Project no. FP7-216556

Instrument: Integrated Project (IP)

Objective ICT-2007.1.2: Service and Software Architectures, Infrastructures and Engineering

Deliverable D.B1a
Scientific Evaluation Report

Keywords:
Web Site, Service Level Agreement, Service-Oriented Infrastructure

Due date of deliverable: 31st July 2011
Actual submission to EC date: 29th July 2011

Start date of project: 1st June 2008
Duration: 38 months

Lead contractor for this deliverable: XLAB

<table>
<thead>
<tr>
<th>Dissemination level</th>
<th>PU</th>
<th>Public</th>
<th>Yes</th>
</tr>
</thead>
</table>

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)
### Document Status

<table>
<thead>
<tr>
<th>Deliverable Lead</th>
<th>Miha Stopar, XLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reviewer 1</td>
<td>Francesco Torelli</td>
</tr>
<tr>
<td>Reviewer 2</td>
<td>Ulrich Winkler</td>
</tr>
<tr>
<td>PMT Reviewer</td>
<td>Klaus-Dieter Platte</td>
</tr>
<tr>
<td>Complete version submitted to reviewers</td>
<td>29.06.2011</td>
</tr>
<tr>
<td>Comments of reviewer 1 received</td>
<td>22.07.2011</td>
</tr>
<tr>
<td>Comments of reviewer 2 received</td>
<td>22.07.2011</td>
</tr>
<tr>
<td>Revised deliverable submitted to PMT</td>
<td>24.07.2011</td>
</tr>
<tr>
<td>PMT Approval</td>
<td>26.07.2011</td>
</tr>
</tbody>
</table>

### Contributors

<table>
<thead>
<tr>
<th>Partner</th>
<th>Contributors</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENG</td>
<td>Francesco Torelli, Paolo Zampognaro, Keven Kearney</td>
</tr>
<tr>
<td>TA</td>
<td>Mark Evenson, Alfred Tellian, Andreas Woratschek</td>
</tr>
<tr>
<td>XLAB</td>
<td>Gregor Berginc, Miha Stopar, Gregor Pipan, Miha Vuk, Primoz Hadalin</td>
</tr>
<tr>
<td>SAP</td>
<td>Tariq Ellahi, Wolfgang Theilmann</td>
</tr>
<tr>
<td>Intel</td>
<td>John Kennedy, Michael Nolan, Joe Butler, Victor Bayon</td>
</tr>
<tr>
<td>TID</td>
<td>Juan Lambea Rueda, Carlos Bueno Royo, Alfonso Castro, S. G. Gomez, Beatriz Fuentes</td>
</tr>
<tr>
<td>UDO</td>
<td>Philipp Wieder, Constantinos Kotsokalis, Miguel Rojas</td>
</tr>
<tr>
<td>FZI</td>
<td>Christof Momm, Franz Brosch</td>
</tr>
<tr>
<td>GPI</td>
<td>Giampaolo Armellin</td>
</tr>
<tr>
<td>City</td>
<td>Davide Lorenzoli, Howard Foster</td>
</tr>
<tr>
<td>PMI</td>
<td>Luciano Baresi</td>
</tr>
</tbody>
</table>

### Notices

The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The
above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2009 by the SLA@SOI consortium.

* Other names and brands may be claimed as the property of others.

This work is licensed under a Creative Commons Attribution 3.0 License.

### Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>15.4.2011</td>
<td>Vuk</td>
<td>Y3 basics</td>
</tr>
<tr>
<td>0.2</td>
<td>3.5.2011</td>
<td>Vuk</td>
<td>Deliverable structure and first draft</td>
</tr>
<tr>
<td>0.3</td>
<td>30.5.2011</td>
<td>Miha Stopar</td>
<td>Graphs updated, evaluation conclusions updated.</td>
</tr>
<tr>
<td>1.0</td>
<td>22.6.2011</td>
<td>Miha Stopar</td>
<td>Finalization.</td>
</tr>
</tbody>
</table>
Executive Summary

The present document is the third version of DB1a. Some of the sections of this document are unchanged with respect to the first two versions, while others are new or updated. The new or changed sections are marked with a left-side blue border.

The main goal of DB1a is to present the requirements collected by work package B1 and to evaluate the scientific results produced by the A-Line with respect to such requirements.

In the first version of this deliverable we presented the process for requirements collection, the main features of the SLA@SOI framework identified during the execution of this process and a preliminary assessment of their implementation in Y1, performed by the A-Line. During Y1 we collected more than 200 requirements from internal and external sources. After a first analysis of the specified use cases against the scientific and technical objectives of the SLA@SOI framework, the B-Line decided to change or extend some of the use case scenarios, in order to better demonstrate the upcoming results from the A-Line. After the consolidation of these requirements, we found that the new scenarios cover all the scientific WP outcomes in a balanced way.

In the second version we added the technical evaluation of Y2 results performed both by the A-Line and by the B-Line use cases.

Following the production of a first version of the Framework components offering the requested features, the A-Line has provided a self-evaluation of the status of the implementation of each feature. The A-Line evaluation has been performed through both automatic tests and a human assessment performed through a survey. The B-Line evaluation of the developed software has also been performed through a specific survey that collects the judgements of each use case work-package on the quality of each one of the developed features. Such judgments are of course based on the experience accumulated by the B-Line during the usage of the framework for the development of the use cases. This evaluation also includes lessons learned and planned next steps.

In this third version of the deliverable we continue the approach from Y2, grounding all activities on the agreed list of features and focusing on requirements still to be implemented in Y3. One source for the identification of such requirements were the results of Y2 feature evaluation, and another source was the feedback received by external parties during the SLA@SOI presentation day. The third and the main source were the requirements expressed explicitly by the use cases during Y3 Requirement Gathering and Consolidation process.

As the technical evaluation is strictly related to the state of the art and to the scientific innovations of the Framework, also this version of the deliverable is accompanied with an updated report [1] on the state of the art. The A-Line specified in its evaluation the relationship of each feature with the scientific innovations.
Table of Contents

Executive Summary ........................................................................................................... 5
1 Introduction .................................................................................................................. 11
  1.1 Context and Scope ......................................................................................... 11
  1.2 Document Overview ...................................................................................... 11
2 State of the Art Analysis .......................................................................................... 13
3 Framework Features and Use Cases ......................................................................... 15
4 Y3 Requirement Process ......................................................................................... 20
  4.1 Requirement Gathering .................................................................................... 22
  4.2 Requirement Consolidation ............................................................................... 23
  4.3 Requirement Clarification & Assignment ....................................................... 23
  4.4 Requirement Implementation ........................................................................... 24
  4.5 Requirement Verification .................................................................................. 24
5 Requirements Analysis ............................................................................................. 24
6 Y3 Requirements and their Status .......................................................................... 28
  6.1 Framework Management .................................................................................. 29
  6.1.1 Framework Configuration & Setup ............................................................. 29
  6.1.2 Framework Model Configuration ............................................................... 29
  6.1.3 Framework Operation ............................................................................... 30
  6.1.4 Framework Access .................................................................................... 30
  6.2 Design & Development .................................................................................... 30
  6.2.1 Reference Software Manageability Components and Configuration .......... 30
  6.2.2 Service properties dependencies coding .................................................... 31
  6.2.3 Designing of QoS predictable systems ....................................................... 31
  6.3 Pre-offering ....................................................................................................... 31
  6.3.1 Customer Registration ............................................................................... 31
  6.3.2 3rd Parties Configuration .......................................................................... 32
  6.4 Service Offering ............................................................................................... 32
  6.4.1 Discovery of Serv. and SLAT .................................................................... 32
  6.4.2 Product Management ............................................................................... 32
  6.4.3 Product Discovery ..................................................................................... 32
  6.4.4 SLAT Definition ....................................................................................... 33
  6.5 Service Negotiation ......................................................................................... 33
  6.5.1 Business Negotiation ............................................................................... 33
  6.5.2 SLAT Customization ................................................................................. 33
  6.5.3 Business SLA planning for negotiation ....................................................... 34
  6.5.4 Automatic orchestration of customizable (re)negotiation protocol .......... 34
  6.5.5 Out of band SLA Registration .................................................................... 34
  6.6 Service Provisioning .......................................................................................... 34
  6.6.1 Coordination of Provisioning ...................................................................... 34
  6.6.2 Dynamic binding setting ............................................................................ 35
  6.6.3 Virtual hw. infrastructure provisioning ....................................................... 35
  6.6.4 Software provisioning ............................................................................... 35
  6.6.5 Software landscape ................................................................................... 36
  6.7 SLA enforcement .............................................................................................. 36
  6.7.1 Automated SLA enforcement coordination ............................................... 36
  6.7.2 Virtual hw. infrastructure Adjustment ....................................................... 36
  6.7.3 Software Adjustment .................................................................................. 36
  6.7.4 Automatic Binding ..................................................................................... 37
  6.7.5 Business adjustment ................................................................................... 37
  6.8 Runtime Prediction ............................................................................................ 37
6.8.1 Runtime infrastructure metrics prediction ........................................... 37
6.8.2 Runtime Software SLAs violations prediction ...................................... 38
6.9 Service Monitoring .................................................................................. 38
6.9.1 Virtual hardware infrastructure monitoring rules extraction ................. 38
6.10 Service Reporting ................................................................................. 39
6.10.1 SLA Management .............................................................................. 39
6.10.2 Push/Pull Business SLA Reporting .................................................... 39
6.10.3 SLA Status Reporting ........................................................................ 39
6.10.4 Virtual hw Infrastructure Reporting .................................................. 40
6.11 New Features ......................................................................................... 40
6.12 Sub-requirements stated by A-Line ......................................................... 40
6.13 External Requirements ........................................................................... 40
7 Evaluation Approach .................................................................................. 41
7.1 Scientific Metrics Evaluation ................................................................... 41
7.2 Testing ..................................................................................................... 42
7.3 Survey ..................................................................................................... 42
8 Feature Evaluation Results ........................................................................ 44
8.1 Framework Management ......................................................................... 52
8.2 Design & Development .......................................................................... 56
8.3 Pre-offering ............................................................................................. 60
8.4 Service Offering ...................................................................................... 61
8.5 Service Negotiation ................................................................................ 66
8.6 Service Provisioning ............................................................................... 72
8.7 SLA Enforcement ................................................................................... 78
8.8 Runtime Prediction ................................................................................ 83
8.9 Service Monitoring ................................................................................ 85
8.10 Service Reporting ................................................................................ 90
9 External Requirements ............................................................................... 94
10 Conclusions ............................................................................................. 98
10.1 Contributions and Achievements ......................................................... 98
10.2 Lessons learned .................................................................................... 98
10.3 Outlook ................................................................................................. 99
References ................................................................................................... 100
Appendix A: Glossary .................................................................................. 101
Appendix B: Abbreviations .......................................................................... 103
Appendix C: Industrial Use Cases Overview ............................................... 105
   Use case B3: ERP Hosting (SAP) ................................................................. 105
   Use case B4: Enterprise IT (Intel) ............................................................... 105
   Use case B5: Service Aggregator (Telekom Austria) ................................ 106
   Use case B6: E-Government (ENG) .......................................................... 106
Appendix D: Overview of External Sources ................................................ 106
Appendix E: Framework Features ............................................................... 107
   Framework Management ......................................................................... 109
   Design & Development .......................................................................... 111
   Pre-offering ............................................................................................. 112
   Service Offering ...................................................................................... 113
   Service Negotiation ................................................................................ 115
   Service Provisioning .............................................................................. 116
   SLA Enforcement ................................................................................... 118
   Runtime Prediction .................................................................................. 120
   Service Monitoring .................................................................................. 120
Table of Figures

Figure 1: Offline Features of the SLA@SOI Framework ........................................ 18
Figure 2: Runtime Features of the SLA@SOI Framework ........................................ 19
Figure 3: Requirement Process .................................................................................. 21
Figure 4: States of Requirement Fulfilment Process .................................................. 22
Figure 5: Count of requirements by their priority ...................................................... 25
Figure 6: Count of requirements by reporting use case ............................................. 25
Figure 7: Count of requirements by responsible work package .................................. 26
Figure 8: Count of requirements by responsible work package and status at the beginning of Y3 ................................................................. 27
Figure 9: Count of requirements by their status at the beginning of Y3 ...................... 27
Figure 10: Count of requirements by responsible work package and status at the end of Y3 .................................................................................................. 28
Figure 11: Count of requirements by their status at the end of Y3 ............................. 28
Figure 12: Average status reported by the A-Line for each category of features. 45
Figure 13: Average quality of the implementation judged by the B-Line for each category in Y2. .................................................................................................. 45
Figure 14: Average quality of the implementation judged by the B-Line for each category in Y3. .................................................................................................. 46
Figure 15: Average level of complexity reported by the B-Line for each category in Y2................................................................................................................. 46
Figure 16: Average level of complexity reported by the B-Line for each category in Y3................................................................................................................. 47
Figure 17: Average quality of the documentation judged by the B-Line for each category in Y2. .................................................................................................. 47
Figure 18: Average quality of the documentation judged by the B-Line for each category in Y3. .................................................................................................. 48
Figure 19: Average quality of the Architecture judged by the B-Line for each category in Y2. .................................................................................................. 49
Figure 20: Average quality of the Architecture judged by the B-Line for each category in Y3. .................................................................................................. 49
Figure 21: Average quality of the APIs judged by the B-Line for each category in Y2. .................................................................................................. 50
Figure 22: Average quality of the APIs judged by the B-Line for each category in Y3. .................................................................................................. 50
Figure 23: Number of issues reported by the B-Line for each category in Y2 ................................. 51
Figure 24: Number of issues reported by the B-Line for each category in Y3 ................................. 51

List of Tables

Table 1: Document changes against previous version ............................................. 12
Table 2: Required Framework Features ......................................................... 17
Table 3: Requirements Wiki Table ................................................................. 24
Table 4: Requirements Evaluation Goals ...................................................... 41
Table 5: Evaluation Survey ........................................................................... 43
Table 6: Required Framework Features (updated to Y3) ............................... 109
Table 7: Technical evaluation of Use Case B3 ............................................. 128
Table 8: Technical evaluation of Use Case B4 ............................................. 130
Table 9: Issues reported on Use Case B4 ................................................... 130
Table 10: Technical evaluation of Use Case B5 ......................................... 134
Table 11: Technical evaluation of Use Case B6 ......................................... 137
Table 12: Issues reported by Use Case B6 ............................................... 137
1 Introduction

1.1 Context and Scope

The present document is the third version of DB1a. Some of the sections of this document are unchanged with respect to the first version, while others are new or updated. The new or changed sections are marked with a left-side blue border.

In the first version we presented the process for requirements collection, the main features of the SLA@SOI framework identified during the execution of this process and a preliminary assessment of their implementation in Y1, performed by the A-Line. In the second version of the deliverable we added the technical evaluation of Y2 results performed both by the A-Line and by the B-Line use cases. This evaluation also includes lessons learned and planned next steps.

The third version of DB1a report on the requirements implemented during Y3 and framework evaluation at the end of the project. The Y2 framework evaluation was followed by a new round of requirements gathering in Y3, based on experience of Y2 framework adoption. The requirements gathering consisted of more phases of enquiries where use cases and external stakeholders (participants at SLA@SOI presentation day) were involved.

As the technical evaluation is strictly related to the state of the art and to the scientific innovations of the Framework, also this version of the deliverable is accompanied with an updated version of the state of the art and the A-Line reported in its evaluation the relationship of each feature with the scientific innovations.

Main achievement

The most important task for the third year of the project was to identify B-line issues in the process of the framework adoption in Y2. These issues have been identified on the base of the evaluation performed at the end of Y2 and have been transformed into new requirements. A special care was taken for the issues that received bad evaluation and for the relevant requirements. These requirements have been successfully implemented and verified as closed by B-line in the Y3.

1.2 Document Overview

This document is organized in the following way: Section 2 summarizes some of the considerations related to the State of the art that have influenced the project decisions. Section 3 describes framework features that are used as default requirement taxonomy. Section 2 has not been changed with respect to the first version and it summarizes main points from the previous version.

Section 4 describes the Y3 requirement process that has evolved from the previous one. All phases are described in details.

Section 5 presents a brief statistical overview of Y3 consolidated requirements, while section 6 presents their full list.

Section 7 describes the evaluation approach used in Y2 and Y3. While in the first version we presented also a sketch of the business evaluation approach, in this versions we describe just the technical evaluation, as in Y2 and Y3 the WP B1 releases a specific deliverable for the business evaluation (DB1b). So any business evaluation is included in that deliverable.
Section 8 presents the results of the evaluation. For each feature both A-Line and B-Line evaluations are reported. As each use case provided its own evaluation, section 8 reports the consolidated results, while the evaluation of each use case is reported in Appendix G.

Section 9 summarizes the external requirements from the beginning of the project and status of their implementation.

Finally, section 10 gives the conclusions and a summary of the future work.

**Table 1: Document changes against previous version**

<table>
<thead>
<tr>
<th>Section</th>
<th>Change overview</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 4</td>
<td>new</td>
</tr>
<tr>
<td>Section 5</td>
<td>new</td>
</tr>
<tr>
<td>Section 6</td>
<td>new</td>
</tr>
<tr>
<td>Section 7</td>
<td>updated</td>
</tr>
<tr>
<td>Section 8</td>
<td>updated</td>
</tr>
<tr>
<td>Section 10</td>
<td>updated</td>
</tr>
</tbody>
</table>
2 State of the Art Analysis

The state of the art analysis of topics relevant to SLA@SOI includes input from all A-line work packages. It comprises 90 contributions to various topics such as SLA modelling, SLA translation, information modelling, business processing, SLA monitoring and prediction. Due to the huge number of works considered, detailed results of the analysis are presented in a separate report [1] accompanying this deliverable.

The report [1] reviews several types of significant works such as research papers, projects and standards. It includes a description of various functional and non-functional requirements taken into account by the analysed works that may be potential candidates for inclusion in the SLA@SOI requirements (see also section 0). Furthermore, the report analyses the possibility of applying the results from the state of the art work to SLA@SOI and anticipates the necessary actions. To do the analysis, not only at a conceptual, but also at an implementation level, the report includes existing frameworks, components or services which realise the work and, where possible, states license model and maturity level.

The analysis has highlighted several limitations that the SLA@SOI project should try to overcome. First of all, we haven’t found any SLA management framework with complete coverage of eContracting, design-time and runtime prediction, software management, infrastructure management, as SLA@SOI attempts.

Covering such a huge set of functionalities in an integrated way requires a comprehensive model shared between the different components of the SLA@SOI platform.

We identified two main proposals pertinent for domain modelling in SLA@SOI: SID (NGOSS Shared Information and Data Model [2]), i.e. a set of standardized information definitions for modelling business entities developed by TeleManagement Forum, and WS-Agreement [3], i.e. an XML model and protocol developed from Open Grid Forum (OGF), for establishing agreement between parties. While the SID suffers of a limited support to SLA related concepts such as guarantee terms, penalties and rewards, WS-agreement is a very open standard that leaves many relevant aspects without a concrete definition both from the semantic and syntactic point of view. Moreover SID is more oriented to eContracting, whilst WS-Agreement is more concerned with software components, explicitly taking into account technical and low level aspects such as the WSDL interface of a service and its XML structure. In a sense the two specifications are complementary and a conceptual integration of the two would be valuable. A conceptual generalization is required also because SLA@SOI tries to apply the concept of service uniformly to all layers of the software stack.

There is no widely adopted solution for modelling an SLA hierarchy corresponding to a service composition hierarchy. Current works are often related to the composition of some specific QoS parameters, without proposing a general approach for relating SLAs between different layers.

The hierarchy of SLAs is strictly tied to the relationships between the software components that implement a service. Moreover the quality constraints expressed in a SLA are related to the quality of such components and of the related resources. Therefore, in order to automatically provision, predict and adjust the quality of services for satisfying SLAs, the SLA@SOI framework must also supply
a model to describe the software components that implement the services and their relationships.

As the SLA@SOI platform is also a software system realized through components, it would be desirable to use such a component model also to describe the SLA@SOI platform itself.

As a unifying standard for architectural components and in particular service components, we selected SCA (Service Component Architecture [4]) as the most promising specification. It has several implementations available and good chance to replace today's component technologies such as Java EE and .Net in the future. This model however does not cover several needs. In particular there is no possibility to associate, to a component, non-functional properties and information needed for prediction of their values. More in general there is no support in SCA for SLA aware management.

While various research efforts have centred on service management based on SLAs, currently there is no widely adopted solution for monitoring of SLAs or for dynamic binding of services based on constrains expressed in SLAs. Solutions need to be found, which are as compatible as possible with current standards and tools supporting SLA aware service management.

A particularly challenging objective is QoS prediction. Some of analysed approaches take into account single system properties only, e.g., CPU load, disk usage or network traffic. Many of the approaches use regression models, trend analysis and machine learning techniques, which all require a training session before being able to predict. To extend the applicability of these methods it is necessary to remove some of the above limitations.

At the infrastructure level we lack flexible and technology-independent models and interfaces through which both functional and non-functional aspects of infrastructure can be described and managed. For instance the Open Virtual Machine Format (OVF) allows describing functional properties of a virtual machine but it does not describe non-functional properties. In contrast, specific technologies such as Ganglia for monitoring can be considered very mature and have been adopted in SLA@SOI without any need for extensions. Also XMPP (set of open XML technologies for presence and real-time communication) has been identified as a good standard for distributed messaging at infrastructural level. No standard or widely adopted solution, however, is available for dynamic provision and re-provisioning.

Details on all the relevant technologies that are being used and improved by each scientific WP are presented in their deliverables.
3 Framework Features and Use Cases

In this section we describe the formulation of the SLA@SOI framework required features, coming from requirement consolidation process and the further feedbacks received in Y2.

The initial list of features was agreed during a dedicated meeting between A-Line and B-Line called “matchmaking meeting” and then refined. Each feature was associated with a specific A-Line WP and a specific step of the service lifecycle. While we maintain the association between the feature and the original collected requirements, the features represent the consolidated version of the relevant requirements and are used as reference for the implementation of the framework, the adoption by use case and the evaluation process.

