Secure Provisioning of Cloud Services
based on SLA Management

SPECS Project - Deliverable 2.3.1

Reference Architecture for Cloud SLA Negotiation: Development and Tests - Prototype

Version no. 1.1
18 February 2016

The activities reported in this deliverable are partially supported
by the European Community's Seventh Framework Programme under grant agreement no. 610795.

Deliverable information

Deliverable no.:	D2.3.1
Deliverable title:	Reference Architecture for Cloud SLA Negotiation: Development and Tests - Prototype
Deliverable nature:	Prototype
Dissemination level:	Public
Contractual delivery:	18 February 2016
Actual delivery date:	18 February 2016
Author(s):	Madalina Erascu (IeAT)
Contributors:	Valentina Casola (CeRICT), Massimiliano Rak (CeRICT), Adrian Spătaru (IeAT)
Reviewers:	Silvio La Porta (EMC), Stefano Marrone (CeRICT)
Task contributing to the deliverable:	T2.3
Total number of pages:	28

Executive summary

The current deliverable is the first of the three deliverables (D2.3.1, D2.3.2, and D2.3.3) reporting the status of development activities of the Negotiation module. The module is in charge of managing the negotiation phase of a Service Level Agreement (SLA) life-cycle. More precisely, it analyses users' security requirements transforming them into a machine-understandable language in order to be usable by the SPECS framework, as well as delivering the result of the negotiation back to the users.

As described in previous deliverables, in order to cover the large number of requirements, the *Negotiation* module is composed by the three components: *SLO Manager, Supply Chain Manager*, and *Security Reasoner* and by a number of different artifacts, mainly conceptual models to cope with the SLA document and format.

In this document, we present the status of development activities related to T2.3 and first reference to install and use the prototype components to be developed in the framework of this task. At this stage, the *SLO Manager* and *Security Reasoner* components are defined. The first component has the role of translating the security requirements specified by the users into a hierarchy of SLAs. The second component evaluates the security levels of different supply chains and compares them with the users' security requirements. They cover a very large number of requirements and they are core components of the SPECS solution, although not transparent to the users.

The analysis of elicited requirements and motivation behind the choice of presented solution are discussed in deliverable D2.1.2. Relevant state-of-the-art and current design are presented in deliverable D2.1.1. Therefore, in this deliverable we focus on implementation status, provide links to repositories, and present guides for installation and usage of the component.

Table of contents

Index of figures

Index of tables

1. Introduction

The prototypes presented in this document demonstrates how the SPECS solution manages the *Negotiation* phase of an SLA life cycle. More precisely, we present the first prototypes of the *SLO Manager* and *Security Reasoner* components of the *Negotiation* module.

SLO Manager receives End-users' security requirements and transforms them into a hierarchy of SLAs, hierarchy determined by the security level of each SLA. In this document, we present how the *SLO Manager* interacts with the other *Negotiation* components (*Supply Chain Manager, Security Reasoner*) and other SPECS modules (*SPECS Application, SLA Platform*) by providing component APIs and interaction protocols (Section 3.2.2). A prototype of the *SLO Manager* can be downloaded from [1].

The *Security Reasoner* is the component devoted to evaluate and rank Security SLAs in order to allow the SPECS customers to choose the best SLA offer. At the first release, the *Security Reasoner* is accessible as a standalone SPECS Application, which can be downloaded from [2]. In the next version, the functionalities will be offered through a REST Interface.

The two components cover around 80% of the requirements (see Section 3.1) needed for presenting complex use cases which will demonstrate the SPECS solution at month 24, namely *Secure Web Container* and *Secure Storage* (see D5.1.1 for more details on these use cases). In case of the *SLO Manager* component, at month 24 we plan to have all the requirements covered and after this date only maintenance work will take place. The development and maintenance work of the *Security Reasoner* will spread until month 30.

Details on implementation, installation and usage of the components are given in Sections 3.2.1 and 3.2.2 (*SLO Manager*), respectively in Sections 3.3.1 and 3.3.2 (*Security Reasoner*).

2. Relationship with other deliverables

The overview of the relationship of this deliverable with others from different WPs is presented in
Figure 1. This deliverable is based on the existing work on architecture, requirements, use cases, etc., developed in the previous deliverables, but also constitutes an input for the deliverables finalizing the architecture, core modules and their interplay, use cases.

