SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	Specification of the governance infrastructure v2
Document version:	1.0
Task code:	T3
Deliverable code:	D3.2.2
WP Leader (organisation):	PetalsLink
Deliverable Leader (organisation):	PetalsLink
Authors (organisations):	Nicolas Salatgé
Date of first version:	07/02/11

Change control

Changes	Author / Entity	Code of version
Creation of the document	Nicolas Salatgé	0.1

Table of Contents

1. Introduction 4
2. List of Acronyms 7
3. Governance Requirements 8
3.1. Introduction 8
3.2. Governance Engine 8
3.3. Governance Registry 8
3.3.1. Governance Policies 10
3.3.2. Governance Standards 11
3.4. Lifecycle Management Policies 12
3.4.1. Service Lifecycle Management Policies 12
3.4.2. Design Time Service Policies 13
3.4.3. Run-Time Service Policies 13
3.4.4. SLA Lifecycle Management Policies 13
3.5. Prototype Specification 15
3.5.1. Core Concepts 15
3.5.2. APIs 18
3.5.3. Frontends 19
3.6. Prototype Implementation 19
3.6.1. Components 19
3.6.2. Sources 20
3.6.3. Installation 21
4. Conclusion 24
5. Bibliography 25

1. Introduction

Governance is the act of governing or administrating. It refers to all measures, rules, decision-making, information and enforcement that ensure the proper functioning and control. In the context of Service-Oriented Architecture (SOA), there are several ways to define governance. We survey, in the following, definitions from the literature to define the concept of SOA Governance:

- Anne Thomas Manes Research Director at Burton Group defines SOA Governance as processes that an enterprise puts in place to ensure that things are done in accordance with best practices, architectural principles, government regulations, laws, and other determining factors [1]. SOA governance refers to the processes used to govern adoption and implementation of SOA, ensuring and validating that assets and artifacts within the architecture are acting as expected and maintaining a certain level of quality.
- According to Paolo Malinverno [2], SOA Governance is about having discipline and making sure that the very important decisions go through to appropriate people, and that these people have the appropriate input to make those decisions.
- SUN [3] states SOA Governance as the ability to organize, enforce, and reconfigure service interactions in an SOA.

Basing on the above definitions, in Soceda governance can be defined as a set of processes, rules, policies, mechanisms of control, enforcement policies, and best practices put in place throughout the lifecycle of services and choreographies (from the design time to run-time stage), in order to ensure the successful achievement of the SOA implementation. SOA governance ask for the definition of policies and supporting tools. Particularly challenging is the definition of governance policies and mechanisms that need to be distributed and controlled according to a decentralized and neutral paradigm. In order to ensure that the integration of independently developed pieces of software is successful, an effort in introducing standardized notations, interfaces, data coding and more recently also semantics (through ontologies) has been undertaken by SOA companies. Nevertheless companies have soon realized that standards alone are not sufficient to ensure interoperability. The socio-technical nature of the SOA world means that it is necessary to consider services as active entities that operate aside or replace humans. To achieve SOA interoperability, it is then necessary to put in place also some social organization to govern the interactions among the participating services, aiming at assuring that everyone abides by the agreed social rules.
In Soceda, the governing rules and procedures should be established and enforced by superpartes entities, which must be trusted and accepted by anyone. This is what SOA Governance is conceived for. Indeed, the apparent flexibility and ease of use of service-oriented application can be only achieved through discipline and an enforced framework of rules, policies and processes.