Each feature is defined in order to assign the responsibility of its development to just one scientific work-package. For this reason, we included features coming from the splitting of the needs of use cases to the various scientific work packages. Therefore the features take into account both the needs of the use cases and the technical dependencies between scientific development tasks.

Some of the features in the category “Framework Management” and some features in “Design and Development” correspond to non-functional aspects or to development facilities. All the other features correspond to functionalities that the framework is expected to offer. Some of these functionalities are internal to the framework, while others are offered to external actors or applications. Some features are specific to a certain layer, while others are cross layer. Therefore when applicable each feature has been associated to a specific external actor (user) and to a specific layer, in accordance with the terminology standardized by the SLA@SOI glossary.

The features have been validated (i.e. accepted) by the A-Line, i.e. each scientific WP has recognized that they are able to understand the feature and the original requirements associated to them.

The following table gives a summary of all the identified features organized according to the decided categories. The table shows the scientific work package associated to the feature and the use cases that will use them according to the last definition of use case scenarios:

<table>
<thead>
<tr>
<th>WP</th>
<th>Feature</th>
<th>Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td>Framework Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Framework Configuration &amp; Setup</td>
<td>X</td>
</tr>
<tr>
<td>A1</td>
<td>Framework Model Configuration</td>
<td></td>
</tr>
</tbody>
</table>
### Use Cases

<table>
<thead>
<tr>
<th>WP</th>
<th>Feature</th>
<th>Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td>A1</td>
<td>Framework Operation</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Framework Access</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Design &amp; Development</strong></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>Reference Software Manageability Components and Configuration</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Service properties dependencies coding</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>Designing of QoS predictable systems</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>Manageability Design</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Pre-offering</strong></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Customer Registration</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>3rd Parties Configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Service Offering</strong></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Discovery of Serv. and SLAT</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>Product Management</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Product Discovery</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Product Definition</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>SLAT Definition</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td><strong>Service Negotiation</strong></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Business Negotiation</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>SLAT Customization</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Business SLA planning for negotiation</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Automatic orchestration of customisable (re)negotiation</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Out-of-band SLA Registration</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Service Provisioning</strong></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Coordination of Provisioning</td>
<td>X</td>
</tr>
<tr>
<td>A5</td>
<td>On the fly deploy of monitoring</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Dynamic binding setting</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>Virtual hardware infrastructure provisioning</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Software provisioning</td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>Software landscape</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td><strong>SLA enforcement</strong></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Automated SLA enforcement coordination</td>
<td>X</td>
</tr>
<tr>
<td>A4</td>
<td>Virtual hw. infrastructure Adjustment</td>
<td>X</td>
</tr>
</tbody>
</table>
Table 2: Required Framework Features

Figure 1 and Figure 2 depict the main relationships between the functional features (i.e., features that represent activities or functionalities) offered by the SLA@SOI Framework and the relationship with the external actors.

Each feature is represented as a UML Use Case. We distinguish between offline features and runtime features. An offline feature represents an activity that the SLA@SOI Framework must support in the phases that precede the provision of the services, such as the design and implementation of the service components and the configuration of the Framework. Such operations typically happen when the Framework is not running. Runtime features are functionalities offered by the Framework at runtime. Different colours are used to distinguish UML Use Cases belonging to different Feature Categories.
Figure 1: Offline Features of the SLA@SOI Framework

The UML diagram for Runtime Features clearly distinguish functionalities that are directly accessed by external actors, from functionalities that are internal to the framework, but needed to perform the functions asked by the external actors. Classical UML Use Cases dependencies are used to show the relationship between the features. A dependency between the features will likely imply a dependency between the implementation of the features at least at interface level.
Figure 2: Runtime Features of the SLA@SOI Framework

The complete description of framework features can be found in Appendix E: .
4 Y3 Requirement Process

SLA@SOI framework development is based on requirements from use cases and external stakeholders. These requirements evolved over time. Therefore we have repeated the requirements gathering in Y3, putting focus on the functionalities that are needed but were not yet properly implemented in Y2. The requirements gathering was based on the set of framework features defined in Y2 – basically the use cases had to specify what is missing or not working properly for each of the framework features. New set of requirements aimed to give a fresh, high quality guideline for a framework development and a better evaluation of framework usability.

The requirement process has been also updated and consists of:

- Collection process
  - Gathering
  - Consolidation
- Fulfilment process
  - Clarification and Assignment
  - Implementation
  - Verification

Requirements gathered from various sources are clarified and consolidated into a single consistent set of requirements that drives the software development. Each consolidated requirement has a reporter (use case(s)) and it has to be assigned to a single developer (A-Line person). During this assignment the requirement has to be clarified in order that assigned developer clearly understands the point of the reporter. When the requirement is implemented, the reporter has to verify it and if he confirms it, the Requirement Manager closes the requirement; otherwise he goes back to the developers to fix the open issues.

Through the third year of the project Requirement Manager was regularly checking the requirements status, asking developers from B-line to test the requirements implementation from A-line and was resolving the conflicts between reporters and assigned developers.
For the Requirement Fulfilment Process, the following requirement states are defined:

- **new** – not yet accepted or rejected
- **accepted** – the responsible A-Line person has confirmed that it will be implemented
- **fixed** – implemented, but not yet tested by the reporter
- **verified** – tested and confirmed by the reporter
- **supp** – already supported by Y2 implementation
- **later** – will be implemented later (possibly after the end of the project)
- **wontfix** – out of the scope of the project
- **invalid** – the requirement does not make sense
- **dup** – already covered by some other requirement

**Figure 3: Requirement Process**
The state diagram is shown in Figure 4.

![State diagram](image)

**Figure 4: States of Requirement Fulfilment Process**

### 4.1 Requirement Gathering

In Y3 we wanted to get the requirements that take the existing Y2 implementation into account – i.e. what is needed but not yet properly implemented. We have gathered the requirements from external (see section 9) and internal sources.

For **external requirement gathering** we have asked the visitors on the SLA@SOI Information Day (Brussels, Sep 2010 – end of Y2) to fill out the brief questionnaire: evaluation of the feature classes and proposing new requirements. We did not receive any requirement that would not be addressed before. On the other hand we have gathered additional requirements from partners of the SLA@SOI consortium and their internal projects, see Section 6.13 for more details.

The major part of the requirements process was focused on the **internal requirement gathering**. It was extensive and has targeted all industrial Use Cases (B3-B6). Questionnaire was based on framework features. For each feature the use cases have answered on:

- How do you see this feature? What does your UC need from this feature?
- What does your UC need in Y3 beyond Y2 feature implementation?

Use cases B4-B6 have provided answers for all features, marking those features that are irrelevant for them. For each requirement they have also defined its **priority** and the **current status of its implementation** as they see it.

Use case B3 was just in the middle of reorganization and redesign and has therefore skipped this requirement gathering.
The statuses are described in Appendix F: while priorities are divided into four categories:

**NOT-USED** – implementation not needed or Y2 implementation is already sufficient

**NICE-TO-HAVE** – implementation appreciated

**REQUIRED** – implementation required, but still possible workaround if not

**ESSENTIAL** – blocker, no reasonable workaround. Implementation absolutely required.

## 4.2 Requirement Consolidation

When the use cases have filled out the questionnaires, the Requirement Manager (B1 expert) has reviewed all inputs and clarified all unclear points and open issues in peer discussions with use case leaders.

Based on clarified inputs the Requirement Manager has identified the **consolidated (generalized) requirements** and set them the **priority**. They were reviewed and approved by the use case leaders, resulting in a consistent set or requirements representing all industrial use cases.

The result of this phase was a word document presenting all consolidated requirements and original inputs (with notes from further clarifications) from use cases. It was sent to all A-Line WP leaders.

## 4.3 Requirement Clarification & Assignment

For the Requirement Fulfilment Process the (consolidated) requirements were moved into a **wiki table** on the project intranet portal. All involved people were able to update this page, but the revision history was stored.

Table 3: Requirements Wiki Table shows an example, how the requirements were presented in the wiki table at the beginning of the process. Every requirement is uniquely identified by Y3_RID that acts as pointer to the requirement and to the related use case inputs. For those requirements that need longer description than wiki table can offer or need a formal clarification, a dedicated ticket was opened on the SLA@SOI Trac and the hyperlink was added to the wiki table.

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>More documentation on OSGI integration</td>
<td>REQ B4</td>
<td>Gabriele, A1</td>
<td>accepted</td>
<td>More clarification about the kind of documentation required.</td>
</tr>
<tr>
<td>1.1.3</td>
<td>WS-Agreement support needed</td>
<td>REQ B5</td>
<td>Peter/Edwin UDO, A5</td>
<td>later</td>
<td>Trying to provide, however not guaranteed due to time constraints and other higher priority requirements.</td>
</tr>
</tbody>
</table>
Table 3: Requirements Wiki Table

When a wiki table was created for each requirement a **responsible person** was proposed - an A-Line responsible developer or a WP leader. Typically it was a person responsible for the related feature. That person had to **clarify** potential open points with the reporter and **accept** or **reject** the requirement. In some rare cases it turned out that the proposed responsible person was not appropriate and the related WP leader was asked to propose more appropriate one.

Once this phase was finished, all requirements were in **accepted**, **supp**, **dup**, **wontfix** or **invalid** state (status).

### 4.4 Requirement Implementation

Once the responsible developer **implemented** and tested the requirement, he changed its status in the wiki table to **fixed** and informed the reporter(s).

Even though the description of this phase is very short, it was the most time consuming.

### 4.5 Requirement Verification

The primary reporter **tests** the solution, taking into account all aspects that he had in mind when the requirement was reported. If the solution covers all aspects, he changes the requirement status to **verified** and informs the Requirement Manager, otherwise he changes the status back to the accepted, writes a note with the reason for rejection and informs the developer.

Other use cases have a chance to review the implementation, the testing results and they can also repeat the tests if needed. All defects and concerns that are found are to be noted on the wiki table and/or the Trac ticket and sent to the responsible developer and primary reporter.

Once the Requirement Manager sees the verification is finished, he closes the requirement.

### 5 Requirements Analysis

At the end of the consolidation process, we counted a total of 67 new Y3 requirements. 66 of these requirements came from use cases, while 1 additional requirement came from A2 as a result of splitting the original requirement. External requirements that have not been implemented at this point have been merged into this new set of requirements (see section 9).

This section presents some basic statistics how Y3 requirements are distributed over **priorities**, **reporting use case**, **responsible work package** and **status**.

Figure 5 shows distribution over priorities assigned by the reporting use cases. The 36 requirements are **required**, 28 are **nice-to-have** and only 2 are **essential** (top priority). It indicates that use cases have a “workaround” for a large number of the requirements already by using the implementation from Y2. The two essential requirements were related to the SLA model extensibility and to the implementation of the custom mechanisms for automated SLA enforcements. The first one was resolved by adding support for custom domain specific SLA vocabulary, the second one can be addressed by the development of the custom
PAC (Provisioning and Adjustment Component), which is part of the skeleton SLAM (Service Level Agreement Manager) and can be extended for different use cases. In this particular case the reporter was Enterprise IT use case and the requirement has been resolved by the development of the custom Infrastructure PAC with some specific rules.

![Priorities](image)

**Figure 5: Count of requirements by their priority**

Figure 6 shows distribution over reporting use cases. The blue bars shows the requirements where the given use case is a primary reporter, while the red bars include all requirements reported by that use case. About 50% of the requirements were reported by B6, 30% by B4 and 20% by B5. B3 was not involved in Y3 requirement gathering, due to use case redesign and reorganization at that time.

![Reporters](image)

**Figure 6: Count of requirements by reporting use case**
Figure 7 shows distribution over responsible (A-Line) work packages. A5 has the most Y3 requirements, A4 the least and other are in between. The estimated amount of work in Y3 (focusing only on the accepted requirements - see Figure 8) shows much more even distribution. It might indicate that A4 did the best dissemination of their Y2 work and A5 did it the worse, but more probably it is related to the nature of the scientific field.

The fact that A5 is the most critical scientific WP with respect to use cases requirements was expected as A5 is a foundation for the other WPs. Most of the A5 requirements concern the SLA Model. Some of these requirements need a relatively simple implementation because they just require adding a particular kind of information to the SLA model. Others, however, are more difficult to implement as they have important repercussions on the other WPs too (e.g. monitoring any new kind of constraint included in the SLA Model). Therefore it is important to appreciate that the number of use case requirements associated with a WP should not be viewed as a measure of the implementation effort of that WP, but as an indication of which is the first WP in the implementation dependency chain.

![Figure 7: Count of requirements by responsible work package](image)
Figure 8: Count of requirements by responsible work package and status at the beginning of Y3

Figure 9 shows that at the end of Y2 nearly half of requirements were accepted, around 25% have been already implemented (supp and fixed) and around 30% have been rejected or postponed.

Figure 9: Count of requirements by their status at the beginning of Y3

Figure 10: Count of requirements by responsible work package and status at the end of Y3 and Figure 11: Count of requirements by their status show that the end of the project all accepted requirements were fixed and verified by a primary reporters.
This section presents the full list of consolidated Y3 requirements. Requirements are grouped by features and presented as on the wiki table used in Requirement Fulfilment Process. Each requirement has:

- short description (summary),
- priority
- reporting use case(s)
• responsible developer and his work package
• (latest) status
• notes

6.1 Framework Management

6.1.1 Framework Configuration & Setup

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.1</td>
<td>More documentation on OSGI integration</td>
<td>REQ B4</td>
<td>Gabriele, A1</td>
<td>verified</td>
<td>Verified. Documentation available on Source Forge, see Developer Resources section.</td>
</tr>
<tr>
<td>1.1.2</td>
<td>Syntax Converter role not clear</td>
<td>B5</td>
<td>Peter UDO, A5</td>
<td>invalid</td>
<td>Clarification has been already provided in the responses to the requirements document. See documentation on Source Forge.</td>
</tr>
<tr>
<td>1.1.3</td>
<td>WS-Agreement support needed</td>
<td>REQ B5</td>
<td>Peter/Edwin UDO, A5</td>
<td>later</td>
<td>Trying to provide, however not guaranteed due to time constraints and other higher priority requirements.</td>
</tr>
<tr>
<td>1.1.4</td>
<td>Tutorial for component configuration setup</td>
<td>NTH B6</td>
<td>Gabriele, A1</td>
<td>verified</td>
<td>Verified. Configuration of all platform modules is centralized, for general configuration needs there is documentation on Source Forge.</td>
</tr>
</tbody>
</table>

6.1.2 Framework Model Configuration

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2.1</td>
<td>SLA Model missing tutorials and examples.</td>
<td>REQ B4, B5</td>
<td>Keven, A5</td>
<td>verified</td>
<td>See: <a href="https://sourceforge.net/apps/trac/sla-at-soi/wiki/SlaModel">https://sourceforge.net/apps/trac/sla-at-soi/wiki/SlaModel</a></td>
</tr>
<tr>
<td>1.2.2</td>
<td>Support for custom domain specific SLA vocabulary</td>
<td>REQ B6</td>
<td>Keven, A5</td>
<td>supp</td>
<td>Verified.</td>
</tr>
<tr>
<td>1.2.3</td>
<td>Simpler configuration procedure for the new SLA model</td>
<td>NTH B6,B4,B5</td>
<td>Keven, A5</td>
<td>wontfix</td>
<td>The new SLA Model is out of the project scope.</td>
</tr>
</tbody>
</table>
6.1.3 Framework Operation

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
</table>
| 1.3.1  | Deployment without OSGi/PAX Runner | NTH B4, B5 | Gabriele, A1 | wontfix | Miguel: not applicable for G-SLAM  
Nawaz: not applicable |
| 1.3.2  | An open choice of application server | NTH B5 | Gabriele, A1 | wontfix | Nawaz: not applicable, requires restructuring of modules. |
| 1.3.3  | Event bus should support hundreds of events per second | REQ B6 | Primož, A1 | verified | Verified. AMQP implementation for the messaging library provided. Based on test results, it is much faster than XMPP implementation. |
| 1.3.4  | Make the integration and testing easier (without OSGi) | NTH B5 | Gabriele, A1 | wontfix | Miguel: not applicable for G-SLAM  
Nawaz: not applicable w.r.t WS standards and web service integration. |

6.1.4 Framework Access

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.4.1</td>
<td>REST interfaces of the framework</td>
<td>NTH B4</td>
<td>Gabriele, A1</td>
<td>invalid</td>
<td>Nawaz: The framework expose java interfaces as both SOAP/REST web services by default using the web service approach explained in guidelines. It is already in use.</td>
</tr>
<tr>
<td>1.4.2</td>
<td>Using JAXB for encoding larger XMLs</td>
<td>NTH B5</td>
<td>Gabriele, A1</td>
<td>invalid</td>
<td>Nawaz: It is not clear if we are confusing xml beans with jaxb which doesn’t hold because it is possible to use jaxb inside osgi, its being integrated and used by partners.</td>
</tr>
<tr>
<td>1.4.3</td>
<td>Web based GUI for negotiation and monitoring</td>
<td>NTH B5</td>
<td>Juan, A2</td>
<td>verified</td>
<td>Verified. Provider has web GUI for negotiation. It is possible to define monitoring-reporting capabilities in the SLAT.</td>
</tr>
</tbody>
</table>

6.2 Design & Development

6.2.1 Reference Software Manageability Components and Configuration

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1.1</td>
<td>Support resources for configuring third party</td>
<td>REQ B6</td>
<td>Sam, A3</td>
<td>verified</td>
<td>Verified.</td>
</tr>
<tr>
<td>Y3_RID</td>
<td>Summary</td>
<td>Priority, Reporter</td>
<td>Responsible</td>
<td>Status</td>
<td>Notes</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------------------------------</td>
<td>--------------------</td>
<td>-------------</td>
<td>--------</td>
<td>---------------------</td>
</tr>
<tr>
<td></td>
<td>providers</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.1.2</td>
<td>Control of BPEL engine (Dynamic binding of component services of BPEL processes).</td>
<td>REQ B6</td>
<td>Paolo, A3</td>
<td>supp</td>
<td>Verified. Extended by: 6.5.1, 6.3.1</td>
</tr>
</tbody>
</table>

**6.2.2 Service properties dependencies coding**

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.1</td>
<td>Improve SCM with better support the dependency of a service from a variable number of other services</td>
<td>NTH B6</td>
<td>Jens, A1</td>
<td>verified</td>
<td>Verified.</td>
</tr>
</tbody>
</table>

**6.2.3 Designing of QoS predictable systems**

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>Predict the performance of service based on time dependant demand (B6)</td>
<td>NTH B6</td>
<td>Franz FZI, A6</td>
<td>verified</td>
<td>Verified.</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Better static prediction model for B6 metrics</td>
<td>REQ B6</td>
<td>Franz FZI, A6</td>
<td>verified</td>
<td>Verified.</td>
</tr>
</tbody>
</table>

**6.3 Pre-offering**

**6.3.1 Customer Registration**

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.1</td>
<td>Customers should be able to register by their own</td>
<td>REQ B4</td>
<td>Juan, A2</td>
<td>supp</td>
<td>Verified. Implemented in Y2</td>
</tr>
<tr>
<td>3.1.2</td>
<td>Customers should be able to be preregistered by administrator</td>
<td>NTH B4</td>
<td>Juan, A2</td>
<td>supp</td>
<td>Verified. Implemented in Y2</td>
</tr>
<tr>
<td>3.1.3</td>
<td>Preconfigured department structure should be supported + required user attributes</td>
<td>REQ B4</td>
<td>Juan, A2</td>
<td>supp</td>
<td>Verified. Implemented in Y2 (user attrib). Department support will not implemented</td>
</tr>
<tr>
<td>3.1.4</td>
<td>User authentication should be supported</td>
<td>REQ B4</td>
<td>Juan, A2</td>
<td>supp</td>
<td>Verified. Implemented in Y2</td>
</tr>
</tbody>
</table>
### 6.3.2 3rd Parties Configuration

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.1</td>
<td>Static 3rd parties configuration</td>
<td>REQ B6</td>
<td>Wasif, A3</td>
<td>wontfix</td>
<td>TID moved out of A3</td>
</tr>
<tr>
<td>3.2.2</td>
<td>SLAT announcements should be based on providerID and its authorization by the Business Manager</td>
<td>REQ B5</td>
<td>Wasif, A3</td>
<td>wontfix</td>
<td>TID moved out of A3</td>
</tr>
</tbody>
</table>

### 6.4 Service Offering

#### 6.4.1 Discovery of Serv. and SLAT

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1</td>
<td>SLAT based search of already established SLAs of third party providers</td>
<td>NTH B6</td>
<td>Keven, A5</td>
<td>verified</td>
<td>Verified. B6 will use the SLATRegistry to store SLAs. This is implemented using the SLATRegistry query mechanisms</td>
</tr>
</tbody>
</table>

#### 6.4.2 Product Management

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.1</td>
<td>Basic support for internal “products” (B4)</td>
<td>NTH B4</td>
<td>Juan, A2</td>
<td>supp</td>
<td>Verified. Implemented in Y2</td>
</tr>
</tbody>
</table>

#### 6.4.3 Product Discovery

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.1</td>
<td>Product discovery should be similar to service and SLAT discovery</td>
<td>REQ B4</td>
<td>Juan, A2</td>
<td>supp</td>
<td>Verified. Implemented in Y2</td>
</tr>
<tr>
<td>4.3.2</td>
<td>A unified approach to searching products, services, SLAs and SLATs</td>
<td>NTH B4</td>
<td>Keven, A5</td>
<td>wontfix</td>
<td>involves also Juan, A2.</td>
</tr>
</tbody>
</table>
### 6.4.4 SLAT Definition