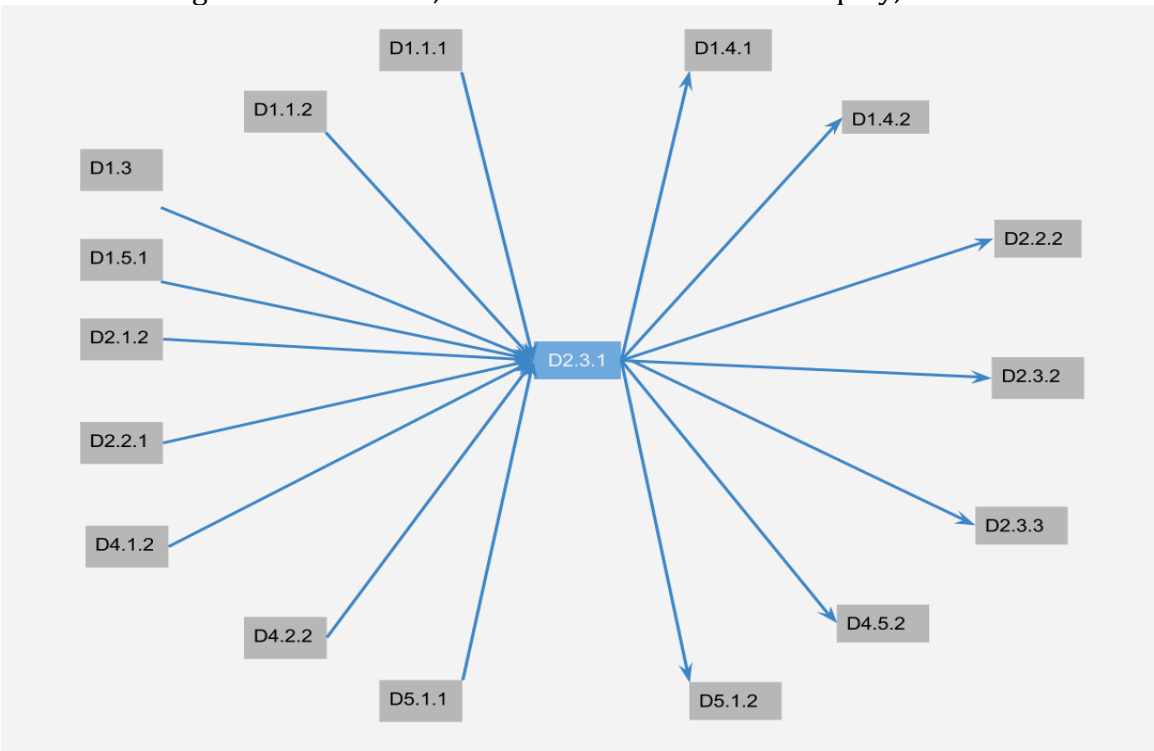


Figure 1. Relationship with other deliverables

The following deliverables are input for D2.1.3:

- D1.1.1: The initial architecture design submitted at M6 is a direct input for D2.3.1 since it sets up the basis for the initial design of the Negotiation Architecture done in D2.3.1.
- D1.1.2: Same as before, the architecture design done in D1.1.2 contributes to D2.3.1, since D1.1.2 establishes the implementation design.
- D1.3: The Module Interaction Protocol is an input in D2.3.1 as it defines how the Negotiation Module will exchange information within the SPECS Platform.
- D1.5.1: Continuous input is taken from the progress in this deliverable as the Negotiation Module will have to conform to the integration scenarios.
- D2.1.2: This deliverable is the final report on requirements for the negotiation and represents the main input regarding negotiation requirements in D2.3.1.
- D2.2.1: The initial report on conceptual framework for Cloud SLA Negotiation is taken into consideration for the design and development in this deliverable.
- D4.1.2: The details on the requirements for renegotiation described in D4.1.2 are considered in D2.3.1.
- D4.2.2: The final design of the Enforcement Module affects D2.3.1 in terms of the post negotiation steps.
- D5.1.1: The validation scenarios considered in WP5 (especially in deliverable D5.1.1) were used to guide the development of the negotiation process and the relationship

among all the entities of the Negotiation Module.

The following are the deliverables and WPs that take as input the results obtained in D2.3.1:

- D1.4.1: Deliverable D2.3.1 will contribute to D1.4.1 with the prototype for the architecture regarding the Negotiation Module and API.
- D1.4.2: Same as before, D2.3.1 will contribute to D1.4.2 with the prototype for the architecture regarding the Negotiation Module and API.
- D2.2.2: The report on Conceptual Framework for Cloud SLA Negotiation will be finalized based on the considerations carried out in D2.3.1.
- D2.3.2: This deliverable represents the final prototype for the Negotiation Module architecture, being the continuation of deliverable D2.3.1.
- D2.3.3: The final report on Cloud SLA Negotiation will be based on deliverable D2.3.1.
- D4.5.2: The prototype developed in D2.3.1 will be subject to testing and validation, the report being delivered as D4.5.2 in M24.
- D5.1.2: For this deliverable, D2.3.1 will provide input for the description of the validation scenarios.

Note that there is no direct interface between Negotiation and Monitoring modules because of the decoupling function of the Enforcement module and the SLA Platform.

3. *Service Level Agreement Negotiation* Module

SPECS Negotiation implies an agreement on the security level of services requested by a Cloud Service Consumer (CSP), referred in the following as End-user (EU), and offered by the CSP, agreement materialized in an SLA. According to deliverables D1.1.1 and D2.1.2, negotiation can be divided into: *negotiation* and *renegotiation*.

- *Negotiation***.** EU starts the negotiation by specifying his/her security requirements through the SPECS Application. At the end of the negotiation if the level of security of the services are accepted, the *SPECS Application* will select the SLA to be signed and will use its own proprietary solution to sign the document and invoke the *SLA Platform* in order to change the state of the SLA to "Signed SLA".
  Renegotiation. As it is defined in D2.1.2, renegotiation can occur due to a change in a previously agreed SLA and is initiated by EU and/or CSP. Renegotiation can occur in the case: (1) the existing SLA has been violated or alerted (2) changes were performed in the signed SLA.

We will focus in the rest of the deliverable on *negotiation*. The following sequence diagram (Figure 2) shows the complete, interaction level, process of a negotiation triggered by an EU and the initial set of messages exchanged. The diagram also depicts the interaction of the three components (*SLO Manager*, *Supply Chain Manager*, *Security Reasoner*) of the *Negotiation* module, as well as the communication of the Negotiation module with other SPECS modules (*SPECS Application*, *SLA Platform*). We will not enter here into details on the negotiation process, as it is described in D1.3, rather we will focus on the communication interfaces of the *SLO Manager* with the other negotiation modules and SPECS components.