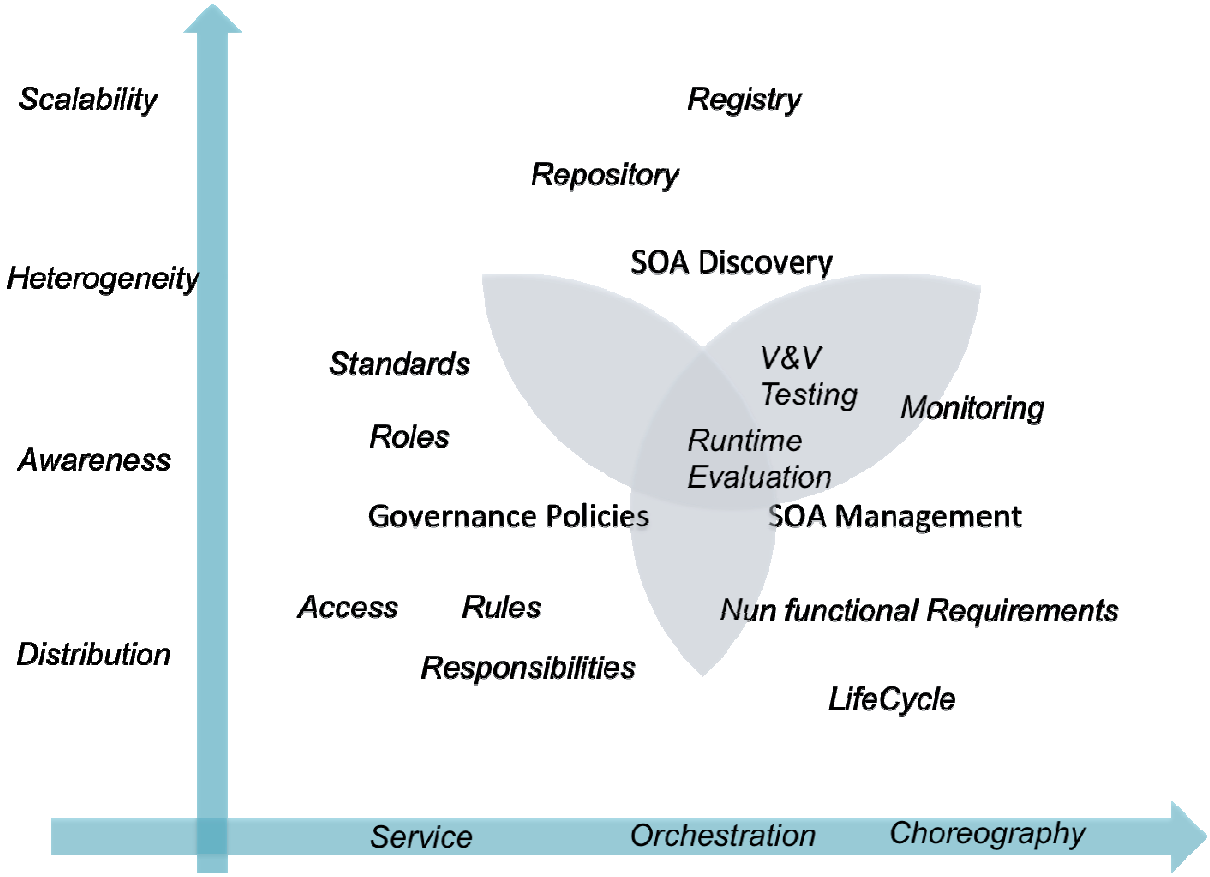

Figure 1: Overview of SOA Governance in Future Internet

So far, SOA Governance has been mainly pursued for achieving service integration within one organization.
In the future vision of services all around us that dynamically connect and disconnect, on demand, towards some business objective, SOA Governance must be meant as a comprehensive management umbrella under which effective interoperability across organizations and platforms is ensured.
However, the definition of decentralized policies does not solve the issues and challenges that the Soceda project will need to address; we then also need means for monitoring, assessing and enforcing policies and rules. This functionality will be made available through specific mechanisms included in the Soceda platform as detailed in the following chapters.
Soceda will investigate both design-time and runtime governance. Governance activities can be seen as a transversal layer that ensures the adoption of the right way of doing things, the right time and by the right persons. Governance is a paradigm underlying the whole service lifecycle and the IT system and at the borderline of three concerns: SOA Discovery, SOA Management, and Governance Policies (as illustrated in Figure 1).

- SOA Discovery: services registries and repositories provide service discovery capabilities at both design and run-time. Governance capabilities as looking for a service, retrieving it, and managing its life cycle are provided on top of a registry/repository mechanism.

- SOA Management: covers the management of the service lifecycle from development until runtime. This includes the definition and following of the service at several stages of their lifecycle.. Moreover, SOA Management can also include the monitoring and runtime evaluation of the non functional requirements of services. Consequently, it ensures the alignment between service consumer requirements and the runtime behavior of services. SOA Management needs to address also the service aspects by defining which rules, policies and standards are more relevant to be applied at different stages of the service lifecycle. Governance Policies: policies are the cornerstone of the governance paradigm. Through the adoption of a set of policies and standards commonly used, SOA governance makes the exposed services compliant with heterogeneous services coming from several platforms. These need to be identified in order to implement the governance framework. In order to ease interoperability and service reuse, best practices and rules are adopted. Both SOA discovery and SOA Management are concerned with governance policies as they define each step of the service life cycle. Each stakeholder involved in the governance process has their roles and responsibilities that need to be identified. The governance process resides also in setting common code conventions and standards. In the Soceda project we elucidate a list of governance policies and rules as being part of the governance framework..

2. List of Acronyms

Acronym	Definition
BSM	Business Service Monitoring
CEP	Complex Event Processing
DCEP	Distributed Complex Event Processing
DSB	Distributed Service Bus
EDA	Event Driven Architecture
ELA	Event Level Agreement
IaaS	Infrastructure as a Service
JSON	JavaScript Object Notation
PaaS	Platform as a Service
QoS	Quality of Service
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SaaS	Software as a Service
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SPARQL	SPARQL Protocol And RDF Query Language (recursive acronym)
USDL	Unified Service Description Language
WSDL	Web Service Description Language
WSN	Web Service Notification

3. Governance Requirements

3.1. Introduction

Since their inception, Service Oriented Architectures (SOA) was revealed as being the de facto paradigm for future systems. They combine best practices paradigms inspired from previous application models. Modularity, encapsulation, fine-grained granularity, publication and discovery help SOA to be widely used by developers and users. As a consequence, enterprise need to move their systems from classical IT to innovative SOA, and essential functions need also to be exported and adapted. Governance is a first and foremost function in IT systems. Governance Framework activities ensures the best interests of an organization to be met and this through corporate decisions from design time to run-time 0.

3.2. Governance Engine

The important component of SOA Governance Engine is the registry functionality, see e.g. 0. Having a robust registry promotes the discovery and reusability of services. Registries serve not only to inventory and catalog service data, but also as places to store metadata about services, necessary to SOA Governance. These metadata go beyond Web Service Description Language (WSDL) documents and include descriptions of their functionalities, capabilities, and the locations of their service contracts. Nevertheless, a good way to provide governance on top of such scalable systems is to set policies and rules. The following sections are devoted to the description of the Governance Registry, Governance Policies and Governance Standards.

3.3. Governance Registry

In Soceda context, the registry functionality is essential since a very large of events and services coming from different sources need to be discovered and governed. Governance Registry enables the management of business services.
The Governance Registry allows the discovery of business services at both design and run-time. First, it provides the ability of managing the business service lifecycle, the creation of service level agreement and their negotiation. A status for the business service is assigned and evolves, as the service is being developed, tested, verified and validated.
Second, the Governance Registry presents also a run-time view of the deployed services. Business services running in the service access middleware, and precisely in the distributed service bus nodes, are discovered and monitored. This way the governance framework is also able of evaluating at run-time if a negotiated service level agreement is respected or not. The Governance Registry is at the center of all the governance activities and it is also linked to the run-time middleware for services. Governance Policies. The Figure 2 depicts the registry component of the Governance Engine.

Third, in order to tackle heterogeneity issues of business services coming from different sources, the governance registry relies on a uniform and common service description language. The USDL 0 enables the expression of the common business services descriptions in a unique agreed way.

