Journal of Universal Computer Science, vol. 22, no. 8 (2016), 1048-1071
submitted: 3/9/15, accepted: 29/7/16, appeared: 1/8/16 © J.UCS

IntelliGOV - A Semantic Approach for Compliance Validation of Service-Oriented Architectures

Haroldo Maria Teixeira Filho
(Federal University of the State of Rio de Janeiro (UNIRIO), Rio de Janeiro, Brazil
haroldo.filho@uniriotec.br)

Leonardo Guerreiro Azevedo
(IBM Research, Rio de Janeiro, Brazil
LGA@br.ibm.com
Federal University of the State of Rio de Janeiro (UNIRIO), Rio de Janeiro, Brazil
azevedo@uniriotec.br)

Sean Wolfgand Matsui Siqueira
(Federal University of the State of Rio de Janeiro (UNIRIO), Rio de Janeiro, Brazil
sean@uniriotec.br)

Abstract: Organizations are adopting Service-Oriented Architecture (SOA) to increase operation's efficiency and flexibility. To accomplish these goals, it is necessary to ensure that the architecture and its evolution are compliant with business goals, best practices, legal and regulatory requirements. However, the diversity of domains and stakeholders involved in SOA solutions demands complex and expensive work to validate the architecture compliance. Hence, it can result in high costs and low quality assessment if the organization does not use an effective approach in this scenario. In addition, it would be important to consider standards and open solutions in order to promote interoperability and reuse of available tools, making it easier to spread throughout the organizations. We propose intelliGov, an architecture that aims to solve these problems by using ontologies, semantic rules and queries in order to simplify the compliance validation process. The architecture employs open standards – OWL, SWRL and SQWRL – in its implementation. A case study, executed in a global energy company that is currently adopting SOA, demonstrates gains in quality and costs of the compliance assessment process using the proposed architecture.

Keywords: Service-Oriented Architecture (SOA), Governance, Ontology, Semantic Rules, Semantic Queries.
Categories: H.0, H.3.5, H.4, I.2.4, M.2, M.9

1 Introduction

Service-Oriented Architecture (SOA) is an approach for building applications by the interconnection of loosely coupled software modules into on-demand business processes [Elgammal et al., 2014]. This goal is achieved by discovering, invoking and composing distributed services to accomplish a task [Papazoglou et al., 2007]. It aims to reduce costs and schedules of applications development through composition of existing services, fostering reuse of existing IT assets [Erl, 2005].

However, in a SOA environment the number of flexible parts of the architecture and of involved independent stakeholders increases, resulting in a higher complexity than traditional approach [Niemann et al., 2010] [Holanda et al., 2009]. It leads to several problems, such as: (i) in the development of several versions of the same service, reducing reuse [Niemann et al., 2010]; (ii) difficulty to expand the architecture to a corporate and global scale, due to the distributed nature of SOA and the variations in the regulatory aspect [Hsiung et al., 2012]; (iii) change management issues in an environment that depends on distributed components and stakeholders [Schepers et al., 2008].

Academy [Hojaji and Shirazi, 2010] [Joachim et al., 2013] [Stantchev and Stantcheva, 2012] and industry [Bennett, 2012] [Brown et al., 2006] points to SOA Governance as a solution for those problems. As described in [Janiesch et al., 2009], SOA Governance is the establishment of structures, processes, policies and metrics to ensure the adoption, implementation, operation and evolution of a SOA aligned with business objectives and compliant with laws, regulations and best practices.

In addition, as stated in [Niemann et al., 2010], the primary goal of SOA Governance is the compliance to intra-company, normative and legal standards. There is also an increasing pressure for regulatory compliance in organizations, due to high-profile bankruptcy cases, safety mishaps and the financial crisis [Rodríguez et al., 2013]. Hence, compliance is a critical aspect in the SOA domain.

A common view of SOA Governance processes was provided in [Teixeira Filho and Azevedo, 2014]. It consolidates industry and academy approaches in 51 processes. These processes require knowledge from several areas, ranging from technical aspects (e.g., service development and monitoring) to strategic activities (e.g., financial models and definition of organizational structures). This holistic characteristic of SOA makes the compliance validation activity a challenge due to the demand of knowledge derived from several domains, which is difficult to formalize [Tran et al., 2011]. It also makes compliance a complex and expensive activity due to the necessity to employ experts to audit compliance rules, collect, analyse and report results [Rodríguez et al., 2013].

Therefore, it is necessary to consider domain representation and tools for supporting the compliance validation. Although there are some proposals for dealing with compliance assessment problems in SOA [see Section 2], they require the development of specific components and extensions which make their use more complex or they are not adequate to describe real world SOA governance. Then, there is a need for supporting standards and available tools while also improving the validation results (in terms of correctness and effort) in more complex real world scenarios.

The work presented in this article proposes intelliGOV, a new approach for compliance evaluation in SOA environments based on ontologies, semantic rules and queries. IntelliGov uses Ontology to formally represent the domain knowledge and semantic rules and queries to express the policies that describe regulatory requirements. A software architecture is proposed to handle these elements, providing mechanisms to extract information from the SOA environment, load this data as instances in the ontology and evaluate compliance using rules described in the ontology. The proposal was implemented and evaluated through a case study conducted in a global energy organization. The results evidence the use of intelliGOV effectively reduces efforts and errors in the compliance validation process. Even though some problems might be similar in Information Technology (IT) governance as a whole, we focus on SOA Governance and, therefore, possible generalization of the proposed approach and its validation on a broader scenario is out of the scope of this work.

The remainder of this work is organized as follows. Section 2 describes related work. Section 3 presents intelliGov approach. Section 4 describes a case study used to validate the solution. Finally, Section 5 presents the conclusions and future work.

2 Related Work

This section reviews related work regarding SOA governance and the compliance assessment problem in this context.

2.1 SOA Governance

The main goal of this paper is to propose a compliance mechanism to simplify SOA governance establishment, which according to Janiesch *et al.* [2009] is the establishment of structures, processes, policies and metrics to ensure the adoption, implementation, operation and evolution of a SOA aligned with business objectives and compliant with laws, regulations and best practices. To reach this goal, several implementation frameworks were proposed to structure SOA governance mechanisms implementation.

Niemann *et al.* [2010] propose model elements to be used for SOA Governance establishment:

- Organization Entities: comprise processes, roles and responsibilities necessary to implement and operate the governance mechanism;
- Governance policies: establish the expected behaviour of architectural elements;
- Best practices catalogue: provides guidelines and acts as a base for future policies and process improvements;
- Compliance check mechanisms: constantly verify adherence to policies, checking if the resulting architecture is aligned to business goals, and to legal and regulatory requirements;
- Maturity evaluation: measures the evolution of the governance mechanisms.

These five elements are defined considering organization's goals and they act on processes and technologies used by the organization to implement and operate the SOA environment. Niemman *et al.* [2010] do not define which processes are necessary to implement SOA governance.

Several works define frameworks specifying the necessary processes to implement, operate and govern a SOA environment. Janiesch *et al.* [2009] propose a framework of processes and compare the proposal with ITIL framework [Taylor *et al.*, 2007]. Hojaji *et al.* [2010] propose a model based on COBIT [ITGI, 2007]. The Open Group [2009] proposes a model that divides processes in two categories: (i) Governed processes category: contains the processes to develop and sustain the SOA environment and are subjects of the SOA Governance Mechanism; and, (ii) Governance processes category: controls policies, compliance evaluation and result reporting applied to the governed processes. Bennet [2012] and Brown [2006] also propose models to implement, operate and govern a SOA.

To compare these models, Teixeira Filho and Azevedo [2014] analysed the proposed governance models and listed the suggested processes of each approach, aiming to identify the processes more frequently used and eventual divergences. The result was the design of a common model, composed by 51 processes to implement SOA governance. Those processes can be classified in four different groups of processes:

- Strategy group: contains processes to define objectives, establish and measure goals, define financial models for service charging and investment, and organizational structure definition;
- Compliance group: establishes mechanisms to define and standardize policies, auditing and exception handling due to non-conformities;
- Execution group: includes processes to implement SOA, considering service and application development, project and portfolio management;
- Support group: includes processes to deal with change management, communication, training and resource development, monitoring and problem and incident management.