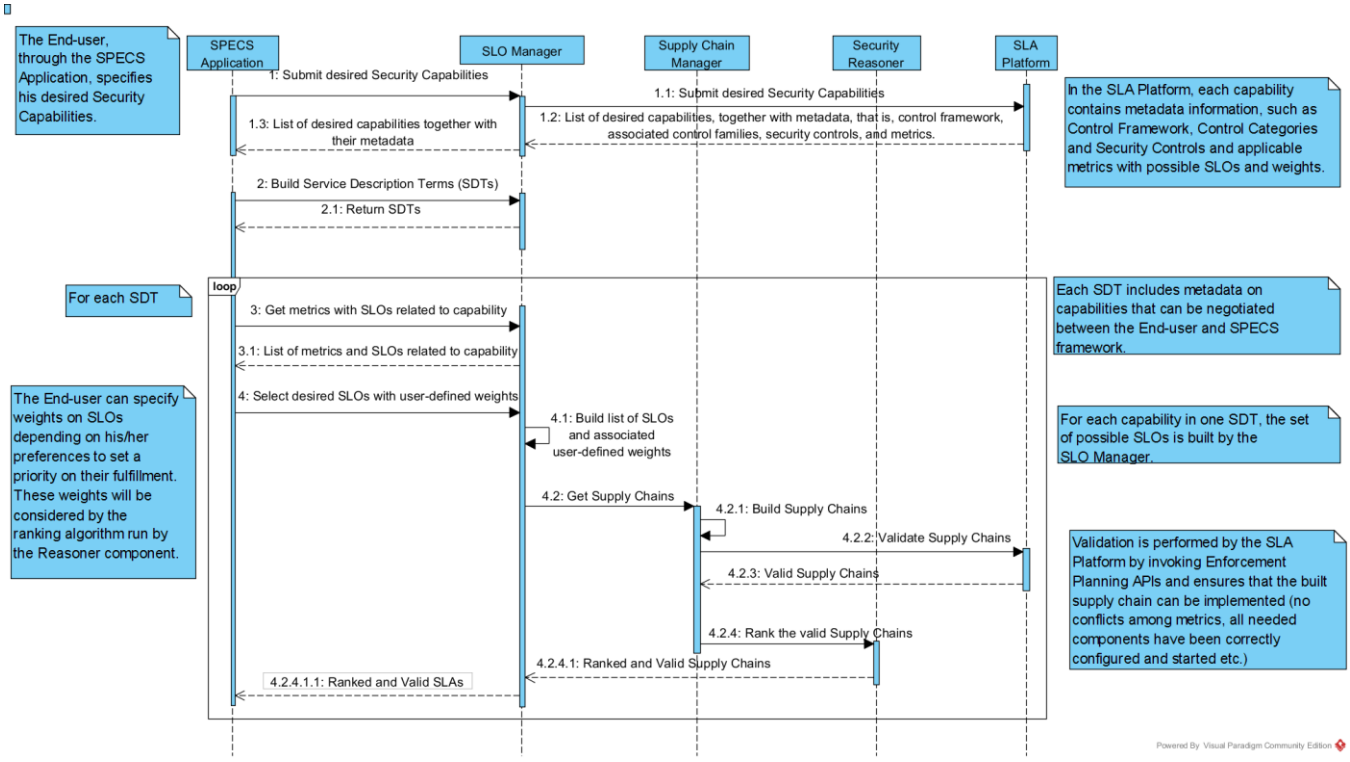


Figure 2 SLA Negotiation Process

Negotiation module (Figure 3) is composed by the following components:

1. *SLO Manager*: receives the EU's security requirements and transforms them into a set of SLAs.
2. *Supply Chain Manager* creates the list of service offers, the so-called supply chains.
3. *Security Reasoner* evaluates the security levels of different supply chains and compares them with the EU's security requirements.

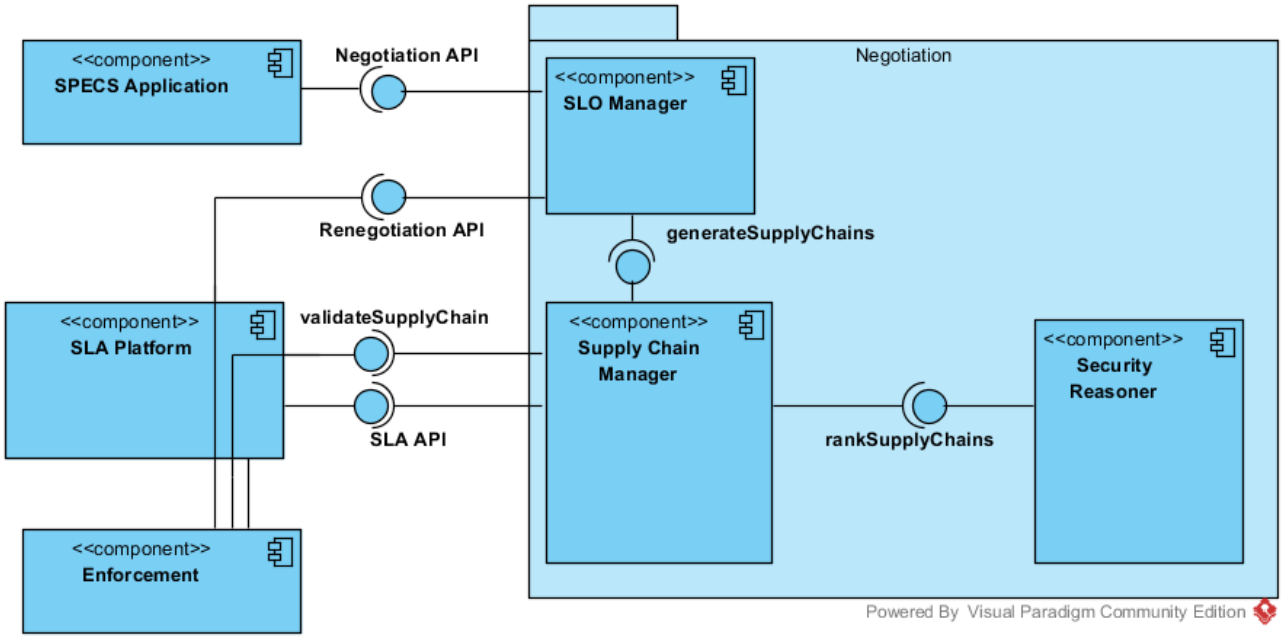


Figure 3 Negotiation Module Architecture

3.1. Status of development activities

In Table 1 we schematically report the list of SPECS software components under development associated to the Negotiation module as discussed in D1.1.2 and D2.1.2, together with the requirements they respectively cover.