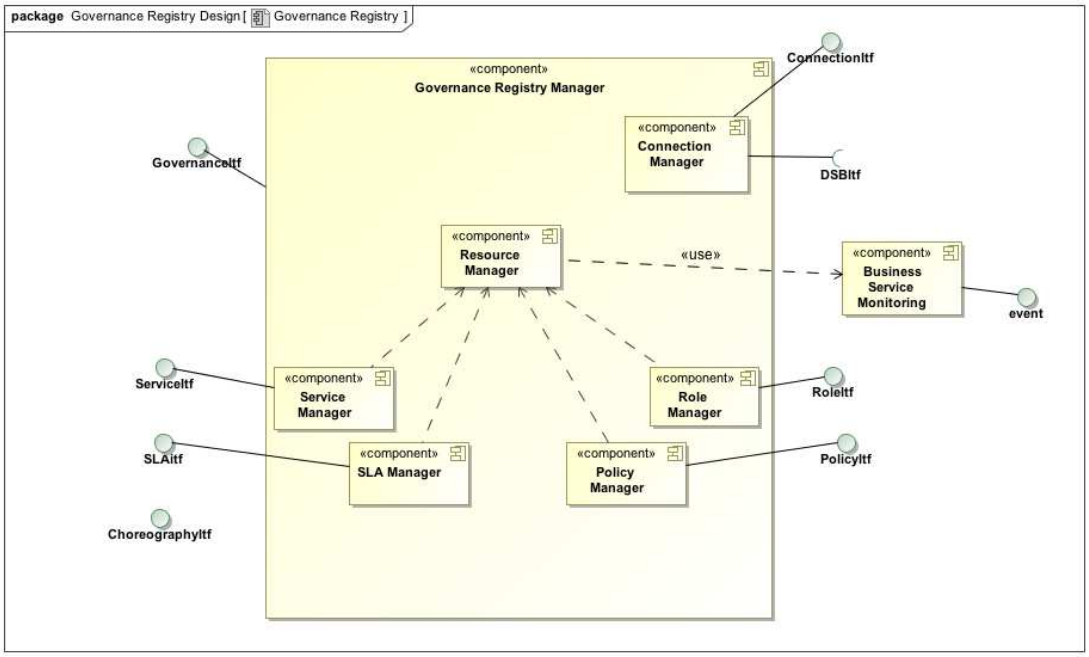

Figure 2: Governance Registry Components

The governance registry is composed of the following components:

- **Resource Manager:** The *Resource Manager* is a generic component that provides capabilities for the management of the lifecycle of the XML Resources. These are handled by the Governance Framework and can be a Service Description, an SLA, a Role or a Policy. The Resource Manager provides the Resource interface and is connected to the Business Service Monitoring to monitor resources and run-time.
- **Service Manager:** The *Service Manager* is a component providing an interface called Service. The Service interface provides abilities for managing the lifecycle of a service. A service can be created, described, and finally retired.
- **SLA Manager:** The *SLA Manager* provides an SLA interface. It provides a list of functionalities facilitating the management of the lifecycle of an SLA such as the creation, reading, negotiation, lookup of the SLA templates attached to a service and finally the termination of an SLA.
- **Policy Manager:** The *Policy Manager* component provides a Policy interface. It is responsible for the management of the lifecycle of a policy. Offered functionalities are the creation, addition, storage and removal of a Policy.
- **Role Manager:** The *Role Manager* component is responsible for providing facilities for the creation, modification and deletion of the governance roles and responsibilities. It provides the Role interface.

- **Connection Manager:** The *Connection Manager* is responsible for synchronizing the Governance with the middleware runtime environment functionalities such as the connection, disconnection, synchronization and the lookup of the available execution environments are offered.

3.3.1. Governance Policies

Defining the life-cycle policies of the resources to be governed is essential. Indeed, Governance activities apply policies and constraint rules at several steps of the lifecycle for ensuring the good behavior of these resources. The **Erreur ! Source du renvoi introuvable.** shows the Policies that are supported by the Governance Registry.

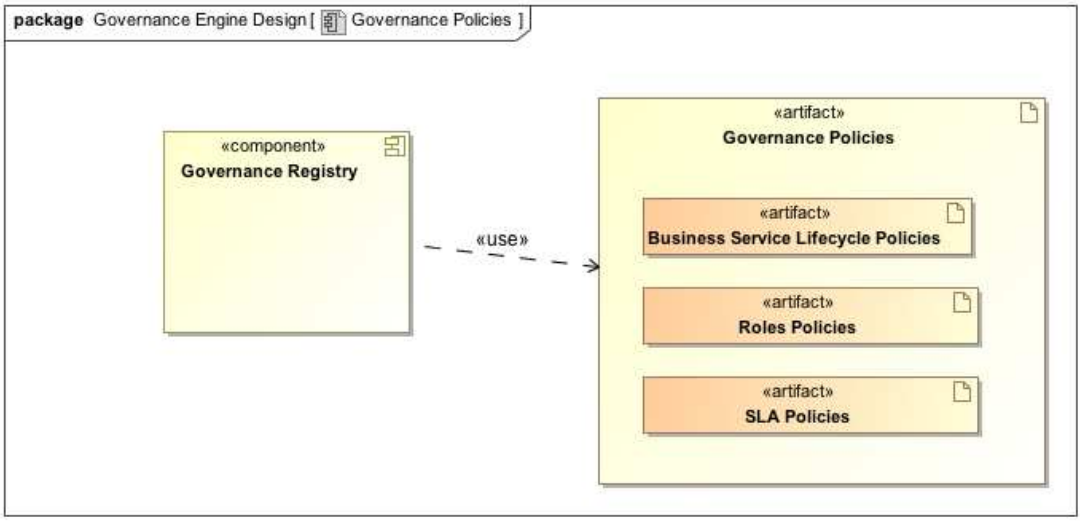

Figure 3: Governance Policies Components

The governance policies are component by the following components:

- **Business Service Lifecycle Policies**: are responsible for controlling the lifecycle of a service from its design to its deployment on the middleware. They ensure and guide the adoption of the best practices for governing the development process of a service.
- **Role Policies**: define the responsibilities for each person or application with regard to the governance framework. It states exactly who can do what and when. This functionality is essential for governance activities. The different identified roles interact with the governance registry and are able according to their access permissions to operate functions.
- **SLA Policies**: refers to the policies that concern the service level agreements. These are related to their definition, publication, negotiation, and monitoring at run-time. The governance registry enables the handling of the SLA lifecycle.

