantics; as a first step toward governance policy definition, enforcement and automatic consistency checking.

3.1 WS-Policy drawbacks

In working for a Public Administration, we were concerned with developing mainstream policies that would avoid ad-hoc solutions. We therefore tried to use WS-Policy; however, when applying it to the previously described documents we encountered the following limitations:

Lack of Context and meta-data (LCD) Governance policies need a rich context to ensure their validity, specifying who enacts the policies and providing additional metadata in order to ensure authorization for policy enactment and the integrity of the policies as enacted, thus avoiding tampering. In using WS-Policy , there is not a single point where we can insert the required information that assures the validity of the policy, such as official seals, a declaration of validity by the enacting authority or a GD's preamble. We call this problem Lack of Context Data (LCD)

Scope Definition Limitations (SDL) Defining a policy P1 as simple as "*All services provide an Availability greater than 99%*" may become a nightmare for SOA practioners since WS–Policy has not been designed keeping in mind that the scope of a policy could be defined by intension. For example, if an endogenous attachment is used then all the services descriptions will need to be changed to incorporate the policy. This intrusive action used to be forbiden in large organizations since it is time–consuming and error prone. In turn, if an exogenous attachment is used, then all the services references will need to be computed and inserted in the `AppliesTo` section of the policy.

It may seem that this kind of policy is well supported by exogenous attachment, but it is not true in our case study. For example, if there was a change in a political decision *All services except S23, S45, . . .*) this would entail a re-working of over a hundred services as well as modifying the content of the scope section. This is not an adequate solution when dealing with a SOA comprising of hundreds of applications and services; where policies are often, even if slightly, changed.

Summarizing, these circumstances make it very appealing to have the possibility to define the policies' scope by intension and not only by extension; which was the only used mechanism to date, without the creation of a DSL language to support this.

The expression of these scope predicates require a rich predicate DSL and the specification of the elements and data sources needed to feed the predicate, in order to effectively evaluate policy scope. This problem is particularly acute for governance policies, since governance relevant information is stored in disparate sources, such as UDDI registries, LDAP directories, *ad hoc* databases, etc. For instance, in our sample GDs there are properties such as "*critical services*" and "*local/external apps & services*" whose values must be obtained from different information sources.

Isolation of Policies (IoP): WS-Policy only defines `Policy`, and `PolicyAttachment` as top level elements, were no logical grouping of cohesive interrelated policies is provided except for the logical compositors. A GD should, however, contain a set of cohesive policies that govern concrete aspects of the architecture; following specific management or business goals in order to perform a proper management of the own governance policies set. For instance, in our sample Gov. Doc. (i), the set of policies that manage critical services are grouped in a unique cohesive document.

WS-Governance Documents address drawbacks described above by incorporating an extensible context to the contained policies (addresing LCD), thus defining a global document structure that contains a set of policies under an umbrella governance scope (addressing IoP). It also describes relevant governance properties and provides mechanisms for incorporating disparate governance-relevant information sources (addressing SDL and some LCM issues). GDs are described in detail in the next section.

Inadequate consistency cheking Due to the specific nature of the governmental authority, a fundamental need in our case-study was to check the consistency of the government policies. The only analysis operation WS-Policy envisioned for this was the intersection of policies. In our case, this analysis was not appropriate due to the syntactical or structural nature of this operation. For example, two identical-meaning policies may prove inconsistent by using the intersection operation. Consequently, a semantical consistency notion is needed.

One of the most valuable features of any language is to dispose of a tool support able to debug programs written in that language as well as to provide some interesting properties. In our user case, one of the most valuable properties was to check the consistency. It is said that a policy is consistent if it has no internal contradictions.

If the automatic management and enforcement of policies is the aim, a formal semantics of governance policies is needed. The open and flexible nature of WS-Policy makes it difficult to provide homogeneous semantics to policies, since each domain specific DSL would have its own semantics. This problem motivates the merely structural-syntactical nature policy intersection operation as defined in WS-Policy, avoiding its usage in diverse scenarios [6, 1], we name this drawback as **Syntax/Structure Driven Semantics (SDS)**. LCD and SDS drawbacks motivate the creation of two general purpose XML-based DSLs for governance policy assertions and SOA modelling described in detail in the next section. Based on those DSLs and the authors' experience providing formal semantics for SLAs specified in WS-Agreement by using CSPs [19, 13, 12], a CSP based semantics for WS-Governance documents using those DSLs is proposed in Sec. 5 addressing SDS. Based on those semantics a consistency property for policies and governance documents is defined.

4 From WS-Policy to Ws-Governance

A WS-Governance Document (GD) provides policies definitions along with contextual metadata, sources-of-governance relevant information, and the specification of properties on which policies definitions is based. In so doing, WS-Governance addresses

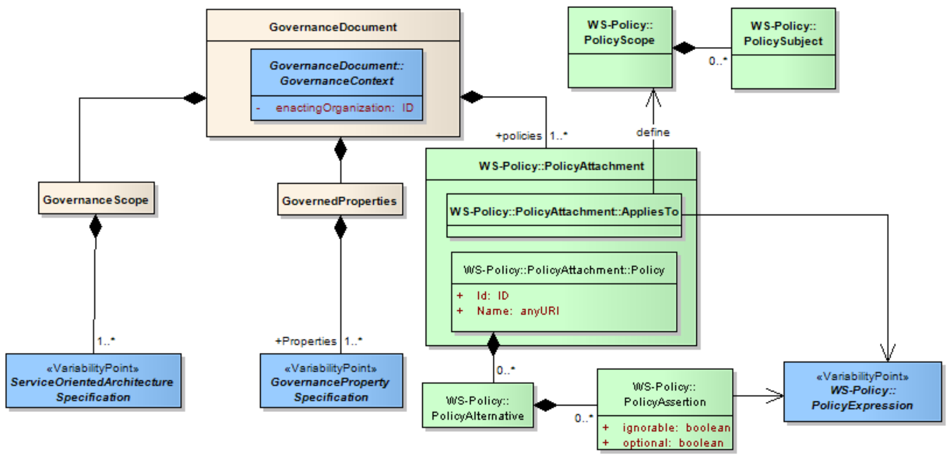


Figure 4: UML Metamodel of WS-Governance

the aforementioned drawbacks of WS-Policy for governance policies definitions. The structure of a GD in WS-Governance comprises of:

- **Governance Document Context:** It currently defines the governing organization but its grammar is left open in order to support the expressions of authorizations to enact policies on this GD, and the data needed to ensure GD authenticity and integrity.