Negotiation Module	**SPECS Software Components**		
Requirements	***SLO Manager***	***Supply Chain Manager***	***Security Reasoner***
SLANEG_R1	X		
SLANEG_R2	X		
SLANEG_R3	X		
SLANEG_R4	X		
SLANEG_R5		X	
SLANEG_R6	X		
SLANEG_R7			X
SLANEG_R8			X
SLANEG_R9			X
SLANEG_R10	X	X	
SLANEG_R11			X
SLANEG_R12	X		
SLANEG_R13	X		
SLANEG_R14	X	X	

SLANEG_R15	X		
SLANEG_R16	X		
SLANEG_R17	X		
SLANEG_R18 (*)			
SLANEG_R19 (*)			
SLANEG_R20 (*)			
SLANEG_R21 (*)			
SLANEG_R22 (*)			
SLANEG_R23 (*)			
SLANEG_R24 (*)			
SLANEG_R25 (*)			
SLANEG_R26 (*)			
SLANEG_R27 (*)			
SLANEG_R28 (*)			
SLANEG_R29 (*)			
SLANEG_R30 (*)			
SLANEG_R31 (*)			
SLANEG_R32 (*)			
SLANEG_R33 (*)			
SLAPL_R10		X	
SLAPL_R14			X
SLAPL_R21		X	
SLAPL_R33	X		
ENF_PLAN_R1		X	
ENF_PLAN_R12		X	

Table 1. SPECS Components related to the Negotiation module and related requirements

There are 33 requirements for the Negotiation module (SLANEG_1 – SLANEG_33), out of which SLANEG_R18 (*) - SLANEG_R33 (*) are not yet covered in the current design of the Negotiation module, an updated version of the design will be available at month 24 and will be included in the next iteration of this report.

In Table 2, we report the status of actual development of all SPECS artifacts associated to the Negotiation module. In particular, these artifacts include both the components and conceptual models that need to be developed in the tasks of WP2 until the end of the project.

Module	Artifacts under development	Status
Negotiation module	component:SLOManager	Available
	component:SupplyChain	Work in progress
	component:SecurityReasoner	Available[2]
	model:SLAConceptualModel	Work in progress
	model:SLAMachineReadableFormat	Work in progress
	model:SecurityMetricsCatalogue	Work in progress
	model:NegotiationProtocol	Work in progress
	model:NegotiationAPI	Work in progress

Table 2. Negotiation Module Implementation Status

The SLO Manager and the Security Reasoner components are publicly available on Bitbucket [1][2] and respectively cover 10 out of 17 of the requirements, that is, about the 30% of

[2] This component is currently available as a standalone application and the core functionalities will be accessible via REST APIs in the next release of this deliverable.

elicited requirements related to the Negotiation module (about 80% related to the SLO Manager component) and 4 out of 5 of the requirements, that is, about the 80% of elicited requirements related to the Security Reasoner component.

We are currently working on the Supply Chain Manager component and to the interface of the Security Reasoner to develop the proper APIs for full integration within the Negotiation module. All these three components use the model artifacts which are not yet finalized. In this deliverable we use the results of the current status which were developed in close relationship with the ongoing tasks of WP2 (*Negotiation*), WP3 (*Monitoring*) and WP4 (*Enforcement*).

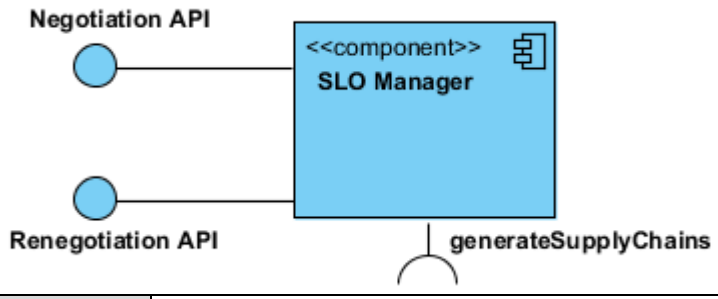
For example, preliminary versions of the *SLAConceptualModel* and *SecurityMetricsCatalogue* are available in D2.2.1. In order to be integrated in the SPECS solution, the *SLAConceptualModel* should be translated into a machine-readable format meaning that the *SLAMachineReadableFormat* model should be available. Currently, the *SLAMachineReadableFormat* model is based on WS-Agreement [3], which was used for formalizing the security features the EU can negotiate and SPECS framework can monitor and enforce, as well as for advertising the capabilities of service providers and creating agreements based on templates.

Moreover, the catalogue will be enriched with metrics which are *monitorable* and enforceable.

3.2. SLO Manager

SLO Manager component consists of a RESTful web service that obtains EU security requirements, translates them into a machine-readable format to be used by the other Negotiation modules components and SPECS modules, and finally translates the result of negotiation into SLA offers, from which the EU must choose one SLA to be signed.

SLO Manager Interfaces definition are presented in Table 3 SLO Manager Interfaces.

<table>
<tr><th></th><th colspan="2">SLO Manager Component (Figure 2)</th></tr>
<tr><td>Functional description</td><td colspan="2">Transforms EU security requirements into a hierarchy of SLAs based on the security levels they offer.</td></tr>
<tr><td>Overview</td><td colspan="2"> 
</td></tr>
<tr><td>Interfaces</td><td>Provide</td><td>Negotiation API offers the following negotiation functionalities:
1) Description: The SLO Manager receives from the EU, through the SPECS Application, his desired security capabilities.
a) Input: List of security capabilities (Step 1).
b) Output: Metadata associated to each security capability. Metadata is information regarding the control framework, associated control families, security controls, and metrics (Step 1.3).</td></tr>
</table>