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5.1</td>
<td>SLA model extensible with UC specific extension</td>
<td>ESS B4</td>
<td>Keven, A5</td>
<td>dup</td>
<td>Verified. See 1.2.2</td>
</tr>
<tr>
<td>4.5.2</td>
<td>SLAT advertisement should work for internal and 3rd party providers</td>
<td>REQ B5</td>
<td>Peter, A5</td>
<td>verified</td>
<td>Verified. SLAT registry is exposed externally through WS</td>
</tr>
<tr>
<td>4.5.3</td>
<td>SLAT editor (GUI)</td>
<td>NTH B5, B6</td>
<td>Peter, A5</td>
<td>verified</td>
<td>Verified. SLA editor provided.</td>
</tr>
<tr>
<td>4.5.4</td>
<td>SLAT announcements should be limited to providers and customers are authorized by Business Manager</td>
<td>REQ B5</td>
<td>Peter, A5</td>
<td>wontfix</td>
<td></td>
</tr>
</tbody>
</table>

### 6.5 Service Negotiation

#### 6.5.1 Business Negotiation

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1.1</td>
<td>Semi-automatic and manual business negotiation</td>
<td>REQ B5, B6</td>
<td>Juan, A2</td>
<td>verified</td>
<td>Verified. Provider capability.</td>
</tr>
<tr>
<td>5.1.2</td>
<td>GUI for manual negotiation/editing of offered SLAs</td>
<td>NTH B6</td>
<td>Juan, A2</td>
<td>verified</td>
<td>Verified. Provider capability.</td>
</tr>
</tbody>
</table>

#### 6.5.2 SLAT Customization

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.2.1</td>
<td>Some SLAT fields can be preset based on customer profile or segmentation (e.g. department). Some of these fields should not be changed during negotiation.</td>
<td>NTH B4</td>
<td>Juan, A2</td>
<td>wontfix</td>
<td>A2 is not responsible of SLA Model. SLAT fields that should be fixed must be annotated or outlined in order customer can understand that these parameters are not modifiables during negotiation.</td>
</tr>
</tbody>
</table>
### 6.5.3 Business SLA planning for negotiation

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.1</td>
<td>Possibility to assert conditions for triggering the negotiation with third parties that depends from the current available resources, the result of a static evaluation and/or from a human decision</td>
<td>REQ B6</td>
<td>Edwin, Kuan, A5</td>
<td>supp</td>
<td>Verified. Domain dependent. Supported by I-POC. Not Supported by SW-POC.</td>
</tr>
</tbody>
</table>

### 6.5.4 Automatic orchestration of customizable (re)negotiation protocol

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4.1</td>
<td>Negotiation protocol that includes also a confirmation step performed by a human</td>
<td>REQ B6</td>
<td>Edwin, A5</td>
<td>invalid</td>
<td>Edwin: This is not related to Negotiation Protocol, but the INegotiation interface. It requires support from SLA Registry to not auto-provision when SLA is created/inserted - rather upon invocation of new signSLA method.</td>
</tr>
</tbody>
</table>

### 6.5.5 Out of band SLA Registration

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.5.1</td>
<td>Adding new sub providers and corresponding out-of-band SLAs in runtime</td>
<td>NTH B6</td>
<td>Miguel, A5</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

### 6.6 Service Provisioning

#### 6.6.1 Coordination of Provisioning

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1.1</td>
<td>Tutorial how to implement provisioning of external resources (not hw/sw services)</td>
<td>REQ B6</td>
<td>Peter, A5</td>
<td>invalid</td>
<td>This feature is domain/use case specific. As such it is not being developed as part of A5. Description of available scenarios (sw, hw) should be used as basis. Most of all a *-ServiceManager will be required for this task, which is</td>
</tr>
</tbody>
</table>
### 6.6.2 Dynamic binding setting

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.3.1</td>
<td>Multiparty binding - possibility to represent in a simple way the invocation of the same service on multiple providers and to dynamically bind the list of these providers</td>
<td>REQ B6</td>
<td>Paolo, A3</td>
<td>verified</td>
<td>Verified. This requirement is implemented by the new feature for multiparty invocation provided by the Y3 version of the DOE.</td>
</tr>
</tbody>
</table>

### 6.6.3 Virtual hw. infrastructure provisioning

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.4.1</td>
<td>Provisioning of storage and network</td>
<td>NTH B4</td>
<td>John, A4</td>
<td>later</td>
<td>Network provisioning is a very large complicated area where generic network management interface standards are not readily available. It is not feasible to address with current resources/timelines. Thus we cannot commit to network-related storage solutions either. Local storage provisioning is in scope. P.S. Network interface settings can be manipulated already... bandwidth etc.</td>
</tr>
<tr>
<td>6.4.2</td>
<td>VM persistence and backup as specified in SLA</td>
<td>NTH B5, B4</td>
<td>MihaS, A4</td>
<td>verified</td>
<td>Verified. Tashi has been extended with a bunch of new SLA terms like VM persistence, isolation, location, backup...</td>
</tr>
</tbody>
</table>

### 6.6.4 Software provisioning

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5.1</td>
<td>Possibility to change the structure (activity diagram) of the BPEL process</td>
<td>NTH B6</td>
<td>Natallia, A3</td>
<td>verified</td>
<td>Verified. A Java-implementation is provided on SourceForge.</td>
</tr>
</tbody>
</table>
### 6.6.5 Software landscape

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.6.1</td>
<td>Software landscape tutorial and documentation</td>
<td>NTH B5</td>
<td>Jens, A3</td>
<td>supp</td>
<td>Verified.</td>
</tr>
</tbody>
</table>

### 6.7 SLA enforcement

#### 6.7.1 Automated SLA enforcement coordination

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.1.1</td>
<td>Allow UCs to implement their custom mechanisms (B4) for automated SLA enforcement</td>
<td>ESS B4</td>
<td>Peter, A5</td>
<td>verified</td>
<td>Verified. Already available via custom I-PAC with specific rules.</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Automated SLA enforcement should cover business, infrastructure and sw adjustment</td>
<td>NTH B5</td>
<td>Peter, A5</td>
<td>supp</td>
<td>Verified. PACs provide these features already for business (B-PAC/violation-penalty), infrastructure (I-PAC). The SW-PAC is under development.</td>
</tr>
</tbody>
</table>

#### 6.7.2 Virtual hw. infrastructure Adjustment

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2.1</td>
<td>Scale up and down virtual infrastructure</td>
<td>NTH B4</td>
<td>John, A4</td>
<td>verified</td>
<td>Verified. Provisioning system supports it.</td>
</tr>
</tbody>
</table>

#### 6.7.3 Software Adjustment

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.3.1</td>
<td>Adjustment of BPEL process structure (activity diagram) and bindings</td>
<td>REQ B6</td>
<td>Natallia (Adjustment of BPEL process structure), Paolo (Bindings) - A3</td>
<td>verified</td>
<td>Verified. Separate project on Source Forge provided for BPEL adjustments.</td>
</tr>
<tr>
<td>7.3.2</td>
<td>SW adjustment tutorial and examples for other kinds of sw</td>
<td>REQ B5</td>
<td>Wasif, A3</td>
<td>verified</td>
<td>Verified. Documentation is available on Source Forge.</td>
</tr>
</tbody>
</table>
### 6.7.4 Automatic Binding

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.4.1</td>
<td>Automatic binding of dynamically selected services on the base of SLAs matching specified SLATs</td>
<td>REQ B6</td>
<td>Paolo, A3</td>
<td>verified</td>
<td>Verified. This requirement is implemented by the automatic binding feature of the DOE, that uses SLAT based rules associated to abstract BPEL processes.</td>
</tr>
</tbody>
</table>

### 6.7.5 Business adjustment

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.5.1</td>
<td>A skeleton/example of PAC &amp; POC implementing the described behavior (see notes)</td>
<td>REQ B6</td>
<td>Miguel, A5</td>
<td>wontfix</td>
<td>Triggering renegotiation with customer and 3rd party providers by runtime prediction warnings. Miguel: The skeleton SLA-Manager provides functionality to create a new SLAM. This includes a basic/empty implementation of PAC and POC</td>
</tr>
</tbody>
</table>

### 6.8 Runtime Prediction

#### 6.8.1 Runtime infrastructure metrics prediction

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.1</td>
<td>Predictions should be stored in history database to identify trends</td>
<td>REQ B4</td>
<td>Victor, A6</td>
<td>supp</td>
<td>verified</td>
</tr>
<tr>
<td>8.1.2</td>
<td>Runtime infrastructure metrics prediction</td>
<td>NTH B6</td>
<td>Victor, A6</td>
<td>supp</td>
<td>verified</td>
</tr>
<tr>
<td>Y3_RID</td>
<td>Summary</td>
<td>Priority, Reporter</td>
<td>Responsible</td>
<td>Status</td>
<td>Notes</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-------------------</td>
<td>-------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td>aware of temporal context (e.g. day of the week)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.1.3</td>
<td>Runtime infrastructure metrics prediction on custom (UC specific metric) data should be supported</td>
<td>NTH B6</td>
<td>Victor, A6</td>
<td>wontfix</td>
<td>out of scope due to significant limitations which could not be avoided: significant compute load, high dependency on workload consistency, effectiveness only for short term predictions.</td>
</tr>
</tbody>
</table>

### 6.8.2 Runtime Software SLAs violations prediction

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2.1</td>
<td>Runtime Software SLAs violations prediction on custom (UC specific metric) data should be supported</td>
<td>REQ B6</td>
<td>Davide, A6</td>
<td>verified</td>
<td>Verified. Runtime prediction supports QoSs that has been already agreed with UC owners (B6).</td>
</tr>
<tr>
<td>8.2.2</td>
<td>Runtime Software SLAs violations prediction aware of temporal context (e.g. day of the week)</td>
<td>NTH B6</td>
<td>Davide, A6</td>
<td>wontfix</td>
<td>It would imply a prediction model modification, which is unfeasible in the short remaining.</td>
</tr>
</tbody>
</table>

### 6.9 Service Monitoring

#### 6.9.1 Virtual hardware infrastructure monitoring rules extraction

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.2.1</td>
<td>Virtual hardware infrastructure monitoring rules/constraints should be derived for agreed SLAs and applied to monitoring configuration of infrastructure service</td>
<td>REQ B4</td>
<td>Damjan, A4</td>
<td>verified</td>
<td>Verified. Dynamic configuration supported.</td>
</tr>
</tbody>
</table>
## 6.10 Service Reporting

### 6.10.1 SLA Management

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.1</td>
<td>GUI and API to configure SLATs and already established SLAs</td>
<td>REQ B6</td>
<td>Keven/Miguel, A5</td>
<td>wontfix</td>
<td>Erasing, modifying and deleting SLATs seems to be missing.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>This requirement is not about monitoring/reporting, but just about managing registries.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Peter: The remaining time in the project does not suffice to implement such a feature.</td>
</tr>
<tr>
<td>10.1.2</td>
<td>Reports and history of SLA status should be available</td>
<td>REQ B6</td>
<td>Davide, A2</td>
<td>verified</td>
<td>Verified. SLA status history held by postsale-reporting system and available through its interface</td>
</tr>
</tbody>
</table>

### 6.10.2 Push/Pull Business SLA Reporting

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.2.1</td>
<td>Summary Business SLA report at the end of billing period</td>
<td>REQ B4</td>
<td>Agustin, A2</td>
<td>verified</td>
<td>Verified.</td>
</tr>
<tr>
<td>10.2.2</td>
<td>XML reports containing current statuses of all guarantee terms + content specified in reporting actions</td>
<td>REQ B6</td>
<td>Davide, A2</td>
<td>verified</td>
<td>Verified. Generic support for adding report terms. One solution is to use introduce auxiliary GTs as in (11.1.1).</td>
</tr>
</tbody>
</table>

### 6.10.3 SLA Status Reporting

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.3.1</td>
<td>Renegotiation reports showing when a SLA was renegotiated and what was changed</td>
<td>REQ B6</td>
<td>Miguel, A5</td>
<td>verified</td>
<td>Verified. The renegotiated status of the SLA is included. Related information can be queried via SLAManagementConsole</td>
</tr>
<tr>
<td>10.3.2</td>
<td>SLA Status mail</td>
<td>REQ</td>
<td>Miguel, A5</td>
<td>verified</td>
<td>Verified. A notification is</td>
</tr>
</tbody>
</table>
### 6.10.4 Virtual hw Infrastructure Reporting

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.4.1</td>
<td>Virtual HW Infrastructure Reporting GUI showing the history graphs</td>
<td>REQ B4</td>
<td>Damjan, A4</td>
<td>verified</td>
<td>Verified. Integrated in B4 GUI.</td>
</tr>
</tbody>
</table>

### 6.11 New Features

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>11.1.1</td>
<td>Expose Internal Resource KPIs to Service Provider (B6)</td>
<td>REQ B6</td>
<td>Davide, A2</td>
<td>verified</td>
<td>Verified. Auxiliary GTs are introduced for these Internal Resource KPIs so that existing reporting/monitoring mechanisms can be used. UC can add new GTs by itself. Present and historical data are provided.</td>
</tr>
</tbody>
</table>

### 6.12 Sub-requirements stated by A-Line

<table>
<thead>
<tr>
<th>Y3_RID</th>
<th>Summary</th>
<th>Priority, Reporter</th>
<th>Responsible</th>
<th>Status</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.1.1</td>
<td>Web GUI for SLA Editor</td>
<td>NTH A2</td>
<td>Peter, A5</td>
<td>wontfix</td>
<td>required by req. 1.4.3. Peter: There will be an Eclipse-based SLAT-editor. It should be possible to transform the Eclipse RCP application to a web-based version though.</td>
</tr>
</tbody>
</table>

### 6.13 External Requirements

Additional requirements from partners of the SLA@SOI consortium and their internal projects have been collected. A careful analysis revealed that these requirements do not constitute completely new requests regarding the SLA@SOI platform. Instead, literally all of them extend/refine the understanding of already existing requirements. This shows that (a) the coverage of features is already
pretty good and stable, and (b) that additional stakeholder feedback is nevertheless helpful to refine the understanding of these. Therefore, these requirements have been merged with existing requirements that were gathered in the process of the internal requirement gathering described below. For example additional infrastructure requirements were gathered from Internal IT Cloud group (Intel) and Gaea+ development team (XLAB), please see requirements such as 6.4.2, 7.2.1 and 9.2.1. Some requirements about integration tutorials came out of TID New Service Delivery Platform team, these were merged in the 1.1.1, 1.1.4 and 1.2.1. Some reporting requirements came out of NetWeaver Lifecycle Management SAP and were merged in the 10.1.2 and 10.2.1.

7 Evaluation Approach

The goal of the evaluation is to evaluate the project results from two points of view: 1) to evaluate how well the framework developed by the scientific work packages implements the use case requirements (evaluation of scientific results) and 2) to evaluate how well the external requirements gathered at the beginning of the project have been satisfied and what is the value of the framework from the view of the involved stakeholders. For information about external requirements gathered at the beginning of the project please see section 9.

In the next section we describe the evaluation process; metrics for the technical evaluation and the business evaluation are described.

7.1 Scientific Metrics Evaluation

Following the Goals, Questions, Metrics methodology we now present the derivation of the metrics that will be used in the evaluation of requirements from the scientific work packages. We start by listing the high level goals we have defined for the evaluation process in Table 4: Requirements Evaluation Goals.

<table>
<thead>
<tr>
<th>ID</th>
<th>Goal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>Intelligibility</td>
<td>Each requirement should be understood by all involved parties.</td>
</tr>
<tr>
<td>G2</td>
<td>Monitoring</td>
<td>The progress towards the implementation of each requirement shall be continuously monitored.</td>
</tr>
<tr>
<td>G3</td>
<td>Traceability</td>
<td>The effects of each requirement in the creation of various artefacts during the implementation process must be traceable back to the originating requirement.</td>
</tr>
<tr>
<td>G4</td>
<td>Implementation</td>
<td>The goal of a requirement is the successful implementation as a software component that satisfies the requirement in the use case application context.</td>
</tr>
</tbody>
</table>

Table 4: Requirements Evaluation Goals
For each of these goals a set of questions was provided to estimate the final implementation of the goal. There is a set of metrics defined for each question. The metrics fall into one of the two categories:

1. Metrics based on the performance of the software features we detail in 7.2 below.
2. Metrics based on a survey after results of the integration are known. We present the survey in the 7.3 below.

### 7.2 Testing

For the software testing part of the metrics, we initially followed the strategy we outlined in M18 of using unit tests to evaluate the convergence of the functionality of individual components to the specified requirements. Using JUnit framework in the SLA@SOI Maven system, we steadily accumulated a number of unit tests, achieving more than 250 such tests by the middle of M27, with 90% of the tests being reported as having successfully passed at that time. In the Y3 the number of tests was increased, but what is more important, all tests are being reported as successful. The Surefire Maven plugin provides a publicly visible online report [16] which is updated regularly by the continuous integration environment introduced to support the overall technical support. By using the continuous availability of the reporting we were able to ensure that every component was covered by at least some amount of tests. The results of these tests are provided per tested component in section 8.

By the end of M27, it was widely felt that while the unit testing process was providing feedback on the development state of individual modules, we were missing a crucial view into the actual behaviour of the SLA@SOI as a platform. We saw in M18 that such an integrated and validated view could be provided by the testing performed by each use case as they integrated the software into their prototypes. With the aggressive development schedule adopted after the M18 review, it quickly became apparent that feedback from the use cases wouldn’t be sufficient and it would be too late to implement the proposed modifications. Fortunately, in the Open Reference Case demonstrator created for WP B2 we had developed three scenarios to verify the integration tests of the software. These integration scenarios covered quite sophisticated negotiation, provisioning, and runtime interactions between all of the major components, with the most complex (Negotiation) comprising over thirty discrete testable steps. For each testable step in the scenario, data was prepared that would constitute the expected result of the tests. Development resources were iteratively redistributed until each of the steps was judged to have successfully passed. While not providing as nice measurable metric as the passing of individual unit tests, it is widely felt throughout the project that the development and maintenance of these integration tests contributed immeasurably more to the quality of the implementation than the unit tests ended up doing. Documentation of the integration tests can be found on Source Forge [17] These tests can serve simply as an overall SLA@SOI framework testing platform, as well as a starting point for the developers who want to adopt the SLA@SOI framework for their use case.

### 7.3 Survey

We now provide an overview of the survey which was completed by each use-case for their domain specific evaluation. The intention was to provide a light-
weight mechanism to collect information for evaluation of the requirement. We initially expected to implement the survey via an online mechanism using a free tool such as SurveyMonkey\(^1\), but in the end we utilized an approach consolidating the collected information into a common spreadsheet.

We solicited feedback to the following questions as outlined in Table 3: Evaluation Survey whose results were consolidated into a spreadsheet. We planned to use integers from 1 to 5 to provide the answers to the questions ranging from “highly satisfactory” to “highly unsatisfactory”, but upon revision of the questions we decided that higher quality information would be solicited if 1) we used a simple true or false response for whether a given feature was needed by the use case, 2) we represented the answers for ‘Qualitative1’ as “insufficient”, “nearly sufficient”, “sufficient”, “good”, and “excellent”, while for the question listed as ‘Qualitative2’, we supplied the values “negligible”, “low”, “medium”, “high”, and “very high” and 4) we provided an opportunity for free form remarks about the implementation to be provided.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Question</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Needed</td>
<td>Whether the feature was used in this iteration of the use case.</td>
<td>Boolean</td>
</tr>
<tr>
<td>Documentation</td>
<td>Does the documentation provided cover in a clear way the aspects needed to know related to this feature and its implementation?</td>
<td>Qualitative1</td>
</tr>
<tr>
<td>Architecture</td>
<td>Does the SLA@SOI architecture associated with this feature satisfy the functional needs for the given feature (and if the concrete Java/WSDL APIs is not suitable to your Use Case due to other constraints?)</td>
<td>Qualitative1</td>
</tr>
<tr>
<td>API</td>
<td>Do the Java/WSDL interfaces satisfy both functional and non-functional needs (and if the use case used a different implementation of the interface as provided from A-line?)</td>
<td>Qualitative1</td>
</tr>
<tr>
<td>Implementation</td>
<td>How complete is the implementation of this feature?</td>
<td>Qualitative1</td>
</tr>
<tr>
<td>Integration Complexity</td>
<td>How much complexity does integration of the feature require for your implementation</td>
<td>Qualitative2</td>
</tr>
</tbody>
</table>

Table 5: Evaluation Survey

\(^1\) http://www.surveymonkey.com/
8 Feature Evaluation Results

In this section the results of the Y3 evaluation are described. For comparison with the Y2 evaluation also graphs from Y2 evaluation are given. The same evaluation methodology as in Y2 was presented also this year.

As anticipated in section 7, several metrics has been identified to assure a good Intelligibility, Monitoring, Traceability and Implementation.

The intelligibility of features was assured during consolidation, clarification and assignment of Y3 requirements.

The metrics related to the other goals have been assessed through a survey distributed to B-Line users and through a self-evaluation of A-Line providers.

The survey measures the quality of the framework from 5 points of view (quality of Documentation, quality of Architecture, quality of API, quality of Implementation and Complexity of Integration). For each one of these aspects, five possible rating was possible: (“insufficient”, “nearly sufficient”, “sufficient”, “good”, “excellent”) for the quality aspects and (“negligible”, “low”, “medium”, “high”, “very high”) for the integration complexity.

The A-Line has evaluated the status of the implementation, by means of both subjective judgements (accepted, designed, implemented, integrated, completed) and automatic tests.

The following sections present the results of the evaluation at the end of the project, as well as the results from the Y2 evaluation.

The next sub-sections show the results of the evaluation for each feature. In this introduction, instead we present the different evaluations aggregated by feature categories. They show the status of the results at high level and gives an interesting insight on the different perception of maturity level of the issues between A-Line and B-Line. To improve readability, all the figures adopt a same colour schema (orange for negative evaluations, blue for sufficient and green for more positive evaluations) for both the bars and the Y-axis tabs.

Figure 12 shows the average status of each feature category as reported by the A-Line at the end of Y3. All features were reported as completed, although there is still some room for improvements.
Figure 12: Average status reported by the A-Line for each category of features.

This evaluation reported by the A-Line may be contrasted with the evaluation of this implementation performed by the B-Line and shown in Figure 14. The results from Y2 are shown in Figure 13.