3.3.2. Governance Standards

The adoption of common agreed technological standards for software design and development eases software interoperability and helps addressing the scalability issue.
In the Soceda context, we deal with an important number of heterogeneous services. Sources, protocols, development paradigms can differ from a service provider to another. The governance framework relies on a uniform service model based on USDL.
In **Erreur ! Source du renvoi introuvable.**, we report the standards for services, SLAs, policies and rules that are commonly used by the SOA community and that we consider in our governance engine. Obviously other standards may be added as we develop the governance framework.

Standard	Description	Ref.
Unified Service Description Language (USDL)	USDL is a generic service description language consolidated from SAP Research projects. It aims to provide a way for users to model services from a business, operational and technical point of view. It defines nine modules related to each other to model of the overall service description: Service, Service Level, Legal, Technical, Functional, Interaction, Participants, Pricing and Foundation.	0
Web Services Description Language (WSDL)	WSDL describes a web service and defines how it works. It defines a standardized syntax for expressing the most relevant information about web services.	0
Web Services Interoperability (WS-I)	WS-I defines a common way to expose business services al- lowing their interoperability. For example, the WS-I standard may set constraints on the WSDL definition.	0
Simple Object Access Protocol (SOAP)	SOAP is a protocol for XML-based messaging over a network as defined by its WSDL file.	0
eXtensible Access Control Markup Language (XACML)	this standard is based on XML and defines an access control mechanism on rules and conditions	0
WS-Agreement	this standard provides a normalized way of expressing service level agreements. An agreement is a contract between a service consumer (client) and a service provider essentially on QoS constraints such as latency or availability	0
Web Service Policy (WS-Policy)	WS-Policy defines an nXML model and syntax to express policies of a Web Service.	0

Table 1: Commonly used standards

3.4. Lifecycle Management Policies

Defining the lifecycle policies of the resources to be governed is essential. Indeed, in the governance framework, we need to define the different steps of the lifecycle of business services, Service Level Agreements. Governance activities apply policies and constraint rules at several steps of the lifecycle for ensuring the good behavior of these resources. The following sections are devoted to the description of the life-cycle management policies for services and Service Level Agreements.

3.4.1. Service Lifecycle Management Policies

Service lifecycle management policies focus on defining rules and regulations on business services. The appropriate SOA Governance policies have to be applied in each phase of the lifecycle to both guarantee and control the access of services. Figure 4 summarizes several policies that can be defined in each phase of the service lifecycle.

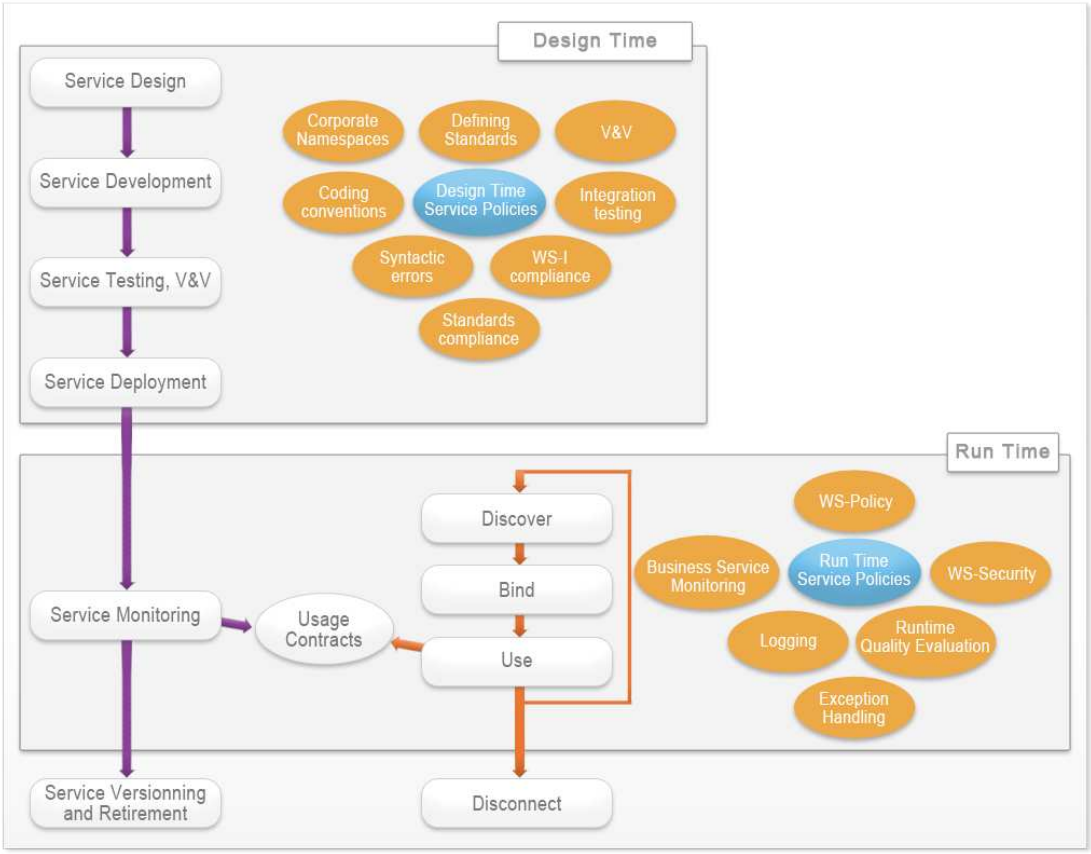

Figure 4: Service Lifecycle Policies

3.4.2. Design Time Service Policies

In a governance approach and at design time, each service passes several phases such as specification and design, development and implementation, testing and V&V before it is deployed.
The service design describes the infrastructure, capabilities and desired features. In this phase, a design time policy might define which, when, and where to use standards and insure compliance between them. The design time policies may also consider the fact of setting out corporate namespace, common coding conventions, identifying syntax errors, etc. Once the design of the service is finished, it is time to develop it respecting in one hand the rules defined in service design and in another hand, by defining appropriate policies such as coding conventions (for example, naming conventions for objects, variables, procedures, etc.) and syntactic errors policies.
Then, to assure the quality of services, service development needs to be coordinated with service testing with respect to integration testing and V&V policies. Services can be combined with mocks or other services and integration tests are performed. Then, development-time conformance tests are applied to verify whether the services play correctly their roles. Differently from the test strategies applied at run-time, at this point of development, the integration and the conformance tests are applied off-line in a development or testing environment.
Finally, policies at deployment stage might require that services in production environment are compliant with requirements of Web Service-Interoperability standard.