Considering those processes, it is possible to identify a diversity of knowledge domains in SOA governance, ranging from technical aspects, like versioning and system monitoring, to management aspects, like goal and organizational structure definition. Niemann *et al.* [2010] also indicates this multidisciplinary approach, defining nine possible domains for SOA governance policies: architecture, project management, finance, service operations, data standards, asset management, technology, security and organization. Zhou *et al.* [2010] state SOA governance mechanisms must deal with heterogeneity and must be able to deal with several types and granularities of policies. Hence, the definition of policies for a SOA governance compliance mechanism demands tools able to represent knowledge from several domains and different types.

2.2 Compliance assessment in SOA

Several authors address the compliance assessment problem in SOA. This section lists these works and their limitations.

The use of the Service Modelling Language (SML) [Pandit *et al.*, 2009] to describe the SOA elements (e.g., services and systems) and ISO Schematron [Jelliffe, 2002] to describe, as schema constraints, the policies that the elements must follow is proposed in [Zhou *et al.*, 2010]. A set of tools verifies if the architecture elements described in SML are valid considering the Schematron restrictions. It added specific components and extensions to the standards to deal with the non-capability of SML and Schematron to work with forward chaining inference and to handle performance issues. These specific components are more complex and difficult to maintain than the use of standards and open tools.

Some works [Birukou *et al.*, 2010] [Tran *et al.*, 2011] propose the use of Complex Event Processing (CEP) [Luckham, 2002] techniques to evaluate compliance. These approaches collect events from the SOA environment, and evaluate compliance using this information through an event correlation mechanism based on rules that specifies the desired behaviour. In this solution, the rules have to contain all the necessary knowledge of the policies and the SOA environment for validation. Considering the cross-domain characteristic of SOA, the use of a CEP approach can lead to rules with high complexity, increasing the chance of errors in the compliance assessment process.

The following concerns were identified in the CEP approach [Tran *et al.*, 2012]: lack of a common vocabulary and difficulty to consolidate data. To solve these problems, they improved the CEP approach with Model Driven Architecture (MDA) [Mellor *et al.*, 2002] practices. They included mechanisms for aggregating the policy definitions into a model describing the service architecture and combining these pieces of information to generate service architecture elements and CEP rules. However, this solution increases the complexity of the CEP proposals and creates architectural concerns. Following the MDA approach requires to generate the rules and the related components of the service architecture whenever a stakeholder alters a policy.

The use of a data warehouse to establish an integrated database with a schema representing SOA concepts (e.g., services and applications) and data representing the existing elements of the architecture is proposed in [Rodriguez *et al.*, 2013]. This solution uses queries to verify compliance. The same problem of the CEP solution exists in this scenario: it uses queries and views in the data warehouse environment to express all the necessary knowledge to validate the SOA policies, leading to complex models.

Considering these works, it is necessary to reduce complexity by aggregating knowledge in the compliance validation process. To ensure the applicability in organizations, the solution needs to be simple, with few building blocks for implementation. In addition, as SOA deals with interoperability of several heterogeneous environments, an approach based on open standards can simplify the integration issues.

Some solutions propose the use of ontologies and rules to deal with compliance validation, making the process of heterogeneous knowledge representations and interoperability between environments easier. The use of ontologies to describe an IT governance domain and the use of axioms and rules in the ontology to describe policies, submitting the items to an inference engine, is proposed in [Spies, 2012]. This approach allows capturing the knowledge of the domains involved in SOA in a formal language and uses inferences to validate compliance.

An approach for using OWL [Hitzler *et al.*, 2012] ontologies to represent domain knowledge and SWRL [Horrocks *et al.*, 2004] to express policies in the SOA domain was proposed in [Teixeira Filho *et al.*, 2014]. However, in both approaches ([Spies, 2012] and [Teixeira Filho *et al.*, 2014]), the use of ontologies and SWRL are not enough to describe real world SOA governance policies. An example is a rule like "minor versions of a service must implement the same service interfaces". To deal with this kind of rule, it is necessary to compare the set of interfaces related to each service version and assert that their contents are equal. OWL and SWRL does not compare an entire set of related elements, demanding operations to handle sets and aggregations. The work presented in this article evolves the ontology based approaches, increasing the types of policies addressed by these solutions, providing a solution to the set manipulation problem, by use of a query language attached to the ontology and rules language.

3 intelliGov – An Approach for Compliance Validation of Service-Oriented Architectures

This section presents the intelliGov conceptual view and details to implement its required components.

3.1 intelliGov Conceptual View

The intelliGOV approach is based on a modular architecture that combines ontologies, semantic rules and queries with an inference engine and data extraction tools to provide an accurate vision of the compliance state of a SOA.

An ontology is an explicit representation of a conceptualization [Gruber, 1993], constituted by objects, concepts and other entities assumed to exist in a domain and the relationship between them. The use of an ontology leads to a unique vocabulary definition capable to represent formally and unambiguously the SOA domain concepts. It guarantees correct interpretation of data by human or computational agents. Hence, it reduces the necessity of experts to define or execute compliance assessment, providing a unified vocabulary to users and computational agents to specify and infer compliance.

For policy definition, it is important to use a mechanism capable of dealing with concepts expressed in the ontology. It is also necessary to consider all the knowledge expressed in the form of axioms in the ontology to simplify the policy description through the reuse of this logic. Finally, both the ontology axioms and the policy specification have to be machine-interpretable in order to automate compliance assessment, thus reducing its execution time and errors.

In order to deal with these aspects, we propose the use of rules and queries to describe governance policies. In this work, rules are assertions that can describe enterprise behaviour, able to express organizational rules, company policies, external regulations and standards [Bajec and Krisper, 2005]. The use of rules combined with queries enhances the expressiveness of the solution, allowing complex operations like aggregations and cardinality verification, which cannot be expressed as rules.

Figure 1 presents the proposed architecture. It is divided in five modules (Data extractor, Ontology, Semantic Rules/Queries, Inference Engine and User Interface). This modular architecture allows customizations and the use of different technologies according to the application environment. The approach to design and implement the architecture focus on the establishment of a set of tools that make easier the manipulation of ontologies, rules and queries.

A Data Extractor module collects data describing the elements that composes the architecture, reading this information from the tools that supports the SOA environment. Afterwards, it converts this data into instances of an ontology, creating objects representing elements like the services that exists in the organization and systems that consume information using these services. Additional descriptive data that cannot be extracted automatically might be inserted in the ontology by a user interface. This ontology contains classes and relationships describing the SOA domain, e.g. services, service contracts, systems and other architecture elements.

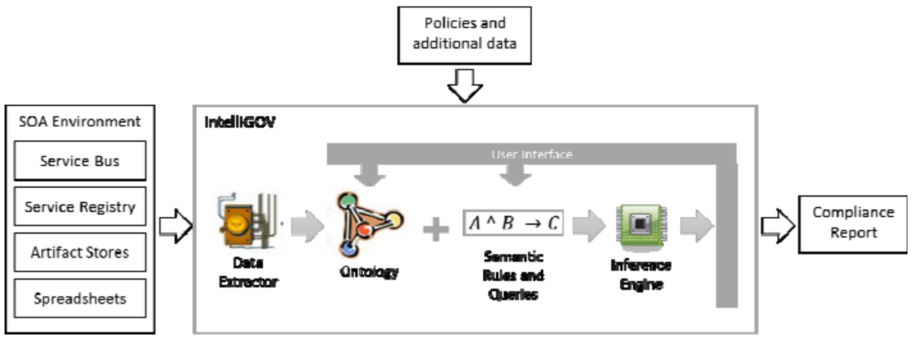

Figure 1: intelliGOV conceptual view

By using the user interface, governance teams can define policies for SOA governance describing them as semantic rules that are added to the ontology. Finally, whenever is necessary to generate a compliance report, the ontology content combined with the semantic rules are submitted to an inference engine, that verifies which instances are compliant with the rules and presents the result on the user interface.