Figure 13: Average quality of the implementation judged by the B-Line for each category in Y2.
As we can see, all features have at least sufficient implementation.

Figure 16 shows that the level of integration complexity is mostly between low and medium, which means a considerable improvement from Y2 (see Figure 15). The most frequent remarks about the simplification of the integration process were about the improved documentation and integration tests available on the Source Forge.

Figure 15: Average level of complexity reported by the B-Line for each category in Y2.
Figure 16: Average level of complexity reported by the B-Line for each category in Y3.

Most of the difficulties faced by the users of the SLA@SOI Framework in Y2 were strictly related to the quality of the documentation (see Figure 17).

Figure 18 shows that the quality of the documentation is significantly improved in Y3.

Figure 17: Average quality of the documentation judged by the B-Line for each category in Y2.
The improved quality of the documentation is also a reason for a better integration complexity evaluation results. In contrast to the Y2 all features have a documentation judged more than sufficient. Moreover, most of the documentation is judged as good. The documentation was considerably improved in the phase of open sourcing the framework.

Figure 20 and Figure 22 show the rating of quality of the architecture and the API. The rating of the Architecture is generally evaluated as at least good. Significant improvements are detected also in the level of API, see Figure 19 and Figure 21 for comparison with Y2 evaluation results.

Figure 23 shows the number of reported issues on each category of features at the end of Y2. The considered issues does not include bugs or low level concerns (that was managed by the project development process), but just high level issues that are more directly tied to the use case requirements. Figure 24 shows the number of issues at the end of Y3. In Y2 the highest number of issues was reported on the category Framework Management, which was an alarm to reduce the complexity of the framework in Y3. The major part of these issues has been successfully resolved during the last year of the project and there were almost no issues reported by B-line for the Framework Management category in the final evaluation.

Also the number of other issues is small, probably because of the fact that issues from Y2 have been transformed into the requirements for the Y3 implementation. Mostly, the issues reported in Y3 are related to the SLA model and SLA(T) customization. For example the following issue was reported by Enterprise IT use case: “Yes SLATs can be defined, but requires expert level knowledge of the SLA model. Perhaps an editor would help with this, as per B3’s suggestion”. These issues will be however resolved by an SLA editor, which is being developed in the final phase of the project as part of the SLA@SOI Studio.
Figure 19: Average quality of the Architecture judged by the B-Line for each category in Y2.

Figure 20: Average quality of the Architecture judged by the B-Line for each category in Y3.
Figure 21: Average quality of the APIs judged by the B-Line for each category in Y2.

Figure 22: Average quality of the APIs judged by the B-Line for each category in Y3.
The following sections are organized by category of features. For each category the evaluation of the corresponding features is reported. Each evaluation is composed by 4 sub-sections. The first one, named “Traceability” specifies what
are the components of the SLA@SOI Architecture that implements the feature and what is the innovation (if any) the feature is related to. When needed both grades are accompanied by a short explanation. The second sub-section is named “A-Line evaluation” and reports the number of automatic tests associated to the features, the number of passed tests and the status of the implementation estimated by the developer of the feature. The third sub—section, named “B-Line Evaluation” reports the quality and complexity assessments performed by the use cases and the reported issues. Finally the last subsection reports the “Lessons learned” for that issue.

Note that as each use case has performed a different evaluation, the results reported in the following subsections are aggregated data. Distinct evaluations and issues reported by each use case are provided in Appendix G:

8.1 Framework Management

Framework Configuration & Setup

Traceability

Component/s: Full Architecture

There is no single component or small set of components that implement this feature. Instead, the different means for configuration, wiring and setup of the framework are manifold and spread over different mechanisms. The possible means are described in deliverable D.A1b [5].

Related to innovation: “Architecture for a consistent SLA-management framework”

The feature is closely related to innovation “Architecture for a consistent SLA-management framework”, as it allows supporting a wide range of stakeholder scenarios which realize different hierarchies of SLA and service managers.

A-Line evaluation

Associated tests: 3 (the integration tests for negotiation, provisioning and runtime)

Passed tests: 100%

Status: completed

This feature is well completed. Of course there are always still aspects that might be further improved.

B-Line evaluation

Documentation: Good

Architecture: Good

API: Good

Implementation: Good
Integration Complexity: Medium

Issues: No issues reported.

Lessons Learned
Overall, the evaluation shows that the developed framework architecture and the concepts for its configuration are well received by the use cases and largely meet their needs. In particular, the adoption guide as well as the tutorial turned out as very usefully instruments.

Framework Model Configuration

Traceability

Component/s: Full Architecture

There is no single component responsible for model configuration but rather the general need that all relevant models can be configured. The main models to which this applies are the SLA model (A5), the service construction model (A1), the business rule models (A2), the software landscape (A3), the infrastructure landscape (A4), and the Palladio component model (A6). Their respective configuration mechanisms are described in deliverable D.A1b [6], D.A2a [7], D.A3a [8], D.A4a [9], D.A5a [10], and D.A6a [11] but also in the adoption guide [13].

Related to innovation: “Architecture for a consistent SLA-management framework”

The feature is closely related to innovation “Architecture for a consistent SLA-management framework”, as it allows supporting a wide range of deployment scenarios which need different meta-models for their domain.

A-Line evaluation

Associated tests: 0 (feature cannot be tested in an automatic fashion)
Passed tests: N/A
Status: completed

The feature implementation is well completed.

B-Line evaluation

Documentation: Good
Architecture: Good
API: Good
Implementation: Good
Integration Complexity: Low

Issues: No issues reported.

Lessons Learned
Overall, the evaluation shows the usefulness of the various information models used throughout the framework and the ways for their usage and configuration in particular use cases. One issue that came up was a better understanding of the inter-linkage between the different model artefacts. This has been addressed explicitly by the adoption guide [13].

Framework Operation

Traceability
Component/s: Full Architecture
Related to innovation: N/A
The feature is not related to any specific innovation but required as an implicit technical aspect by the use cases.

A-Line evaluation
Associated tests: N/A
Passed tests: N/A
Status: completed
The core of the operation feature is fully completed.

B-Line evaluation
Documentation: Good
Architecture: Good
API: Excellent
Implementation: Good
Integration Complexity: Low
Issues: No issues reported.

Lessons Learned
Overall, the evaluation shows that this feature is well realized.
**Framework Access**

**Traceability**

*Component/s:* Full Architecture

*Related to innovation:* N/A

The feature is not related to any specific innovation but required as an implicit technical aspect by the use cases.

**A-Line evaluation**

*Associated tests:* 0 (not applicable)

*Passed tests:* not applicable

*Status:* completed

The core of the access feature is fully completed. However, no explicit support is given for access from C/.Net nor for integration with DMTF CIM management models.

**B-Line evaluation**

*Documentation:* Sufficient

*Architecture:* Good

*API:* Good

*Implementation:* Good

*Integration Complexity:* Medium

*Issues:* No issues reported.

**Lessons Learned**

Overall, the evaluation shows that the provided mechanisms for framework access are well realized and complete.
8.2 Design & Development

Reference Software Manageability Components and Configuration

Traceability

Component/s: N/A

This is an interface which should be implemented by the domain specific ManageabilityAgents to be able to integrate with the SoftwareServiceManager.

Related to innovation: Unified Manageability Interface and Manageability Infrastructure for Service-Oriented Systems.

A-Line evaluation

Associated tests: 1

Passed tests: 100%

Status: Completed

The interface is specified and is ready for the use cases to be implemented with their custom manageability agents. Reference implementations of this interface for the DOE and for Axis2 containers are also available.

B-Line evaluation

Documentation: Good

Architecture: Good

API: Good

Implementation: Sufficient

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned

Overall, the provided reference implementations for the DOE and generic Axis2 containers, together with the comprehensive documentation, have been well received by the adopting use cases.

Service properties dependencies coding

Traceability
**Component/s:** Service Manager, Service Construction Model

This feature is realized by the service construction model, which specifies the relations between service types, service implementations (incl. their dependencies) and service instances. It’s heavily used in the interface of the Software Service Manager, however could be also reused for other Service Manager domains. Details are described in deliverable D.A1b[6].

**Related to innovation:** “Architecture for a consistent SLA-management framework”

The feature is closely related to innovation “Architecture for a consistent SLA-management framework”, as it is a core prerequisite for supporting multi-layered SLA management.

**A-Line evaluation**

**Associated tests:** 1  
**Passed tests:** 100%  
**Status:** Completed  

The model and the related Software Service Manager is fully completed, and successfully used.

**B-Line evaluation**

**Documentation:** Excellent  
**Architecture:** Excellent  
**API:** Good  
**Implementation:** Good  
**Integration Complexity:** Low  
**Issues:** No issues reported.

**Lessons Learned**

Overall, the evaluation shows that this feature is well realized and the editor supports the easy creation of correct models.

**Designing of QoS predictable systems**

**Traceability**

**Component/s:** ServiceEvaluation

The ServiceEvaluation component is used by SLAManager components at each level (business, software, infrastructure) to determine a-priori evaluation of
service quality parameters. ServiceEvaluation offers its functionality via the IServiceEvaluation interface. For more details, refer to Deliverable D.A6a.

**Related to innovation: Software Performance and Reliability Prediction**

The approach considers architectural influences on the performance and reliability of a service-based system in a comprehensive way, namely, service component implementation, service usage, quality of required services, and execution environment (physical resources). Other approaches are generally limited in their expressiveness to a subset of these factors.

**A-Line evaluation**

**Associated tests:** 10

**Passed tests:** 100%

**Status:** Completed

While the general Service Evaluation component is abstract, a concrete implementation for the case of software service performance has been provided and integrated into the open reference use case scenario. In the third project year, an implementation for software service reliability has been provided.

**B-Line evaluation**

**Documentation:** Good

**Architecture:** Excellent

**API:** Excellent

**Implementation:** Good

**Integration Complexity:** Low

**Issues:** Usage profiles are important with regards to the provisioning of services.

**Lessons Learned**

Overall, the feature is well received by the industrial use cases B3, and B6, which make use of the feature. However, it turned out that in case of B3, the model-driven performance prediction approach is insufficient. Therefore, we extended the approach with measurement-based information gathering, in order to fulfil the needs of WP B3.

**Manageability Design**

**Traceability**
**Component/s:** The manageability modelling tool has been developed as an EMF-based Eclipse plugin. It allows the designer to specify high-level KPIs that he/she wants the system to be able to provide at runtime, as well as the adjustment strategies that should be available at runtime. The tool allows for the automatic synthesis of instrumentation directives for deploying appropriate sensors to the executing application, and for instrumenting the services to support runtime adjustment. The tool is an external component with respect to the overall SLA@SOI platform. More details can be found in deliverable D.A6a [11].

**Related to innovation:** We provide an explicit support of service-based systems (web service compositions), including the configuration of monitoring and control capabilities. Complete instrumentation for monitoring is supported with white-box services (i.e., instrumentation of the internal BPEL code). We also support instrumentation of BPEL processes so that they support dynamic binding and runtime control- and data-flows adjustments.

**A-Line evaluation**

**Associated tests:** 1

**Passed tests:** 100%

**Status:** Completed

The implementation is complete. The tool has been extended, in particular to include support for new control-related meta-models and for event correlation.

**B-Line evaluation**

**Documentation:** Good

**Architecture:** Sufficient

**API:** Good

**Implementation:** Good

**Integration Complexity:** Low

**Issues:** No issues reported.

**Lessons Learned**

The final evaluation shows that the approach is usable. The documentation is sufficient, yet a wider variety of tutorial examples could be useful.
8.3 Pre-offering

Customer registration

Traceability

Component/s: Business Manager is the component that implements this functionality. It offers two ways for customer registration. A web service interface that can be used by external systems framework and GUI inside Business Web Tools (part of the Business Manager). It is possible to look up in A2 official deliverable [7].

Related to innovation: "Comprehensive Business Management suite for e-contracting". This feature is part of the novel marketplace through which customers are enabled to contract and provision services.

A-Line evaluation

Associated tests: 7

Passed tests: 100%
Status: Completed

B-Line evaluation

Documentation: Good

Architecture: Good

API: Good

Implementation: Sufficient

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned

This feature is completed and some use cases have integrated it.

3rd Parties Configuration

Traceability

Component/s: N/A

The overall architecture of the SLA@SOI framework was designed from the beginning to be able to be used in a multi-provider environment. The main impact
of the inclusion of external providers would be at the business layer. Therefore, database model designed as part of A2 work package (see deliverable D.A2a [7]) has been extended so to include a registry of the 3rd parties

*Related to innovation*: N/A

**A-Line evaluation**

*Associated tests*: 6

*Passed tests*: 100%

*Status*: Completed

Database model designed as part of A2 work package (see deliverable D.A2a [7]) has been extended so to include a registry of the 3rd parties

**B-Line evaluation**

*Documentation*: Good

*Architecture*: Sufficient

*API*: Good

*Implementation*: Good

*Integration Complexity*: Medium

*Issues*: No issues reported.

**Lessons Learned**

Overall, the evaluation shows that this feature meets the requirements of the use cases that need a multi-provider environment.

**8.4 Service Offering**

**Discovery of Services and SLAT**

*Component/s*: SLAT Registry, PSS Utilities

The feature is implemented by the SLA Template registry, which provides persistent storage for the templates, and the Publish/Subscribe System Utilities, a client to the Pub/Sub system for template advertisements. Once a template is stored in the registry, it is automatically advertised making use of the PSS utilities. Receiving parties can then store it locally and use it for discovery purposes. The process is detailed in D.A5a [10].

*Related to innovation*: SLA Template-based Service Discovery

In SLA@SOI, we assume that the use of a service is always regulated by an (implicit or explicit) SLA between service provider & customer. This SLA contains
both a functional description of the offered service, together with various non-functional terms, including:

- quality of service (QoS) parameters,
- behaviors that the customer or provider contractually guarantee to the other party,
- penalties to pay in case of infringements,
- monitoring policies relating to the detection of infringements

In this context, the concept of service discovery takes on added significance. In the traditional view, a customer typically searches for a service endpoint providing specific functional properties. In an SLA context, however, the customer searches for a provider able to supply a specific kind of service and also willing to make certain guarantees and agree to certain terms. In other words, we assume that in real business scenarios, customers search for a specific kind of SLA – rather than just a specific kind of service.

Most service registries in common use (such as UDDI), however, do not provide any standard means to represent SLA Templates, and in particular they are not capable of matching semantically equivalent SLA terms expressed in different syntactic forms. Indeed, the problem of semantically matching customer queries against advertised SLA Templates is non-trivial & is best handled by logical reasoning & constraint-satisfaction mechanisms – as opposed to simple syntactic pattern matching.

**A-Line evaluation**

**Associated tests: 14**

**Passed tests:** 100%

**Status:** Completed

**B-Line evaluation**

**Documentation:** Sufficient

**Architecture:** Good

**API:** Good

**Implementation:** Good

**Integration Complexity:** Low

**Issues:** No issues reported.

**Lessons Learned**

Overall, the B-line partners have found the feature implemented and documented sufficiently well. The fact that these are generic components that do not require customization, seems to be well-received. Next steps include the completion of
template queries based on offered guarantees, and filtering of incoming advertisements based on custom preferences.

**Product Management**

**Traceability**

**Component/s:** Business Manager is the component that implements this functionality. It offers two ways for the product management. An internal API and a GUI inside Business Web Tools (part of the Business Manager). It is possible to look up in A2 official deliverable [7].

**Related to innovation:** “Comprehensive Business Management suite for e-contracting”. This feature is part of the novel e-services marketplace through which the commercial offer can be activated and configured.

**A-Line evaluation**

**Associated tests:** 3  
**Passed tests:** 100%  
**Status:** Completed

**B-Line evaluation**

**Documentation:** Good  
**Architecture:** Good  
**API:** Good  
**Implementation:** Good  
**Integration Complexity:** Low  
**Issues:** No issues reported.

**Lessons Learned**

The needs of the use cases have been addressed in order to facilitate the integration of the components.

**Product Discovery**

**Traceability**

**Component/s:** Business Manager is the component that implements this functionality. It offers two ways for product discovery. A web service interface that can be used by external systems framework and a GUI inside Business Web
Tools (part of the Business Manager). It is possible to look up in A2 official deliverable [7].

**Related to innovation:** “Comprehensive Business Management suite for e-contracting”. This feature is part of the novel e-services marketplace through which customers can look for the products offered in the platform.

**A-Line evaluation**
**Associated tests:** 1  
**Passed tests:** 100%  
**Status:** Completed

**B-Line evaluation**
**Documentation:** Sufficient  
**Architecture:** Good  
**API:** Good  
**Implementation:** Good  
**Integration Complexity:** Low

**Lessons Learned**
Product searching based on categories has been used in use cases.

**Product Definition**

**Traceability**
**Component/s:** Business Manager is the component that implements this functionality. It offers two ways for the product definition. An internal API and a GUI inside Business Web Tools (part of the Business Manager). It is possible to look up in A2 official deliverable [7].

**Related to innovation:** “Comprehensive Business Management suite for e-contracting”. This feature is part of the novel e-services marketplace through which the services provide and configure the commercial offer (products, prices, SLAs, etc.).

**A-Line evaluation**
**Associated tests:** 10 (1 specific and 9 common tests)

**Passed tests:** 100%
Status: Completed

B-Line evaluation
Documentation: Good
Architecture: Good
API: Good
Implementation: Good
Integration Complexity: Low
Issues: No issues reported.

Lessons Learned
Documentation was improved.

SLAT Definition

Traceability

Component/s: SLAT Registry
The SLAT Registry provides persistent storage for SLA templates. It can be queried using templates as input and yielding all compatible (matching) templates as output. Search is possible either using template content (e.g. negotiable properties), or metadata.

Related to innovation: SLA Model
Describing a Service Level Agreement using machine-readable formats can be a very complex task. Balancing between expressiveness equal to the natural language used for describing traditional (paper) contracts, and computability of a representation, is challenging and requires careful design. The present work, committed as part of task TA5.1, takes into account the requirements of the project’s use cases, as well as existing State of the Art, to create an extensible model that is technology-independent and suits very diverse application scenarios. Existing prior art is typically bound to specific technologies for describing the SLA (XML in most cases). This is an approach we chose to avoid, the reason being that XML is not an efficient representation for certain activities within the scope of the project, such as reasoning over templates during discovery; negotiation; conformance checking; etc. Therefore, a more flexible approach was preferred; the SLA model needs to support (XML) serialization, but not be limited by it. Similarly, most efforts are specifically targeting Web Services, while SLA@SOI wishes not to be bound by specific service-orientation technologies.

A-Line evaluation
Associated tests: 14
Passed tests: 100%
Status: Completed

B-Line evaluation
Documentation: Good
Architecture: Excellent
API: Excellent
Implementation: Good
Integration Complexity: Low

Issues: Yes SLATs can be defined, but requires expert level knowledge of the SLA model. Perhaps an editor would help with this, as per B3’s suggestion. The SLA Model does not support explicit dependency between SLAs.

Lessons Learned
The industrial use cases seem to be satisfied by the expressiveness of the SLA model, but on the same time find it to be very complex when creating a template. Specifically, partners seemingly had problems creating valid templates (using the XML rendering). Due to these reasons, the possibility of a graphical editor that supports the task of SLAT creation is evaluated (this is significant effort that was not foreseen by the DoW and as such requires some planning).

One more comment received was a request to allow explicit SLA dependencies as part of the model. This is a B6 requirement, due to the need for regulatory and governmental authorities to be able to monitor SLAs together. As B6 used Y2 model, there was no high priority for this requirement and will be a subject of future research.

8.5 Service Negotiation

Business SLA Negotiation

Traceability
Component/s: The Business SLA Negotiation functionality is covered by Business Manager and the BSLAM. The web UI supports the definition of prices, promotions, offers, etc. and the Generic SLAM interface is reused to trigger the negotiation processes. More details in A2 official deliverable [7]. Details on generic negotiation can be found within A5 deliverable [10].

Related to innovation: “Dynamic Negotiation of agreements“. This feature implements the protocols and strategies for the negotiation of SLAs from the business point of view (prices, offers, discounts, penalties, etc.)
**A-Line evaluation**

Associated tests: 7  
Passed tests: 100%  
Status: Completed  

**B-Line evaluation**

Documentation: Good  
Architecture: Good  
API: Good  
Implementation: Sufficient  
Integration Complexity: Low  
Issues: No issues reported.

**Lessons Learned**

Provider business negotiation has been implemented allowing SLAT changes manually.

**SLAT Customization**

**Traceability**

Component/s: Business Manager  
Business Manager is the component that implements this functionality. It offers an internal API to deliver this functionality. It is possible to look up in A2 official deliverable [7].

Related to innovation: “Customization of Business SLA definitions taking into account the requirement of customer and service provider”. This feature provides the intelligence to assess and adapt the definition of the SLATs offered to different customers taking into account their characteristics and preferences, using pre-defined policies.

**A-Line evaluation**

Associated tests: 1  
Passed tests: 100%  
Status: Completed.

**B-Line evaluation**

Documentation: Medium
Architecture: Good

API: Good

Implementation: Good

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned
Customer profile recovery has been widely explained and documented in A2 deliverable [7].

Business SLA planning for negotiation

Traceability

Component/s: Business SLA Manager

Business SLA Manager is the component that implements this functionality but the behaviour is driven by the Business Manager. It offers web services to achieve a business negotiation. The web services used are the defined in generic negotiation component and the business negotiation is customized from the Business Manager. It is possible to look up in A2 official deliverable [7].

Related to innovation: “Dynamic negotiation of agreements”. Proposal and experimentation of different negotiation protocols and strategies for multiparty negotiation at the different phases of the service delivery.

A-Line evaluation

Associated tests: The tests of “Business Negotiation” issue are valid for this feature.

Passed tests:

Status: Completed

B-Line evaluation

Feature: Business SLA planning for negotiation

Documentation: Sufficient

Architecture: Good

API: Sufficient

Implementation: Good

Integration Complexity: Low
**Issues:** No issues reported.