		2) *Description*: The *SPECS Application* asks the *SLO Manager* to submit the Service Description Terms (SDTs)[3] of its services. a) *Input*: List of desired EU security capabilities together with their metadata (Step 2). b) *Output*: A list of SDTs[5] (Step 2.1). 3) *Description*: For each SDT, the *SLO Manager* sends to the *SPECS Application* the metrics and associated SLOs that SPECS can guarantee. a) *Input*: List of capabilities (Step 3). b) *Output*: List of metrics and SLOs associated to each capability (Step 3.1). 4) *Description*: For each metric and associated SLO, the *SLO Manager* builds SLAs by using the ranked and valid supply chains offered by the other modules (*Enforcement* Module) and components (*Security Reasoner*). a) *Input*: List of metrics, SLOs and desired weights (Step 4). b) *Output*: List of SLOs and associated user-defined weights (Step 4.2.4.1.1).
	Consume	The *SLO Manager* needs the following functionalities: 1) *Description*: *SLO Manager* receives from the *SLA Platform* the security capabilities desired by the EU together with their metadata. a) *Input*: List of security capabilities (Step 1.1). b) *Output*: For each security capability, associated metadata (Step 1.2). 2) *Description*: The *SLO Manager* receives from the *Security Reasoner* ranked and valid supply chains which will be further used to construct the corresponding SLAs (SLA offers). a) *Input*: Ranked and valid supply chains (Step 4.2.4). b) *Output*: Ranked and valid SLAs (Step 4.2.4.1).
Relationship with external modules	1. *SPECS Application* 2. *SLA Platform*	

Table 3 SLO Manager Interfaces

3.2.1. Installation

The *SLO Manager* component is a Java web application and has to be deployed on an application server.

We give here brief instalation guides, however, a readme file is available at [1] which will be kept updated if feedback from the users of the prototype is received.

[3] Service Description Terms (SDTs) are elements of the WS-Agreement language schema which give a functional description of the service to provide. Therefore, the service description terms contain a domain specific description of the service.

[5] For one secured Target Service, at least one SDT is constructed.

Prerequisites for using the prototype are:

- Java 1.7 [4] ;
- J2EE application server (e.g., Apache Tomcat 7 [5]);
- Apache Maven Project [6].

For installing the items above, please refer to the corresponding user manuals.

Installation steps:

1. Using a terminal change directory into the root of *SLOManager* directory.
2. Run the command:

   ```
   ~$mvn package
   ```

   This will create a *.war* file in the directory named `target` named SLOManager.war.
3. Deploy the war file to your application server [7].

3.2.2. Usage

The *SLO Manager* can be accessed through a RESTful interface which is compliant with the API guidelines presented in D1.3. After the successful deployment of the web application, the REST API can be accessed at:

```
%TOMCAT_APPLICATION_PATH%/rest/*
```

In the case of a deployment on the local machine with the application path `slomanager`, the API can be accessed at:

```
localhost:8080/slomanager/rest/*
```

This prototype of the *SLO Manager* exposes the following resources in a RESTful manner:

1. Service Description Term

A Service Description Term (SDT) is returned for a capability or list of capabilities. An SDT provides a full or partial functional description of a service. One or more SDTs can be related to a service.

Methods:

- `POST /sdts`
  - Consumes: `application/json`
  - Produces: `application/json`
  - Returns: an id for the Service Description Term containing information about security capabilities and security metrics related to a list of capabilities taken as input from the request body.

  (see Appendix 1. Example of Input/Output of Service Description Term REST Access Methods
- `GET /sdts/{id}`
  - Consumes: `text/html`
  - Produces: `application/json`
  - Returns: the Service Description Term associated with *id* containing information about the security capabilities and security metrics requested a priori.

  (See Appendix 1. Example of Input/Output of Service Description Term REST Access MethodsExample 2).

2. Service Level Objectives

A list service level objectives (SLOs) is returned for a capability, or for a list of capabilities.

Methods:

- `POST /slos`
  - Consumes: `application/json`
  - Produces: `application/json`
  - Returns: - the combined list of SLOs, for all the capabilities in the list taken as input from the request body.

  (See Appendix 2. Example of Input/Output of Service Level Objectives REST Access MethodsExample 1)

3. Service Level Agreements

A list of Service Level Agreements (SLAs) are returned based on users preferences. In the next version of the prototype, the SLAs will be *ranked and validated* since the *SLO Manager* will be integrated with the *Supply Chain Manager* and the *Security Reasoner.* The information which will be contained into an SLA is presented schematically in Table 4 and Table 5). This information is not yet validated, nor ranked.

SLA entry	#1	#2
Capability	WebPool	TLS
Metric	M1	M4
SLOs	M1>3	M4=no
Resources	AMI-xxx, HW: t1-micro, Amazon, US-EAST	

Table 4 SLA Offer 1

SLA entry	#1	#2
Capability	WebPool	TLS
Metric	M1	M4
SLOs	M1>4	M4=yes
Resources	AMI-xxx, HW: t1-micro, Amazon, US-EAST	

Table 5 SLA Offer 2

Out of this two SLA Offers, the EU has to choose one for signing.

One can observe that additionally to the security capabilities and their metadata, the negotiated SLAs contain also information on the *resources* on top of which the security capabilities are installed. These information will be delivered by the *Supply Chain Manager*.

Methods:

- `POST /offers`
  - Consumes: `application/json`
  - Produces: `application/json`
  - Returns: A list containing ranked and valid SLAs. Each SLA is constructed from the set of SLOs taken as input, and an associated SDT, whose id should be found in the request body. The request body should have the following form:

```
{
        "stdId" : "86",
        "slos" : [*SLOs chosen by the EU*]
}
```

  - **Remark**: We strongly recommend that this call is made in an asynchronous manner, since the SLAs have to be ranked and validated, which is time consuming.

(See Appendix 3. Example of Input/Output of Service Level Agreement Offers REST Access MethodsExample 1)

3.3. Security Reasoner Standalone Application

The *Security Reasoner* is the component devoted to evaluate and rank Security SLAs in order to allow the SPECS customers to choose the best SLA offer. In Deliverable D2.2.1 we presented two different evaluation techniques to evaluate and rank the Security SLA offers from different CSP. The current implementation of the Security Reasoner is based on the Reference Evaluation Methodology (REM) evaluation technique [11] and it uses the Consensus Assessments Initiative Questionnaire (CAIQ) [10] proposed by CSA as a reference template to represent the security provisions in the SLA .

