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: 30th May 2009
Actual submission to EC date: 29th June 2009
Resubmission to EC date: 18th November 2009

Start date of project: 1st June 2008
Duration: 36 months

Lead contractor for this deliverable: Engineering Ingegneria Informatica Spa
Revision: V.0.59 (5th Dic 2009)

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

<table>
<thead>
<tr>
<th>Dissemination level</th>
</tr>
</thead>
<tbody>
<tr>
<td>PU</td>
</tr>
<tr>
<td>Public</td>
</tr>
<tr>
<td>Yes</td>
</tr>
</tbody>
</table>
### Document Status

| Deliverable Lead | ENG  
| Reviewer 1 | George Spanoudakis  
| Reviewer 2 | Sinan Toprak  
| PMT Reviewer | Ramin Yahyapour  
| Complete version submitted to reviewers |  
| Comments of reviewer 1 received |  
| Comments of reviewer 2 received |  
| Revised deliverable submitted to PMT |  
| PMT Approval |  
| Resubmission Reviewer 1 | George Spanoudakis  
| Resubmission Reviewer 2 | Joe Butler  
| Resubmission PMT Reviewer | Ramin Yahyapour  
| Comments of reviewer received |  

### Contributors

<table>
<thead>
<tr>
<th>Partner</th>
<th>Contributors</th>
</tr>
</thead>
</table>
| ENG | Francesco Torelli, Paolo Zampognaro  
| TA | Mark Evenson, Alfred Tellian, Andreas Woratschek  
| XLAB | Gregor Berginc, Miha Stopar, Gregor Pipan, Miha Vuk  
| SAP | Tariq Ellahi, Wolfgang Theilmann  
| Intel | John Kennedy, Michael Nolan, Joe Butler, Victor Bayon  
| TID | Juan Lambea Rueda, Carlos Bueno Royo, Alfonso Castro  
| UDO | Philipp Wieder, Constantinos Kotsokalis  
| FZI | Christof Momm, Franz Brosh  
| GPI | Giampaolo Armellin  
| City | Davide Lorenzoli  