**Lessons Learned**
Business negotiation example was implemented in A2 work package.

**Automatic orchestration of customisable (re)negotiation protocol**

**Traceability**

*Component/s:* Protocol Engine

The Protocol Engine (PE) implements a state machine that can orchestrate different protocols (i.e. message exchanges) for the purpose of negotiation. The PE is closely tied to the Planning/Optimization Component (POC) of the SLAM to which it belongs, as the latter takes the decisions while the PE takes care of negotiation mechanics. This effectively decouples stateless decision making from stateful message exchange, and allows the easy integration of different algorithms and different negotiation protocols.

**Related to innovation:** SLA Negotiation

Currently there is a large gap between implemented SLA solutions and scientific proposals. Almost all solutions use the above-mentioned simple protocol (one exception is BREIN which uses the FIPA Contract Net protocol in an agent-based environment), although a number of (SLA) negotiation proposals exist. Task A5.4 evaluated those in the light of its use cases and worked together with the global community towards a generic SLA negotiation standard.

**A-Line evaluation**

*Associated tests:* 9  
*Passed tests:* 9  
*Status:* Completed

**B-Line evaluation**

*Documentation:* Good  
*Architecture:* Sufficient  
*API:* Good  
*Implementation:* Good  
*Integration Complexity:* Low  
*Issues:* No issues reported.
Lessons Learned

Evaluation of the feature came only from B3 and B6 use cases. According to them, the feature is sufficiently well implemented and documented.

Several projects and solutions were considered by Task A5.4 in order to understand the state of the art on SLA Negotiations. As a result, it was realised that the use cases which SLA@SOI focuses on specifically target bilateral negotiations which can potentially be part of hierarchies. It was understood that the Year 2 approach of keeping protocol governed interaction behaviour decoupled with the negotiation strategy and communication mechanism is the most flexible way to provide automated negotiations for a host of possible scenarios including those of SLA@SOI use cases. We learned from several design aspects of the SecSe project which also caters for bilateral negotiations but couples negotiation strategy related rules with behavioural ones - this surfaced as an engineering difference between SLA@SOI approach of writing protocols and SecSe as well as many other projects. Shortcomings from recent projects like Kasbah, ASAPM and CAAT were also studied. Kasbah for instance does not have a well-defined counter-offer support, ASAPM and CAAT as well as BREin couple low level communication mechanisms (i.e. synchronous and asynchronous modes of messaging) with protocol description – a requirement that SLA@SOI defines as strictly separable in order to not limit adoption for scenarios that favour e.g., web services based communication over the FIPA CNP styled asynchronous communication. Approaches from earlier projects like OPELIX, Inspire, Aspire and e-Agora were also considered. We reached an understanding that there cannot be a single protocol that would amicably serve the needs for all possible scenarios.

We found strength in SLA@SOI’s design principles of developing as generic an approach as possible to encode protocols that concretely define only agent interaction behaviour, while loosely coupling the negotiation intelligence in domain specific components – the POCs which are consulted during negotiation time by the Protocol Engine. Similar lessons have been drawn by several other works that strive for a generic protocol description and execution. Thirdly, we also abstract over the underlying communication mode that could be either synchronous (as was required by all SLA@SOI use cases) or asynchronous. For the latter or any other communication mode per se, merely a separate implementation of SyntaxConverter component could be used which provides for message invocation and delivery in addition to SLA model based transformations among various serializable forms – needed to marshal parameters.

Exiting SLA standards were also considered. Although a one for all global standards does not exist, the closest work is that delivered by the OGF in the form of WSAG and WSAGN. These have emerged as de-facto standards to represent an SLA (agreement) template and negotiate using given protocol. Since SLA@SOI has developed its own model, the challenge lies in 1) bridging the two models and 2) allowing a compatibility of SLA@SOI based projects to be able to interoperate with WSAG/WSAGN based projects. In regards with 1, reasonable effort was invested to be able to transform essential elements and terms of WSAG based templates to those of SLA@SOI based. For 2, a feasibility study was conducted with more interest towards WSAGN which supports multi-round negotiation capability that is of basic importance in SLA negotiations. It is worth noting here that SLA@SOI negotiation platform caters for several protocol level parameters as part of protocol description which are not available in WSAG/WSAGN. These enrich negotiations by allowing a pre-negotiation mechanism that fixes protocol parameters between negotiating parties in a mutually consensual manner. WSAGN on the other hand advocates a host of signalling scenarios which allow parties to communicate with each other using a certain mode of communication. Among these, the client-server asymmetric signalling scenario in principle can easily be made interoperable with SLA@SOI
negotiation platform since all required engineering artefacts are mostly available. The ongoing work on WSAGN also posed a concern for interoperability efforts, nevertheless our timely assessment and resultant influence on SLA@SOI negotiation platform benefits us in terms of situating our work and furthering the state of art.

**Out-of-band SLA registration**

**Traceability**

**Component/s:** SLA Registry

The SLA Registry offers persistent storage for SLAs and their state changes. It includes interfaces that may be invoked either internally within a SLAM, after a negotiation has concluded successfully, or externally for storing an already existing SLA. This feature refers to the latter. It allows that a framework manager can store a SLA that has been negotiated among humans, has been formatted according to the SLA model and some respective rendering, and can be stored after invoking the exposed interfaces.

**Related to innovation:** None.

Does not apply.

**A-Line evaluation**

**Associated tests:** 2

**Passed tests:** 2

**Status:** Completed

Fully implemented and tested (IRegister interface for inserting and updating an SLA).

**B-Line evaluation**

**Documentation:** Good

**Architecture:** Good

**API:** Good

**Implementation:** Sufficient

**Integration Complexity:** Low

**Issues:** No issues reported.

**Lessons Learned**
Feedback was that the feature is sufficiently well implemented and documented. Use case B5 provided a lower rating for the API of the feature; as such, in Y3 WP A5 has worked with B5 to enhance the API accordingly.

The feature is fully implemented.

8.6 Service Provisioning

Coordination of Provisioning

Traceability

Component/s: Provisioning/Adjustment Component

The Provisioning/Adjustment Component (PAC) addresses the requirement to handle provisioning-time tasks separately from negotiation-time tasks. Although provisioning MAY take place at the end of the negotiation, it may well be the case that it can happen much later. For instance, someone may require a number of VMs for a few days later, but wants to establish the SLA (and therefore, reserve the resources) well in advance. Should the VMs be activated right after the negotiation ends successfully, those resources would be wasted until the service described in the SLA is supposed to become active.

In addition, there is the need to synchronize complex provisioning tasks, where the order of provisioned s/w and h/w is critical (e.g., a VM running a DB image must be available before the DB can be configured to a particular scenario). This implies a “provisioning hierarchy” that reflects, to a large extent, the service hierarchy. The PAC, and this feature in specific, is expected to address this requirement using the respective interfaces as defined and explained in D.A5a [10].

Related to innovation: N/A

N/A

A-Line evaluation

Associated tests: 11

Passed tests: 11

Status: Completed

While the Provisioning and Adjustment Component is domain-specific, a generic implementation has been provided, aiming to be easily reused/adapted by the use cases. Furthermore, concrete software and infrastructure components for the provisioning functionality have been implemented and integrated into the Open Reference Case basic scenarios.

B-Line evaluation

Documentation: Sufficient

Architecture: Good

API: Sufficient
Implementation: Good

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned
Overall, the evaluation shows that the software provisioning feature implemented as part of the PAC component largely meets the requirements of the use cases. In particular, the architecture, implementation and facility of integration are well received. Documentation has been improved significantly in the open source release of the SLA@SOI framework.

On the fly Deploy of Monitoring

Traceability

Component/s: Monitoring Manager

The Monitoring Manager is the SLAM component that decides what are the proper monitoring framework components that must be deployed, as part of the provisioning process, in order to properly monitor an established SLA. This involves a complex process to “translate” from the SLA model to a respective event-calculus based description, which is provided as input to the monitoring system. Deliverable D.A5a [10] provides detailed information about the produced output, the translation process and the general functionality of the Monitoring Manager.

Related to innovation: N/A

N/A

A-Line evaluation

Associated tests: 3

Passed tests: 3

Status: Completed

The Monitoring Manager can process SLAs according to the SLA model, and produce monitoring configurations for software services.

B-Line evaluation

Documentation: Sufficient

Architecture: Excellent

API: Good
Implementation: Sufficient
Integration Complexity: Medium
Issues: No issues reported.

Lessons Learned
Partners have received the feature sufficiently well, overall. Some further work needs to be done with regard to the B4 use case, which relies heavily on A4 and for which the monitoring features (the input to the Monitoring Manager) have not been defined. Given those definitions, B4 will integrate the feature as well.

In Y3 the Monitoring Manager implementation has been extended according to SLA model changes. Further documentation has been provided to better help users of the platform use the feature to its full extent.

Dynamic Binding Setting

Traceability
Component/s: Dynamic Orchestration Engine (DOE)
This feature is implemented within the Dynamic Orchestration Engine which is a specialized form of ManageabilityAgent.

Related to innovation: Development of a Dynamic Orchestration Engine (DOE): this feature allows to set, at runtime, different bindings for different instances of a same deployed BPEL process. In Y3 the DOE has been extended to support a special type of invocation activity for multi-party conversation (simultaneous invocation of different partners implementing the same interface). The results of the invocation of each partner participating in a same multi-party invocation are automatically aggregated by the DOE according to merging criteria specified at design time. Also the binding of multi-party invocations can be set at runtime in a way similar to normal bi-party invocations.

A-Line evaluation
Associated tests: 1
Passed tests: 100%
Status: Completed
The feature is completely implemented.

B-Line evaluation
Documentation: Good
Architecture: Good
API: Good
Implementation: Sufficient

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned
Evaluation of the feature comes from B5 and B6 use cases. According to them, the feature is sufficiently well implemented and documented.

Possible improvements could be applied to support more sophisticated merging criteria for additional use cases.

Virtual Hardware Infrastructure Provisioning

Traceability
Component/s: Infrastructure Service Manager.

The Infrastructure Service Manager is the lowest level component responsible for actual provisioning of virtual infrastructure. It works very closely with the Infrastructure SLA Manager, which manages all Infrastructure SLA-level concerns. Infrastructure Service Managers have been developed on top of both Apache Tashi and OpenNebula, both of which exposing the OCCI interface. These are described in deliverable D.A4a [9].

Related to innovation: SLA-enabling Infrastructure Providers, Harmonized interface to access heterogeneous infrastructure resources.

Regarding the state of the art, there remain no providers that offer machine readable SLAs, and there remains no alternative open standard for harmonised access to heterogeneous infrastructure resources. Hence, the consortium’s attention and co-chairing of the recently published OCCI standard.

A-Line evaluation
Associated tests: 112 Unit tests, 49 integration tests

Passed tests: 100%

Status: Completed

Initial implementations of the Infrastructure Service Manager have been completed. They are built on top of the Apache Tashi and OpenNebula provisioning systems.

B-Line evaluation
Documentation: Good

Architecture: Good
API: Excellent

Implementation: Good

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned

The cloud computing industry is still unsure about providing a consistent virtual interface to heterogeneous infrastructure, despite the call for interoperability from customers. The OCCI provides a unique standard to help deliver the interoperability necessary, and has received a significant amount of attention from SLA@SOI, and is gaining attention from the cloud community.

Although OCCI addresses the manipulation of virtual infrastructure, other significant challenges remain, including the portability of virtual machine images and the ability to dynamically reprovision arbitrary VMs across cloud providers. Cloud platform providers will be keen to support dynamic reprovisioning between instances of their cloud platform, but complete interoperability is something that may need to be encouraged at a higher level before it is addressed.

Software Provisioning

Traceability

Component/s: SoftwareServiceManager

SoftwareServiceManager provides a default implementation for ORC based software provisioning. However, to accommodate use case specific custom provisioning logic, an interface is defined which can be implemented by use cases to plug-in use case specific software provisioning logic

Related to innovation: N/A

A-Line evaluation

Associated tests: 6

Passed tests: 100%

Status: Completed

B-Line evaluation

Documentation: Good

Architecture: Excellent

API: Good
Implementation: Good

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned
Software Provisioning evaluation has shown that the overall architecture was sufficient to support a generic provisioning methodology which can be customized for domain specific provisioning processes.

Software Landscape

Traceability

Component/s: SoftwareLandscape
Software Landscape is implemented in the SoftwareLandscape component and is accessible through the interface exposed by the SoftwareLandscapeManager entity. SoftwareLandscape is available as part of the SoftwareServiceManager

Related to innovation: Software Landscape Modelling
The main innovation of the activity is the capability to capture model elements beyond the core entities i.e. software, service and their relationships. Landscape metamodel designed in the task enables the service providers to represent monitoring configurations, service packaging and service dependencies.

A-Line evaluation

Associated tests: 3
Passed tests: 100%
Status: Completed
Software Landscape has been implemented and integrated into the SoftwareServiceManager component.

B-Line evaluation

Documentation: Excellent

Architecture: Excellent

API: Good

Implementation: Good

Integration Complexity: Low
Lessons Learned
SoftwareLandscape provides consistent information in the form of landscape models which can be used during various phases of the lifecycle. Having landscape models stored in SoftwareLandscape ensures that the service operations can be automated and performed with marginalized error.

8.7 SLA Enforcement

Automated SLA Enforcement Coordination

Traceability

Component/s: Provisioning/Adjustment and Planning/Optimization Components

The enforcement of a SLA, following its establishment and provisioning, is of great importance both to providers and to customers. Naturally, during runtime there may be circumstances that affect the service, and therefore the SLA with the agreed service parameters. Therefore, it is important to have mechanisms that can react if a monitoring event arrives that indicates a violation, or even a warning that a violation is imminent. The Provisioning/Adjustment Component (PAC) does exactly that. When something is found to be wrong, it tries to locate the exact problem, and take a corrective action e.g. by reconfiguring the service. When this is not possible, but there are additional actions that must be taken, the PAC may consult with the Planning/Optimization Component (POC), which is mostly used during the SLA negotiation. The POC will try to create a new plan and submit it to the PAC for provisioning (an original plan has been created and submitted after the SLA was established). If it is not possible to construct a new plan, re-negotiation may take place, initiated by the POC.

Related to innovation: N/A

A-Line evaluation
Associated tests: 10
Passed tests: 10
Status: Completed

A generic implementation of the Provisioning and Adjustment Component has been provided, where the adjustment decisions rely on a rule engine. This facilitates the adaptation needed by the use cases. Furthermore, concrete software and infrastructure Adjustment components have been implemented and integrated into the Open Reference Case basic scenarios.

B-Line evaluation
Documentation: Good
Architecture: Good
API: Good

Implementation: Good

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned

The evaluation shows that the developed Adjustment component has been useful for all the use cases. In particular, the easy integration of this component has been well evaluated. API definition and implementation have been refined and improved in the third year. Documentation has been centralized and improved with more detailed descriptions and examples.

Virtual Hardware Infrastructure Adjustment

Traceability

Component/s: Infrastructure SLA Manager - Infrastructure PAC, Infrastructure Service Manager

The Infrastructure PAC is responsible for making SLA-aware adjustment decisions based on the results from infrastructure monitoring. The Infrastructure Service Manager includes internal resource-aware optimisation. For more information please see deliverable D.A4a SLA Aware Infrastructure Management [9].

Related to innovation: SLA-enabling Infrastructure Providers, Predictive Compression and Decompression of Infrastructure

Although other providers allow infrastructure adjustment to be made, adjustments are typically manually driven. VMware has recently demonstrated runtime compression and decompression of virtual infrastructure based on current load - the ability of it to respond to predictions has not yet been established. SLA@SOI has implemented vertical and horizontal scaling, and resource-utilisation ratios are used to identify potential bottlenecks.

A-Line evaluation

Associated tests: 14

Passed tests: 100%

Status: Completed

An initial version of the Infrastructure PAC has been implemented. It drives the SLA-aware re-provisioning of infrastructure based on alerts from the monitoring system.

B-Line evaluation
**Lessons Learned**

Useful workload-aware prediction is a challenging area that has scope for further research. However, SLA@SOI has demonstrated that techniques such as resource-utilisation ratios can give a useful indication of whether adjustment would be useful.

**Software Adjustment**

**Traceability**

*Component/s:* SoftwarePAC

It is used in the Provisioning and Adjustment Components inside the SLA Managers. Being the PAC domain specific, there are several implementations: reference implementation focused on the Open Reference Case (ORC), both software and infrastructure, as well as implementations for the industrial use cases.

*Related to innovation:* N/A

**A-Line evaluation**

*Associated tests:* 5

*Passed tests:* 100%

*Status:* Completed

**B-Line evaluation**

**Documentation:** Good

**Architecture:** Good

**API:** Good

**Implementation:** Good

**Integration Complexity:** Low

**Issues:** No issues reported.
Lessons Learned

Overall, the evaluation shows that the Software Adjustment component developed within A3 work package has been well considered for the use cases. In particular, the architecture, API, implementation and the easy integration of this component has been well evaluated. Documentation has been well improved in the last year of the project.

Automatic Binding

Traceability

Component/s: Dynamic Orchestration Engine (DOE)

This feature is implemented within the Dynamic Orchestration Engine which is a specialized form of ManageabilityAgent.

Related to innovation: Development of a Dynamic Orchestration Engine (DOE): this feature allows to execute abstract processes described by using standard WS-BPEL language for process description and a decoupled SLAT based representation for describing the component services. In Y3 the implementation has been extended to support also multi-party invocations, and to take into account the monitored quality during the automatic selection of services (the best service is selected according to an order criteria specified in the binding rule).

A-Line evaluation

Associated tests: 1

Passed tests: 100%

Status: Completed

B-Line evaluation

Documentation: Good

Architecture: Good

API: Good

Implementation: Sufficient

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned

The evaluation shows that the implemented feature meets the requirements of B5 and B6 use cases.
Possible improvements could be applied to support more sophisticated binding rules for additional use cases.

**Business Adjustment**

**Traceability**

*Component/s:* BusinessPAC

This feature is implemented as part of the Business Manager component, and takes care of the economical impact of the service performance. It receives information of the violations produced at lower levels (software and infrastructure), calculates the penalties to be paid based on the SLA, and triggers actions aim to reduce the cost due to the malfunctioning of the service. See deliverable DA2a [7].

**Related to innovation:** Business SLA post-sale management

**A-Line evaluation**

*Associated tests:* 6

*Passed tests:* 100%

*Status:* Completed

Business adjustment feature is implemented in the ViolationPenalty component within the Business Manager. It uses the information coming from the lower level PACs through the <<control/track>> interaction, and the information from the database. This component is used by the ORC scenario and some of the use cases.

**B-Line evaluation**

*Documentation:* Sufficient

*Architecture:* Good

*API:* Good

*Implementation:* Sufficient

*Integration Complexity:* Low

*Issues:* No issues reported.

**Lessons Learned**

The evaluation shows that the Business Adjustment component has been useful for some use cases. In particular, the easy integration of this component has been well evaluated.
8.8 Runtime Prediction

Runtime Infrastructure Metrics Prediction

Traceability

Component/s: PredictionClient, PredictionService

This feature is implemented as a couple of independent services that can be deployed according to the specific requirements of the use case. PredictionClient gathers local instrumentation metrics and provides computed predictive values over those metrics. The computation of prediction is done using different techniques following approaches such as heuristics and/or machine learning, according to requirements. As the prediction is a resource intensive operation, the computational part of prediction can be offloaded to a dedicated PredictionService(s) nodes.

PredictionClient and PredictionService implement a message passing actor model to exchange data and provision prediction computations. PredictionService is also configurable to be able to offer interfaces for the consumption of prediction data.

Related to innovation: Predictable Systems Engineering

A-Line evaluation

Associated tests: 4

Passed tests: 100%

Status: Completed

B-Line evaluation

Documentation: Good

Architecture: Good

API: Excellent

Implementation: Good

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned

Infrastructure Runtime Prediction is a challenging area for resource planning: it is computationally intensive, the results are dependent on consistency of the workload, and accurate predictions are typically not possible beyond a short time
frame. Some potential for future improvement remains. First, the implementation could be improved to allow the dynamic configuration of prediction algorithms which should be chosen at deployment time. Also, for Machine Learning based prediction implementations the window of time and data re-sampling that is used for training the prediction models is fixed at deployment time and it could be made dynamic and configurable also during runtime. One area also of improvement for runtime prediction is to integrate the underlying implementation to standardised scalable backends such as Hadoop/Mahout.

Runtime Software SLA Violations Prediction

Traceability

**Component/s:** Software Level Prediction System (SLPS)

The SLPS is part of the Software Level Monitoring System (SLMS). Building element of the SLMS is the Reasoning Component. A reasoning component (aka Reasoning Component Gateway (RCG)) can perform monitoring, prediction, or it can combine both monitoring and prediction.

A RCG receives a reasoning component configuration (that is part of a monitoring system configuration) and translate it into monitoring and/or prediction specification. The SLMS is described in deliverable D.A3a whilst the prediction system is described in deliverable D.A6a [11].

Related to innovation:

Three limitations of existing techniques that make them falling short of providing adequate support for run-time prediction of SLA violations are:

- they tend to focus on system infrastructure properties (e.g., network and server properties) rather than service level application based properties (e.g., service throughput, mean time to failure).
- they tend to focus on the prediction of specific types of properties without providing a more generic framework for building predictors that can cover a wide or even the whole spectrum of service properties that can be part of an SLA
- they are not integrated with environments for monitoring SLAs for service-based systems

We addresses the above key issues by realising a framework that provides an integrated architecture for SLA monitoring and prediction that is capable of supporting the latter activity through the deployment of a built-in set of model-based predictors. The framework doesn’t focus on specific QoS properties; on contrary it provides the infrastructure for creating user-defined QoS predictors.