- **Governance Properties:** It defines all properties that are relevant for governance policies. Following the philosophy of WS-Policy, its grammar is left open, allowing the use of XML-based DSLs for specifying those properties.

- **Governance Scope:** It provides information about the SOA where policies are established. Different information sources could be used in this section, from UDDI registries to ad hoc databases, since any SOA element could be a governance policy subject; such as projects, developers, organizations, messages, XML-schemas or applications servers.

- **Governance Policies:** It defines the policies that conform the governance. Those polices are WS-Policy compliant, where the exogenous policy attachment mechanism is mandatory. Assertion and scope definition grammar is left open, allowing the use of XML-based DSLs.

The UML class diagram shown in Fig. 4 represents our proposal of metamodel for WS-Governance documents.

4.1 DSLs in WS-Governance

Some elements of WS-Governance are intentionally left open for extension in order to allow a high degree of flexibility. This flexibility is based on the use of XML-based DSLs in some variability points, allowing the creation of a whole family of governance languages. In Fig. 4 XML-based DSL variability points are decorated with a *VariabilityPoint* UML stereotype. A brief description of these variability points is provided as follows:

- **Context DSL:** the GD metadata in the context element can be extended with any information needed by means of the nesting of new XML elements and attributes.
- **SOA Specification:** The architecture and elements to govern must be described in order to define unambiguous policies.
- **Governance Property Specification:** A description of the properties of the governed elements of the SOA is needed in order to define expressive policies. Those properties must be expressed using a XML-Based DSL.
- **Policy Expression Specification:** Policy scope and assertions can be expressed using any predicate-oriented DSL.

In order to define effective governance documents, those DSLs must be set. In our proposal we provide two DSLs that allow the creation of service-focused governance policies, *i.e.* policies that specify assertions defined on service properties and their directly related elements such as consumers, providers, and governance relevant information such as organizational structure. Those DSLs are *Service Oriented Architecture Modelling Language (SAML)*, addressing the SOA Specificaton variation point, and *Govenance Assertion Language (GAL)*, addressing the Governance Property Specification and Policy Expression Specification variation points.

The UML Class Diagrams in Figs. 5 and 6 depicts the metamodel of SAML and GAL respectively.

4.2 SOA Modeling with SAML

SAML has been designed to model the SOA state and structure, making our proposal independent of the specific governance information sources available on each SOA, such as UDDI Registries, LDAP directories, *ad hoc* databases, etc. SAML describes both the SOA structure as elements, and its state as the corresponding governance properties value to those elements. In this paper we focus on service-related governance policies, so SAML mainly contains elements related with services; however SAML is extensible, supporting the use of any XML-based construct as sub-elements of its basic structural elements. Specifically, structural elements in SAML are described as follows:

- Service Oriented Architectures are networks of participants providing and consuming services to fulfill a purpose. In SAML these participants are specified as organizations and applications.
- Organizations are participants with governance relevant identity and properties, tracing an organizational boundary on their owned applications and services. Organizations are arranged hierarchically, where an organization can contain various sub-organizations (*e.g.* departments) and have a unique parent.
- Applications represent business processes, related capabilities and software packages. They allow the arrangement of software artifacts and capabilities independently of the organizational hierarchy in a governance-meaningful way. Applications are owned by a unique organization. Applications have a set of provided and consumed services.

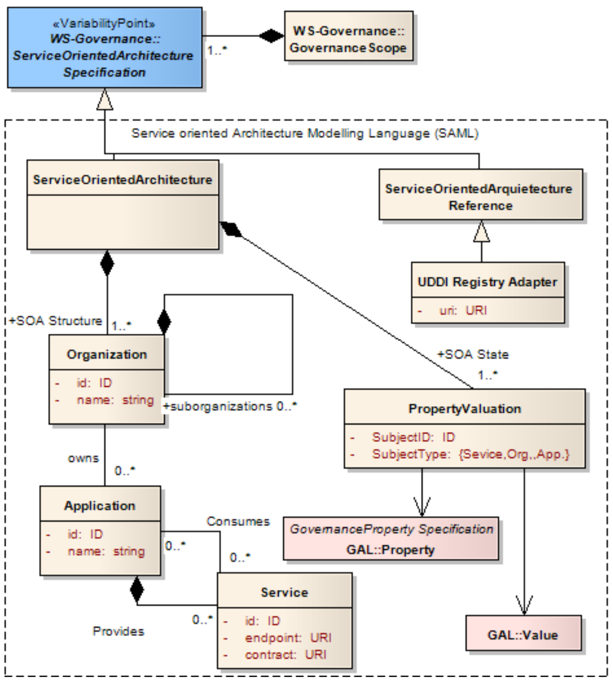


Figure 5: UML Metamodel of SAML

- Services represent capabilities that participants provide and consume.

Regarding SOA state description, SOAML allows the specification of property values for all the aforementioned elements based on GAL.