3.2 intelliGOV components

The intelliGOV architecture requires a set of tools and technologies. Our implementation employs open standards, aiming interoperability and easiness of use.

The approach uses the Ontology Web Language (OWL) [Hitzler *et al.*, 2012] for ontology description. OWL is an open standard, recommended by W3C and has a large set of available tools for modelling, programming and reasoning. OWL is based on XML technologies, which are also usual in a SOA environment, reducing the learning curve of the approach.

We used Semantic Web Rule Language (SWRL) [Horrocks *et al.*, 2004] to describe policies rules. SWRL is an open standard to describe rules using classes and properties represented in an OWL ontology. It exploits all the knowledge formally expressed in ontologies, considering vocabulary, axioms and all types of relationships defined in OWL.

Finally, we used Semantic Query-Enhanced Web Rule Language (SQWRL) [O'Connor and Das, 2009] to express the queries. It exploits knowledge described in the OWL ontology and associated rules in SWRL. It adds the capability to aggregate data, obtaining measures like average and counting of elements, classify and order data, and evaluate cardinality in relations. It fosters the increasing of the solution expressiveness.

We used the components described in Figure 2 to implement the intelliGOV prototype according to the conceptual view presented in section 3.1 (Figure 1).

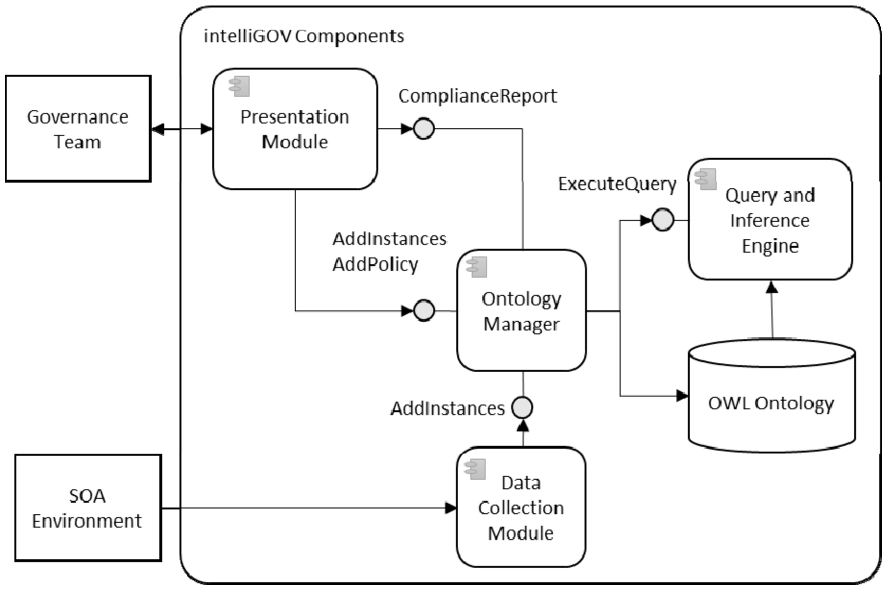

Figure 2: intelliGOV architecture components

The users of this architecture are members of SOA governance teams, responsible for defining policies and evaluating compliance. These users are able to execute these tasks through a presentation module that accesses an ontology manager that creates rules and instances on the ontology. The ontology manager accesses a query and inference engine to evaluate the rules and classify the elements in the ontology. Using these components, the users add rules and queries representing the policies in the ontology and, using the inference engine, executes queries and inferences to evaluate compliance. Finally, the Data Collection Module uses the ontology manager to create instances to represent the SOA environment objects. It gathers data from organization SOA tools, e.g., service bus, service repository or artefact repository.

The modules were implemented in Java and deployed on the JBoss application server. The Protegé API [1]was used to access the ontology. It provides an open API to the ontology management component. Drools[2] is used as query and inference engine, due to it compatibility with SQWRL/SWRL and with the Protégé API.

4 The Case Study

Although the proposal followed a design science research [Hevner *et al.*, 2004], in the evaluation we executed a case study in a global company of the energy sector. In this section, we present the case study and discuss its results.

[1] Protegé API – protegewiki.stanford.edu/wiki/ProgrammingWithProtegeOWL
[2] Drools rules engine – www.drools.org

4.1 Case study scenario

The case study was executed in a Brazilian energy company with global operations, currently engaged in a SOA initiative. The company has 118 services implemented in different technologies, including ABAP, Java, Lotus Notes, and NET. These services are published in Oracle Service Bus and documented in Oracle Enterprise Repository. A Clear CASE repository stores the source code versions.

The governance and publication of services in the service bus is executed by a team named as Integration Competency Center (ICC). Several teams develop and maintain services, using both traditional and agile development processes. There are internal development teams distributed in six different states of the country and outsourced development executed by external software factories.

A case study considering only one case is applicable to several situations, such as the scenario of this work [Yin, 2009]. As the company acts in a global scale, has a broad spectrum of technologies used in other organizations and must interoperate with other enterprises, we can consider this scenario as representative.

4.2 Definition of the policies for the case study

The first step of the case study was to raise governance processes that require improvements according to organization business challenges. Then it would be possible to choose important processes considering the literature and the impact on the organization, restricting the scope of the case study for governance policies. Two senior members of the ICC accomplished this task. They both have worked more than five years within the organization and they have more than five years of experience in SOA. In order to map the organization governance processes to the proposals of the literature, the ICC member chosen the processes considering a model that consolidates the main proposals of SOA governance processes recommended by industry and academia [Teixeira Filho and Azevedo, 2014]. As a result, they selected service modelling and version control processes, due to its complexity and importance for the organization.

Security and reuse are the main challenges for the service modelling process. The organization classifies information type in security levels, distributed in five ordered classes (A, B, C, D and E, with A being the less strict category). These levels specify constraints and requirements to store data in systems and transport it among applications. This classification was reviewed in 2013 in order to comply with a new transparency law defined by the Brazilian government resulting in new security levels that now coexist with the legacy classification. The levels names were changed to Public, 1, 2, 3 and 4, representing respectively the former levels A, B, C, D and E. Therefore, it brought the requirement for considering this mapping in service modelling.

The second challenge deals with reuse. Developers in the company need to know the existing set of services to include in their SOA solutions. Although documentation is explicit in Oracle Enterprise Repository, a new classification schema was required to enhance searches. The solution defined by the corporate architecture team was the definition of a taxonomy to classify data based on Subject Areas, representing the business areas related to the data. The subject areas are organized in a hierarchical way, reflecting the organization hierarchy. This classification scheme is currently under development and several information and services are not classified yet. Therefore, every service must be associated to the correct subject area, aiming to optimize the search for reuse.

Finally, regarding version control, several existing services are variations of the same function that coexists due to the lack of a defined versioning policy, leading to multiplication of unnecessary service endpoints. There is also a history of operational problems due to contract changes in services already consumed by applications. To solve this problem, the ICC proposed a version control policy based on two elements: service releases and variants. Service releases are versions of the same service but with different interface. They potentially affect the behaviour of existing consumers. Each new service release generates a new asset in the production environment, with a new endpoint. The legacy service releases may coexist to fulfil legacy service contracts and they have to become active until these contracts expire. Variants are service versions that do not have different interfaces, but changes in the service functionality. Usually variants result from bug fixes or performance improvements.

Considering the scenario, the specialists described policies for these processes. They also prioritized the policies in three levels of importance (Low, Medium and High), informed if automatic verification mechanisms are possible and if evidences were available from automatic collection. The results were filtered, considering only high priority and automatically verifiable policies with existing evidences to validation, leading to a set of policies, described on Table 1.