A-Line evaluation

**Associated tests:** 0
**Passed tests:** N/A
**Status:** Completed
**B-Line evaluation**

**Documentation:** Good

**Architecture:** Sufficient

**API:** Good

**Implementation:** Sufficient

**Integration Complexity:** Low

**Issues:** No issues reported.

**Lessons Learned**

A generic prediction model for mean QoS constraints (or, equivalently, SLA guarantee terms in the SLA@SOI terminology) has been developed. The experimental evaluation of the model with several such properties (e.g., MTTR, MTTF, mean service completion time) has shown positive results in terms of accuracy and completeness. Separate tests have also been performed with some infrastructure properties (e.g. web server response times) outside the scope of SLA@SOI. Future work should investigate the enhancement of the model with the fitting of multi-variable probability distribution functions and the improvement of the configurability of the model (e.g. selection of historical data sample).

**8.9 Service Monitoring**

**Monitoring Coordination**

**Traceability**

**Component/s:** MonitoringManager

The MonitoringManager (MM) coordinates the generation of a monitoring configuration of the system, given a set of Agreement and Guarantee Terms in an SLA model. A monitoring configuration describes which components to configure and how their configurations can be used to obtain events from service monitoring. See deliverable D.A3a (SLA Aware Service Management) for more information. The monitoring configuration supports monitoring agents in the monitoring architecture by expressing output ids of events for reasoning about guaranteed states.

**Related to innovation:** Advanced SOA Management

**A-Line evaluation**

**Associated tests:** 3 (incremental versions of the A4 SLA Template for testing)

**Passed tests:** 100%

**Status:** Completed
**B-Line evaluation**

**Documentation:** Good

**Architecture:** Excellent

**API:** Excellent

**Implementation:** Good

**Integration Complexity:** Low

**Issues:** No issues reported.

---

**Lessons Learned**

The MonitoringManager component has been successfully tested to extract, parse and configure monitoring components from terms expressed as Guaranteed States in the current examples of SLA templates and models. As part of this test we have generated example Monitoring System Configurations which are used by monitoring agents to instantiate and respond to monitoring events in the monitoring system.

---

**Virtual Hardware Infrastructure Observation and Violation Detection**

---

**Traceability**

**Component/s:** Low Level Monitoring System (LLMS).

The Low Level Monitoring System receives monitoring configuration information, deploys the appropriate monitors and gathers, analyses and propagates violations and warnings. The LLMS is described in Deliverable D.A4a [9].

**Related to innovation:** Standardisation of Infrastructure Monitoring Interfaces, Self-Provisioned Distributed Infrastructure Monitoring.

---

**A-Line evaluation**

**Associated tests:** 15

**Passed tests:** 100%

**Status:** Completed

An initial Low Level Monitoring System has been implemented and deployed on top of Apache Tashi. It can send monitoring events to the Historical Database, to the Infrastructure Provisioning Adjustment Component (IPAC), and to other subscribed components.

---

**B-Line evaluation**

**Documentation:** Good
**Architecture:** Good

**API:** Sufficient

**Implementation:** Good

**Integration Complexity:** Medium

**Issues:** No issues reported.

**Lessons Learned**

This feature is essential not only for infrastructure based use cases, but also when a service is deployed as a set of virtual appliances. We have learned that infrastructure instrumentation should focus on available virtual hardware resources (e.g. CPU speed, memory size, disk size, network throughput) which would be sufficient for most purposes, as these are the parameters that the service user understands and should have guaranteed in the SLA.

**Software Violation Detection**

**Traceability**

**Component/s:** RCG, EVEREST

**Related to innovation:** Supporting detection of violations of SLAs expressed in SLA(T) at the software level.

The developed RCG supports the transformation of SLA(T) onto EC-Assertion. Hence, it provides a possible formal foundation for SLA(T) itself since EC-Assertion is a formal version of Event Calculus (first order temporal logic language).

The implementation of EVEREST has also been extended to provide support for more complex data aggregation functions and basic relational conditions, as well as the use of time expressions in fluent related predicates.

**A-Line evaluation**

**Associated tests:** 6

**Passed tests:** 100%

**Status:** Completed

75 percent completion of implementation/integration. The components have been compiled with the rest of the SLA@SOI framework in Maven but its availability as an OSGi service has not been fully tested yet.

**B-Line evaluation**

**Documentation:** Good

**Architecture:** Good
**API:** Sufficient

**Implementation:** Sufficient

**Integration Complexity:** Medium

**Issues:** No issues reported.

**Lessons Learned**

The implementation of EVEREST has also been extended to provide support for more complex data aggregation functions. Further extensions to the RCG EVEREST have been provided to support also guaranteed actions.

**Software monitoring rules extraction from SLA**

**Traceability**

**Component/s:** MonitoringManager

The MonitoringManager (MM) coordinates the generation of a monitoring configuration of the system, given a set of Agreement and Guarantee Terms in an SLA model. It decides, for an SLA model instance it receives, which is the most appropriate monitoring configuration according to a monitoring component selection criteria. A monitoring configuration describes which components to configure and how their configurations can be used to obtain events from service monitoring. See deliverable D.A3a [8] for more information.

**Related to innovation:** Advanced SOA Management

**A-Line evaluation**

**Associated tests:** 3 (incremental versions of the A4 SLA Template for testing)

**Passed tests:** 100%

**Status:** Completed

**B-Line evaluation**

**Documentation:** Good

**Architecture:** Sufficient

**API:** Good

**Implementation:** Sufficient

**Integration Complexity:** Medium

**Issues:** No issues reported.
Lessons Learned

The MonitoringManager component has been successfully tested to extract, parse and configure monitoring components from terms expressed as Guarantee States in the current examples of SLA templates and models. The selection process of these components has been extended to perform some complex selection based upon preferential criteria (potentially expressed in the SLA itself, e.g. cost optimisation, provider-based optimisation etc).

Virtual Hardware Monitoring Rules Extraction from SLA

Traceability

Component/s: SLA Level Monitoring System

The SLA Level Monitoring System receives SLAs and extracts monitoring configuration information to evaluate which QoS terms should be analysed. See Deliverable D.A4a SLA Aware Infrastructure Management for more information.

Related to innovation: Self-Provisioned Distributed Infrastructure Monitoring.

A-Line evaluation

Associated tests: 14
Passed tests: 100%
Status: Completed

B-Line evaluation

Documentation: Sufficient
Architecture: Good
API: Good
Implementation: Sufficient
Integration Complexity: Medium
Issues: No issues reported.

Lessons Learned

Infrastructure Monitoring is a key requirement for SLA-aware infrastructure and thus generic interfaces need to provide a generic means to understand and manipulate monitors that are required as per SLAs. OCCI extensions have been developed to support this functionality. Further enhancements to describe and configure infrastructure monitoring are being considered in future work.
8.10 Service Reporting

SLA Management

Traceability
Component/s: Business Manager
This feature makes available all the administrative information related with the lifecycle of a given SLA. In particular, the detailed information of the content of the Agreement, any extension and revision to it, and the history of its performance should be persistently stored and accessible to both the service provider and the customer.

It is possible to look up in A2 official deliverable [7].

Related to innovation: Business SLA post-sale management.

A-Line evaluation
Associated tests: 12
Passed tests: 100%
Status: Completed

B-Line evaluation
Documentation: Good
Architecture: Excellent
API: Good
Implementation: Good
Integration Complexity: Low
Issues: No issues reported.

Lessons Learned
The mean evaluation results show that the feature is adequate to be used by the use cases. It has been done a closer interaction with the leaders of the B-line work packages to understand the specific requirements they have regarding the specification, implementation and documentation of this functionality.
**Push/Pull Business SLA Reporting**

**Traceability**

**Component/s:** Reporting

This feature allows the service provider to generate reports about the administrative information of an SLA, including the description of the purchased service, validity dates and the conditions of the agreement, as well as information about the performance of the service, in terms of violations and penalties occurred. The generated report is periodically sent to the customer, and also an interface is offered so the customer can request the business information on demand.

The Reporting component of the Business Manager is the one that implements this functionality, and offers its functionality through different interfaces: an internal API, web services and a GUI inside Business Web Tools (part of the Business Manager). It is possible to look up in A2 official deliverable [7].

**Related to innovation:** Business SLA post-sale management

**A-Line evaluation**

**Associated tests:** 9

**Passed tests:** 100%

**Status:** Completed

**B-Line evaluation**

**Documentation:** Good

**Architecture:** Good

**API:** Excellent

**Implementation:** Excellent

**Integration Complexity:** Low

**Issues:** No issues reported.

**Lessons Learned**

SLA Reporting fulfils the more complex requirements of the use cases. Significant improvements were included in order to allow the dynamic configuration of the content of the reports. Also different methods to deliver the reports has been considered, apart from the already implemented web services (for the push interface) and the email (for the pull one).
**SLA Status Reporting**

**Traceability**

*Component/s:* PAC, SLA Registry and SLA Management Console

During runtime, as SLAs are being established, provisioned, monitored, renegotiated and eventually expired or terminated, their status is persistently stored in the SLA Registry, after the PAC receives it from the monitoring system and forwards it accordingly. The information on each SLA includes how exactly it relates to other SLAs, i.e. the dependencies among them. Based on this information, it is possible e.g. for the PAC to address violations taking into account dependent SLAs. Status information is also necessary to see how SLAs are performing, and extract conclusions about the resources/infrastructure used – which may be very useful later on for reasons of capacity management.

The information within the SLA Registry is also used by the SLA Management Console, for visualization purposes. It is therefore possible for a framework manager to see how many SLAs are active, their content, their status, etc, and filter them based on various criteria to get an overview of the whole SLA ecosystem within the framework he/she manages.

**Related to innovation:** N/A

N/A

**A-Line evaluation**

*Associated tests:* 15  
*Passed tests:* 15  
*Status:* Completed

**B-Line evaluation**

*Documentation:* Sufficient  
*Architecture:* Good  
*API:* Good  
*Implementation:* Good  
*Integration Complexity:* Low  
*Issues:* No issues reported.

**Lessons Learned**

The feature has been evaluated by use cases B3 and B6. The feedback was that overall implementation is sufficient; while documentation of the feature has been reported as to be improved in the Y2 evaluation. This has been improved in Y3, in parallel with extensive modifications to the console so that it can handle arbitrary SLAs as well as possible, and display them intuitively.
Virtual Hardware Infrastructure Reporting

Traceability

Component/s: Low Level Monitoring System (LLMS).

The Low Level Monitoring System receives monitoring configuration information, deploys the appropriate monitors, and gathers, analyses and propagates violations and warnings. It also stores monitoring information in an historical database. The LLMS is described in Deliverable D.A4a [9].

Related to innovation: Standardisation of Infrastructure Monitoring Interfaces, Self-Provisioned Distributed Infrastructure Monitoring.

A-Line evaluation

Associated tests: 16 (15 gathering, analysing, escalating, storing data; 1 extensive test covering reporting data in historical database)

Passed tests: 100%

Status: Completed

The historical database stores monitoring data in a structured way, and an API is provided to allow this data be queried as needed by the use case, by message bus or via a client library.

B-Line evaluation

Documentation: Good

Architecture: Good

API: Good

Implementation: Good

Integration Complexity: Low

Issues: No issues reported.

Lessons Learned

Infrastructure Reporting is important for all use cases were the service user is aware of the infrastructure. The reporting should offer insight into the historical trends of the metrics used in SLA guarantees. As the GUI is use case specific, this feature provides just a common interface to observed metrics.
# External Requirements

In Y1 several external research projects have been asked to provide their requirements for the SLA@SOI framework. See Appendix D: for the description of these sources. These requirements (see table below) have affected the design and specifications for the SLA@SOI framework. In the process of requirements gathering at the beginning of Y3 these requirements have been checked again and some of them which have not been implemented yet have been merged into the new set of requirements. The majority of the requirements have been successfully implemented, the details can be found in the deliverables of the relevant work packages.

<table>
<thead>
<tr>
<th>Ticket</th>
<th>Summary</th>
<th>Requirement Sources (Collaboration)</th>
<th>Priority</th>
<th>Action Line A - Responsible WP</th>
</tr>
</thead>
<tbody>
<tr>
<td>#45</td>
<td>Negotiation API must support renegotiation during active service lifetime</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A2 - Business Management</td>
</tr>
<tr>
<td>#54</td>
<td>SLA Model must support penalty fees depending from the kind and amount of violation.</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#56</td>
<td>Framework must support translation of Business SLA conditions into requirements for the different service layers (software / infrastructure) and vice versa</td>
<td>TEXO/Spillner</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#60</td>
<td>Service Provider must be able to customize how to react to SLA violations (independently from what is fixed in SLAs).</td>
<td>PAT - Politiche Sociali</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#87</td>
<td>SLA@SOI framework must be decomposable into individually deployable components</td>
<td>TEXO/Spillner</td>
<td>blocker</td>
<td>A1 - Architecture &amp; Integration</td>
</tr>
<tr>
<td>#125</td>
<td>SLA template must support both negotiable and non-negotiable terms</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#126</td>
<td>Templates should allow the expression of rights and obligations</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#128</td>
<td>SLA templates must be able to express functional</td>
<td>Q-ImPrESS</td>
<td>blocker</td>
<td>A5 - SLA Management &amp;</td>
</tr>
<tr>
<td>#</td>
<td>Dependencies</td>
<td>Foundations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------</td>
<td>--------------------------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>#130</td>
<td>Multi-party negotiation modalities, at least auctions, must be supported</td>
<td>SOA4ALL/Ripa, TEXO/Spillner</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#131</td>
<td>(Re-)Negotiations must be able to be initiated by both providers and users at any time</td>
<td>SOA4ALL/Ripa, TEXO/Spillner</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#132</td>
<td>Both manual and automated approaches to negotiation must be supported</td>
<td>SOA4ALL/Ripa</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#135</td>
<td>It must be possible to start SLA enactment immediately after negotiation or at a future time</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#136</td>
<td>&quot;Monitoring of Composed Services must identify violations of SLAs of component services responsible for violations of SLAs of the composed service. Automatic SLA Management of composed Services (negotiation, adjustment) must automatically query third parties for SLATs and automatically perform with them an optimized (re)negotiation of SLA of component services in order to satisfy the SLA of the composed service.&quot;</td>
<td>SOA4ALL/Ripa</td>
<td>blocker</td>
<td>A2 - Business Management</td>
</tr>
<tr>
<td>#137</td>
<td>SLA must identify the party responsible for violation</td>
<td>Q-ImPrESS</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#139</td>
<td>SLA status must be viewable by all allowed parties</td>
<td>TEXO/Spillner, TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#142</td>
<td>Providers/Consumers must be able to monitor services</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#143</td>
<td>Monitoring must be available to third parties</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#144</td>
<td>Events must be able to follow both a push and a subscription based messaging pattern</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------------------------------------------------------</td>
<td>-------------</td>
<td>---------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>#145</td>
<td>Events must be able to trigger automatic management actions</td>
<td>SOA4ALL/Ripa</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#146</td>
<td>Dependencies between SLAs must be taken into account by monitoring</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#147</td>
<td>Resources for services must be automatically adapted based on SLA constraints</td>
<td>TEXO/Winkler</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#191</td>
<td>Customer must be able to request for additional data storage (other than the basic compute unit).</td>
<td>37/57/66/125/160/161</td>
<td>blocker</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#141</td>
<td>Service parameters (minimally response time and execution time) must be automatically adjusted to restore SLA compliance</td>
<td>TEXO/Winkler</td>
<td>critical</td>
<td>A3 - Service Management</td>
</tr>
<tr>
<td>#148</td>
<td>QoS of composite services must be automatically derivable from the QoS of their components</td>
<td>TEXO/Winkler</td>
<td>critical</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#149</td>
<td>Service design tool must allow prediction for validation of composite services</td>
<td>TEXO/Winkler</td>
<td>critical</td>
<td>A6 - Predictable Systems Engineering</td>
</tr>
<tr>
<td>#150</td>
<td>Deviations from prediction versus actual must be available in monitoring</td>
<td>TEXO/Spillner</td>
<td>critical</td>
<td>A4 - Infrastructure Management</td>
</tr>
<tr>
<td>#153</td>
<td>Monitoring must be exposed via WSDL/SOAP interface</td>
<td>TEXO/Winkler</td>
<td>critical</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#134</td>
<td>SLA terms must allow pricing models</td>
<td>TEXO/Winkler</td>
<td>major</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>#138</td>
<td>SLA applications components must be accompanied by prediction models</td>
<td>Q-ImPrESS</td>
<td>major</td>
<td>A6 - Predictable Systems Engineering</td>
</tr>
<tr>
<td>#</td>
<td>Requirement</td>
<td>Responsible Party</td>
<td>Role</td>
<td>Category</td>
</tr>
<tr>
<td>-----</td>
<td>------------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>-----------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>151</td>
<td>WSDL/SOAP must be used for access to data for third parties</td>
<td>TEXO/Winkler</td>
<td>major</td>
<td>A1 - Architecture &amp; Integration</td>
</tr>
<tr>
<td>154</td>
<td>SLAs across heterogeneous execution platform must be comprehensively managed</td>
<td>TEXO/Spillner</td>
<td>major</td>
<td>A1 - Architecture &amp; Integration</td>
</tr>
<tr>
<td>25</td>
<td>Service Consumer in the concertation with Service Provider must be able to design orchestration/composition rules</td>
<td>PAT Politiche Sociali</td>
<td>minor</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>129</td>
<td>SLOs must support also pre-conditions dependent on SLA status of used services</td>
<td>Q-ImPrESS</td>
<td>minor</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>140</td>
<td>Contract associated with SLA must be identifiable</td>
<td>TEXO/Winkler</td>
<td>minor</td>
<td>A2 - Business Management</td>
</tr>
<tr>
<td>152</td>
<td>SLA Template registry must be accessible by WSDL/SOAP</td>
<td>TEXO/Winkler</td>
<td>minor</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>38</td>
<td>Service Customer must be able to propose the agreement</td>
<td>TEXO/Winkler</td>
<td>undefined</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>127</td>
<td>SLA registries must maintain metadata as required by law per country</td>
<td>TEXO/Spillner</td>
<td>undefined</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>133</td>
<td>SLA terms must include concepts from logistics (see description)</td>
<td>TEXO/Winkler</td>
<td>undefined</td>
<td>A5 - SLA Management &amp; Foundations</td>
</tr>
<tr>
<td>208</td>
<td>TMF SID concepts should be used as one of the basis of the SLA data model for entities and attributes</td>
<td>SotA/1.1/TMF SID</td>
<td>trivial</td>
<td>A2 - Business Management</td>
</tr>
</tbody>
</table>
10 Conclusions

10.1 Contributions and Achievements

During Y1 we collected SLA@SOI framework requirements. The result of such a process consists of 42 features organized in 10 categories.

In Y2 we performed a detailed evaluation both at level of single features and at level of feature categories. The evaluation has been performed by both the A-Line and B-Line. We found that while there is some significant difference in the perception of the quality of the feature implementation between the A-Line and B-Line and not all features have been the same level of experimentation, the judgments of B-Line are generally aligned with the level of completeness of the implementation reported by the A-Line.

At the beginning of Y3 we have repeated the requirement collection to better target the framework development in the last year of the project. We focused on the features that were missing in the Y2 implementation and were explicitly required by use cases in the requirement process in Y3. All requirements that were gathered in Y3 and were accepted by A-line have been fixed through the last year of the project. Also, all of them have been evaluated as at least sufficient in all aspects by all use cases. The progress from Y2 is significant in regards to implementation status, simplicity of the integration, improved documentation, architecture and API.

Important issues have been identified at the end of Y2 that were needed to be solved. One of these was related to monitoring features, which was successfully solved in Y3. A dynamic configuration of the monitoring (through the whole stack – from the Monitoring Manager to the Software Monitoring and down to the Infrastructure Monitoring) has been fully provided and implemented. Moreover, integration complexity – a serious issue from Y2 has been resolved with a simplified usage of the OSGi technology. Also, a detailed documentation and step-by-step tutorials related to the integration of the components have been provided.

10.2 Lessons learned

The requirements gathering for such a complex project is a critical task. The following lessons learned have been experienced throughout the project.

Lesson 1: A special emphasis should be put on the development of the simple, but working framework prototype. Later on new requirements should be gathered how to extend the prototype.

It can cause many problems if the first implementation aims to be too complex. The implementation as well as requirements gathering should be incremental. The first set of requirements should cover the overall picture of a framework, later on additional more detailed requirements should be collected.

Lesson 2: Integration of such a complex framework should be simple and should be based on proven technologies.

Integration of a large number of components covering different areas is far from being trivial. Special set of requirements should be gathered for the integration process.
Lesson 3: Having a technical manager who would be responsible for checking consolidated requirements with the framework architecture and implementation would be helpful.

Technical manager who would periodically (perhaps every month) check the requirements with the architecture and current implementation could be of a great help. Such verifications would require also execution of the framework tests and/or forcing people to provide tests for the most critical business features.

10.3 Outlook

Important development and integration issues have been resolved in Y3. Further improvements are obviously still possible. Some requirements were not accepted in the requirement process in Y3 and were left to the open source community. Extensive documentation on the SourceForge should make further development relatively simple and fast. In particular SLA@SOI adoption guide [13] and SLA@SOI tutorial [14] should be of great help when developing, extending or integrating SLA@SOI components. SLA@SOI studio – an Eclipse plugin developed to simplify the SLA@SOI adoption for an arbitrary use case should be helpful and a good starting point when adopting SLA@SOI framework. The documentation about SLA@SOI studio can be found on the SourceForge [15].


References


[8] SLA@SOI project: Deliverable D.A3a SLA-aware service management. July 2011


Appendix A: Glossary

The following list shows the most important entries of the SLA@SOI glossary. Note that terms that are specific for the current document and not part of the overall project wide glossary are marked with an asterix *.

Agreement Initiator: An agreement initiator is a party to a service level agreement. The initiator creates and manages an agreement on the availability of a service on behalf of either the service customer or service provider, depending on the domain-specific signalling requirements.

Agreement Offer: An offer is the description of the agreement relationship that is sent from agreement initiator to agreement responder during agreement creation, indicating the relationship which the initiator would like to form.

Agreement Responder: The agreement responder is a party to a service level agreement. The responder implements and exposes an agreement on behalf of either the service provider or service customer, depending on the domain-specific signalling requirements.