The first release of the *Security Reasoner* is accessible as a standalone SPECS Application and it does not use any other *SPECS Core services* except the SLA repositories. It currently offers the capability to evaluate and compare different SLAs and security related terms.

Figure 4 summarizes the architecture of the application, which uses the *Security Reasoner interface* (*Evaluation Interface*), described in Deliverable D1.3. The *Evaluation Interface* offers, among the other, the *rankSupplyChain* method, in accordance with the component diagram reported in Figure 3. In the next version, the *Evaluation API* will be offered through a REST Interface.

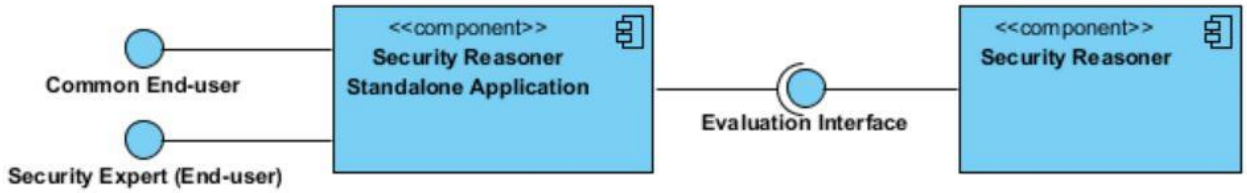
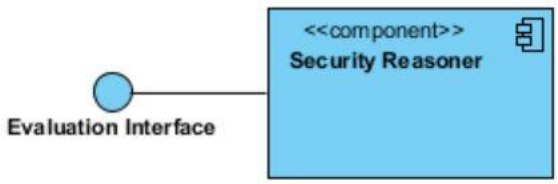


Figure 4 Security Reasoner Standalone Application

Table 6 summarizes the main features of the *Security Reasoner* application.

	Security Reasoner
Functional description	Evaluate a CSP according to the REM methodology. This function takes in consideration: a) the description of the target of evaluation (currently with the REM we can evaluate any service, CSP, SLA or a supply chain), b) the description of EU preferences. Description of the target of evaluation is expressed through an XML file that codes the features to be evaluated (see [8], [9] for further details). In particular, state of art implementation supports an XML conceptual model representing the CSP parameters as reported in CSA CAIQ questionnaire [10]. Description of EU preferences are expressed in an additional XML file having the same structure of the previous XML file, but containing an additional label with declared importance (weights) for each node of the xml file.
Overview	

Interfaces	**Evaluation Interface**	The interface offered by the *Security Reasoner* is currently a draft prototype in Java. It will be subject to changes in the next release of the component development.
Relationship with external modules	*None*	

Table 6 Security Reasoner Component

3.3.1. Installation

Security Reasoner standalone application runs as a Web Application based on servlet. The application is available for download on the SPECS Repository at [2].

Prerequisites for using the prototype are:

- Tomcat v7

Install Application on Tomcat:

1. Download war file from https://bitbucket.org/specs-team/specs-negotiation-securityreasoner-rem-standalone/downloads/specs-negotiation-SecurityReasoner-REM-standalone.war
2. Deploy the war in Tomcat using Tomcat admin.

3.3.2. Usage

The *Security Reasoner Application* enables a user to access based on his role. According to the requirements elicited and discussed in D2.1.2, we assume the following two roles:

- Not-expert end-user, he/she can evaluate a target (e.g., a CSP) according to an already available template of SLO;
- Security expert, he/she is able to build and/or upload and edit a SLA based on the CAIQs template, furthermore he/she can express evaluation criteria and give importance to each security domain and security controls declared (for details on these concepts, see [8], [9], [11]).

In order to access the application, a user needs to access to the URL where it has been deployed.

The functionalities provided to a not-expert user are:

- `Single Evaluation`, which enables a user to select a CSP among a list of available providers and use the evaluation criteria proposed (this has been proposed by Security Experts and embedded in the application code, it cannot be modified by a not-expert user). Furthermore this function enables both the evaluation of the whole SLA and the evaluation of single domains and controls included in the SLA. Since we represented the SLA as an XML file, the evaluation function returns a tree and it is possible to evaluate both the whole SLA (with an aggregated function) and the single security domains (tree internal nodes). According to the REM methodology, the result of the ranking is a value between 0 (lower security level) and 3 (higher security level).

- `Comparison Evaluation`, which enables a user to select two or more CSPs among a list of available providers. The evaluation function returns a Bar graph

comparing the evaluation of the CSPs with values among 0 (lower security level) and 3 (higher security level). Even in this case, it is possible to compare both the whole SLA (with an aggregated function) and the single security domains (tree internal nodes).

As said, the `Evaluation` founds on the availability of CAIQ on the system and on the definition of an evaluation criteria, which can be configured and/or customized only by a Security Expert.

The functionalities provided to an expert user are:

- `Upload the CAIQ template`, it can be uploaded in csv format (that can be downloaded as an excel files from the CSA web site [10]). Furthermore, during the evaluation, if the CAIQ is not correctly/completely compiled, a form is proposed with CSPs replies that enable the security expert to `Edit the CAIQ` and set a Yes/No value for the selected field. The Security Expert can verify (and alter) the CAIQ answers and edit them, according to their considerations and evaluations on the declarations.

`Upload/Edit weights`, the Security Expert can upload a precompiled XML file containing the level of importance (weight) he/she gives, according to the end-user preferences, for each security domain of the CAIQ. Such weights can be edited online through a dedicated interface defining the weights associated to each domain and control declared in the CAIQ.

Conclusions

This document presents the first implementation of the *SLO Manager* and *Security Reasoner* components of the *Negotiation* module.