### 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](https://creativecommons.org/licenses/by/3.0/).
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Author</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>07th April</td>
<td>Francesco Torelli</td>
<td>Template</td>
</tr>
<tr>
<td>0.2</td>
<td>15th April</td>
<td>Philipp Wieder</td>
<td>Filled sections 2 and 0</td>
</tr>
<tr>
<td>0.3</td>
<td>15th April</td>
<td>Francesco Torelli</td>
<td>Updated examples</td>
</tr>
<tr>
<td>0.4</td>
<td>16th April</td>
<td>Juan Lambea</td>
<td>Updated section 5.2</td>
</tr>
<tr>
<td>0.5</td>
<td>16th April</td>
<td>Francesco Torelli</td>
<td>Added a specific subsection for topics (originally included in the section of Accepted requirements)</td>
</tr>
<tr>
<td>0.6</td>
<td>16th April</td>
<td>Mark Evenson</td>
<td>started to fill content of sections 3 and 5.</td>
</tr>
<tr>
<td>0.7</td>
<td>22nd April</td>
<td>Wolfgang Theilmann</td>
<td>A1 inputs added to section 4.2</td>
</tr>
<tr>
<td>0.8</td>
<td>23rd April</td>
<td>Wolfgang Theilmann</td>
<td>Improvement of section 4.2</td>
</tr>
<tr>
<td>0.9</td>
<td>24th April</td>
<td>John Kennedy</td>
<td>A4 inputs added to section 4.5</td>
</tr>
<tr>
<td>0.10</td>
<td>24th April</td>
<td>Constantino Kotsokalis</td>
<td>Filled sections 4.6.3 - 4.6.5</td>
</tr>
<tr>
<td>0.11</td>
<td>27th April</td>
<td>Tariq Ellahi</td>
<td>A3 inputs added to section 4.4</td>
</tr>
<tr>
<td>0.12</td>
<td>27th April</td>
<td>Juan Lambea</td>
<td>Updated section 5.2</td>
</tr>
<tr>
<td>0.13</td>
<td>27th April</td>
<td>Christof Momm</td>
<td>A6 inputs added to section 4.7</td>
</tr>
<tr>
<td>0.14</td>
<td>28th April</td>
<td>Davide Lorenzoli</td>
<td>Added run-time prediction at service level related subsection in chapter A6 – Predictable Systems Engineering</td>
</tr>
<tr>
<td>0.15</td>
<td>30th April</td>
<td>Francesco Torelli</td>
<td>Added overview of SotA limits and simplified A5 tables.</td>
</tr>
<tr>
<td>0.16</td>
<td>05th May</td>
<td>Mark Evenson</td>
<td>Updated section 3.</td>
</tr>
<tr>
<td>0.17</td>
<td>20th May</td>
<td>Francesco Torelli</td>
<td>Improved sections 1 and 2. Added references.</td>
</tr>
<tr>
<td>0.18</td>
<td>25th May</td>
<td>Wolfgang Theilman</td>
<td>Updated section 4.2 and references.</td>
</tr>
<tr>
<td>0.19</td>
<td>26th May</td>
<td>Constantino Kotsokalis</td>
<td>Updated sections 4.6.2 and 4.6.3.</td>
</tr>
<tr>
<td>0.20</td>
<td>04th Jun</td>
<td>Juan Lambea</td>
<td>Updated section 4.3 and Appendix B.</td>
</tr>
<tr>
<td>0.21</td>
<td>05th Jun</td>
<td>Gregor Berginc</td>
<td>Updated Section 3 (improved requirement gathering process and added overview of use cases) and Appendix B.</td>
</tr>
<tr>
<td>0.22</td>
<td>08th Jun</td>
<td>John Kennedy, Christoff Mom, Davide Lorenzoli</td>
<td>Updated section 4.7. Added references.</td>
</tr>
<tr>
<td>0.23</td>
<td>16th Jun</td>
<td>Francesco Torelli</td>
<td>Updated section 3</td>
</tr>
<tr>
<td>0.24</td>
<td>26th Jun</td>
<td>Francesco Torelli</td>
<td>Updated section 4 and section 3</td>
</tr>
<tr>
<td>0.25</td>
<td>03rd Nov</td>
<td>Francesco Torelli</td>
<td>Restructured document for M18 resubmission.</td>
</tr>
<tr>
<td>0.26</td>
<td>03rd Nov</td>
<td>Michael Nolan</td>
<td>Updated updated section 3.2 to reflect changes made to the</td>
</tr>
</tbody>
</table>