Finally, SAML provides a generic element for the specification of the concrete governance data sources as references; such as UDDI registries, that should be queried to obtain the governance-relevant SOA structure and state in order to check properties and test policies adherence. By creating adapters that query those data sources and create a SOAML compliant SOA model, our proposal becomes independent of those specific data-sources, thus semantics of GDs are based on explicit SOAML models.

4.3 Specifying Governance properties, and policy assertions with GAL

Governance Assertion Language GAL is a generic and expressive language designed to declare governance properties and assertions. Property definitions in GAL have a name and an identifier as attributes, comprising of: (i) type definition, where basic XML-Schema [18] types are supported, (ii) an optional domain definition that restricts the space of valid values of the property; where it could be described as a GAL assertion (by intension) or as a set of values (by extension); and (iii) an optional SAML governance subject declaration, that defines the type of SOA element that can present the property (service, organization, policy, all, etc.). Through GAL constraints we provide a suitable language to specify policy assertions on governance properties. Assertions can be composed using WS-Policy composition operators: All ($\odot$), ExactlyOne ($\otimes$) and PolicyAlternative ($\oplus$).

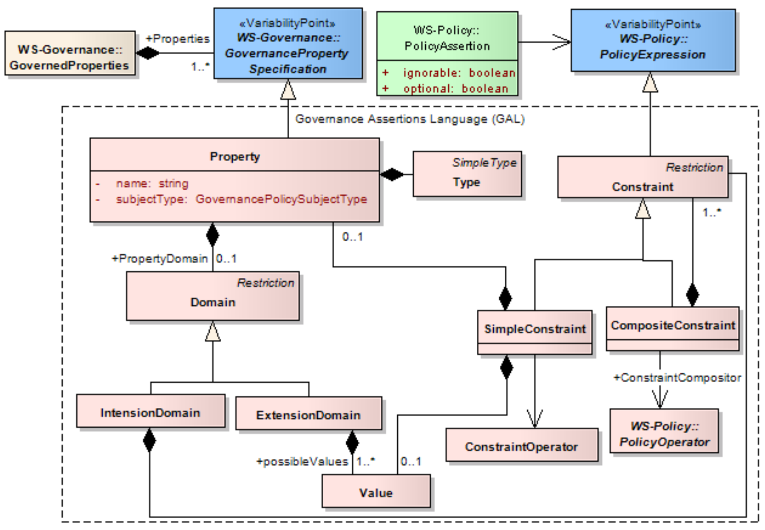


Figure 6: UML Metamodel of GAL

In order to allow consistency checking, in this paper we use a subset of WS-Governance a bit less expressive, called WS-Governance*. A WS-Governance* document must use SAML to describe the SOA to govern and GAL to define governance properties, policy scopes and policy assertions.

A WS-Governance* document (ρ) comprises of:

- Governance Scope defines the set of organizations O, applications A and services S to govern, and their relationships, namely: consumption of services by applications and organizations, provision of services by applications and organizations, ownership of applications by organizations and hierarchy of organizations. Only those elements and relationship functions are used to define policies.

- Governance Vocabulary must define the set of all properties V used in the guarantee terms.

- For each governance policy both scope (s) and assertion (a) must be defined as GAL assertions on the properties defined in the governance vocabulary section, and only to those applied to the sets and relationship functions defined in Governance Scope.

- Policy assertions can be composed using the compositors defined in WS-Policy: All ($\odot$), ExactlyOne ($\otimes$) and PolicyAlternative ($\oplus$).

XML-Schemas that model WS-Governance* documents conforming the previously described syntax are available at [17]. Although readable for humans, XML is not as understandable as plain text, even if it is formatted for that purpose. Consequently in Table 2 the structure of a WS-Governance* document is described in a plain text language, named WS-Gov4People, that is equivalent to the XML-Schemas. The mapping of schema elements onto its corresponding WS-Gov4People sentences is shown in Table 2.

A WS-Gov4People document describing the policies specified in figure 3 and a simple SOA structure is shown in the first column of table 4.

WS-Governance/SAML/GAL XML Element	WS-Gov4People Document Structure
<wsg:GovernanceDocument Name='*name?*' Id='*Id?*'>	***Governance Document*** - *Name* (*Id*)
<wsg:Governor id='*Org. Id?*' name='*Org. Name?*'/>	**Governor:** *Org. Name?*(*Org. Id?*)
<wsg:GovernanceScope>	**Scope:**
{<saml:ServiceOrientedArchitectureReference>...</...> |	{SOA Registry Reference|
<saml:ServiceOrientedArchitecture>...</...>}+	SOA Governance Model}+
</wsg:GovernanceScope>	
<wsg:GovernanceVocabulary>	**Vocabulary:**
<gal:Property> ...</gal:Property>+	{ Property: ... }+
</wsg:Vocabulary>	
<wsg:GovernancePolicies>	**Policies**:
{<wsp:PolicyAttachment>	{**Policy** *name?* (*Id?*)
<wsp:AppliesTo>	
<gal:QuantifiedAssertion>	
{<gal:Quantifier type='*Exists*|*ForAll*' varname='*VarName?*'	{*forall*|*Exists VarName?*
in='*Service*|*Org*|*App*'>}+	in (*Servs*|*Orgs*|*Apps)*}+
<gal:Assertion>*Scope expr*?</gal:Assertion>	**Scope**: *Scope expr*?
</gal:QuantifiedAssertion>	
</wsp:AppliesTo>	
<wsp:Policy name='*name?*' id='*Id?*'>*Assertion expr?*</...>	**Assertion**: *Assertion expr?*
</wsp:PolicyAttachment>}+	}+
</wsg:GovernancePolicies>	
<wsg:GovernanceDocument>	
<saml:ServiceOrientedArchitecture>	**SOA Governance Model:**
	STRUCTURE:
{<saml:Organization name='*Org. Name?*' > id='*Org. Id?*'>	{**Organization:** *Org. Name?* (*Org. Id?*)
<saml:SubOrganizations>	**SubOrganizations:** *Org. Id1?,...,Org. IdN?*
{<saml:Organization ...> ... </saml:Organization>}*	
</saml:SubOrganizations>	
<saml:Applications>	**Applications:**
{<saml:Application name='*App. Name?* id='*App. Id?*'>	{Application: *App. Name?* (*App. Id?*)
<saml:ProvidedServices>	Provides:
{<saml:Serivce name='*Serv. Name?* id='*Serv. Id?*'/>}*	{Service: *Serv. Name?* (*Serv. Id?*)}*
</saml:ProvidedServices>	
<saml:ConsumedServices>	Consumes:
{<saml:Serivce name='*Serv. Name?* id='*Serv. Id?*'/>}*	{Service: *Serv. Name?* (*Serv. Id?*)}*
</saml:ConsumedServices>	
</saml:Applications>}*	}*
}+	STATE:
{<gal:PropertyValue property='IdP' subject='IdS' value='*Val. Expr?*'/ >	*Val. Expr?**
</saml:ServiceOrientedArchitecture>	}+
<gal:Property name='*Prop. Name?*'	**Property:** *Prop. Name?* (*Prop. Id?*)
id='*Prop. Id?*' subjectType='*Serv*|*Org*|*App*'>	
<gal:Type>*Type. Expr.?*</gal:Type>	**for** {*Servs*|*Orgs*|*Apps*}
<gal:Domain><gcl:Constraint>*Domain Expr.?*	Type: *Type. Expr.?*
</gal:Constraint></gal:Domain>	
</gal:Property>	Domain: *Domain Expr.?*