3.4.3. Run-Time Service Policies

Policies at run-time stage come into play once the service is deployed and monitors operational aspects of a service to get full control of SOA Governance. It is most effective to define run-time policies in a Web Service Management (WSM) system to provide a common way to access and exchange information. Besides, run-time policies might require all deployed services to be managed and use the Web Service Security (WS-Security) standard and Web Service Policy (WS-Policy). Moreover, at run-time, it is ultimately necessary to enforce and manage the SLA contract elaborated between the service provider and the service consumer, according to defined policies.

3.4.4. SLA Lifecycle Management Policies

The Service Level Agreements are a commonly used way for designing non-functional objectives to be reached by the business service once deployed. They are also needed for defining the responsible entities to be alerted if a constraint is violated. Besides, the service level agreement is useful in order to achieve the run-time quality evaluation of running business services. They are used later in order to calculate the choreography level agreement. In this section we present the SLA lifecycle, as we will consider in the governance framework. The SLA lifecycle includes the states of discovery, negotiation, creation, monitoring, termination, and enforcing consequences for non-compliance. Figure 5 provides an overview of the following SLA life-cycle stages as defined by Sun Microsystems.

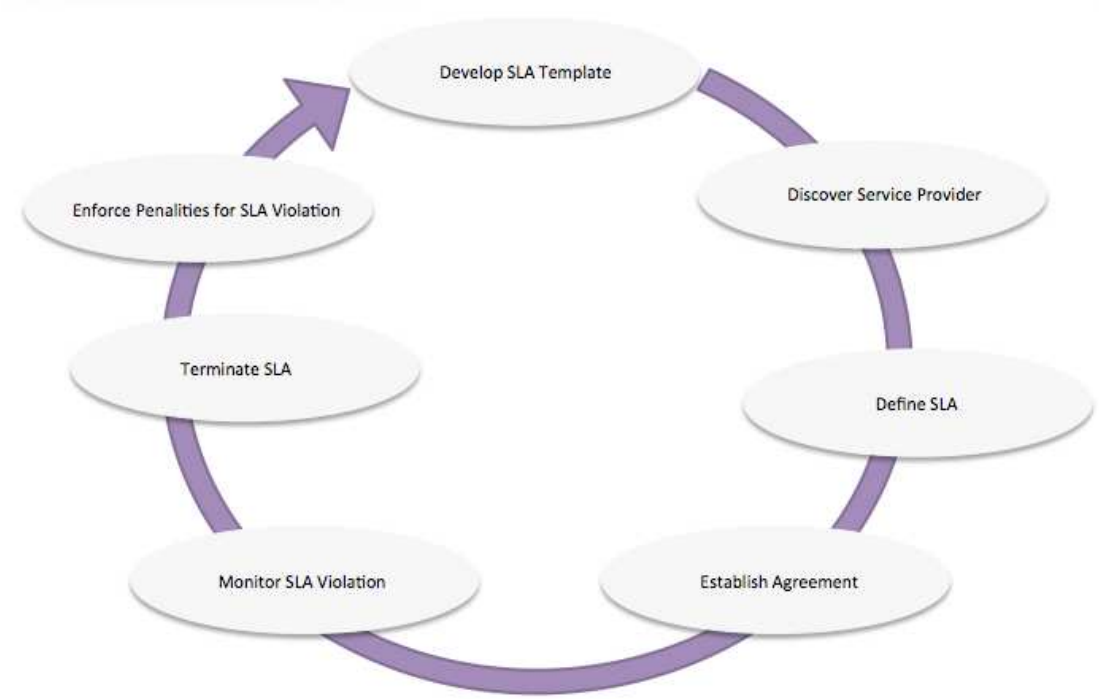

Figure 5 Service Lifecycle Policies

1) **Develop SLA Template**: the whole lifecycle of the SLA is based on a template or model used for the representation of its various clauses. This first phase consists in the specification and modeling of the template;

2) **Discover Service Provider**: the SLA lifecycle starts with discovering resources that could satisfy the requirements of the service consumer;

3) **Define SLA**: once the service provider(s) has (ve) been discovered, it is important to specify and model the required quality levels of the agreement;

4) **Establish Agreement**: in this phase the SLA template is created and is accomplished by signing a service level agreement by the client and the provider.

5) **Monitor SLA Violation**: this phase consists in monitoring the obligations defined in the SLA to ensure that all contract clauses have been achieved or violated by one of the parties or both of them.

6) **Terminate SLA**: This phase consists on the termination of the SLA.

7) **Enforce Penalties for SLA Violation**: In case of non-compliance, it is important to specify the consequences and penalties.

3.5. Prototype Specification

In 3.4, we presented all the important points for governing a service oriented architecture and so an event oriented architecture we target in the Soceda project. The first version of the governance prototype will take these basis concepts to provide a first implementation focusing on the following event-based aspects: How to detect event producers, how to handle agreements and how to provide frontends allowing user interaction.

3.5.1. Core Concepts

3.5.1.1. Events management

The main goal of the first governance prototype is to manage events producers and consumers. To achieve this, we need to define how to model events. Like in all the other platform components, the governance is strongly based on open standards. For events, we chose to use the OASIS Web service Notification specification collection 0 which is open and extensible. We explain in the next sections how we create the mapping between events and WSN specifications.