Deleted: 85
Deleted: 91
Deleted: 85
<table>
<thead>
<tr>
<th>Date</th>
<th>Author</th>
<th>Action Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>09th Nov 09</td>
<td>Alfred Tellian</td>
<td>added comments concerning requirement process and statistics</td>
</tr>
<tr>
<td>10th Nov 09</td>
<td>Francesco Torelli</td>
<td>Updated introduction of section 4</td>
</tr>
<tr>
<td>12th Nov 09</td>
<td>Alfred Tellian</td>
<td>Drafted chapter 3.8 Statistics</td>
</tr>
<tr>
<td>12th Nov 09</td>
<td>Mark Evenson</td>
<td>Added draft of section 5, Evaluation Plan</td>
</tr>
<tr>
<td>12th Nov 09</td>
<td>Francesco Torelli</td>
<td>Added comments on requirement process section</td>
</tr>
<tr>
<td>13th Nov 09</td>
<td>Alfred Tellian</td>
<td>Updated Process description</td>
</tr>
<tr>
<td>13th Nov 09</td>
<td>Francesco Torelli</td>
<td>Added comments</td>
</tr>
<tr>
<td>20th Nov 09</td>
<td>Andreas Woratschek</td>
<td>Updated Process description</td>
</tr>
<tr>
<td>21st Nov 09</td>
<td>Tariq Ellahi</td>
<td>WP A3 Implementation Status w.r.t Framework Features</td>
</tr>
<tr>
<td>23rd Nov 09</td>
<td>Wolfgang Theilmann</td>
<td>added integration features; updated A1 section</td>
</tr>
<tr>
<td>23rd Nov 09</td>
<td>Franz Brosh</td>
<td>updated Section 4.6 about WP A6 feature status</td>
</tr>
<tr>
<td>24th Nov 09</td>
<td>Miha Vuk</td>
<td>Updated chapter 3.6</td>
</tr>
<tr>
<td>24th Nov 09</td>
<td>Paolo Zampognaro</td>
<td>Updated description of automatic binding implementation.</td>
</tr>
<tr>
<td>24th Nov 09</td>
<td>Joe Butler</td>
<td>1st draft of business metrics</td>
</tr>
<tr>
<td>25th Nov 09</td>
<td>Juan Lambea</td>
<td>Updated A2 section</td>
</tr>
<tr>
<td>26th Nov 09</td>
<td>John Kennedy</td>
<td>A4 section added</td>
</tr>
<tr>
<td>26th Nov 09</td>
<td>Carlos Royo</td>
<td>Updated</td>
</tr>
<tr>
<td>26th Nov 09</td>
<td>Miha Vuk</td>
<td>Updated sections 4.4 and 3.6</td>
</tr>
<tr>
<td>27th Nov 09</td>
<td>Francesco Torelli</td>
<td>added description of business features; updated description of infrastructure features.</td>
</tr>
<tr>
<td>30th Nov 09</td>
<td>Miha Vuk</td>
<td>updated section 3.7</td>
</tr>
<tr>
<td>01st Dec 09</td>
<td>Mark Evenson</td>
<td>Added updated requirements figure</td>
</tr>
<tr>
<td>01st Dec 09</td>
<td>Constantinos Kotsokalis</td>
<td>Updated A5 section</td>
</tr>
<tr>
<td>01st Dec 09</td>
<td>Francesco Torelli</td>
<td>added comments on evaluation process</td>
</tr>
<tr>
<td>02nd Dec 09</td>
<td>Tariq Ellahi</td>
<td>B3 Metric Revision</td>
</tr>
<tr>
<td>02nd Dec 09</td>
<td>Juan Lambea</td>
<td>3.7 and 4.2.1 sections updated</td>
</tr>
<tr>
<td>03rd Dec 09</td>
<td>Juan Lambea</td>
<td>updated A2 section</td>
</tr>
<tr>
<td>04th Dec 09</td>
<td>Alfonso Castro</td>
<td>Business Adjustment feature updated in section 4.2.2</td>
</tr>
<tr>
<td>04th Dec 09</td>
<td>Victor Bayon</td>
<td>Updated 3.7.3 with “Design Time”; Updated “Runtime” for Infra. and Software (service) Prediction</td>
</tr>
<tr>
<td>Date</td>
<td>Author</td>
<td>Changes</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>04th Dec 09</td>
<td>Alfred Tellian</td>
<td>Updated statistics 3.8</td>
</tr>
<tr>
<td>04th Dec 09</td>
<td>Joe Butler</td>
<td>Updated Business Metrics</td>
</tr>
<tr>
<td>04th Dec 09</td>
<td>John Kennedy</td>
<td>A4 updates</td>
</tr>
<tr>
<td>05th Dec 09</td>
<td>Miha Vuk</td>
<td>Updated integration features description</td>
</tr>
<tr>
<td>05th Dec 09</td>
<td>Francesco Torelli</td>
<td>Addition of summary table on features, addition of conclusions, revision of several sections and general reformatting.</td>
</tr>
<tr>
<td>09th Dec 09</td>
<td>Francesco Torelli</td>
<td>Corrected errors in table on features</td>
</tr>
<tr>
<td>10th Dec 09</td>
<td>Alfred Tellian</td>
<td>Update chapter 3.7 statistics</td>
</tr>
<tr>
<td>11th Dec 09</td>
<td>Miha Vuk</td>
<td>Fixed issues on feature descriptions and figure captions.</td>
</tr>
<tr>
<td>11th Dec 09</td>
<td>Mark Evenson</td>
<td>Updated requirements process and metrics sections.</td>
</tr>
<tr>
<td>11th Dec 09</td>
<td>Juan Lambea</td>
<td>Updated section 5.5.2</td>
</tr>
<tr>
<td>14th Dec 09</td>
<td>Franz Brosh</td>
<td>Updated A6 section</td>
</tr>
<tr>
<td>15th Dec 09</td>
<td>Ellahi Tariq</td>
<td>Updated A3 section</td>
</tr>
<tr>
<td>16th Dec 09</td>
<td>Alfred Tellian</td>
<td>Added appendix C</td>
</tr>
<tr>
<td>17th Dec 09</td>
<td>Francesco Torelli</td>
<td>Extended introduction on consolidated features and final reformatting.</td>
</tr>
</tbody>
</table>
Executive Summary

The main goal of this document is to give an overview of the relationships between the B-Line requirements collected by work package B1 and the scientific results produced by the A-Line.