Table 2: Mapping from WS-Governance, GAL and SAML to WS-Gov4People

5 Property checking through CSPs

In this section, we show how CP can help governance policy definition in a highly distributed cooperative environment by checking for consistency.

5.1 CSPs in a nutshell

A constraint is a relationship among several variables, each of which ranges over a given domain. Thus, a constraint restricts the values of its variables. Their most important feature is their declarative nature, i.e. they specify what relationships must be held without specifying a computational procedure to enforce them. The idea of CP is to solve problems by stating constraints about the problem area and, consequently, finding a solution that satisfies all of the constraints. This task is carried out by so-called solvers. A problem expressed as a set of constraints is formalized as a *Constraint Satisfaction Problem* (CSP). A CSP is defined as a set of variables and a set of constraints specifying which combinations of variables and values are acceptable.

Definition (CSP) A CSP is a three-tuple of the form (V, D, C) where $V \neq \oslash$ is a finite set of variables, $D \neq \oslash$ is a finite set of domains (one for each variable), and C is a set of constraints defined on V.

A solution σ to a CSP consists of an assignment in which each variable gets a value from its corresponding domain, as long as it satisfies each constraint.

Definition (Solution Space) Let ψ be a CSP of the form (V, D, C), its solution space, denoted as $sol(\psi)$, is composed of all its possible solutions.

$$sol(\psi) = \{\sigma \in V \to D | \sigma(C)\}$$

where $\sigma(C)$ holds iff each assigment in σ satisfies every constraint in C.

Definition (Satisfiability) Let ψ be a CSP of the form (V, D, C), it is satisfiable, denoted as $sat(\psi)$, iff its solution space is not empty

$$sat(\psi) \Leftrightarrow sol(\psi) \neq \oslash$$

For instance, being $\psi = (\{u, v, x, y\}, \{\{0, 1, 2\}, \{0, 1, 2\}, \{0, 1, 2\}, \{0, 1, 2\}\},$ $\{\{u < v\}\{x < y\}\})$, then $sol(\psi) = \{\{0, 1, 0, 1\}, \{0, 1, 1, 2\}, \{1, 2, 0, 1\}, \{1, 2, 1, 2\}\}$.

5.2 Semantics of GDs* based on CSPs

As shown in [23], the definition of the semantics of a language can be accomplished through the definition of a mapping between the language itself and another language with well-defined semantics such as Abstract State Machines, Petri Nets, rewriting logic or CSPs. These semantic mappings between semantic domains are very useful not only to provide precise semantics to DSLs, but also to be able to simulate, analyze or reason about them using the logical and semantical framework available in the target domain.In this section we define the mappings that tranform GDs* onto CSPs that provide their precise semantics, allowing the usage of CSP solvers to reason about policies and complete GDs*.

Mapping a GD* into CSPs. The mapping ($\mu^{GD} : \rho \rightarrow \psi$) of a WS-Governance* GD (ρ) to a CSP (ψ) is performed in two steps as follows:

1. For each policy $p_i = (s, a)$ in the GD*, p_i is mapped for the concrete governance scope (SOA model in terms of services, applications, organizations and its relationships) into a constraint p_i^C that contains only variables and literals composed using logical and algebraic operators. This constraint is constructed so that it tells exactly when the policy holds in the given governance scope. This transformation is performed by the explicit enumeration of the sets and relationships (ownership, provision, consumption, and organizational hierarchy) on the governance scope for each quantifier, combining the resulting constraints using logical AND ($\wedge$) operators for universal quantifiers and logical OR ($\vee$) operators for existential quantifiers.