Agreement Template: An agreement template is an XML document used by the agreement responder to advertise the types of offers it is willing to accept.

Agreement Term: Agreement terms define the content of a service level agreement.

Business Service: A business service is exposed/invoked via at least some non IT elements.

Business Manager: A specialization of service provider: person that defines the SLATs of products and joins available services in a product.

External Service: External services are exposed across the boundaries of an organization, i.e. across at least two administrative domains.

Framework Administrator: A specialization of service provider: person that configures/adapts the SLA@SOI framework for a specific application.

Guarantee Term: Guarantee terms define the assurance on service quality associated with the service described by the service definition terms. They refer to the service description that is the subject of the agreement and define service level objectives, qualifying conditions and business value expressing the importance of the service level objectives.

Hybrid Service: A hybrid service is a set or bundle of other services where all these services are exposed to the customer but have different service interface types (e.g. an IT service and a business service).

Infrastructure Manager: A specialization of infrastructure provider: person/system that is interested to measure and control infrastructure properties.

Infrastructure Provider: A specific kind of service provider that focuses on the provisioning of infrastructure services.
<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infrastructure Service</td>
<td>An infrastructure service is a specific IT service which exposes resource/hardware-centric capabilities.</td>
</tr>
<tr>
<td>Internal Service</td>
<td>Internal services are exposed within the boundaries of an organization, i.e. within one administrative domain.</td>
</tr>
<tr>
<td>IT Service</td>
<td>An IT service is exposed/invoked by means of information technology. Specific classes of IT services may be software services, infrastructure services or media services.</td>
</tr>
<tr>
<td>Offered Service</td>
<td>An abstract service (more precisely: service type) which is offered by a specific Service Provider to its Service Customers.</td>
</tr>
<tr>
<td>Operation Level Agreement</td>
<td>A specification of the conditions under which an internal service or a component is to be used by its “customer”.</td>
</tr>
<tr>
<td>Service</td>
<td>A means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks. See also service interface type, service concreteness, service exposure</td>
</tr>
<tr>
<td>Service Concreteness</td>
<td>The stage a service reaches over time from a fully abstract type to actually instantiated. See also service type, offered service, service implementation, service instance</td>
</tr>
<tr>
<td>Service Consumer</td>
<td>Person(s) who actually consume/use the provided services. Typically they belong to the service customer.</td>
</tr>
<tr>
<td>Service Customer</td>
<td>Someone (person or group) who orders/buys services and defines and agrees the service level targets.</td>
</tr>
<tr>
<td>Service Description Term</td>
<td>Service Description Terms describe the functionality that will be delivered under the service level agreement. The agreement description may include also other non-functional items referring to the service description terms.</td>
</tr>
<tr>
<td>Service Exposure</td>
<td>Services can be exposed either internally (within the same administrative domain) or externally. See also internal service, external service</td>
</tr>
<tr>
<td>Service Implementation</td>
<td>A service implementation is a possible concrete realization of a given service type.</td>
</tr>
<tr>
<td>Service Instance</td>
<td>A concrete realization of an offered service which is ready for consumption by service users. It relies on the instantiations of all the resources required for a given service implementation.</td>
</tr>
<tr>
<td>Service Interface Type</td>
<td>Describes the nature of an actually exposed service, i.e. about the nature of his invocation interface. See also business service, IT service, hybrid service</td>
</tr>
<tr>
<td>Service Level Consequence</td>
<td>An action that takes place in the event that a service level objective is not met.</td>
</tr>
<tr>
<td>Service Level Agreement</td>
<td>An agreement defines a dynamically-established and dynamically managed relationship between parties. The object of this relationship is the delivery of a service by one of the parties within the context of the agreement. The management of this delivery is achieved by agreeing on the respective roles, rights and obligations of the parties. The agreement may specify not only functional properties for identification or creation of the service, but also non-functional properties of the service such as performance or</td>
</tr>
</tbody>
</table>
availability. Entities can dynamically establish and manage agreements via Web service interfaces.

Service Level Objective  Service Level Objective represents the quality of service aspect of the agreement. Syntactically, it is an assertion over the agreement terms of the agreement as well as such qualities as date and time.

Service Provider  An organization supplying services to one or more internal customers or external customers.

SLA Manager  A specialization of service provider: person/system that is responsible for managing SLATs and SLA relationships.

Software Designer  A specialization of software provider: person that designs/develops the architecture and components of a specific SLA based application.

Software Manager  A specialization of service provider: person that defines software-based services, takes care of their management and supports the SLA manager in creating appropriate SLA templates.

Software Provider  An organization producing software components which might be used by a service provider to assemble actual services.

Software Service  A software service is a specific IT service which is exposed/invoked by means of software entities such as Web services, user interfaces, or software-based business processes.

Software Component  Software components are the entities produced at design-time by a software provider.

Service Type  A service type (or abstract service) specifies the external interface of a service possibly including non-functional aspects. It does not specify any means (components, resources) which are needed for the actual provisioning of that service.

Appendix B: Abbreviations

AOP  Aspect Oriented Programming
BM  Business Manager
B-SLAM  Business SLA Manager
EMF  Eclipse Modelling Framework
ERP  Enterprise Resource Planning
IE  Interaction Event
FCR  Finite capacity regions
ISLAM  Infrastructure SLA Manager
ISM  Infrastructure Service Manager
IoC  Inversion of Control
KPI  Key Performance Indicator
LLMS  Low Level Monitoring System
LQN  Layered Queueing Networks
MA  Manageability Agent
MRE  Monitoring Result Event
MVC  Model View Controller
NFP  Non-functional property
ORC  Open Reference Case
OVF  Open Virtualization Format
QoS  Quality of Service
QPN  Queueing Petri Nets
PAC  Provisioning and Adjustment Component
POC  Planning and Optimization Component
POJO  Plain Old Java Objects
SaaS  Software as a Service
SE  Service Evaluation
SLA  Service Level Agreement
SLAM  SLA Manager
SLAT  Service Level Agreement Template
SM  Service Manager
SME  Small and Medium-sized Enterprise
SOA  Service Oriented Architecture
SW-SLAM  Software SLA Manager
SW-SM  Software Service Manager
TCO  Total Cost of Ownership
TOGAF  The Open Group Architecture Framework
Appendix C: Industrial Use Cases Overview

In order to fulfil the scientific goals set by the SLA@SOI consortium, a number of industrial use cases were included in the project. Industrial use cases were chosen with a very broad field of interested stakeholders, including infrastructure providers, large service providers; social and communication services. In the following sections, we provide a brief introduction to all these use cases.

Use case B3: ERP Hosting (SAP)

The Enterprise Resource Planning Hosting (ERP Hosting) solution is targeted at SMEs not able to afford expensive ERP solutions including software, hardware and constant support. The service provider provides enterprise solutions based on applications as services (SaaS, Software as a Service) using an online portal. The portal also provides customers with tools for specification of business requirements (providing functional and non-functional information) as well as SLA parameters.

Based on this input, the service provider plans the capacity required to satisfy all requirements, especially the Quality of Service guarantees. Once terms are formally and legally agreed by both entities, the service provider provisions the required infrastructure. It also provides monitoring capabilities for all components (infrastructure, middleware, applications, and business services) to facilitate appropriate adjustment.

Use case B4: Enterprise IT (Intel)

The Enterprise IT use case deals with SLA aware dynamic provisioning through Service Oriented Architectures. It will examine SLA negotiation and automation and how IT can dynamically reflect the priorities of the business.

Three scenarios are considered. The first scenario, titled “Provisioning”, responds to the issue of efficient allocation of new services on IT infrastructure, SLA negotiation and provisioning of new services in the environment. The second scenario, “Run Time”, deals with day-to-day, point in time operational efficiency decisions within the environment. These decisions maximise the value from the infrastructure investment. The final scenario “Investment Governance” builds on the first two to demonstrate how they feed back into future business decisions. Taking a holistic cost view, it provides fine grained SLA based data to influence future investment decisions based on capital, security, compute power and energy efficiency.

2 This deliverable does not consider the use case B7, (“Financial Grid”) as insufficient progress has been achieved in its definition.
**Use case B5: Service Aggregator (Telekom Austria)**

The service aggregator use case will demonstrate the aggregation of SLA-aware telecommunication and third party web-based services. The use case will involve the construction of a telecommunication as a service (TaaS) platform over telco and compute infrastructures. The TaaS will then be used to examine scenarios in which multi-party, multi-domain SLAs for aggregated services are offered to customers. The use of information provided by customer relationship management systems will be utilised to personalise the business aspects of the SLA lifecycle to improve the quality and satisfaction of the aggregated service.

**Use case B6: E-Government (ENG)**

The E-Government use case is intended to apply automatic SLA management to social services that mixes activities with human based activities. While most government services cannot be automatic, several management activities such as monitoring, resource planning, selection, negotiation can be at least partially automated thanks to the adoption of formal SLAs.

The e-Government use case aims to integrate two kind of social services: the medical treatment services and the mobility services for elderly people. Three scenarios will be developed. The first one will demonstrate the application of the monitoring framework of SLA@SOI to the integration of human based and automatic services. The second scenario will show how third party providers can be dynamically and automatically chosen by the system on the base of the citizen needs and of pre-signed SLAs. Finally the third scenario will apply automatic resource adjustment and automatic SLA negotiation with third parties to adapt to unforeseen changes in consumer demand.

**Appendix D: Overview of External Sources**

In the second round of requirements gathering in Y1 a shorter questionnaire was prepared and distributed through established collaboration channels to external entities. We’ve received and included the following projects:

- **SOA4ALL**, a Large-Scale Integrated Project funded by the European Seventh Framework Programme. It aims at realizing a world where billions of parties are exposing and consuming services via advanced Web technology.

- **Q-ImPRESS** aims to bring service orientation to critical application domains, such as industrial production control, telecommunication and critical enterprise applications, where guaranteed end-to-end quality of service is particularly important.

- **TEXO** contributes to service economy by creating infrastructure components for Business Webs in the Internet of Services.

Moreover other external requirements were identified by analysis of the literature [1]. In particular we selected requirements coming from the following sources:
- SCENE, a service composition execution environment supporting binding, re-binding and self-reconfiguration operations. SCENE is part of SeCSE.
- SeCSE, a European IP, which ended in November 2008 and was aimed at creating new methods, tools and techniques to support the cost-effective development and use of dependable services and service-centric applications.
- TMF SID, a set of comprehensive standardized information definitions, developed by the TeleManagement Forum (TMF), acting as the common language for building easy to integrate OSS (Operational Support System) and BSS (Business Support System) solutions.
- TMF SDF, Service Delivery Framework. A work aimed to facilitate service syndication and end-to-end management of services.
- TM Forum SLA Management Handbook, a handbook to assist two parties in developing a Service Level Agreement, by providing a practical view of the fundamental issues.
- WBEM, Web-based Enterprise Management (WBEM) Standards. A set of standards aimed to unify the management of distributed computing environments.
- JMX, Java Management Extensions. A standard developed by Sun Microsystems, part of the Java Platform Standard Edition, used to manage resources such as applications, devices and services with the Java Programming Language.
- DIANE, an approach and a language developed by University of Jena and University of Karlsruhe for Automated Service Discovery, Matchmaking and Composition.
- CentraSite, a standards-based SOA registry and repository jointly developed by Fujitsu and Software AG that aims at greater visibility and control of integrated SOA based applications, better support on decision-making, and increased productivity.

### Appendix E: Framework Features

In this section the complete description on framework features is given.

The following table gives a summary of all the identified features organized according to the decided categories. The table shows the scientific work package associated to the feature and the use cases that use them according to the last definition of use case scenarios:

<table>
<thead>
<tr>
<th>WP</th>
<th>Feature</th>
<th>Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td></td>
<td>Framework Management</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Framework Configuration &amp; Setup</td>
<td>X</td>
</tr>
<tr>
<td>WP</td>
<td>Feature</td>
<td>Use Cases</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td>A1</td>
<td>Framework Model Configuration</td>
<td>X</td>
</tr>
<tr>
<td>A1</td>
<td>Framework Operation</td>
<td>X</td>
</tr>
<tr>
<td>A1</td>
<td>Framework Access</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td><strong>Design &amp; Development</strong></td>
<td></td>
</tr>
<tr>
<td>A3</td>
<td>Reference Software Manageability Components and Configuration</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>Service properties dependencies coding</td>
<td>X</td>
</tr>
<tr>
<td>A6</td>
<td>Designing of QoS predictable systems</td>
<td>X</td>
</tr>
<tr>
<td>A6</td>
<td>Manageability Design</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Pre-offering</strong></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Customer Registration</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>3rd Parties Configuration</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Service Offering</strong></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Discovery of Serv. and SLAT</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>Product Management</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Product Discovery</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>Product Definition</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>SLAT Definition</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td><strong>Service Negotiation</strong></td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Business Negotiation</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>SLAT Customization</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Business SLA planning for negotiation</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Automatic orchestration of customisable (re)negotiation protocol</td>
<td>X</td>
</tr>
<tr>
<td>A5</td>
<td>Out-of-band SLA Registration</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Service Provisioning</strong></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Coordination of Provisioning</td>
<td>X</td>
</tr>
<tr>
<td>A5</td>
<td>On the fly deploy of monitoring</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Dynamic binding setting</td>
<td></td>
</tr>
<tr>
<td>A4</td>
<td>Virtual hardware infrastructure provisioning</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Software provisioning</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Software landscape</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td><strong>SLA enforcement</strong></td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Automated SLA enforcement coordination</td>
<td>X</td>
</tr>
</tbody>
</table>
### Table 6: Required Framework Features (updated to Y3)

<table>
<thead>
<tr>
<th>WP</th>
<th>Feature</th>
<th>Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>B3</td>
</tr>
<tr>
<td>A4</td>
<td>Virtual hw. infrastructure adjustment</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Software Adjustment</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Automatic Binding</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>Business adjustment</td>
<td></td>
</tr>
</tbody>
</table>

#### Runtime Prediction
- Runtime infrastructure metrics prediction | X  | X  |
- Runtime Software SLAs violations prediction | X  | X  |

#### Service Monitoring
- Monitoring coordination | X  | X  | X  |
- Virtual hardware infrastructure monitoring rules extraction from SLA | X  | X  | X  |
- Virtual hardware infrastructure observation and violation detection | X  | X  | X  |
- Software violation detection | X  | X  | X  |
- Software monitoring rules extraction from SLA | X  | X  | X  | X  |

#### Service Reporting
- SLA Management | X  | X  | X  |
- Push/Pull Business SLA Reporting | X  | X  | X  |
- SLA Status & Resource Reporting | X  | X  |
- Virtual hardware Infrastructure Reporting | X  | X  | X  |

---

### Framework Management

#### Framework Configuration & Setup

**Responsible WP:** A1  
**External Actors:** Framework Architect  
**SLA Layers:** all  
**Overall priority:** Critical

The feature “Framework Configuration & Setup” comprises a set of four validated requirements describing the use case needs on how the overall SLA framework can be configured and set up. The main need in this area is about the flexibility of using the framework in different ways, i.e. the framework configurability in terms
of (a) which components are used, (b) the usage of components in multiple instances and (c) the replacement of components with domain-specific instances. Furthermore, use cases require multi-domain setups, including 3rd parties and the operation of a service registry by the service provider.

**Framework Model Configuration**

**Responsible WP:** A1  
**External Actors:** Framework Architect  
**SLA Layers:** all  
**Overall priority:** Blocker

The feature “Framework Model Configuration” comprises two validated requirements describing the use case needs on how to parameterise the basic data models of the framework. The main need in this area is about flexibility in adapting and extending the basic SLA model with additional basic metrics and domain specific parameters (on guarantee terms, guaranteed actions, etc.).

**Framework Operation**

**Responsible WP:** A1  
**External Actors:** Framework Architect  
**SLA Layers:** all  
**Overall priority:** Blocker

The feature “Framework Operation” comprises two validated requirements describing the use case needs on how the overall SLA framework shall operate. Basically, the framework shall work under Linux, Unix and Windows operating systems and should ideally also come with default implementation support for managing common middleware such as Apache or Tomcat.

**Framework Access**

**Responsible WP:** A1  
**External Actors:** Framework Architect  
**SLA Layers:** all  
**Overall priority:** Blocker

The feature “Framework Access” comprises a set of five validated requirements describing the use case needs on how the overall SLA framework can be accessed by different kinds of internal and external clients. Main needs in this area include basic library access via Java, remote interfaces for building blocks via WSDL/SOAP and browser-based graphical user interfaces.
Design & Development

Reference Software Manageability Components and Configuration

Responsible WP: A3
External Actors: Service Provider
SLA Layers: all
Overall priority: Blocker

This feature must offer modelling and development solutions to simplify the monitoring of common SOA software resources. This must include the implementation of specific instrumentations for the monitoring of service workflows and web services. This is needed in particular to simplify the adoption of the platform for applications that need to monitor composed services such as use case B6.

Service properties dependencies coding

Responsible WP: A1
External Actors: Software Designer
SLA Layers: Software Layer
Overall priority: Blocker

This feature, used by the software designer at service development time, allows the encoding of how a service depends on other services.

It is needed to support automatic negotiation and planning, especially with lower level SLA managers and third parties, and for determining responsible parties for violations of the composed SLA.

Designing of QoS predictable systems

Responsible WP: A6
External Actors: Software Designer
SLA Layers: Software Layer
Overall priority: Blocker

This feature is aimed at the creation of QoS-enabled architectural models, including performance annotations that can be used for early design-time prediction of component-based software. The feature includes a design-time model repository for facilitating the reuse of QoS aware models, Eclipse base tooling to support the design and development process and methodologies for calibration and validation of component’s performance.
Manageability Design

**Responsible WP:** A6  
**External Actors:** Software Designer  
**SLA Layers:** Software Layer  
**Overall priority:** Major

This feature will enable software developers to specify basic and event-based monitoring capabilities for software components at design type, resulting in a monitoring-enabled architectural design. The models will also present management (long term) and control (short term) interfaces in order to support the manageability life-cycle of the software layer.

Pre-offering

Customer registration

**Responsible WP:** A2  
**External Actors:** Service Customer  
**SLA Layers:** Business  
**Overall priority:** Critical

This feature corresponds to the requirement that the customers must be able to register themselves in the platform, through some GUI. The registration is a prerequisite for participating in negotiation. The customers define their contact details, billing information, information about their preferences and reporting content.

3rd Parties Configuration

**Responsible WP:** A2  
**External Actors:** Framework Manager  
**SLA Layers:** Business, Software  
**Overall priority:** Blocker

This feature allows the Framework Manager to register third party service providers in the platform. The specification of third party providers is needed to enable, at software level, the dynamic binding of abstract composed services and to enable the business manager to statically compose new services to offer as products. Requirements corresponding to this feature require that the framework allows registering for each third party provider at least the endpoints of the provided services, the associated SLATs and the SLAs already established without the support of the framework.
Service Offering

Discovery of Services and SLAT

**Responsible WP:** A5  
**External Actors:** no  
**SLA Layers:** all  
**Overall priority:** Major

This feature corresponds to an internal functionality of the framework that allows all components to access, through a specific API, the list of available services and corresponding SLATs, from both internal and third party providers. This feature groups technical requirements such as the possibility to access this information using a WSDL/SOAP interface and the possibility to restrict queries to specific service providers.

Product Management

**Responsible WP:** A2  
**External Actors:** Business Manager  
**SLA Layer:** Business  
**Overall priority:** Blocker

This feature is intended to manage all the information on a product during its entire lifecycle. In particular this feature included the features “Product Discovery” and “Product Definition” and adds the possibility of defining rules to conduct negotiation and to provision the service. It also includes the possibility for the provider to activate/deactivate a service for a certain customer and to manually configure the related SLA. This feature is required by the use cases B4, B5 and B6. For instance, use case B6 needs to register the rules to use for the automatic negotiation with 3rd party providers and to manually set SLAs negotiated without the platform.

Product Discovery

**Responsible WP:** A2  
**External Actors:** Business Manager, Service Customer  
**SLA Layer:** Business  
**Overall priority:** Major

This feature allows the end customer and also the provider to query for available products based on product description, categorization and other specific information from the SLAT. The requirements of this feature require to permit querying just to registered users, to allow searching for providers based on SLATs expressed by the customer, and to allow the browsing of existing services
through a GUI. This feature depends on the feature “Discovery of Services and SLAT”.

**Product Definition**

**Responsible WP:** A2  
**External Actors:** Business Manager  
**SLA Layers:** Business  
**Overall priority:** Blocker

This feature allows the Business Manager to define the offered products. Use Cases asked to be able to add/change service descriptions and related SLATs. A Product may be defined by composing other available services and related SLATs. The composed service can be just an aggregate of other services or correspond to a composition done through a specific language such as BPEL. Component services of a composed service may be delegated to third parties. The provider and SLA of component services can be statically specified on the base of rule specified at product definition time.

Product definition leverages on the feature “SLAT Definition”.

**SLAT Definition**

**Responsible WP:** A5  
**External Actors:** No  
**SLA Layers:** all  
**Overall priority:** Blocker

This feature corresponds to the internal functionality of the framework used to register SLA Templates associated to services. Numerous requirements from use cases constrain the content of the SLAT that this functionality must support. All use cases ask to be able to customize an SLA both with respect to the offered functionalities and with respect to the QoS. In particular all ask for the support of “classic” QoS parameters such as Availability, Cost, Reliability and for the possibility to specify: specific responsible parties for each SLA term, context dependent constraints, time dependent ranges, statistical constraints and penalties. Moreover use cases B3, B4, B5 ask for the possibility to constraint in SLAs more specific terms such as: number of computer units, data storage, compute power, period of peak hours, geographical region of data receiver, maximum budget per month. Use cases B3 and B5 ask for the support of the TMF Model terms. Other QoS parameters are instead very specific of each use case. Use cases B4, B6 ask for the possibility to express SLOs that depend on other SLOs and B5 ask for the possibility to express SLOs preconditioned by the satisfaction of third party SLAs. B6 asks for the possibility to express constraints depending on I/O data exchanged with the service. Use Cases B3, B4, B6 ask for the possibility to specify the receiver of the monitoring reports. Use Case B4 asks for the possibility to assign a priority to the terms in a SLA.
Service Negotiation

Business SLA Negotiation

Responsible WP: A2
External Actors: Service Customer, Business Manager
SLA Layers: Business
Overall priority: Blocker

This feature of the framework offers a Web interface and an API to the customer to select and negotiate offered products. Such interfaces allow the customer to offer SLAs that conform to the selected product SLAT. This feature includes feature “SLAT Customization” and is coordinated in an automatic way by the feature “Business SLA planning for negotiation”. The behaviour of the business negotiation is constrained by some use case specific requirements. In particular use cases B3 and B5 ask the possibility for the provider to offer to the customer a specific SLAT, selected on the base of the registered customer profile. Use Cases B4 and B6 add the precondition that a Customer must already have a pre-contract with the provider in order to participate to the negotiation (in particular in the case of automatic negotiation on the side of the provider).