Policy	**Policy Domain**	**Policy Description**
1	Service Modelling	Every service must be classified accordingly to the corresponding subject area of the information handled by the service.
2	Service Modelling	Every service must be classified accordingly to the corporate information security levels of the organization.
3	Service Modelling	Every service that handles sensitive information must use channel cryptography to transport data.
4	Service Versioning	All the variants of a service release must have the same input and output interfaces.
5	Service Versioning	Service releases of a service must have different input or output interfaces.
6	Service Versioning	Only one service variant of a service release must be active on the service bus.
7	Service Versioning	Every service release that has an active service contract must be active in the service bus.

Table 1: Policy set for the case study

4.3 Manual evaluation

The research team distributed the policies to seven other analysts with different skills and experience. All analysts were ICC members with experience in software development. In order to characterize the analysis, they were classified in two dimensions: the first measuring the experience in the organization, obtained in previous projects in the company; the second, based on the experience in service-oriented architectures. Table 2 presents the distribution of the experience of the team: rows represent dimensions, and columns represent ranges of number of years of experience. Cells represent the quantity of people related to a type of knowledge against the years of experience.

Type of Expertise	**Years of Experience**		
	1 - 2	3 – 5	5+
Organization	4	1	2
SOA	2	2	3

Table 2: Distribution of experience of the team

Each member evaluated a number of services and checked the compliance, describing any reasons identified for non-compliance. The number of evaluated services per person varied between two to five services according to the complexity of the service interfaces and the number of existing versions, leading to an equal workload to each team member.

In order to define a validation template, the research team evaluated the policies, and a member of ICC, with more than five years of experience both in the organization and in SOA, validated this work.

During the manual execution, 25 services were evaluated, corresponding to approximately 21% of the services in the organization. The services were selected randomly and the quantity was selected in a way that each analyst would expend a two hours of work, without affecting their other activities. Although it does not cover all the services, it covers all the technologies used in the organization, including scenarios of communication with other companies or agencies, allowing the evaluation of external providers' scenario, and data access services based on legacy systems.

After the evaluation, the results expressed by each analyst were compared with the template. For each divergence, an evaluation error was counted. The quantity of errors was divided by the total number of evaluations, obtaining an error rate per policy. The time spent by each analyst was also measured, leading to a total time spent to validate the 25 services and an average time per policy per service.

In this time measurement, it was included the time to find and obtain the required information, the time to take the decision of compliant or not, and the time to register the result, emulating the same steps that intelliGOV does.

4.4 Ontology and Rules definition

The policy set defined for the case study acted as a base to model the ontology for our proposed solution, using an adaptation of the 101 methodology [Noy and McGuinness, 2001]. This methodology was selected due to its simplicity to allow the validation of the proposed approach.

The resulting ontology is presented in Figure 3, and described as follows. It is based on the Service-Oriented Architecture Ontology Version 2.0 [The Open Group, 2014], considering versioning and specific elements from the organization. Classes in white are reused from the Open Group ontology and classes in grey were created to fit the requirements of this case study (*i.e.*, they were not present in Open Group ontology). For any other enterprise interested in adopting the proposed approach, it would be possible to reuse the Open Group ontology and/or the case study ontology as well as any other related ontology. The aim of the ontology employed in our case study was to analyze the applicability of the proposed approach and, therefore, a reference ontology for SOA governance was out of scope of this work.

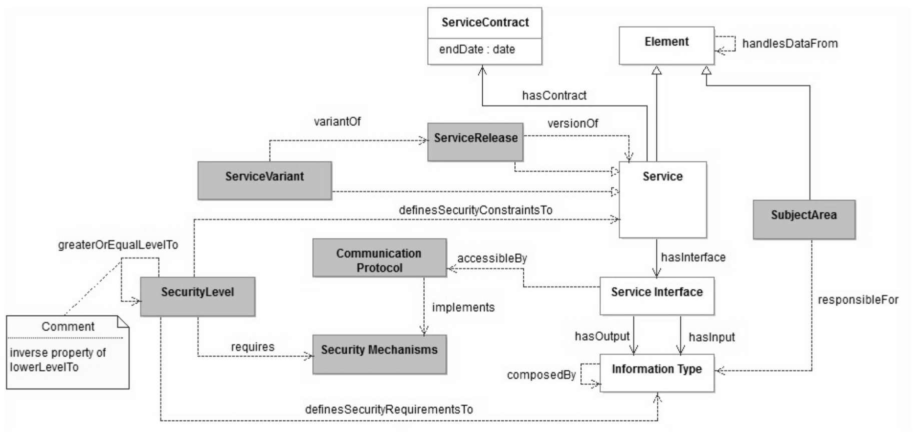

Figure 3: Case study ontology

Element represents any item of the SOA environment. It is specialized in **Services** that represents functionalities available in the organization. **Service Contract** establishes the agreement of a service use. A **Service** has a **Service Interface** that requires (*hasInput*) and provides (*hasOutput*) data from different types (**Information Types)**. This data can be atomic or complex (represented by the **composedBy** object property)

The extensions to the Open Group ontology created to handle the organization specific requirements were the following.

First, to deal with information classification, a class named **SubjectArea** was included to represent organizations' information areas. **SubjectArea** is a subclass of **Element**. **SubjectAreas** classifies **Information Types** by the object property **responsibleFor**. **Element**'s object property **handlesDataFrom** represents the possibility of data manipulation from any element.

To handle security, two more classes were included: **Communication Protocol**, representing possible protocols, like HTTP and JMS, used to access service interfaces; and **Security Mechanism**, representing mechanisms used to ensure security, like channel and content cryptography. In both cases, object properties were added: **accessibleBy** to relate **Service Interface with Communication Protocol**; and, **implements** to relate **Communication Protocol** with **Security Mechanism**. **Security Level** class was created to represent the security levels of the organization. It is related to **Service** by a **definesSecurityConstraintsTo** object property and to an

Information Type by the **definesSecurityRequirementsTo** object property. To express the ordering of the security levels, an object property named **greaterOrEqualTo** and its inverse property **lowerLevelTo** represent the ordering between the security levels.

Finally, to deal with version management, the **Service Release** class was included to represent releases of a service that exists in the organization. On the other hand, **Service Variant** was included to represent the concept of variant proposed by the ICC, relating to the service release by a **variantOf** object property. They are both services, inheriting relations to service interfaces, allowing validation of the versioning policies.

A SQWRL statement was created for each policy. We used ontology terms to build a logical expression representing the policy statement that would result in TRUE whenever all policy conditions are met and FALSE otherwise. To illustrate this process, consider the policy 1, presented below, and the steps for creating the corresponding SQWRL as follows.

Every service must be classified accordingly to the corresponding subject area of the information handled by the service

The first step to create the SQWRL statement corresponds to the definition of the ontology elements to be used by the policy. The first element is a service ***s***. To define the information handled by the service using the ontology terms, we must consider its service interface ***si***, and the data types of the inputs and outputs of the service interface, denoted as ***input*** and ***output*** *InformationType*. Finally, we have to compare the service subject area ***saServ*** to input and output service areas*,* (***saInput*** and **saOut**, respectively) that composes the service interface. With these definitions, the first part of the statement is defined as:

soa:Service(?s)Λ soa:ServiceInterface(?si)Λ soa:InformationType(?input) Λ soa:InformationType(?output) Λ igov:SubjectArea(?saServ) Λ igov:SubjectArea(?saInp) Λ igov:SubjectArea(?saOut)

The second step corresponds to the establishment of the relations between the elements, by setting the object properties as follows:

- **Service** and its **Subject Area** are related by a *igov:handlesDataFrom* property;
- **Service** and its **Service Interface** are related by a *soa:hasInterface* property;
- **Service Interface** is related to inputs and outputs **Information Type** by *soa:hasInput* and *soa:hasOutput* properties;
- **Subject Areas** of inputs and outputs are related by *igov:isResponsibleFor* property.