We have collected more than 200 requirements from internal and external sources. Internal sources are the SLA@SOI industrial use cases (WP B3 – B7), while external sources are other projects we collaborate with and the literature considered during the state of the art analysis [1].

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 results upcoming from the A-Line. The priorities and description of the requirements have been updated accordingly. After the consolidation of this requirements, we found that the new scenarios covers all the scientific WP outcomes in a balanced way.

All the requirements have been categorized according to different criteria, including the phase of the service lifecycle in which they are considered, the effected layer, the involved user type, and the main scientific work package that will take care of their implementation. All collected requirements have been finally consolidated in a set of 42 features that the SLA@SOI framework must implement to satisfy all the needs of the use cases currently identified.

Detailed descriptions of the process used to collect requirements and of the results of this effort are given in the Section 3 and 4 respectively.

The identified features are now under further analysis from the scientific work packages (action line A) to produce a detailed design for their implementation. Section 5 reports the current state of this analysis. In summary, although the development in the first year of the project was driven mainly by the needs of the ORC, several of the identified features have in fact already been partially implemented in the current results of the A-Line.

The analysis of the requirements from the scientific work packages has been performed in parallel to the analysis of the state of the art (summarized in Section 2). We are convinced that the satisfaction of the collected requirements should also drive scientific relevant improvements. For this reason in Section 5 each scientific work package also reported the advancements with respect to the state of the art achieved during the implementation of the current framework components.

Following the production of a first version of the Framework components offering the requested features, the next phase will be the evaluation of these results. The goals of this evaluation are: (1) to measure how well the scientific results from A-Line implement the technical requirements of B-Line and (2) to assess what is the business value of the applications developed with the help of the SLA@SOI framework. The first kind of evaluation will be performed through both automatic tests and human evaluations of the developed software. The second evaluation will be based on metrics derived from the business values defined by each use
Section 6 describes the process that will be followed for the evaluation and the metrics currently identified.
# Table of Contents

1. **Introduction** ........................................................................................................... 12
2. **State of the Art Analysis** ....................................................................................... 13
3. **Requirements Process** ............................................................................................. 15
   3.1 **Requirements Collection** .................................................................................... 16
   3.2 **Requirements Validation** .................................................................................... 18
   3.3 **Requirements Consolidation** .............................................................................. 20
   3.4 **Requirements Implementation** ............................................................................. 22
   3.5 **Industrial use case overview** .............................................................................. 22
      3.5.1 Use case B3: ERP Hosting (SAP) ..................................................................... 23
      3.5.2 Use case B4: Enterprise IT (Intel) ................................................................. 23
      3.5.3 Use case B5: Service Aggregator (Telekom Austria) .................................... 23
      3.5.4 Use case B6: E-Government (ENG) ............................................................... 23
   3.6 **Overview of External Sources** ............................................................................. 24
   3.7 **Requirements Analysis** .................................................................................... 25
4. **Consolidated Requirements** .................................................................................... 28
   4.1.1 Framework Management .................................................................................. 33
   4.1.2 Design & Development ..................................................................................... 34
   4.1.3 Pre-offering ........................................................................................................ 35
   4.1.4 Service Offering .................................................................................................. 35
   4.1.5 Service Negotiation ............................................................................................ 37
   4.1.6 Service Provisioning ............................................................................................ 39
   4.1.7 SLA Enforcement ................................................................................................ 40
   4.1.8 Runtime Prediction ............................................................................................... 41
   4.1.9 Service Monitoring .............................................................................................. 42
   4.1.10 Service Reporting .............................................................................................. 43
5. **Status of Implementation** ........................................................................................ 45
   5.1 **A1 - Architecture & Integration** ........................................................................ 46
      5.1.1 Current state ....................................................................................................... 46
      5.1.2 Advances with respect to the state of the art ................................................... 47
      5.1.3 Lessons learned & next steps .......................................................................... 48
   5.2 **A2 - Business Management** ............................................................................... 48
      5.2.1 Current state ....................................................................................................... 49
      5.2.2 Advances with respect to the state of the art ................................................... 50
      5.2.3 Lessons learned & next steps .......................................................................... 51
   5.3 **A3 - Service Management** ................................................................................ 53
      5.3.1 Current state ....................................................................................................... 55
      5.3.2 Advances with respect to the state of the art ................................................... 56
      5.3.3 Lessons learned & next steps .......................................................................... 57
   5.4 **A4 - Infrastructure Management** ....................................................................... 59
      5.4.1 Current state ....................................................................................................... 59
      5.4.2 Advances with respect to the state of the art ................................................... 61
      5.4.3 Lessons learned & next steps .......................................................................... 61
   5.5 **A5 - SLA Management & Foundations** .............................................................. 62
      5.5.1 Current state ....................................................................................................... 62
      1.1.1 Advances with respect to the state of the art ................................................... 65
      1.1.2 Lessons learned & next steps .......................................................................... 66
   5.6 **A6 - Predictable Systems Engineering** ............................................................... 66
      5.6.1 Current state ....................................................................................................... 66
      5.6.2 Advances with respect to the state of the art ................................................... 68
      5.6.3 Lessons learned & next steps .......................................................................... 69
6. **Requirement Evaluation** ........................................................................................ 71
   6.1 Evaluation Process .................................................................................................. 71
   6.2 **Scientific Metrics Evaluation** .............................................................................. 73
      6.2.1 Testing ................................................................................................................ 75
Table of Figures