Business negotiation, as with negotiation at other layers, is based on the “Automatic orchestration of customisable (re)negotiation protocol”. Moreover it depends on “Product Managements” for the configuration of negotiation behaviour.

SLAT Customization

Responsible WP: A2
External Actors: Business Manager
SLA Layers: Business
Overall priority: Blocker

This feature is to automatically adapt the template sent to an end customer based on his profile or segmentation. This feature is asked by use cases B3 and B5.

Business SLA planning for negotiation

Responsible WP: A2
External Actors: Service Customer, Business Manager
SLA Layers: Business
Overall priority: Blocker

This is automatic coordination of the Business SLA Negotiation. The Business manager defines the parameters to achieve an automatic negotiation. Only in the
case that the parameters cannot be filled, the manager must be allowed to use a GUI to try to achieve a manual negotiation. Such an automatism is required by use cases .B4, B5, B6.

**Automatic orchestration of customisable (re)negotiation protocol**

*Responsible WP:* A5  
*External Actors:* No  
*SLA Layers:* all  
*Overall priority:* Blocker

This feature corresponds to a customisable functionality of the framework to automatically orchestrate the negotiation between provider and customer. Each layer can customise this functionality adding specific negotiation protocols. This internal functionality of the framework is constrained by several requirements. All the use cases ask for the possibility of the provider to make contra offers, for the customer to reject the offer of the provider, and for both parties to be allowed to start re-negotiation at any time. Moreover use cases B3, B4 and B5 ask that the provider always reply to a customer offer. Support for both manual and automatic negotiation is asked by B3, B5, B6. Use Cases B5 and B6 ask for customer negotiation through API and for the support of multiparty negotiation (needed for composed services with third party providers). B3 and B6 ask the support for the classic quotation process.

**Out-of-band SLA registration**

*Responsible WP:* A5  
*External Actors:* No  
*SLA Layers:* All  
*Overall priority:* Blocker

This feature asked by use case B5 and B6, allows to register an SLA negotiated without the platform. Such a contract must be treated as the ones negotiated with the platform.

**Service Provisioning**

**Coordination of Provisioning**

*Responsible WP:* A5  
*External Actors:* Service Provider  
*SLA Layers:* all  
*Overall priority:* Blocker
This feature is based on the requirement that several layers, providers and components must be involved during provisioning of SLA. It will orchestrate the provisioning of SLAs in a recursive manner, depending on the SLA hierarchy of the involved services.

**On the fly Deploy of Monitoring**

**Responsible WP:** A5  
**External Actors:** Service Provider  
**SLA Layers:** Software, Infrastructure  
**Overall priority:** Critical

This feature corresponds to the requirement that runtime configuration of monitoring should be supported. It also requires coordination of the provisioning of all monitoring modules. Based on the Guarantee Terms in the SLAs, the involved monitoring modules must be provisioned on the fly.

**Dynamic Binding Setting**

**Responsible WP:** A3  
**External Actors:** Service Provider  
**SLA Layers:** Software  
**Overall priority:** Critical

This feature must allow the addition/replacement/removal of binding rules for invocation activities of already deployed WS BPEL processes. This is used to support the monitoring and deployment of processes independently from service providers chosen during the negotiation or execution phase of services.

**Virtual Hardware Infrastructure Provisioning**

**Responsible WP:** A4  
**External Actors:** Infrastructure Provider  
**SLA Layers:** Infrastructure  
**Overall priority:** Blocker

This feature aggregates requirements about provisioning of physical and virtual low level resources, in particular compute and storage resources. The creation/deletion/migration of virtual machines must be supported, as well as the configuration of local networks. The lower level must support different virtualisation (e.g. Xen) and Cloud computing systems (OpenNebula, etc.). The provisioning must be performed on request from **Provisioning coordination**.
Software Provisioning

**Responsible WP:** A3  
**External Actors:** Service Provider  
**SLA Layers:** Software  
**Overall priority:** Blocker

This feature corresponds to requirements for provisioning, deployment and configuration of software artefacts, which are required for service functionality. As the services are mostly customised, these mechanisms must be generic and extendable to support all such kinds of services.

Software Landscape

**Responsible WP:** A3  
**External Actors:** Service Provider  
**SLA Layers:** Software  
**Overall priority:** Blocker

This feature corresponds to the requirement that each layer must support some kind of registry for all service and resource related information. These are required during the operation of the framework (negotiation, provisioning, adjustment, etc.).

SLA Enforcement

Automated SLA Enforcement Coordination

**Responsible WP:** A5  
**External Actors:** Service Provider  
**SLA Layers:** all  
**Overall priority:** Blocker

This feature corresponds to a requirement that the resources and sub-services must be automatically adjusted when required to ensure the SLA. Besides changing the parameters, also migration to other resources and sub-services should be supported. Those decisions are based on policies expressed in configuration.

Virtual Hardware Infrastructure Adjustment

**Responsible WP:** A4  
**External Actors:** Infrastructure Provider
**SLA Layers:** Infrastructure  
**Overall priority:** Blocker

This feature implements the mechanism used by *Automated SLA enforcement coordination* to perform the adjustment of the Infrastructure. It is about runtime re-provisioning of compute resources to minimise SLA violations and maximise compliance with internal provider policies.

**Software Adjustment**  
**Responsible WP:** A3  
**External Actors:** Service Provider  
**SLA Layers:** Software  
**Overall priority:** Blocker

This feature implements the mechanism used by *Automated SLA enforcement coordination* to perform the adjustment of the Software. It is about runtime reconfiguration of software resources to correct situations where SLA violations have occurred and restore the normal functioning of the system.

**Automatic Binding**  
**Responsible WP:** A3  
**External Actors:** Service Provider  
**SLA Layers:** Software  
**Overall priority:** Major

This feature corresponds to the requirement to support the deployment and execution of a WS-BPEL process having some service invocation not explicitly bound, but determined at runtime using associated rules. This feature depends on the feature “Dynamic Binding Setting”.

**Business Adjustment**  
**Responsible WP:** A2  
**External Actors:** Business Manager  
**SLA Layers:** Business  
**Overall priority:** Blocker

This feature implements the requirements related to business aspects of SLA violations. The penalties/bonuses must be calculated and the actions must be taken (renegotiation, change the 3rd party, etc.) to optimise the business value of the SLA status and enhance the customer perception of the service.
Runtime Prediction

Runtime Infrastructure Metrics Prediction

Responsible WP: A6
External Actors: no
SLA Layers: Infrastructure
Overall priority: Blocker

This feature allows to get warnings at run-time when infrastructure quality goals are likely to be violated in the near future. This is needed to automatically adapt resources for services based on SLA constraints.

Runtime Software SLA Violations Prediction

Responsible WP: A6
External Actors: no
SLA Layers: Software
Overall priority: Major

This feature allows to get warnings at run-time when software service quality goals are likely to be violated in the near future. As the corresponding feature at infrastructure level, this is needed to automatically adapt software resources for services based on SLA constraints.

Service Monitoring

Monitoring Coordination

Responsible WP: A3
External Actors: Service Provider
SLA Layers: all
Overall priority: Blocker

This feature corresponds to general requirements related to provisioning of monitoring system. Several monitoring modules (even 3rd party) are involved in this process, so coordination is required. The aim of the monitoring system is to observe KPIs, detect undesired states, trigger automatic managements and adjustment actions and offer/report these data to all interested parties.
Virtual Hardware Infrastructure Observation and Violation Detection

**Responsible WP:** A4  
**External Actors:** Infrastructure Provider  
**SLA Layers:** Infrastructure  
**Overall priority:** Blocker

This feature corresponds to a requirement that arbitrary infrastructure related metrics (CPU, memory, response time, failures, etc.) must be monitored. Comparisons with predicted values must also be supported. The granularity of measurements must be configurable in order to detect the violations of predefined conditions at hardware infrastructure level. Those conditions are derived from SLAs and other sources and are used to support adjustment and reporting. Derived/aggregate monitoring metrics must also be supported in conditions.

Software Violation Detection

**Responsible WP:** A3  
**External Actors:** Service Provider  
**SLA Layers:** Software  
**Overall priority:** Blocker

This feature corresponds to requirements about monitoring provisioned software and detecting violations. Real-time alerting of violations of configured conditions must be supported. The supported monitoring mechanisms must include SOAP message interception and BPEL process instrumentation. The granularity of measurements must be configurable. Aggregate monitoring metrics must also be supported.

Monitoring Rules Extraction from SLA

**Responsible WP:** A3, A4  
**External Actors:** Service Provider  
**SLA Layers:** all  
**Overall priority:** Blocker

The configuration of the monitoring system is implied by the SLA. This feature requires the capability to extract rules from the agreed SLA that determines the provisioning of the monitoring system. This feature is replicated both in A3 and A4 to extract both software and infrastructure monitoring rules.
Service Reporting

SLA Management

**Responsible WP:** A2  
**External Actors:** Service Customer, Business Manager  
**SLA Layers:** Business  
**Overall priority:** Major

This feature is asked by all the use cases, and allows both the customer and business manager to see and manage any contract related to the signed SLAs.

Push/Pull Business SLA Reporting

**Responsible WP:** A2  
**External Actors:** Business Manager  
**SLA Layers:** Business  
**Overall priority:** Blocker

This feature generates reports in both push and pull ways. Requirements ask to communicate to the end-customer monitoring data and SLA status including violations and billing.

SLA Status & Resource Reporting

**Responsible WP:** A5  
**External Actors:** Service Provider  
**SLA Layers:** all  
**Overall priority:** Blocker

This feature is asked by all the use cases and allows any Service Provider to see the history of the status of all provided SLAs. This information should be available to all parties of the SLA. This feature also allows the manager of each layer to see the properties of each resource and their history, in particular the allocation of resources to their services. The history information is aggregated on more coarse grained timescales as we go further into the past.

Virtual Hardware Infrastructure Reporting

**Responsible WP:** A4  
**External Actors:** Service Provider  
**SLA Layers:** all  
**Overall priority:** Blocker
This feature requires the possibility to inform interested components of the values of measured compute infrastructure resources properties and their history. This is needed both for composing monitoring reports and more in general to provide information needed for prediction and adjustment.
Appendix F: Status Definitions Used in TRAC

The following status values were defined on the Trac system:

- new
- accepted
- fixed
- invalid
- wontfix
- duplicate
- worksforme

Trac ticket is just an auxiliary requirement record. The main record is always the requirement row in the wiki table where each requirement has assigned a status in Requirement Fulfilment Process. The following requirement states are defined:

- **new** – not yet accepted or rejected
- **accepted** – the responsible A-Line person has confirmed that it will be implemented
- **fixed** – implemented, but not yet tested by the reporter
- **verified** – tested and confirmed by the reporter
- **supp** – already supported
- **later** – will be implemented later (possibly after the end of the project)
- **wontfix** – out of the scope of the project
- **invalid** – the requirement does not make sense
- **dup** – already covered by some other requirement

Both status scales are similar and the mapping between them is quite straightforward (see below). We have defined Requirement Fulfilment Process states that fulfil our need, but were not allowed to adapt the SourceForge Trac states, it turned out that joint use of both status scales is an acceptable solution.

<table>
<thead>
<tr>
<th>Req. Fulfilment Process status</th>
<th>SourceForge Trac status</th>
</tr>
</thead>
<tbody>
<tr>
<td>new</td>
<td>new</td>
</tr>
<tr>
<td>accepted</td>
<td>accepted</td>
</tr>
<tr>
<td>fixed</td>
<td>fixed</td>
</tr>
<tr>
<td>verified</td>
<td>fixed, but added comment</td>
</tr>
<tr>
<td>supp</td>
<td>worksforme</td>
</tr>
<tr>
<td>later</td>
<td>wontfix + postponement comment about wontfix</td>
</tr>
<tr>
<td>---------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>wontfix</td>
<td>wontfix</td>
</tr>
<tr>
<td>invalid</td>
<td>invalid</td>
</tr>
<tr>
<td>dup</td>
<td>duplicate</td>
</tr>
</tbody>
</table>
Appendix G: Technical evaluation of Use Cases

The following tables report the technical evaluation of each Framework feature provided by each use case. Each use case was requested to perform a complete evaluation of at least the features used in its scenario implemented in Y3. An un-complete evaluation was allowed for the other features.

Each feature was evaluated from 5 points of view (Documentation, Architecture, API, Implementation, Integration Complexity). In a separate table for each use case the issues (problems to be solved or possible improvements to be applied by the open-source community) identified by each use case are reported.

Technical evaluation of Use Case B3

<table>
<thead>
<tr>
<th>Feature</th>
<th>Documentation</th>
<th>Architecture</th>
<th>API</th>
<th>Implementation</th>
<th>Integration Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework Configuration &amp; Setup</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Framework Model Configuration</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Framework Operation</td>
<td>Good</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Framework Access</td>
<td>Sufficient</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Service properties dependencies coding</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Designing of QoS predictable systems</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Discovery of Serv. and SLAT</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Product Discovery</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Feature</td>
<td>Documentation</td>
<td>Architecture</td>
<td>API</td>
<td>Implementation</td>
<td>Integration Complexity</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>----------</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Product Definition</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>SLAT Definition</td>
<td>Good</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Business Negotiation</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Low</td>
</tr>
<tr>
<td>Automatic orchestration of customisable (re)negotiation protocol</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Coordination of Provisioning</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>On the fly deploy of monitoring</td>
<td>Sufficient</td>
<td>Excellent</td>
<td>Good</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hw. infrastructure provisioning</td>
<td>Good</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software provisioning</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software landscape</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Automated SLA enforcement coordination</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Virtual hw. infrastructure Adjustment</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software Adjustment</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Runtime Software SLAs violations prediction</td>
<td>Good</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Monitoring coordination</td>
<td>Good</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Monitoring rules extraction from SLA</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>SLA Management</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Feature</td>
<td>Documentation</td>
<td>Architecture</td>
<td>API</td>
<td>Implementation</td>
<td>Integration Complexity</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>---------</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Push/Pull Business SLA Reporting</td>
<td>Good</td>
<td>Good</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>SLA Status Reporting</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Virtual hw Infrastructure Reporting</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 7: Technical evaluation of Use Case B3

---

**Technical evaluation of Use Case B4**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Documentation</th>
<th>Architecture</th>
<th>API</th>
<th>Implementation</th>
<th>Integration Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework Configuration &amp; Setup</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Excellent</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
<tr>
<td>Framework Model Configuration</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Feature</td>
<td>Documentation</td>
<td>Architecture</td>
<td>API</td>
<td>Implementation</td>
<td>Integration Complexity</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>-----------</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Framework Operation</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Framework Access</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Customer Registration</td>
<td>Excellent</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
<tr>
<td>Discovery of Serv. and SLAT</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Low</td>
</tr>
<tr>
<td>Product Management</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Product Discovery</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Product Definition</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>SLAT Definition</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Coordination of Provisioning</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>On the fly deploy of monitoring</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hw. infrastructure provisioning</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Automated SLA enforcement coordination</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hw. infrastructure Adjustment</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Runtime infrastructure metrics prediction</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hardware infrastructure monitoring</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
</tbody>
</table>
### Table 8: Technical evaluation of Use Case B4

<table>
<thead>
<tr>
<th>Feature</th>
<th>Documentation</th>
<th>Architecture</th>
<th>API</th>
<th>Implementation</th>
<th>Integration Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual hardware infrastructure observation and violation detection</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Monitoring rules extraction from SLA</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hw Infrastructure Reporting</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
</tbody>
</table>

### Issues reported from Use Case B4

<table>
<thead>
<tr>
<th>Feature</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Designing of QoS predictable systems</td>
<td>Usage profiles are important with regards to the provisioning of services</td>
</tr>
<tr>
<td>Product Discovery</td>
<td>Cannot search SLAs by their internal agreement terms, only by ID’s and/or other generated metadata, which is not as useful</td>
</tr>
<tr>
<td>SLAT Definition</td>
<td>Yes SLATs can be defined, but requires expert level knowledge of the SLA model. Perhaps an editor would help with this, as per B3's suggestion</td>
</tr>
</tbody>
</table>

### Table 9: Issues reported on Use Case B4
## Technical evaluation of Use Case B5

<table>
<thead>
<tr>
<th>Feature</th>
<th>Documentation</th>
<th>Architecture</th>
<th>API</th>
<th>Implementation</th>
<th>Integration Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework Configuration &amp; Setup</td>
<td>Good</td>
<td>Sufficient</td>
<td></td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Framework Model Configuration</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>High</td>
</tr>
<tr>
<td>Framework Operation</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>High</td>
</tr>
<tr>
<td>Framework Access</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
<tr>
<td>Reference Software Manageability Components and Configuration</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Service properties dependencies coding</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Manageability Design</td>
<td>Nearly Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Customer Registration</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Negligible</td>
</tr>
<tr>
<td>Feature</td>
<td>Documentation</td>
<td>Architecture</td>
<td>API</td>
<td>Implementation</td>
<td>Integration Complexity</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>-----------</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>3rd Parties Configuration</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Negligible</td>
</tr>
<tr>
<td>Discovery of Serv. and SLAT</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Product Management</td>
<td>Good</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Product Discovery</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Product Definition</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>SLAT Definition</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Business Negotiation</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>SLAT Customization</td>
<td>Good</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Business SLA planning for negotiation</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Out of band SLA Registration</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Low</td>
</tr>
<tr>
<td>Coordination of Provisioning</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>On the fly deploy of monitoring</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Dynamic binding setting</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hw. infrastructure provisioning</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software provisioning</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Feature</td>
<td>Documentation</td>
<td>Architecture</td>
<td>API</td>
<td>Implementation</td>
<td>Integration Complexity</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>---------</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Software landscape</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
<tr>
<td>Automated SLA enforcement coordination</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Virtual hw. infrastructure Adjustment</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software Adjustment</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Automatic Binding</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Business adjustment</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Runtime infrastructure metrics prediction</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Runtime Software SLAs violations prediction</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Monitoring coordination</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hardware infrastructure monitoring rules extraction</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Virtual hardware infrastructure observation and violation detection</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Software violation detection</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Feature</td>
<td>Documentation</td>
<td>Architecture</td>
<td>API</td>
<td>Implementation</td>
<td>Integration Complexity</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>-------</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Monitoring rules extraction from SLA</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>SLA Management</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Push/Pull Business SLA Reporting</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Excellent</td>
<td>Low</td>
</tr>
<tr>
<td>Virtual hw Infrastructure Reporting</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Table 10: Technical evaluation of Use Case B5

Technical evaluation of Use Case B6
<table>
<thead>
<tr>
<th>Feature</th>
<th>Documentation</th>
<th>Architecture</th>
<th>API</th>
<th>Implementation</th>
<th>Integration Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework Configuration &amp; Setup</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Framework Model Configuration</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Framework Operation</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Low</td>
</tr>
<tr>
<td>Framework Access</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Reference Software Manageability Components and Configuration</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Negligible</td>
</tr>
<tr>
<td>Service properties dependencies coding</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
<tr>
<td>Designing of QoS predictable systems</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Customer Registration</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Low</td>
</tr>
<tr>
<td>3rd Parties Configuration</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Discovery of Serv. and SLAT</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Product Definition</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>SLAT Definition</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Business Negotiation</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Business SLA planning for negotiation</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td></td>
</tr>
<tr>
<td>Automatic orchestration of</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Feature</td>
<td>Documentation</td>
<td>Architecture</td>
<td>API</td>
<td>Implementation</td>
<td>Integration Complexity</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>---------------</td>
<td>--------------</td>
<td>-----</td>
<td>----------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>customisable (re)negotiation protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out of band SLA Registration</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Coordination of Provisioning</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>On the fly deploy of monitoring</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Dynamic binding setting</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software provisioning</td>
<td>Good</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Automated SLA enforcement coordination</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software Adjustment</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Automatic Binding</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Business adjustment</td>
<td>Sufficient</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Low</td>
</tr>
<tr>
<td>Runtime Software SLAs violations prediction</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Monitoring coordination</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>Software violation detection</td>
<td>Good</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Medium</td>
</tr>
<tr>
<td>Monitoring rules extraction from SLA</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Sufficient</td>
<td>Medium</td>
</tr>
<tr>
<td>SLA Management</td>
<td>Good</td>
<td>Sufficient</td>
<td>Good</td>
<td>Sufficient</td>
<td>Low</td>
</tr>
</tbody>
</table>
### Feature Documentation

<table>
<thead>
<tr>
<th>Feature</th>
<th>Documentation</th>
<th>Architecture</th>
<th>API</th>
<th>Implementation</th>
<th>Integration Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push/Pull Business SLA Reporting</td>
<td>Good</td>
<td>Good</td>
<td>Excellent</td>
<td>Good</td>
<td>Low</td>
</tr>
<tr>
<td>SLA Status Reporting</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Good</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Table 11: Technical evaluation of Use Case B6**

### Issues reported from Use Case B6

<table>
<thead>
<tr>
<th>Feature</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Push/Pull Business SLA Reporting</td>
<td>Specification of eMail based reporting not fully compliant to the SLA Model.</td>
</tr>
</tbody>
</table>

**Table 12: Issues reported by Use Case B6**