Considering these elements, the second block of the statement can be written as:

igov:handlesDataFrom(?s, ?saServ) Λ soa:hasInterface(?s, ?si) Λ soa:hasInput(?si, ?input) Λ soa:hasOutput(?si, ?output) Λ igov:responsibleFor(?saInp, ?input) Λ igov:responsibleFor(?saOut, ?output)

The third step corresponds to the creation of sets that group subject areas related to the service, and its inputs and outputs while also consolidates these groups by service, providing, for each service, the subject areas related to the service itself and the data used in the service interface. The operator *sqwrl:makeSet* builds the set and the operator *sqwrl:groupBy* groups the set by service, allowing the creation of three sets: *setSubServ*, representing the subject areas related to the service, and *setSubInp* and *setSubOut*, respectively related to inputs' and outputs' subject areas. Considering these terms, the third block of the expression can be described as follows.

sqwrl:makeSet(?setSubServ,?saServ) Λ qqwrl:makeSet(?setSubInp, ?saInp) Λ sqwrl:makeSet(?setSubOut, ?saOut) Λ sqwrl:groupBy(?setSubServ, ?s) Λ sqwrl:groupBy(?setSubInp, ?s) Λ sqwrl:groupBy(?setSubOut, ?s)

Finally, the last block of the expression compares, service by service, the content of the sets *setSubServ* and the sets *setSubInp* and *setSubOut*, considering true only the services where all elements of the input/output sets are contained in the subject areas listed in the service set. Initially, the operator *sqwrl:equal* was used to try to compare the sets. However*, sqwrl:egual* operator does not consider the grouping operation. To solve this problem, we calculated the difference between the sets using the *sqwrl:difference* operator and checked if the size of the resulting set was equal to zero, using the *sqwrl:size* and the *swrlb:equal* operators. The resulting block was defined as follows.

sqwrl:difference(?dif1, ?setSubOut, ?setSubServ) Λ sqwrl:difference(?dif2, ?setSubInp, ?setSubServ) Λ sqwrl:size(?siz1, ?dif1) Λ sqwrl:size(?siz2, ?dif2) Λ swrlb:equal(?siz1, 0) Λ swrlb:equal(?siz2, 0)

The combination of the four terms leads to a logical instruction that results TRUE whenever a service is compliant and FALSE otherwise. The result of the query of the services **s** that are compliant with the definition is obtained using the *sqwrl:select* operator. Combining all these elements, the final expression is described as follows.

soa:Service(?s)Λ soa:ServiceInterface(?si)Λ soa:InformationType(?input) Λ soa:InformationType(?output) Λ igov:SubjectArea(?saServ) Λ igov:SubjectArea(?saInp) Λ igov:SubjectArea(?saOut) Λ igov:handlesDataFrom(?s, ?saServ) Λ soa:hasInterface(?s, ?si) Λ soa:hasInput(?si, ?input) Λ soa:hasOutput(?si, ?output) Λ igov:responsibleFor(?saInp, ?input) Λ igov:responsibleFor(?saOut, ?output) Λ sqwrl:makeSet(?setSubServ,?saServ) Λ qqwrl:makeSet(?setSubInp, ?saInp) Λ sqwrl:makeSet(?setSubOut, ?saOut) Λ sqwrl:groupBy(?setSubServ, ?s) Λ sqwrl:groupBy(?setSubInp, ?s) Λ sqwrl:groupBy(?setSubOut, ?s) Λ sqwrl:difference(?dif1, ?setSubOut, ?setSubServ) Λ sqwrl:difference(?dif2, ?setSubInp, ?setSubServ) Λ sqwrl:size(?siz1, ?dif1) Λ sqwrl:size(?siz2, ?dif2) Λ swrlb:equal(?siz1, 0) Λ swrlb:equal(?siz2, 0) → sqwrl:select(?s)

The same process was applied to the other policies defined by the ICC team. Some of the queries demanded the definition of other SWRL rules in the ontology to establish OR relations and optimize memory usage by reducing the number of elements handled by the statement. Table 3 presents these rules.

To implement the policies, it was necessary to define what characteristics of a service handle secure information. After consulting experts from the organization, it was identified that these services are classified as security level 2 or higher in the security classification level. Therefore, rule 1 checks if the service is above level 2 and if it uses Channel Cryptography. Rule 2 verifies if the service does not handle secure information, making it compliant whatever security mechanisms is used. To group the services that complies to one of the two assertives, a new class denoted **CompliantSecurityService** was created, allowing the implementation of an OR clause.

Rule 3 evaluates if a service contract is inactive, classifying it in a class denoted **InactiveContracts**. The execution of this rule reduces the number of service contracts evaluated on policy 7, reducing evaluation time and memory consumption.

Rule	**SWRL Rule Statement**	**Objective**
1	igov:definesSecurityConstraintsTo(?sl, ?s) ∧ igov:lowerLevelTo(?sll, ?sl) ∧ sameAs(?sll, igov:2) ∧ soa:hasInterface(?s, ?si) ∧ igov:accessableBy(?si, ?cp) ∧ igov:implements(?cp, igov:ChannelCriptography) → igov:CompliantSecurityService(?s)	To obtain services that deals with secure information and demands channel cryptography, according to Policy 3.
2	igov:definesSecurityConstraintsTo(?sl, ?s) ∧ igov:greaterOrEqualLevelTo(?sll, ?sl) ∧ sameAs(?sll, igov:3) → igov:CompliantSecurityService(?s)	To obtain services that do not deal with secure information and do not require security mechanisms on the compliant set
3	soa:ServiceContract(?sc) ∧ igov:endDate(?sc, ?dt) ∧ temporal:after("now", ?dt) → igov:InactiveContract(?sc)	To defines the set of Inactive Contracts by checking the expected ending date

Table 3: Supporting SWRL rules

The resulting queries representing the policies 2 to 7 are presented in Table 4.

4.5 intelliGOV execution

After the definition of the ontology and rules, we load data from the organizations's SOA environment. Some manual procedures were also performed. The equivalence between the legacy and current security classification was included in the ontology, using the sameAs property, relating the instances representing the equivalent levels. The information regarding the relation between security mechanisms and communication protocols was obtained by interviews with experts and included in the ontology. Finally, data content was anonymized to preserve organization confidentiality