3.5.1.1.1. Event Definition

Events are types messages. To define an event, we use Topic Namespace 0 like in the following XML snippet:

```
<wstop:TopicNamespace name="CrisisEvents"
        targetNamespace="http://play-project.eu/crisis"
        xmlns:fireman_event="http://play-project.eu/crisis/v1/deliver_iodine/Fireman/"
        xmlns:wstop="http://docs.oasis-open.org/wsn/t-1"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://docs.oasis-open.org/wsn/t-1 http://docs.oasis-open.org/wsn/t-1.xsd
http://play-project.eu/crisis/v1/deliver_iodine/Fireman/ ../deliver_iodine/Fireman.wsdl">
  <wstop:Topic name="cardiacRythmTopic" messageTypes="fireman_event:cardiacRythmEvent"/>
  <wstop:Topic name="temperatureTopic" messageTypes="fireman_event:temperatureEvent"/>
  <wstop:Topic name="symptomTopic" messageTypes="fireman_event:symptomEvent"/>
</wstop:TopicNamespace>
```

In the snippet above, three topics are defined: *cardiacRythmTopic*, *temperatureTopic* and *symptomTopic*. Each topic corresponds to a message defined in the WSDL document *Fireman.wsdl*. For example:

```
<wsdl:message name="cardiacRythmEvent">
        <wsdl:part name="parameters" element="tns:cardiacRythm"></wsdl:part>
</wsdl:message>
<wsdl:message name="temperatureEvent">
        <wsdl:part name="parameters" element="tns:temperature"></wsdl:part>
</wsdl:message>
<wsdl:message name="symptomEvent">
        <wsdl:part name="parameters" element="tns:symptom"></wsdl:part>
</wsdl:message>
```

Each message contains one part that references an element in a XML schema. This element defines the data format of the events. For example, the schema for the *cardiacRythm* element can be defined as:

```
<xsd:element name="cardiacRythm">
	<xsd:complexType>
		<xsd:sequence>
			<xsd:element name="fireman" type="Q1:FiremanType" />
			<xsd:element name="value" type="xsd:int" />
		</xsd:sequence>
	</xsd:complexType>
</xsd:element>
<xsd:complexType name="FiremanType">
	<xsd:sequence>
		<xsd:element name="name" type="string" />
		<xsd:element ref="wsa:EndpointReference" />
	</xsd:sequence>
</xsd:complexType>
```

3.5.1.1.2. Event Producer Detection

As mentioned earlier, the governance platform may be able to detect event producers. To achieve this, we chose to 'force' event producers to be described as WSDL and to implement the NotificationProducer WSN interface[1].
The interface contains the following operations:

- ***Subscribe***: Subscribe to a topic.
- ***GetCurrentMessage***: Get the last notification which has been sent to subscribers which subscribed using the Subscribe operation above.
- ***GetResourceProperty***: This generic operation is used to retrieve the topics supported by the event producer.

As an example, one may be able to get the list of topics supported by an event producer by invoking the GetResourceProperty with the following message:

```
<wsrf-rp:GetResourceProperty xmlns:wsrf-rp="http://docs.oasis-open.org/wsrf/rp-2"
	xmlns:t="http://docs.oasis-open.org/wsn/t-1" >
		t:TopicSet
</wsrf-rp:GetResourceProperty>
```

The resulting TopicSet contains the set of topics supported by the producer. In the example below, the producer implements two topics from the Topic Namespace definition shown before:

```
<wsrf-rp:GetResourcePropertyResponse xmlns:wsrf-rp="http://docs.oasis-open.org/wsrf/rp-2"
  xmlns:tns1=" http://play-project.eu/crisis" >
  <wstop:TopicSet xmlns:wstop="http://docs.oasis-open.org/wsn/t-1">
    <tns1:cardiacRythmTopic wstop:topic="true" />
    <tns1:temperatureTopic wstop:topic="true" />
  </wstop:TopicSet>
</wsrf-rp:GetResourcePropertyResponse>
```

[1] https://github.com/play-project/play-resources/tree/master/deliverables/D5.3.1/ws-notification

3.5.1.2. Platform Integration

The governance engine is completely part of the Soceda platform and interacts with it using several communication approaches. One of the main goals is to detect new services, new event producers and to manage them. In order to detect all the platform actors, the governance component is integrated with the PEtALS Distributed Service Bus using the following event-based approach:

1. Each time a new service is bound to the service bus, the service bus sends a message to the governance platform with all the service information. The communication between the service bus and the governance platform is event-based
2. The governance engine process the incoming message as follows
    a. If the service description is a WSDL file and if it contains notification producer primitives, it is registered as event actor
    b. Else it is registered as 'standard' service

This approach maintains the coherence between the governance knowledge and the real services available in the platform since the same approach is used when services are removed from the platform.
In order to keep both governance and service bus synchronized if a component fails, the governance may also be able to query the service bus when needed by calling an operation to retrieve all the current services deployed in the platform.
This integration between the governance and the service bus is described in the following sequence diagram:

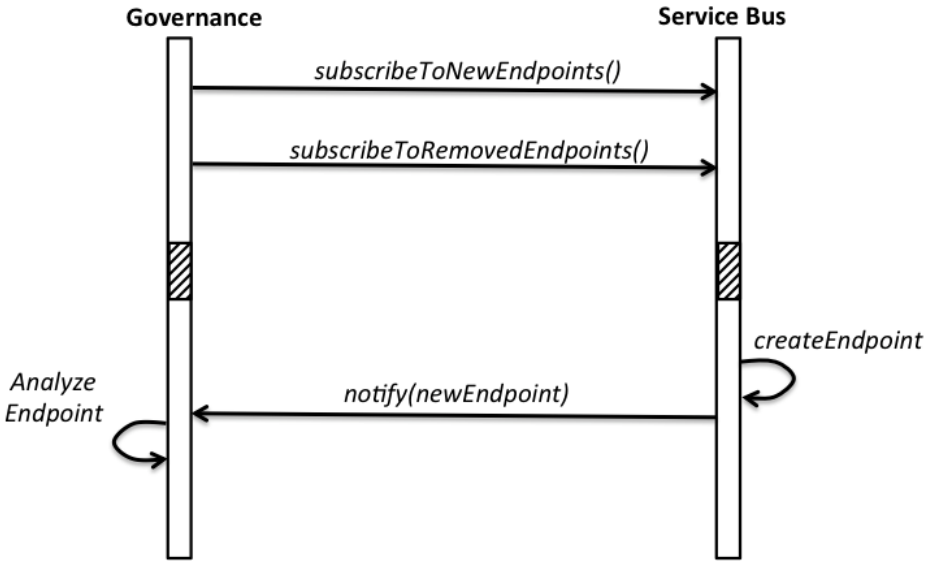

Figure 6 Governance and Service Bus integration

3.5.2. APIs

The governance platform provides several APIs to manage services and events producers. The current section describes the minimal set of interfaces needed to create and use the governance framework.
These APIs follows the open standards approach used in the Soceda project: It uses WSDL as description language and will be exposed as Web services by the governance engine.