2. The set of constraints $p^C = \{p_i^C\}_{i=1}^n$ and original GD* ρ are mapped into a CSP $\psi = \{V, D, C\}$ by creating:
   - a variable $v^x_{s_i|o_i|a_i}$ in V for each property x and corresponding element in the SOA (service, organization and application).
   - variables $v^{supOrg}_{o_i}$,$v^{prov}_{s_i}$,$v^{cons}_{s_i}$ and $v^{own}_{a_i}$ in V for the relation functions $supOrg$ (hierarchical relationship among organizations), $provider$ and $consumer$ (relationship among services and applications) and $owner$. Additionally, their domain of organizations, applications, and services are created.
   - constraints $\{v^{supOrg}_{o_i} = o_j\}$, $\{v^{prov}_{s_i} = a_k\}$, $\{v^{cons}_{s_i} = a_k\}$, and $\{v^{own}_{a_i} = o_l\}$ in C for each variable created in the previous step specifying the values of the relationships$supOrg$, $provider$,$consumer$ and $owner$. Those constraints and variables express the SAML SOA structural model of the governance scope.
   - a constraint $\{v^x_{s_i|o_i|a_i} = \{\mathit{Value\ Expr?}\}\}$ in C for each property valuation specified in the state section of the governance scope in ρ.

   Finally, for each constraint in p^C property invocation functions $X(s_i|o_j|a_j)$ are exchanged by their corresponding variables $v^x_{s_i|o_i|a_i}$, and the resulting constraint is added to C.

The mapping of different elements of a GD* is shown in Table 3.

In order to exemplify the use of GD*, in the left column of table 4 we can see the expression of the policies contained in the governance documents found in the case study (Figure 3). In the table we can see the four different parts of a GD* document: Context, SOA Model, Vocabulary and Policies. In this context, the SOA Model part has been enriched with the structure information of a subset in the organization: on the one hand, Department 1 has one A1 application that provides s1 service and on the other hand, Department 2 has two applications (Zeus and A3). The former consumes s1 and s2 and provides s3 service; the latter only provides s2. It is important to highlight that

WS-Governance* Element	CSP Mapping
Governance Document - *Name* (*Id*) **Governor:** *Org. Name?*(*Org. Id?*)	GD Name, Id and Governor Org. are not mapped to CSP
SOA Governance Model: STRUCTURE: {**Organization:** *Org. Name?* (*Org. Id?*) **SubOrganizations:** *Org. Id1?,...,Org. IdN?* **Applications:** Provides: {Service: *Serv. Name?* (*Serv. Id?*)}* Consumes: {Service: *Serv. Name?* (*Serv. Id?*)}* }* }+ STATE: *{Property val. Expr?}**	For each organization o_i a variable $v_{o_i}^{suporg}$ is created denoting the parent org. of o_i and a domain $d_i^{suporg} = \{o_1, \dots{}_n\}$ is added to D a constraint $\{C_{o_i}^{prov} = o_j\}$ is created encoding the orgs. hierarchy For each application a_j a variable $v_{a_j}^{own}$ is created denoting the owner org. of a_j, a domain $d_j^{own} = \{o_1, \dots{}_n\}$ is added to D and a constraint $\{C_{a_i}^{own} = o_j\}$ is created encoding the app. ownership For each service s_k a variable $v_{s_k}^{prov}$ is created , a domain $d_k^{prov} = \{a_1, \dots, a_m\}$ is added to D and a constraint $\{v_{s_k}^{prov} = a_l\}$ is created encoding the provisioning Add Constraint: $v^x_{s_i\|o_i\|a_i}$[*Proverty val. Expr.?*]
Vocabulary: **Property:** *X* **for** *Services* **for** $\{Servs\|Orgs\|Apps\}$ Type: *boolean* Domain: *Domain Expr.?*	Add variables & domains: for each property x we create a variable $v^x_{s_i\|o_i\|a_i}$ for each element in its corresponding set $S\|O\|A$ Add Constraint: $v^x_{s_i\|o_i\|a_i}$[*Domain Expr.?*]
Policies: **Policy** *P1* {**forall** *y* in $(Servs\|Orgs\|Apps)$}+ {**exists** *y* in $(Servs\|Orgs\|Apps)$}+ **Scope**: *Scope expr*? , **Assertion**: *Assertion expr* |	Add contraint: $\bigwedge^{S\|O\|A}_{v_{s_i\|o_i\|a_i}}$ (*Scope expr*)$[y/v^y_{s_i\|o_i\|a_i}]$ $\Rightarrow$ (*Assertion expr*)$[y/v^y_{s_i\|o_i\|a_i}]$ $\bigvee^{S\|O\|A}_{v_{s_i\|o_i\|a_i}}$ ((*Scope expr*)$[y/v^y_{s_i\|o_i\|a_i}]$ $\Rightarrow$ (*Assertion expr*)$[y/v^y_{s_i\|o_i\|a_i}]$ For each constraint $c \in C$ do: $(c)[v^y_{s_i\|o_i\|a_i}/v^x_{s_i\|o_i\|a_i}]$
where E[x/y] means: ' the expression E, but with occurrences of x replaced by y'	

Table 3: Mapping of WS-Governance* elements onto CSPs

the policy found in the second document fragment of the example is not translated into a policy in the GD* document but it represents structural information expressed in the SOA Model section.

Following the mapping described in this section, in the right side of Table 3 the CSP transformation of the GD* is presented.

5.3 Checking for Consistency

Checking a GD ρ written in WS-Governance* for consistency lets us know whether it has internal contradictions or not. The root of the inconsistencies can be: (i) that a policy in ρ is intrinsically inconsistent; (ii) that the set of policies in ρ are inconsistent; or (iii) even when the set of policies in ρ, $P^\rho = \{\rho_1, \dots, \rho_n\}$, are initially consistent, then ρ can be inconsistent due to the additional information added by the SAML SOA state and structure specified in it. For instance, the GD* shown in the first column of table 4 is inconsistent. The cause of the inconsistency is that policies ρ_2='*Critical services availability should be "'24x7' '*' and ρ_3='*Service s_1 availability is "window" if it is consumed by external applications*' are inconsistent, since s_1 is both consumed by application a_2