Policy	**Policy Query**
2	soa:Service(?oServ) ∧ igov:SecurityLevel(?oLevel) ∧ igov:definesSecurityConstraintsTo(?oLevel, ?oServ) → sqwrl:select(?oServ, ?oLevel)
3	igov:CompliantSecurityService(?s) → sqwrl:select(?s)
4	igov:variantOf(?minor, ?major) ∧ soa:hasInterface(?minor, ?minInt) ∧ soa:hasInterface(?major, ?majInt) ∧ soa:hasInput(?minInt, ?minInput) ∧ soa:hasInput(?majInt, ?majInput) ∧ soa:hasOutput(?minInt, ?minOutput) ∧ soa:hasOutput(?majInt, ?majOutput) ∧ sqwrl:makeSet(?setMinInput, ?minInput) ∧ sqwrl:makeSet(?setMajInput, ?majInput) ∧ sqwrl:makeSet(?setMajOutput, ?majOutput) ∧ sqwrl:makeSet(?setMinOutput, ?minOutput) ∧ sqwrl:groupBy(?setMajInput, ?minor) ∧ sqwrl:groupBy(?setMinInput, ?minor) ∧ qqwrl:groupBy(?setMajOutput, ?minor) ∧ sqwrl:groupBy(?setMinOutput, ?minor) ∧ sqwrl:difference(?setDif1, ?setMajInput, ?setMinInput) ∧ sqwrl:difference(?setDif2, ?setMinInput, ?setMajInput) ∧ sqwrl:size(?tamDif1, ?setDif1) ∧ sqwrl:difference(?setDif3, ?setMajOutput, ?setMinOutput) ∧ sqwrl:difference(?setDif4, ?setMinOutput, ?setMajOutput) ∧ sqwrl:size(?tamDif2, ?setDif2) ∧ sqwrl:size(?tamDif3, ?setDif3) ∧ sqwrl:size(?tamDif4, ?setDif4) ∧ swrlb:equal(?tamDif1, 0) ∧ swrlb:equal(?tamDif2, 0) ∧ swrlb:equal(?tamDif3, 0) ∧ swrlb:equal(?tamDif4, 0) → sqwrl:select(?minor)
5	igov:versionOf(?ver1, ?serv) ∧ igov:versionOf(?ver2, ?serv) ∧ soa:hasInterface(?ver1, ?int1) ∧ soa:hasInterface(?ver2, ?int2) ∧ soa:hasInput(?int1, ?inp1) ∧ soa:hasInput(?int2, ?inp2) ∧ soa:hasOutput(?int2, ?out2) ∧ soa:hasOutput(?int1, ?out1) ∧ sqwrl:makeSet(?setIn1, ?inp1) ∧ sqwrl:groupBy(?setIn1, ?ver1) ∧ sqwrl:makeSet(?setIn2, ?inp2) ∧ sqwrl:groupBy(?setIn2, ?ver2) ∧ sqwrl:makeSet(?setOut1, ?out1) ∧ sqwrl:groupBy(?setOut1, ?ver1) ∧ sqwrl:makeSet(?setOut2, ?out2) ∧ sqwrl:groupBy(?setOut2, ?ver2) ∧ sqwrl:makeSet(?setVer, ?ver1) ∧ sqwrl:groupBy(?setVer, ?serv) ∧ sqwrl:size(?szVer, ?setVer) ∧ sqwrl:difference(?setDif1, ?setIn2, ?setIn1) ∧ sqwrl:difference(?setDif2, ?setIn1, ?setIn2) ∧ sqwrl:difference(?setDif3, ?setOut1, ?setOut2) ∧ sqwrl:difference(?setDif4, ?setOut2, ?setOut1) ∧ sqwrl:size(?szDif1, ?setDif1) ∧ sqwrl:size(?szDif2, ?setDif2) ∧ sqwrl:size(?szDif3, ?setDif3) ∧ sqwrl:size(?szDif4, ?setDif4) ∧ swrlb:divide(?rd, 1, ?szVer) ∧ swrlb:add(?r, ?szDif1, ?szDif2, ?szDif3, ?szDif4, ?rd) ∧ swrlb:greaterThanOrEqual(?r, 1) → sqwrl:select(?ver1)
6	igov:variantOf(?var, ?ver) ∧ igov:hasState(?var, igov:Operational) ∧ sqwrl:makeSet(?setVar, ?var) ∧ sqwrl:groupBy(?setVar, ?ver) ∧ sqwrl:size(?szVar, ?setVar) ∧ swrlb:equal(?szVar, 1) → sqwrl:select(?ver, ?var)
7	igov:hasState(?serv, igov:Operational) ∧ soa:isContractFor(?sc, ?serv) ∧ igov:InactiveContract(?isc) ∧ sqwrl:makeSet(?setCont, ?sc) ∧ sqwrl:groupBy(?setCont, ?serv) ∧ sqwrl:makeSet(?setInat, ?isc) ∧ sqwrl:difference(?setAtiv, ?setCont, ?setInat) ∧ sqwrl:size(?szAt, ?setAtiv) ∧ swrlb:greaterThan(?szAt, 0) → sqwrl:select(?serv)

Table 4: SQWRL queries representing the policies of the case study

4.6 intelliGOV execution

After the definition of the ontology and rules, we load data from the organizations's SOA environment. Some manual procedures were also performed. The equivalence between the legacy and current security classification was included in the ontology, using the sameAs property, relating the instances representing the equivalent levels. The information regarding the relation between security mechanisms and communication protocols was obtained by interviews with experts and included in the ontology. Finally, data content was anonymized to preserve organization confidentiality.

During intelliGOV execution, some important points were identified:

- As explained in Section 4.4, some queries were originally written using the sqwrl:equal (?s1, ?s2) operator, that compares the contents of two sets ?s1 and ?s2, and returns true if they are equal or false, otherwise. However, when combined with the sqwrl:makeset (?s, ?e) ^ sqwrl:groupBy (?s, ?re) operators, the sqwrl:equal operator always returns true, ignoring the grouping condition. The solution was the evaluation of the difference set, counting the number of elements and verifying if it is equals to 0. The use of these operations produced the desired result, but increased rule complexity. This situation happened on policies 1, 4 and 5. The queries presented previously in Table 4 considered the adapted policies;
- Due to this complexity, the execution of policies 4 and 5 led to out of memory exceptions when all the services were considered. To handle that, a statement was included in each query to limit the execution to one service at a time. As an example, in policy 5, a statement sameAs(?serv, [service instance]) was included in the beginning of the policy and the [service instance] parameter was filled in a loop that executed the query sequentially for the entire service set.

After execution, the number of errors and the execution time for each policy were registered. To measure execution times, the time before and after query execution were stored and the difference between these two moments was considered as the execution time. For policies 4 and 5, due to the use of queries per service strategy, resulting in a loop of query executions, the execution time of each step of the loop was registered and the total execution time for the policy was considered as the sum of the individual executions, leading to a higher execution time. For the other policies that could be evaluated for the entire set, the total execution time was directly measured.

4.7 Case study results

To evaluate the results, the data measured in the case study was consolidated and the research team promoted a meeting with the participants to discuss the results. During this meeting, the template was validated by the entire group. The results are presented in Table 5.

Policy	**Manual Evaluation**				**intelliGOV Execution**			
	quantity of errors	error Rate (%)	execution time (sec)	average execution time per service (s)	quantity of errors	Error Rate (%)	execution time (s)	average execution time per service (s)
1	6	24%	570	23	0	0	3.7	0.1
2	8	32%	405	16	0	0	2.7	0.1
3	2	8%	445	18	0	0	3.4	2.9
4	3	12%	400	16	0	0	72.5	2.8
5	8	32%	365	15	0	0	70	0.1
6	4	16%	305	12	0	0	2.8	0.1
7	6	24%	315	13	0	0	2.8	0.1

Table 5: Results of manual and intelliGOV execution

For policy 1, the problem described by the participants was the difficulty to identify the correct domain of information in the context of the organization, especially when the classification of the company diverges from the practice of the market. For instance, the workforce service, which deals with employee data, was correctly classified in the Human Resources area, since the organization classifies this information in the same manner as the market. However, the service SalesOrder was classified as a Sales service, despite the fact that the company classifies that as a Downstream service.

The participants also had difficulty to identify the classification of hierarchies of data. As an example, consider the Supplier Payment service that handles queries of the set of payments received by a supplier, composed by two groups of data: Supplier Data, which describes specific supplier information, like identification, name and contacts; and Payment Data, containing the value of the payment, banking information and accounting information. Each one of these groups of data belongs to one or more distinct subject areas. Supplier Data is contained in the Supplies area and Payment contains information handled by both Finance and Accounting areas. Hence, the data classification of this service is the composition of these subject areas. In the manual evaluation, the analyst considered only the Supplies area on the analysis and marked the service as compliant.

These errors were avoided using intelliGOV through the transitive property handlesDataFrom. In the Supplier Payment Service, all the levels of information were considered, leading to an expected classification composed by the Supplies, Contract, Finance and Accounting areas. The compliance assessment errors were originated due to the lack of knowledge of the analysts about the structure of the company and not from a technical aspect. This knowledge was registered in the ontology, leading to better results using intelliGOV.

For policy 2, the problem was the mapping between the old security classification and the new one. The analysts with less experience in the organization considered the services classified in the legacy classification as non-compliant, instead of interpreting the mapping between the two classifications. intelliGOV considered the mapping of the two classification model by means of the sameAs relation between instances representing each level of the model. One important point is that no logic to map the levels needed to be expressed in the SQWRL queries, thus reducing the complexity of the query.

The difference between the security classification taxonomies also caused the wrong evaluation on policy 3, since errors identified by the team were caused by errors in the classification that led to a wrong protocol selection. However, because this policy depended more on a technical knowledge (given a security level, which communication protocol is correct), the error rate was lower than the one measured in policy 2. Considering intelliGOV, it reused the knowledge of the mapping between the legacy and new levels of information security, allowing a precise conformity identification. For policies 4 to 7, the problem was the difficulty to separate the concepts of variants and releases in the context of the company. The concept of variant is closer to the traditional way that teams develops software in the IT area, what justifies smaller expected and actual number of non-conformities and smaller error rates. However, the concept of releases as expressed by the ICC is less common, leading to higher error numbers. The participants identified that it was very complex to identify whenever a service specification changed due to a new version in the case of changes in the service interface.