Figure 1: Steps Of the Requirement Process .................................................. 16
Figure 2: Percentage of requirements by source ........................................... 28
Figure 3: Percentage of significant requirements ......................................... 28
Figure 4: Number of requirements by number of requests ........................... 28
Figure 5: Use cases distribution of requirements ......................................... 28
Figure 6: Percentage of high and medium requirements by Action Line WPs ... 28
Figure 7: Scientific work package organisation ........................................... 46
Figure 8: Requirement Evaluation Process .................................................. 71

List of Tables

Table 1: Required Framework Features ......................................................... 30
Table 1: Requirements Evaluation Goals ...................................................... 73
Table 2: Evaluation Survey ........................................................................ 77
Table 3: Business Metrics of Use Case B3 .................................................... 77
Table 4: Business Metrics of Use Case B4 .................................................... 78
Table 5: Business Metrics of Use Case B5 (1) .............................................. 79
Table 5: Business Metrics of Use Case B5 (2) .............................................. 80
Table 5: Business Metrics of Use Case B6 .................................................... 81
1 Introduction

This document is intended to evaluate the relationships between the B-Line requirements collected by work package B1 and the scientific results produced by the A-Line. Whereas the specific objectives and results of each project work package are described in detail in specific deliverables, here we will give a description of the relevance and status of such results with respect to the requirements of the project use cases.

Requirements have been collected by internal and external sources. By internal requirements we mean requirements collected from the SLA@SOI industrial use cases (WP B3 – B7) while the external requirements are requirements collected by external projects we collaborate with or as an output of the state of the art analysis.

State of the art analysis is an on going process that will continue throughout the project, especially as part of the work of action line A. Due to the large number of the considered strands of work, the state of the art is described in a separate accompanying document [1]. Section 2 summarizes some of the considerations that have influenced the project decisions, while Section 5 describes, for each scientific WP, the advances with respect to the state of the art.

Section 3 describes the process used to collect requirements. The consolidated requirements (requested framework features) resulting from the process are described in section 4.

Section 5 reports, for each scientific WP, the status of the implementation of the features associated to that WP. The same section shows the relationships between features and the state of the art. Lessons learned during the first year of development are also described.

Finally section 6 describes the evaluation process that will be used during year 2 and 3 to assess the project results from both the scientific/functional and industrial/business point of view.
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
3.5). 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 implements 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 to describe 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 given in Section 5 "Status of Implementation".
3 Requirements Process

The requirement process for SLA@SOI consists of four sequential steps:

1. Collection
2. Validation
3. Consolidation
4. Implementation

During the collection phase we sought to gather the requirements from four sources:

1. Internal Questionnaire
2. Internal requirements from Use Cases
3. External requirements (partners)
4. External requirements (SotA)

A first order plausibility check was then performed, comprising the categorization of these requirements on a web-based database system. By placing the requirements online we sought to facilitate communication between the consortium members.