The *SLO Manager* has been fully developed and implemented, and we do not plan to present new functionalities of it in the future. On the other side, as discussed in Section 3.1, the implementation will be extended and all enhancements will be presented in the next iteration of the prototype deliverable D2.3.2 and in the Final Report D2.3.3 at month 30. We plan to have all enhancements implemented at month 24, after this date only maintenance activities will take place.

The *Security Reasoner* is currently accessible as a standalone SPECS Application, however at the next release (month 30) its functionalities will be offered through a REST Interface. Moreover, the *Security Reasoner* functionalities will be enhanced in order to satisfy other components requests.

Based on user stories and elicited EU requirements as well as coverage of requirements, the implementation plan for the components of the *Negotiation* module is illustrated in the following table. The plan ensures that the most valuable requirements will be implemented by the end of the project.

Module	Component	M07 – M12	M13 – M18	M19 – M30
Negotiation	*SLO Manager*			
	Supply Chain Manager			
	Security Reasoner			

Table 7 Negotiation Module Components Implementation Plan

Module	Model	M07 – M12	M13 – M18	M19 – M30
Negotiation	SLAConceptualModel			
	SLAMachineReadableFormat			
	SecurityMetricsCatalogue			
	NegotiationProtocol			
	NegotiationAPI			

Table 8 Negotiation Module Models Implementation Plan

Bibliography

[1] SPECS, "Negotiation Module. SLO Manager Component," 2015. [Online]. Available: https://bitbucket.org/specs-team/specs-negotiation-slomanager.

[2] SPECS, "SPECS Security Reasoner Application." [Online]. Available: https://bitbucket.org/specs-team/specs-negotiation-securityreasoner-rem-standalone.

[3] "WS-Agreement." [Online]. Available: https://packcs-e0.scai.fraunhofer.de/wsag4j/wsag/overview.html.

[4] "Java Platform, Standard Edition." [Online]. Available: http://www.oracle.com/technetwork/java/javase/downloads/java-se-jre-7-download-432155.html.

[5] "Apache Tomcat 7," 2015. [Online]. Available: https://tomcat.apache.org/download-70.cgi.

[6] "Apache Maven Project." [Online]. Available: https://maven.apache.org/.

[7] "Apache Tomcat 7 Deployment." [Online]. Available: https://tomcat.apache.org/tomcat-7.0-doc/deployer-howto.html.

[8] Cloud Security Alliance, "Consensus Assessments Initiative Questionnaire STAR Program." [Online]. Available: https://cloudsecurityalliance.org/research/cai/.

[9] Alfieri Giuseppe, Casola Valentina, Rak Massimiliano, "A Cloud Application for Security Service Level Agreement Evaluation," in *4th International Conference on Cloud Computing and Services Science (CLOSER 2014)*, 2014.

[10] Cloud Security Alliance, "CSA CAIQ Questionnaire." [Online]. Available: https://cloudsecurityalliance.org/star/?r=4376#_registry.

[11] Casola V., Mazzeo A., Mazzocca N., Vittorini V., "A policy-based methodology for security evaluation: A Security Metric for Public Key Infrastructures", In the Journal of Computer Security, Volume 15 n.2, pp IosPress, 2007.

Appendix 1. Example of Input/Output of Service Description Term REST Access Methods

Example 1

Method: `POST /sdts`
Request body: `["TLS"]`
Output: `{"stdId" : "86"}`

Example 2

Method: `GET /sdts/86`
Output:

```
{
    "serviceDescription": {
        "capabilities": [{
            "control_framework": [{
                "security_control": [{
                    "id": "EKM-01",
                    "name": null,
                    "otherAttributes": {}
                }, {
                    "control_description": "",
                    "importance_weight": "MEDIUM",
                    "id": "EKM-03",
                    "name": null,
                    "otherAttributes": {}
                }, {
                    "id": "IAM-02",
                    "name": null,
                    "otherAttributes": {}
                }, {
                    "id": "IAM-09",
                    "name": null,
                    "otherAttributes": {}
                }, {
                    "id": "AAC-03",
                    "name": null,
                    "otherAttributes": {}
                }],
                "id": "CCM",
                "frameworkName": "CCM3.0"
            }],
            "id": "TLS",
            "name": "TLS",
            "description": null
        }],
        "security_metrics": [{
                "name": "M3",
                "metricDefinition": {}
            }, {
                "name": "M4",
                "metricDefinition": {
                    "unit": "level",
                    "scale": {
                        "Qualitative": "ORDINAL"
                    }
                }
            },
            /* Other metrics descriptions here*/
        ]
    }
}
```

Appendix 2. Example of Input/Output of Service Level Objectives REST Access Methods

Example 1

Method: `POST /slos`
Request body: `["TLS", "WebPool", "E2EE"]`
Output:

```
{
    "SLO": [{
        "name": "M1"
    }, {
        "name": "M2"
    }, {
        "name": "M15"
    }, {
        "name": "M3"
    }, {
        "name": "M4"
    }, {
        "name": "M5"
    }, {
        "name": "M6"
    }, {
        "name": "M7"
    }, {
        "name": "M10"
    }, {
        "name": "M15"
    }]
}
```

Appendix 3. Example of Input/Output of Service Level Agreement Offers REST Access Methods

Example 1

Method: `POST /offers`
Request body:

```
{
      "sdtId" : "86",
      "slos": [{
              "expression": {
                  "value": "4",
                  "op": "ge"
              },
              "name": "specs_e2ee_M15"
          }, {
              "expression": {
                  "value": "3",
                  "op": "ge"
              },
              "name": "specs_diversity_M1"
          }, {
              "expression": {
                  "value": "forbidden",
                  "op": "eq"
              },
              "name": "specs_tls_M4"
          }]
}
```