Considering specifically policy 7, the analysts with less experience in the company did not considered the concept of existing active contracts to define whenever or not an old release must be active, representing actual service consumers that depend of the old service interface. This fact led to errors in compliance identification and explained several operational problems that occurred in the ICC regarding deactivation of releases with active consumers. In this context, the definitions were technical (*i.e.*, from the IT area).

Considering this analysis, it is possible to identify that intelliGOV gains were obtained due to the use of all necessary knowledge available in the ontology. This knowledge includes detailed and specific rules like mapping of security levels, data and organizational structure information and version control policies. These rules and concepts were not used correctly by part of the analysts during the manual evaluation, leading to a higher error rate. IntelliGOV also presented gains in performance due to the automation of the processes of finding and integrating information and reasoning about compliance.

Another important point is related to the distribution of the error rate of the manual execution according to the different types of expertise of the professionals. Figure 4 presents this compilation, considering the average error rate of the evaluation. In Figure 4.a, the error rate is distributed according to the experience of the professionals in the organization. Figure 4.b presents the error rate according to the experience in SOA technology. From these results, the participants with more experience in the organization have lower error ratings. It shows the importance of intelliGOV solution. Due to knowledge captured in the ontology, the professionals with more experience can be released to act on more strategic tasks, after supporting the modelling of the ontology to be used.

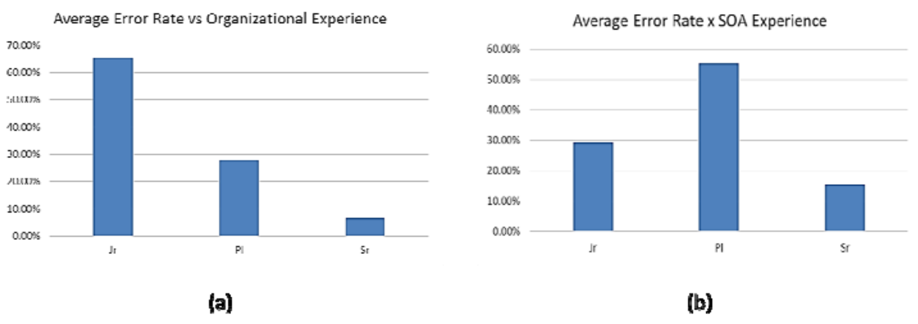

Figure 4: Comparison of error rate distributed by organizational and SOA experience

5 Conclusions and Future Work

Compliance evaluation of SOA is an expensive and complex activity. To reduce these factors, we proposed intelliGOV, an approach based on ontologies, semantic rules and semantic queries. To implement a software toolset to support the approach, OWL, SWRL and SQWRL were used, aiming to increase interoperability and flexibility. The contributions are a new approach to validate compliance in SOA environments and a toolset that automates this process.

To evaluate results, a case study was executed in a global energy company, considering services that deals with developed applications, packages, legacy systems and intercompany communications. Two variables were measured: error rates, representing the percentage of wrong compliance evaluations, and execution times, representing the necessary time to evaluate a policy. Two kinds of evaluation were executed: a manual evaluation, executed by seven analysts of the organization, with different levels of experience both in the organization and in SOA; and an evaluation executed by intelliGOV.

The manual evaluation led to an error rate that varied from 8% to 32% and an execution time for evaluation of one policy considering the entire set of elements of the SOA that varied between approximately 5 and 10 minutes. It was also possible to identify that experience in the organization contributes to decrease this error rate, based on the experience of the participants of the case study. Using intelliGOV, the error rate was reduced to zero and the execution time varied between 2.7 and 70 seconds, even with processing limitations of the SQWRL expressions.

The proposed solution may be used to deal with business concepts (like business structure organization and corporate security policies) as well as technical issues (*e.g.*, version control). Hence, it can provide results for several compliance evaluation scenarios.

However, some limitations were identified and should be stressed in future works. First, the process list used in the evaluation: despite of the importance of the selected processes, the list could be extended to evaluate other critical areas in SOA, like service composition and billing. Second, the domains considered for the ontology definition. The study was limited to the domains of structure, security and technology (*e.g.,* services, contracts, inputs and outputs) according to the selected processes by the ICC team. These aspects could be evaluated applying intelliGOV in additional processes and different domains.

Another limitation was the case study team, composed only by IT people. So, a future work is to broaden the audience to include business stakeholders. Then, identifying their issues in the policy definition step, capturing eventual non-existing terms in the current ontology, and assessing the use of intelliGOV for compliance checking in this scenario.

Other limitation is the use of a simple ontology engineering methodology to design the ontology used in the case. However, the use of more complete approaches could enhance the results, due to more efficient ontology design and could be a new line of work to explore. One possible study would be the conception of a complete ontology for the SOA governance domain and strategies to merge it with other domain ontologies that capture the concepts and rules that relates to the SOA governance mechanism.

Another work could be the use of computational agents to act on the identification of non-conformity, leading to agility and more stability of the organization. A computational agent could run the compliance analysis and, based on the result, act to control the environment or to suggest possible actions.

Finally, some aspects regarding the SQWRL language can be explored. Limitations on comparison operators of SQWRL were detected and the resolution of these issues can simplify the description of the rules and promote more reductions of processing time. Another point considers the process of transcription of policies in natural language to SQWRL. This activity can become complex as policy complexity rises, leading to extensive queries. Further studies regarding the processing of rule transcription and natural language processing can simplify this process and reduce complexity.

References

[Bajec and Krisper, 2005] Bajec, M., Krisper, M.: “A methodology and tool support for managing business rules in organisations”; Information Systems, 30, 6 (2005), 423-443.

[Bennett, 2012] Bennett, S. G.: “A Framework for SOA Governance, Release 3.2”; Oracle Practitioner Guide, Oracle, (February 2012), also appeared as electronic version in http://www.oracle.com/technetwork/topics/entarch/oracle-pg-soa-governance-fmwrk-r3-2-1561703.pdf

[Birukou et al., 2010] Birukou, A., D’Andrea, V., Leymann, F., Serafinski, J., Silveira, P., Strauch, S., Tluczek, M.: “An integrated solution for runtime compliance governance in SOA”; Service-Oriented Computing; Springer Berlin Heidelberg, (2010), 122-136.

[Brown et al., 2006] Brown, W. A., Moore, G., Tegan, W.: “SOA Governance-IBM’s Approach”; Effective governance through the IBM SOA Governance Management Method approach, White paper, August 2006, also appeared as electronic version in ftp://170.225.15.40/software/soa/pdf/SOA_Gov_Process_Overview.pdf.

[Elgammal et al., 2014] Elgammal A., Sebahi, S., Turetken, O., Hacid, M., Papazoglou, M.P., van den Heuvel, W.J. :”Business Process Compliance Management: ad integrated proactive approach"; Proceedings of the 24th International Business Information Management Association Conference, Milan (2014), p.764, 781

[Erl, 2005] Erl, T.: "Service-Oriented Architecture (SOA): Concepts, Technology, and Design"; Prentice Hall, Upper Saddle River, NJ, USA (2005).

[Gruber, 1993] Gruber, T. R.: "Toward Principles for the Design of Ontologies Used for Knowledge Sharing"; Knowledge acquisition, 5, 2 (1993), 199–220.

[Hevner et al., 2004] Hevner, A. R., March, S. T., Park, J., Ram, S.: "Design science in information systems research"; MISQ (Management Information Systems Quarterly), 28, 1 (2004), 75-105.