The purpose of the validation phase was to improve the quality of the requirements by normalizing the values in relationship to the other requirements and to get stable and cross-checked requirements from B-line.

The goal of the consolidation step in the process was to assign requirements to a defined feature or feature-group. Following which, the requirements were checked by A-line, and their feasibility verified.

The aim of the implementation phase is to have implemented the features which are necessary for the satisfaction of the requirements.
3.1 Requirements Collection

This section describes the requirements gathering process through the first year of the project. Requirements were collected from sources both internal and external to the consortium covering as broad range of use cases as possible.

The complete process consisted of four phases, namely:

- Collection of requirements from internal industrial use cases
- Collection of requirements from external entities (research community, industry etc.) and SoTA analysis
- Incorporation into a ticket tracking system (Trac)
- Review of requirements made by action line A and B work packages.

The process is described in detail below. It is evident that this is an iterative process of constantly refining and improving the catalogue of requirements.

For the internal requirements, we started by collecting a list of questions and topics for each scientific work package. Due to many overlaps in the list, we performed an additional clustering and categorization of questions to create a questionnaire which consisted of about 140 questions. Questions were divided into two main categories, namely Business View and Technological View. Business View questions were concerned with an end user perspective and the involved stakeholders. It consisted of questions about involved stakeholders, the nature of
services, the IT systems, user stories, service discovery, modelling assumptions, monitoring aspects and design time prediction. Technological questions solicited information on the customer profile, the service description standards, infrastructure implementation details, ability for dynamic re-provisioning, interaction with third-party providers, and service manageability. The purpose of the questionnaire was to elicit expectations of technologies and features for implementations of industrial use cases which would start in M12.

The results of the surveys as filled out by each industrial use case partner were then collated into a single document. Out of the collation of the summarized results, 116 specific requirements were synthesized, each described in detail and listing relevant use cases and an educated guess as to which A-line work package would be involved in the implementation of these requirements. To capture use case specific requirements that could not be elicited directly by such a general survey, the individual use case specifications were analysed as an additional source of requirements (in particular: internal report D.B[3-6].1a released at M6, draft of D.B[3-6].1 available at M10, and direct communication from use case leaders).

External requirements were gathered from both the state-of-the-art survey [1] and by seeking feedback from projects which had been identified as potential collaborators.

All this information was entered into an online database associated with the shared code repository of the consortium and accessible to it. The software we used was a customized version of the Open Source project management and feature tracking solution Trac [20]. We customized this database to fit the data model shown in Figure 2. The purpose of this customization was to differentiate the requirements by the affected work package and to facilitate the assignment of priorities to each requirement. When we entered the collected requirements, we sought to “collapse” requirements as much as possible by consolidating requirements where it made sense. We intended that this online, shared solution should be a lightweight coordination mechanism, which would be easier to track than something like a centrally maintained spreadsheet. As a bonus, we intend to use the database to track the actual software issues as the project enters the phase where these requirements are to be implemented.
Following online collation of the raw requirements, the industrial use case partners then went over all the collected information, clarifying the description and understanding of the requirements as summarized, and noting the revised dependencies.

### 3.2 Requirements Validation

After the completion of the requirement collection, the requirement validation process was instigated to improve the quality of the requirements. This process entailed that the collected requirements again be checked by all B-line partners. In particular, the B-line WP-partners set priorities for each requirement, expressing the importance of the requirement for each B-line WP.

The first step in the validation process was to export the data, converting the requirements from the TRAC-system into a spreadsheet including all fields. A
report based on this spreadsheet was used for statistical and monitoring purposes. From the spreadsheet a statistical analysis concerning the status of the validation was done. This information was sent to the WP-leads.

The B-line WP-leads set again priorities of requirements on Trac. A new priority value "zero" was then added by which we could denote that a use case had evaluated the requirement but had no need for the fulfilment of the requirement for their work. For each requirement that a use case-partner was interested in, the email address of the nominated partner was added to the field "CC" in Trac. If there were any changes or new comments in the Trac concerning the requirement, the nominated partners contained in this field were sent an automatic notification via e-mail. The comment-field in Trac was then used for collaborating with other use cases to improve the description of the requirement.