Sample SGDL4People Document	Corresponding CSP
Governance Document - *Critial Services Management (GD1)*	$\phi = \{V, D, C\}$ where $C = \{C^D \cup C^{SOA} \cup C^{\rho}\}$
Governor: (o_x)	$V = \{V^{Prop} \cup V^W \cup V^{\rho}\}, D = \{D^{Prop} \cup D^{SOA}\}$
Scope:	V^{SOA}=$\{V^{prov} \cup V^{cons} \cup V^{own} \cup V^{supOrg}\}$
STRUCTURE:	$V^{prov} = \{v^{prov}_{s_1}, v^{prov}_{s_2}, v^{prov}_{s_3}, v^{prvO}_{s_1}, v^{prvO}_{s_2}, v^{prvO}_{s_3}\}$
Organization: *E-gov. Org. X* (o_x) **Suborganizations:** o_1,o_2	$V^{cons} = \{v^{cons}_{s_1}, v^{cons}_{s_2}, v^{cons}_{s_3}, v^{cnsO}_{s_1}, v^{cnsO}_{s_2}, v^{cnsO}_{s_3}\}$
Organization: *Department 1* (o_1)	$V^{own} = \{v^{own}_{a_1}, v^{own}_{a_2}, v^{own}_{a_3}\}$
Applications:	$V^{suporg} = \{v^{suporg}_{o_1}, v^{suporg}_{o_2}\}$
Application:*A1 App* (a_1)	$D^{SOA} = \{D^{prov} \cup D^{cons} \cup D^{own} \cup D^{suporg}\}$
Provides: *s1* (s_1)	$, A = \{a_1, a_2, a_3\}, O = \{o_1, o_2\}$
Organization: *Department 2* (o_2)	$C^{soa} = \{C^{Struct} \cup C^{Stat}\}$
Applications:	$C^{Struct} = \{\{v^{prov}_{s_1} = a_1\}, \{v^{prov}_{s_2} = a_3\}, \{v^{prvO}_{s_1} = o_1\}, \{v^{prov}_{s_3} = a_2\}, \{v^{prvO}_{s_2} = o_3\}\}$
Application:*Zeus* (a_2)	$\{v^{prov}_{s_3} = a_2\}, \{v^{prvO}_{s_3} = o_2\}, \{v^{cons}_{s_1} = a_2\}, \{v^{cons}_{s_2} = a_2\}, \{v^{cnsO}_{s_1} = o_2\},$
Consumes: *s1* (s_1) *s2* (s_2) Provides: *s3* (s_3)	$\{\{v^{cnsO}_{s_2} = o_2\}, v^{cons}_{s_3} = \oslash\}, \{v^{own}_{a_1} = o_1\}, \{v^{own}_{a_2} = o_2\},$
Application:*A3 App* (a_3)	$\{v^{own}_{a_3} = o_2\}, \{v^{supOrg}_{o1} = o_x\}, \{v^{supOrg}_{o_2} = o_x\}\}$
Provides: *s2* (s_2)	$C^{Stat} = \{\{v^{Crit}_{s_1} = true\}\}$
STATE:	$D^{own} = \{O, O, O\}, D^{supOrg} = \{O, O\}$
Critical(s_1)=true	$D^{prov} = D^{cons} = \{A, A, A, O, O, O\}$
Vocabulary:	$V^{Prop} = \{v^{Crit}_{s_1}, v^{Crit}_{s_2}, v^{Crit}_{s_3} v^{Avail}_{s_1}, v^{Avail}_{s_2}, v^{Avail}_{s_3}\}$
Property: *Critical* **for** *Services* Type: *boolean*	$D^{Prop} = \{\{true, false\}, \{true, false\}, \{true, false\},$
Property: *Availability* **for** *Services* Type: *enum*	$\{'24x7','office'\}, \{'24x7','office'\}, \{'24x7','office'\}\}$
Domain: *'24x7','office'*	$C^D = \emptyset$ since there is no domain constraints
Policies:	$C^{\rho} = \{C^{\rho_1}, C^{\rho_2}, C^{\rho_3}\}$
Policy *P1* (ρ_1) **forall** *s* in Services	$C^{\rho_1} = \{\wedge^3_{i=1}((v^{crit}_{s_i} = true) \Rightarrow (v^{prvO}_{s_i} = v^{consO}_{s_i}))\} = \{((v^{crit}_{s_1} = true) \Rightarrow (v^{prvO}_{s_1} = v^{cnsO}_{s_1}))\wedge$
Scope: *Critical(s)=true* **Assertion**: *ProviderO(s)=ConsumerO(s)*	$((v^{crit}_{s_2} = true) \Rightarrow (v^{prvO}_{s_2} = v^{cnsO}_{s_2})) \wedge ((v^{crit}_{s_3} = true) \Rightarrow (v^{prvO}_{s_3} = v^{cnsO}_{s_3}))\}$
Policy *P2* (ρ_2) **forall** *s* in Services	$C^{\rho_2} = \{\wedge^3_{i=1}((v^{crit}_{s_i} = true) \Rightarrow (v^{Avail}_{s_i} =' 24x7'))\} = \{((v^{crit}_{s_1} = true) \Rightarrow (v^{Avail}_{s_1} =' 24x7'))\wedge$
Scope: *Critical(s)=true* **Assertion**: *Availability(s)=' 24x7'*	$((v^{crit}_{s_2} = true) \Rightarrow (v^{Avail}_{s_2} =' 24x7')) \wedge ((v^{crit}_{s_2} = true) \Rightarrow (v^{Avail}_{s_2} =' 24x7'))\}$
Policy *P3* (ρ_3) **forall** *a* in Applications	$C^{\rho_3} = \{\wedge^3_{i=1}((s^{cons}_{s_1} = a_i \wedge v^{prov}_{s_1} \neq v^{own}_{a_i}) \Rightarrow (v^{Avail}_{s_1} =' office'))\} =$
Scope: $a \in Consumer(s_1) \wedge Owner(a) \neq Owner(Provider(s_1))$	$= \{((s^{cons}_{s_1} = a_i \wedge v^{prov}_{s_1} \neq v^{own}_{a_1}) \Rightarrow (v^{Avail}_{s_1} =' office')) \wedge ((s^{cons}_{s_1} = a_2 \wedge v^{prov}_{s_1} \neq v^{own}_{a_2}) \Rightarrow$
Assertion: *Availability*(s_1)=*'office'*	$\Rightarrow (v^{Avail}_{s_1} =' office')) \wedge ((s^{cons}_{s_1} = a_3 \wedge v^{prov}_{s_1} \neq v^{own}_{a_3}) \Rightarrow (v^{Avail}_{s_1} =' office'))\}$