3.5.2.1. Service interface

The governance engine provides a set of operations to create and manipulate registered services as described in the full WSDL description.
We describe here the main operations of this WSDL service description:

- **publishService(WSDL)**: Register a new service by giving its WSDL description.
- **List<ServiceID> findServices(Query)**: Find a list of services ID based on the input parameters.
- **Service getService(ServiceID)**: Get the service description based on its ID.

Note that the API also allows the retrieval and manipulation of services based on their operations, parameters and faults definition. This provides complex manipulations which are not used in the current prototype.

3.5.2.2. Event interface

The events interface provides a set of operations to manipulate events producers and consumers as described in its WSDL description.
We describe the main operations of this interface:

- **findEventProducerByElements**: Retrieve a list of event producers by giving the message type supported by the producer
- **findEventsProducerByTopics**: Retrieve a list of event producers which supports the given topic
- **findTopicsByElement**: Find a list of topics which support the given message type
- **publishTopicNamespaceFromDOM**: Create a new topic namespace from an XML document
- **publishTopicNamespaceFromURL**: Create a new topic namespace from a XML document located at the given URL.

By using this API and the event definition, the governance platform is completely integrated with the system and allows to manipulate event producers in an efficient way.

3.5.3. Frontends

Frontends are defined as ways to easy the user interaction with the platform. The governance is a central part of the Soceda platform and will need many user interactions. In order to achieve this and to be 'Cloud ready', the governance frontend will need to be available as a Web application at the Software as a Service (SaaS) level.
The frontend needs to provide access to all the governance operation defined earlier through forms and simple web pages for example. This will be possible by connecting the user interface to the governance APIs.

3.6. Prototype Implementation

The first prototype of the governance platform is implemented as part of the EasierGov product **Erreur ! Source du renvoi introuvable.** developed by EBM Websourcing. This governance engine follows the requirements and specifications described in section 3.2.

3.6.1. Components

3.6.1.1. EasierGov engine

As defined previously, the engine is the core component of the governance platform. This software components implements all the processes and operations used to manage services and events producers.

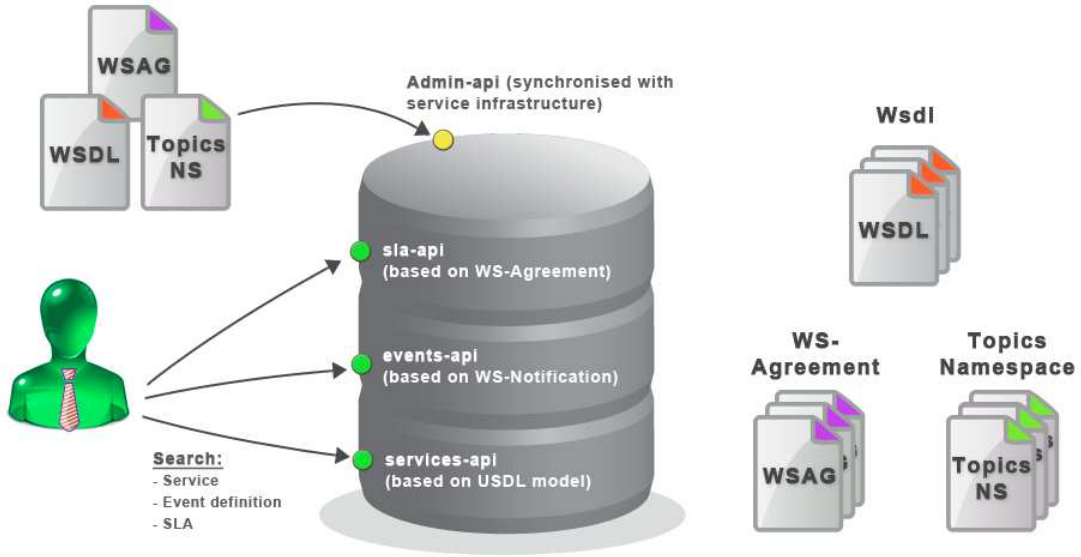

Figure 7 EasierGov components

The easierGov engine provides a set of APIs defined as:

- **Admin API**: Allows to configure specific parameters of the governance platform and gives some essential information to user such as the service list, topic list, …

- **Connexion API**: Services involved in composition are accessed using enterprise service bus capabilities (PEtALS DSB in the Soceda context). This API provides integration mechanisms and techniques for making heterogeneous services communicate and interoperate. It is based on a service discovery protocol for retrieving and querying services in the runtime context : The runtime environment and the governance framework are fully synchronized.
- **Event API**: Depends on Service API. It is able to detect event producers deployed in the service infrastructure. For each producer, it allows to retrieve the supported topicsd. It is also possible to get the message data model associated to the topic.
- **Resources API**: Allows to manipulate resources in the 'CRUD way': Create, read, update, delete. A resource can be defined as a basic concept in all governance framework and is represented as XML.
- **Services API**: Manages services based on their WSDL definition.
- **SLA API**: Manages all the Service and Event Level Agreement part. The API allows to inject agreement in the framework and so to handle all the user-defined SLAs.
-

3.6.1.2. EasierGov frontend

The current version of the EasierGov engine does not provide any User Interface as a frontend. We chose to focus developments on APIs instead of UI in the first prototype to validate governance concepts. Having a solid and well-defined API will just lead us to easily create an efficient user interface.
Actually, we can say that the frontend is the SOAP APIs which are fully integrated in the Soceda platform using the services approach.