The improvement of the description of the requirements has been done in collaboration with other use cases that shared the need for the same requirement. On Trac, a comment to the requirement which should be improved was then added for producing an improved description. Trac automatically sends an email to all interested parties when any aspect of the specific requirement is modified. Any receiver of this notification has the ability to reply by adding another comment on TrAC. If all related use cases accept the proposed modification, the use case initiator of the proposed change then modifies the description text. The overall priority is set once the description is judged sufficient. Otherwise the priority is set to zero and additional requirements proposed if necessary.

Any missing requirements were directly added on TRAC from B-line WP-partners, with the responsible WP-partner for the new requirement adding a ticket in TRAC, including a precise description.

The monitoring of the progress during the validation of the requirements was done via TRAC. Every week a spreadsheet with the requirements from TRAC was created, and an evaluation of the current state of each requirement performed. The results were documented and depicted in a report (including statistics) and sent to all responsible A- and B-line WP-partners. The status field was set depending on the B-line WP-priorities from B1 (checked or cancelled) during the weekly TRAC-reports (monitoring). The requirements with status “checked” were then assigned to a “responsible” A-line-WP and signalled to A-Line.

At that point A-Line continued with the next step of the requirement process: the “consolidation”. In the case that all B-line-WP-priorities were set to “zero” the status of the requirement was set to “cancelled” (B1 TA) after a check and acceptance from B1.

**General conditions:**

The following prioritisations were defined in Trac and chosen from each B-line WP:

- Zero (not relevant for this use case)
- Low (nice to have)
- Medium (must-have for year 3)
- High (must-have year 2)
Before the start of the validation of the requirements all priorities in Trac were set to “none” to get a defined initial state. The first time a requirement was entered into the Trac-system, its status was set to “proposed”.

During validation, the "topic expert" updated also the following fields in Trac:

- Check/adaptation of the short description
- Priority (B-line-WP-priority)
- Role/candidate role
- Scientific WP
- Key words (name of the corresponding features)
- Component (could be overwritten by A-line)
- Framework lifecycle (could be overwritten by A-line)
- SLA layer/Subject

The update of the fields ‘subject’, ‘priority’, ‘layer’ and ‘role’ was made mandatory in order to collect information necessary for the normalization of the requirements into features.

A definition of the candidate roles can be found in the D.A1-1a glossary. Each requirement is related to a candidate role within Trac.

The following status values were used in Trac to capture the current state of each requirement:

- proposed
- checked
- accepted
- rejected
- designed
- implemented
- tested
- issues
- approved
- cancelled
- failed

A definition/description of the above status values that are used in TRAC can be found in the appendix C.

### 3.3 Requirements Consolidation

Once all priorities and fields in TRAC were filled accordingly, the status of the requirement was set to “checked” during the monitoring/reporting from B1 (TA).

Based on the validated requirements from B-line (requirements in state “checked”) an analysis from A-line was done. The result of the evaluation from A-line was a set of features, which are necessary for an implementation of the requirement.

The features were clustered in groups called topics. The following topics were defined:

1. Business
2. Provisioning, Monitoring and Adjustments
3. Prediction
4. Integration (including modularization, configuration, SW development guidelines)

The resulting feature set and topics gathered from the analyses of the requirements formed the basis for the further consolidation of the requirements. For each topic, a topic expert, belonging to B1 WP, was nominated. As B1 partners are also A-Line partners, they have the knowledge to take the responsibility of scientific topics. The topic experts have the responsibility for the quality and coverage of requirements for their topic.

The nominated persons assigned by B1 are the same persons who have already researched or were going to research the relevant topic in the A-Line. These persons were committed to participate in B1 and were in charge of performing the consolidation process, which is a B1 responsibility. In this way A-Line people have one single point of contact for each topic who participates also to the B1 work and is able to directly answer to A-Line questions and/or contact relevant use case leaders to resolve open issues.

Tasks of Topic Experts:

During the task of correlation of requirements to features, the topic experts also checked if all (or at least the mandatory) fields in Trac were filled. The topic experts could also decide to split a requirement if that would improve the intelligibility of the requirement.

If there were duplicated requirements in Trac, the topic experts should decide to cancel one of them. In this case a reference in the Trac field ‘keywords’ is used to record the cancellation. The field ‘keywords’ should be used to note that a duplicated requirement has been cancelled and reference the remaining requirement.