[Hitzler et al., 2012] Hitzler, P. Krötzsch, M., Parsia, B., Patel-Schneider, P. F., Rudolph, S.: "OWL 2 Web Ontology Language Primer (Second Edition)"; W3C Recommendation 11 December 2012 (2012), also appeared as electronic version in http://www.w3.org/TR/2012/REC-owl2-primer-20121211/

[Hojaji and Shirazi, 2010] Hojaji, F. & Shirazi, M. R.: "A Comprehensive SOA Governance Framework Based on COBIT"; Proc. 6th World Congress on in Services (SERVICES-1), IEEE Computer Society, Miami (2010), 407–414.

[Holanda et al., 2009] Holanda, H. J. A., Barroso, G. C., Serra, A. B.: "Performance Analysis of Service Oriented Software". iSys - Revista Brasileira de Sistemas de Informação (Brazilian Journal of Information Systems), 2, 1 (2009), also appeared as electronic version in http://www.seer.unirio.br/index.php/isys/article/view/336/392

[Horrocks et al., 2004] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M.: "SWRL: A semantic web rule language combining OWL and RuleML"; W3C Member Submission 21 May 2004 (2004), also appeared as electronic version in http://www.w3.org/Submission/SWRL/

[Hsiung et al., 2012] Hsiung, A., Rivelli, G., Huttenegger, G.: "How to design a global SOA infrastructure: Coping with challenges in a global context"; Proc. ICWS 2012 - IEEE 19th International Conference on Web Services, IEEE, Honolulu (2012), 536–543.

[ITGI, 2007] ITGI, Control Objectives for Information and related Technology (COBIT), Ver 4.1, Apr 2007.

[Janiesch et al., 2009] Janiesch, C., Korthaus, A., Rosemann, M.: "Conceptualisation and facilitation of SOA governance"; Proc. ACIS 2009 - 20th Australasian Conference on Information Systems, Melbourne (2009), 154–163.

[Jelliffe, 2002] Jelliffe, R.: "The Schematron Assertion Language 1.6"; Academia Sinica Computing Centre (2002), also appeared as electronic version in http://xml.ascc.net/resource/schematron/Schematron2000.html

[Joachim et al., 2013] Joachim, N., Beimborn, D., Weitzel, T.: "The influence of SOA governance mechanisms on IT flexibility and service reuse"; The Journal of Strategic Information Systems, 22, 1 (2013), 86-101.

[Luckham, 2002] Luckham, D. C.: "The power of events - An Introduction to Complex Event Processing in Distributed Enterprise Systems"; Addison-Wesley Reading (2002), also appeared as electronic version in https://sisis.rz.htw-berlin.de/inh2010/12375999.pdf

[Mellor et al., 2002] Mellor, S. J., Scott, K., Uhl, A., Weise, D.: "Model-driven architecture"; In Advances in Object-Oriented Information Systems; Lecture Notes in Computer Science, 2426; Springer, Berlin Heidelberg (2002), 290-297.

[Niemann et al., 2010] Niemann, M., Miede, A., Johannsen, W., Repp, N., Steinmetz, R.: “Structuring SOA Governance”; IJITBAG (International Journal of IT/Business Alignment and Governance), 1, 1 (2010), 58-75.

[Noy and McGuinness, 2001] Noy, N. F., McGuinness, D. L.: “Ontology development 101: A guide to creating your first ontology”; Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford medical informatics technical report SMI-2001-0880 (2001), also appeared as electronic version in
http://liris.cnrs.fr/alain.mille/enseignements/Ecole_Centrale/What%20is%20an%20ontology%20and%20why%20we%20need%20it.htm

[O’Connor and Das, 2009] O'Connor, M. J., Das, A. K.: “SQWRL: A Query Language for OWL”; OWLED - Sixth International Workshop on OWL: Experiences and Directions, Chantilly, Virginia, USA (2009), also appeared as electronic version in
http://www.webont.org/owled/2009/papers/owled2009_submission_42.pdf

[Pandit et al., 2009] Pandit, B., Popescu, V., Smith, V.: “Service Modeling Language, Version 1.1”; W3C Recommendation 12 May 2009 (2009), also appeared as electronic version in http://www.w3.org/TR/sml/

[Papazoglou et al., 2007] Papazoglou, M. P., Traverso, P., Dustdar, S., Leymann, F.: “Service-oriented computing: state of the art and research challenges”; IEEE Computer, 40, 11 (2007), 38–45.

[Rodríguez et al., 2013] Rodríguez, C., Schleicher, D., Daniel, F., Casati, F., Leymann, F., Wagner, S.: “SOA-enabled compliance management: instrumenting, assessing, and analyzing service-based business processes”; Service Oriented Computing and Applications, 7, 4 (2013), 275-292.

[Schepers et al., 2008] Schepers, T. G., Iacob, M. E., Van Eck, P. A.: “A lifecycle approach to SOA governance”; Proc. 2008 ACM Symposium on Applied Computing, ACM, Fortaleza, Brazil (2008), 1055-1061.

[Spies, 2012] Spies, M.: “Continous Monitoring for IT Governance with Domain Ontologies”; Proc. 23rd International Workshop on Database and Expert Systems Applications (DEXA), IEEE, Vienna, Austria (2012), 43 –47.

[Stantchev and Stantcheva, 2012] Stantchev, V., Stantcheva, L.: “Extending traditional it-governance knowledge towards soa and cloud governance”; IJKSR (International Journal of Knowledge Society Research), 3, 2 (2012), 30-43.

[Taylor et al., 2007] Taylor, S., Lacy, S., Macfarlane, I. : “ITIL v3.0 Publication Framework: ITIL Service Support, Service Desk, Service Strategy, Service Design, Service Transition, Service Operation, and Continual Service Improvement”, Office of Governance Commerce (OGC) (2007)

[Teixeira Filho and Azevedo, 2014] Teixeira Filho, H. M., Azevedo, L.G.: “Governance of Service-Oriented Architecture through the CommonGov Approach”; IJCISIM (International Journal of Computer Information Systems and Industrial Management Applications), 6, 1 (2014), 505-514.

[Teixeira Filho et al., 2014] Teixeira Filho, H. M., Azevedo, L.G., Siqueira, S. W. M.: “Applicability Analysis of Using Ontologies and Semantic Rules for Supporting Compliance Assessment in the Context of Service Oriented Architectures” (in Portuguese: “Análise de Aplicabilidade do Uso de Ontologias e Regras Semânticas para Apoiar a Verificação de Conformidade no Contexto de Arquiteturas Orientadas a Serviço”); Proc. IX Simpósio Brasileiro de Sistemas de Informação, SBC, João Pessoa, Brazil.

[The Open Group, 2009] The Open Group: “SOA Governance Framework”, Open Group Standard (2009), 96p, United Kingdom, Reference C093, ISBN 1-931624-82-8, also appeared as electronic version in https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?catalogno=c093

[The Open Group, 2014] The Open Group: “Service-Oriented Architecture Ontology, Version 2.0”; Open Group Standard (2014), 89p., United Kingdom. Reference C144, ISBN 1-937218-50-8, also appeared as electronic version in https://www2.opengroup.org/ogsys/jsp/publications/PublicationDetails.jsp?publicationid=13543

[Tran et al., 2011] Tran, H., Holmes, T. I., Oberortner, E., Mulo, E., Cavalcante, A. B., Serafinski, J., Dustdar, S.: “An end-to-end framework for business compliance in process-driven SOAs”; Proc SYNASC (Symbolic and Numeric Algorithms for Scientific Computing), IEEE, Timisoara, Romania (2011), 407-414.

[Tran et al., 2012] Tran, H., Zdun, U., Holmes, T. I., Oberortner, E., Mulo, E., Dustdar, S.: “Compliance in service-oriented architectures: A model-driven and view-based approach”; Information and Software Technology, 54, 6 (2012), 531-552.

[Yin, 2009] Yin, R. K.: “Case study research: Design and methods”; Sage, 5 (2009).

[Zhou et al., 2010] Zhou, Y. C., Liu, X. P., Wang, X. N., Xue, L., Tian, C., Liang, X. X.: “Context model based soa policy framework”; Proc. ICWS (International Conference on Web Services), IEEE, (2010), 608-615.