Table 4: Sample WS-Governance4People onto CSP mapping

’Zeus’ of department 2 and critical. This information, service consumption and criticality of services are specified by the SOA Structure and State section of the GDs*, thus the corresponding constraint $Availability(s_1) =' 24x7' \wedge Availability(s_1) =' window'$ is obviously unsatisfiable due to the SOA state and structure, not because of an incosistency of policies *per se*, and consequently ρ_2 and ρ_3 are inconsistent in ρ.

Definition Internal Consistency A GD* ρ is said to be consistent iff its corresponding equivalent CSP is satisfiable.

$$consistent(\rho) \Leftrightarrow sat(\psi^{\rho})$$

Definition Consistency. A non empty set of GDs in WS-Governance* $P = \{\rho_1, \ldots, \rho_n\}$ is said to be consistent iff its corresponding equivalent CSPs are simultaneously satisfiable.

$$consistent(P) \Leftrightarrow sat(\bigwedge_{i=1}^{n} \psi^{\rho}{}_{i})$$

As an example, our approach found an additional inconsistence: in the mapping of the GD* shown in Table 4, the corresponding CSP ψ contains among others the following constraints: $\{v_{s1}^{crit} = true \Rightarrow v_{s_1}^{provO} = v_{s_1}^{consO}\}$, $\{v_{s2}^{crit} = true\}$, $\{v_{s_1}^{consO} = o_2\}$, and $\{v_{s1}^{provO} = o1\}$. This set of constraints in unsatisfiable, since $v_{s_1}^{prov} = o_2$ and $v_{s_2}^{prov} = o_1$ are unsatisfiable, and consequently the GD* is inconsistent.

5.4 Prototype implementation

We have developed an implementation of our approach using the Choco constraint solver [9]. This prototype receives two XML documents as input: a WS-Governcance* document ρ^D, and a SAML document that provides governance relevant information Γ^D, simulating a UDDI registry enriched with metadata to support governance policies reasoning. After mapping the ρ^D and Γ^D to the equivalent CSP, our proof-of-concept prototype processes the CSP and returns a report showing the results of the following checks: intrinsic consistency of each isolated policy of ρ^D, consistency of the whole set of policies of ρ^D and the general consistency of ρ^D on Γ^D. Our propotype implementation is available for download at http://www.isa.us.es/GDA.

6 Related Work

Concerning policy definition, Ponder is the pioneer and probably most widely used language. Ponder [3] is a declarative, object oriented language for the specification of management polices in distributed object systems. Additionally, Ponder provides structuring techniques for policy administration in large scenarios and systems. WS-Governance incorporates similar concepts by the explicit declaration of governor and document context, and uses SAML and GAL assertions to model scope. The usage of

WS-Policy as the base policy expression construct, and explicit declaration of governance relevant information sources, makes WS-Governance better suited for SOA governance policies declaration. However, an interesting capability supported by Ponder that we plan to add to WS-Governance in the future is the declaration of policy types as templates. Rei and KAoS [22], both based on semantic web concepts are proposals oriented to the definition of policies for expressing web services capabilities policies. However, is WS-Policy the proposal that has more successful in industrial scenarios, thus we have chosen it for extension in order to define SOA governance policies.

Our proposal provides a richer consistency notion allowing the detection of semantical inconsistences in policies with complex interaction as shown in this paper. General policy conflict analysis is not a novel problem [10], but its application in the context of SOA governance policies in this paper is original. Several approaches have been proposed for policy expression and conflict analysis in the context of network management [20], and security [5], but they are based on Binary Decision Diagrams (BDD), forcing the reasoning with less expressive policies than our CSP based proposal.

7 Conclusions and Future Work

In this paper, based on the specific needs of our case study, a novel XML-based language called WS-Governance has been proposed. WS-Governance takes WS-Policy as a starting point and extends it with two new languages (GAL and SAML), maintaining the compatibility with this standard. The main benefit of WS-Governance is to contextualize policies and integrate governance data sources, enabling effective governance since it provides a formal semantics that supports automated consistency checking. This paper represents a significant improvement in the context of SOA Governance, since a vendor-independent governance language with precise semantics is proposed, allowing the specification of expressive Governance Documents independently of the underlying infrastructure. Furthermore, the GD semantics and consistency operation paves the way for the building of a new generation of governance tools. In this scenario governance policies are created in a distributed, independent but collaborative way, maintaining global consistency. This collaborative process helps governance boards on the creation of the essential normative framework needed for the achievement of the SOA promise.

There are some challenges we have to face in the near feature, namely: i) define extensions of the GD context variability point in order to effectively ensure authentication and authorization of the enacting organization for defining policies and avoid tampering (currently this extension is supported by the language through the variability point but those mechanisms are not specified); (ii) extend WS-Governance to support the definition of policy templates improving reuse and usability; (iii) developing an appropriate tooling for GDs edition; iv) improve WS-Governance and the mapping onto CSP in order to support advanced temporal concerns such as described in [14]; v) carry out a performance analysis of our implementation in order to study the influences of the SOA model, type of governance properties and number and complexity of governance policies.