Output:

```
[ /* SLA Offer 1*/ {
        "wsag:Name": "Y2-SPECS-APP",
        "wsag:Context": {
            "wsag:AgreementInitiator": "$SPECS-CUSTOMER",
            "wsag:AgreementResponder": "$SPECS-APPLICATION",
            "wsag:ServiceProvider": "AgreementResponder",
            "wsag:ExpirationTime": 1429948387712,
            "wsag:TemplateName": "Y2-APP-TEMPLATE"
        },
        "wsag:Terms": {
            "all": [
                /*GuaranteeTerm */
                {
                    "serviceLevelObjective": {
                        "wsag:CustomServiceLevel": {
                            "objectiveList": {
                                "SLO": [{
                                    "expression": {
                                        "value": "3",
                                        "op": "ge"
                                    },
                                    "name": "specs_diversity_M1"
                                }]
                            }
                        }
                    },
                    "ServiceScope": {
                        "wsag:ServiceName": "SecureWebServer"
                    },
```

```
                    "QualifyingCondition": true
                },
                /*ServiceProperties*/
                {
                    "wsag:Name": "//specs:capability[@id='WebPool']",
                    "Wsag:ServiceName": "SecureWebServer",
                    "wsag:VariableSet": [{
                        "wsag:Name": "specs_diversity_M1",
                        "wsag:Metric": "M1",
                        "wsag:Location":
"//specs:security_control[@id='DIVERSITY_BCR-01']"
                    }]
                },
                /*GuaranteeTerm*/
                {
                    "serviceLevelObjective": {
                        "wsag:CustomServiceLevel": {
                            "objectiveList": {
                                "SLO": [{
                                    "expression": {
                                        "value": "4",
                                        "op": "g3"
                                    },
                                    "name": "specs_e2ee_M15"
                                }]
                            }
                        }
                    },
                    "ServiceScope": {
                        "wsag:ServiceName": "SecureWebServer"
                    },
                    "QualifyingCondition": true
                },
                /*ServiceProperties*/
                {
                    "wsag:Name": "//specs:capability[@id='E2EE']",
                    "Wsag:ServiceName": "SecureWebServer",
                    "wsag:VariableSet": [{
                        "wsag:Name": "specs_e2ee_M15",
                        "wsag:Metric": "M15",
                        "wsag:Location":
"//specs:security_control[@id='E2EE_EKM-01']"
                    }]
                },
                /*GuaranteeTerm*/
                {
                    "serviceLevelObjective": {
                        "wsag:CustomServiceLevel": {
                            "objectiveList": {
                                "SLO": [{
                                    "expression": {
                                        "value": "forbidden",
                                        "op": "eq"
                                    },
                                    "name": "specs_tls_M4"
                                }]
                            }
                        }
                    },
                    "ServiceScope": {
                        "wsag:ServiceName": "SecureWebServer"
                    },
```

```
                    "QualifyingCondition": true
                },
                /*ServiceProperties*/
                {
                    "wsag:Name": "//specs:capability[@id='TLS']",
                    "Wsag:ServiceName": "SecureWebServer",
                    "wsag:VariableSet": [{
                        "wsag:Name": "specs_tls_M4",
                        "wsag:Metric": "M4",
                        "wsag:Location": "//specs:security_control[@id='TLS_EKM-
03']"
                    }]
                }, /*ServiceDescriptionTerm*/ {
                    "wsag:Name": "Delivery",
                    "wsag:ServiceName": "SecureWebServer",
                    "specs:serviceDescription": {
                        "capabilities": [{
                            "control_framework": [{
                                "security_control": [{
                                    "id": "DIVERSITY_BCR-01",
                                    "name": null,
                                    "otherAttributes": {}
                                }],
                                "id": "CCM",
                                "frameworkName": "CCM 3.0"
                            }],
                            "id": "Diversity",
                            "name": "WebPool",
                            "description": null
                        }, {
                            "control_framework": [{
                                "security_control": [{
                                    "id": "E2EE_EKM-01",
                                    "name": null,
                                    "otherAttributes": {}
                                }],
                                "id": "CCM",
                                "frameworkName": "CCM 3.0"
                            }],
                            "id": "E2EE",
                            "name": "E2EE",
                            "description": null
                        }, {
                            "control_framework": [{
                                "security_control": [{
                                    "id": "EKM-01",
                                    "name": null,
                                    "otherAttributes": {}
                                }, {
                                    "control_description": " ",
                                    "importance_weight": "MEDIUM",
                                    "id": "TLS_EKM-03",
                                    "name": null,
                                    "otherAttributes": {}
                                }, {
                                    "id": "TLS_IAM-02",
                                    "name": null,
                                    "otherAttributes": {}
                                }, {
                                    "id": "TLS_IAM-09",
                                    "name": null,
                                    "otherAttributes": {}
```

```
                                    }, {
                                        "id": "TLS_AAC-03",
                                        "name": null,
                                        "otherAttributes": {}
                                    }],
                                    "id": "CCM",
                                    "frameworkName": "CCM 3.0"
                                }],
                                "id": "TLS",
                                "name": "TLS",
                                "description": null
                            }],
                            "security_metrics": [{
                                "name": "M1",
                                "metricDefinition": {}
                            }, {
                                "name": "M2",
                                "metricDefinition": {
                                    "unit": "level",
                                    "scale": {
                                        "Qualitative": "ORDINAL"
                                    }
                                }
                            }, {
                                "name": "M15",
                                "metricDefinition": {}
                            }, {
                                "name": "M3",
                                "metricDefinition": {}
                            }, {
                                "name": "M4",
                                "metricDefinition": {
                                    "unit": "level",
                                    "scale": {
                                        "Qualitative": "ORDINAL"
                                    }
                                }
                            }, {
                                "name": "M5",
                                "metricDefinition": {}
                            }, {
                                "name": "M6",
                                "metricDefinition": {}
                            }, {
                                "name": "M7",
                                "metricDefinition": {}
                            }, {
                                "name": "M10",
                                "metricDefinition": {}
                            }, {
                                "name": "M15",
                                "metricDefinition": {}
                            }]
                        }
                    }
                ]
            }
        }
        /*... SLA Offer n */
]
```