3.6.2. Sources

The sources are available in a subversion repository located at https://svn.petalslink.com/svnroot/trunk/research/dev/experimental/easiergov.
easierGov is implemented in Java language and uses the Apache Maven project management tool. To get and build the sources, you have to follow these instructions (from a unix-based system):

```
svn co
https://svn.petalslink.org/svnroot/trunk/research/dev/experimental/easiergov/
cd easiergov
mvn install
```

The sources are decomposed as modules and are organized as follows:

- **Admin-api**: Management API
- **Admin-impl**: Management API implementation
- **Events-api**: Events API

- **Events-impl**: Events API implementation
- **Events-ui**: Events User interface
- **Gov-launcher**: Bootstrap used to launch easierGov with all the required dependencies
- **Gov-ws-client**: A Java Web service client
- **Gov-ws-distribution**: A complete distribution of easierGov packaged as archive with all the libraries, resources and configuration files
- **Kernel-api**: The engine API. This is used by all modules and APIs to interact with the framework.
- **Kernel-impl**: The kernel API implementation
- **Resources-api**: The resources definition. Resources are used in all the modules to handle data.
- **Resources-impl**: The resources API implementation
- **Services-api**: The services API
- **Services-impl**: The services API implementation
- **Services-ui**: The services User interface
- **Services-usdl-api**: The services API as USDL

3.6.3. Installation

The last version of the governance platform is available as open source and can be directly downloaded from the easierGov project page at:
http://research.petalslink.org/display/easiergov/Binaries.
The binary is provided as a graphical installer which allows to tune the easierGov installation. Depending on the system platform, you can double-click on the downloaded file or launch it with the '*java –jar*' command.

Figure 8 EasierGov installer

Once installed, you can go to the selected folder and launch EasierGov from a terminal like (on UNIX-like system):

```
cd $INSTALL_FOLDER/bin
chmod a+x *.sh
./startup.sh
```

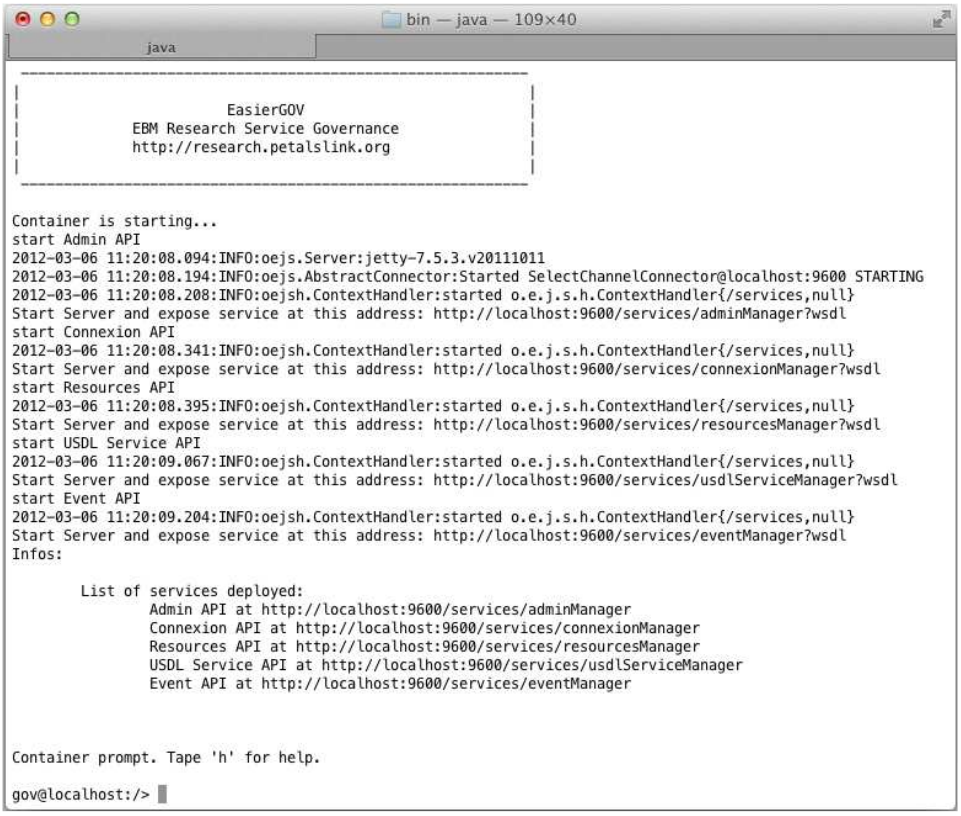

Figure 9 easierGov prompt

4. Conclusion

In this deliverable, we specified and described the second version of the Soceda Governance.
In the Soceda projects, this component have two main functionalities. It is in charge of provide an API to discover any event producers from a given topic and it allows to an event consumer and an event provider to negociate an agreement. This governance framework is currently integrated in the version 1 of packaging of Soceda project.

5. Bibliography

[1] Eric A. Marks, Service-Oriented Architecture Governance for the Services Driven Enterprise, John Wiley & Sons, Inc. Editions, 2008
[2] eBay Open Source. Turmeric repository. https://www.ebayopensource.org/wiki/display/TURMERICDOC/Repository.
[3] USDL: Unified Service Description Language. http://www.internet-of-services.com/index.php?id=288&L=0.
[4] WSDL: Web Service Description Language. http://www.w3.org/TR/wsdl.
[5] WS-I: Web Service Interoperability. http://www.ws-i.org/.
[6] SOAP: Simple Object Access Protocol. http://www.w3.org/TR/soap/.
[7] XACML: eXtensible Access Control Markup Language. http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf.
[8] WS-Agreement: Web Service Agreement. www.ogf.org/documents/GFD.107.pdf.
[9] WS-Policy: Web Service Policy – Framework. http://www.w3.org/TR/ws-policy.
[10] OASIS Web service Notification specification. http://www.oasis-open.org/committees/wsn/.
[11] EasierGov Governance Platform. http://research.petalslink.org/display/easiergov/.