8 Acnowledgements*

This work was partially supported by the European Commission (FEDER) and Spanish Government under Web-Factories (TIN2006-00472) and SETI (TIN 2009-07366) projects, and by the Andalusian Government under project ISABEL (TIC-2533) and THEOS (TIC-5906).

http://www.micinn.es/

http://www.juntadeandalucia.es/organismos/economiainnovacionyciencia.html

References

[1] Anne H. Anderson. Domain-independent, composable web services policy assertions. In *POLICY*, pages 149–152, 2006.

[2] Jan Bernhardt and Detlef Seese. A conceptual framework for the governance of service-oriented architectures. In *Service-Oriented Computing, ICSOC 2008 Workshops*, pages 327–338. Springer, 2009.

[3] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *POLICY '01: International Workshop on Policies for Distributed Systems and Networks*, pages 18–38. Springer, 2001.

[4] Patricia Derler and Rainer Weinreich. Models and tools for soa governance. In *Trends in Enterprise Application Architecture*, pages 112–126. Springer, 2007.

[5] Hazem H. Hamed, Ehab S. Al-Shaer, and Will Marrero. Modeling and verification of ipsec and vpn security policies. In *ICNP*, pages 259–278, 2005.

[6] Bernhard Hollunder. Domain-specific processing of policies or: Ws-policy intersection revisited. In *ICWS*, pages 246–253, 2009.

[7] L. Frank Kenney and Daryl C. Plummer. Magic quadrant for integrated soa governance technology sets. Technical report, Gartner RAS Core Research, March 2009. Available at http://mediaproducts.gartner.com/reprints/oracle/article65/article65.html.

[8] Kostas Kontogiannis, Grace A. Lewis, and Dennis B. Smith. A research agenda for service-oriented architecture. In *SDSOA '08: Proceedings of the 2nd international workshop on Systems development in SOA environments*, pages 1–6, New York, NY, USA, 2008. ACM.

[9] Francois Laburthe, Narendra Jussien, Guillaume Rochart, Hadrien Cambazard, Charles Prud'homme, Arnaud Malapert, and Julien Menana. Choco, java library for constraint satisfaction problems (csp), constraint programming (cp) and explanation-based constraint solving (e-cp). One Source Software (BSD license) available online.

[10] Emil Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Trans. Software Eng.*, 25(6):852–869, 1999.

[11] Eric A. Marks. *Service-Oriented Architecture Governance for the Services Driven Enterprise*. John Wiley & Sons, 2008.

[12] Carlos Müller, Manuel Resinas, and Antonio Ruiz-Cortés. Explaining the non-compliance between templates and agreement offers in ws-agreement*. In *Proc. of the 7th International Conference on Service Oriented Computing (ICSOC)*, volume 5900, pages 237–252, Sweden, Stockholm, Nov 2009. Springer Verlag.

[13] Carlos Müller, Antonio Ruiz-Cortés, and Manuel Resinas. An initial approach to explaining sla inconsistencies. In *6th. Int. Conf. on Service-Oriented Computing (ICSOC)*, volume 5364 of *LNCS*, pages 394–406, Sidney, Australia, Dec 2008. Springer Verlag.

[14] Carlos Müller, Octavio Martín-Díaz, Antonio Ruiz Cortés, Manuel Resinas, and Pablo Fernandez. Improving temporal-awareness of ws-agreement. In *ICSOC*, pages 193–206, 2007.

[15] M. P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-oriented computing: State of the art and research challenges. *IEEE Computer*, 40(11):38–45, November 2007.

[16] J. A. Parejo, Pablo Fernández, and Antonio Ruiz-Cortés. Towards automated sla-based governance policy enforcement. In *International Joint Conference on Service Oriented Computing (ICSOC)*, 2009.

[17] José Antonio Parejo, Pablo Fernandez, and Antonio Ruiz-Cortés. Soa governance languages schemas. including schemas for soa governance description language (sgdl), soa modelling languange (saml), and governance assertions language (gal). Online XML Schemas, 06 2010.

[18] David Peterson, Shudi (Sandy) Gao, Ashok Malhotra, C. M. Sperberg-McQuee, and Henry S. Thompson. W3c xml schema definition language (xsd) 1.1 part 2: Datatypes. W3C Working Draft, 12 2009.

[19] Antonio Ruiz-Cortés, Octavio Martín-Díaz, Amador Durán, and Miguel Toro. Improving the automatic procurement of web services using constraint programming. *International Journal of Cooperative Information Systems*, 14(4):439–467, Dec 2005.

[20] Taghrid Samak, Ehab Al-Shaer, and Hong Li. Qos policy modeling and conflict analysis. In *POLICY*, pages 19–26, 2008.

[21] T. G. J. Schepers, M. E. Iacob, and P. A. T. Van Eck. A lifecycle approach to soa governance. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1055–1061, New York, NY, USA, 2008. ACM.

[22] A. Uszok, J.M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, and S. Aitken. Kaos policy management for semantic web services. *Intelligent Systems, IEEE*, 19(4):32–41, Jul-Aug 2004.

[23] Antonio Vallecillo. A journey through the secret life of models. Schloss Dagstuhl - Leibniz-Zentrum fÃ¼r Informatik, 2008.

[24] Asir S Vedamuthu, David Orchard, Frederick Hirsch, Maryann Hondo, Prasad Yendluri, Toufic Boubez, and Ümit Yalçinalp. Web services policy 1.5 - attachment. W3C Recommendation, 09 2007.

[25] Asir S Vedamuthu, David Orchard, Frederick Hirsch, Maryann Hondo, Prasad Yendluri, Toufic Boubez, and Ümit Yalçinalp. Web services policy 1.5 framework. W3C Recommendation, September 2007.