If during the analysis, topic experts noticed that a requirement was missing, they could decide to add a new requirement in TRAC. If non functional features needed to be added (as current features represent functionalities of the framework) this was done by adding them on the feature-list and notifying B1.

The topic experts were individually responsible for deciding how to make the consolidation. The main constraint was the further use of TRAC for the requirement process. When the screening of the requirements by the topic experts was completed, the A-line was informed to check the feasibility of the requirements.

Each responsible A-line-WP first checked that they understood the requirements. If the responsible A-line WP-partner did not understand a checked requirement (s)he added a comment to it, containing or referring to a request for clarification to the use cases and setting the status of the requirement to “issues”. The clarification was done by contacting the responsible B-line partner. If this was not possible, the clarification happened via the topic expert. The topic expert was the single point of contact for A-line-WP-partners in this case. If a requirement involved more than one component, A-line created a new sub-requirement for each component and related it to the original B-Line requirement. The related WP also associated with each sub-requirement a specific unit-test. During the consolidation phase, the responsible A-line workpackage may discover that several B-Line requirements are specializations of a more general requirement. In this case the A-line added on Trac such a generalized requirement, related the B-line requirements to it and specified an implementation for the generalized requirement. When the status of a generalized requirement was changed by A-
Line, its related B-Line requirements were also be set by A-line to the same value.

After its evaluation by the A-line, a requirement was set to “rejected” if an implementation of the requirement was considered not to be possible or if the realisation had no impact to the architecture from the perspective of A-line. In the other case, the status of the requirement was instead set to “accepted” by the A-line.

The consolidation phase was completed after all requirements were checked and assigned to a defined feature.

### 3.4 Requirements Implementation

The implementation phase follows the first cycle of requirements consolidation. The requirements have been checked. The analyses of the requirements provided a stable feature-list (see Section 4). The responsible A-line WP-partner decides how to realize the requirements associated to a same feature and add a comment that directly specifies or refers to a wiki-page or SVN document providing details on the use of the feature. Suggested details for functional features are the following:

- a. Components which realize that requirements.
- b. If some configuration of the framework or some special deployment option must be adopted from the use case that has the requirements.
- c. An UML sequence diagram, pseudo code or some other equivalent representation that specifies how the use case must interact with the framework interfaces in order to access the feature.
- d. An integration test for checking if the requirements has been satisfied.

After acceptance of the proposed architecture (defined through step (a) – (d)) by B-line, A-line incorporates the requirement in the feature-list and sets its status to “designed” once A-line has completed the implementation of the requirement. Its status in TRAC is set to “implemented” by A-line. The process then continues with the evaluation process (see Section 6).

If during the implementation phase a feature split, additional/new requirements emerge, or an adaptation of requirements is deemed necessary, the process for these requirements will start again with the consolidation step.

### 3.5 Industrial use case 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 and communication services. In the following sections, we provide a brief introduction to all these use cases.

---

1 This deliverable does not consider the use case B7, (“Financial Grid”) as insufficient progress has been achieved in its definition.
3.5.1 **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 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, services) to facilitate appropriate adjustment.

3.5.2 **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.

3.5.3 **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 telecommunications 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.

3.5.4 **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 can not 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 will 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.

3.6 **Overview of External Sources**

In the second round of requirements gathering 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.

3.7 Requirements Analysis

At the end of the consolidation process, we counted a total of 233 requirements. Fifteen percent (15%) of these requirements came from external sources (collected through direct interaction with external parties or by analysis of the literature identified as part of the state of the art review). Figure 3 shows the percentage of requirements identified through the different types of targeted resources.

![Figure 3: Percentage of requirements by source](image)

During the revision steps, 48 requirements were closed as they were considered out of scope after further analysis. If all priorities of a requirement had been set to “zero” by all B-line WPs, the status of the requirement had been set to “closed” after a check from B1-WP-lead and A-line.
Of the remaining 185 requirements, 70 were asked by all the use cases (although with different priority), and 51 were considered useful by at least 3 use cases.

An overall priority was assigned to each requirement. Possible values for the “overall priorities” were:

- Blocker: at least one use case assigned high priority
- Critical: at least two use cases assigned medium priority
- Major: at least one use case assigned medium priority
- Low: at least one use case assigned minor priority

We defined as “significant” the requirements whose priority highest then Low. As 16 requirements are considered of Low priority, we collected 169 significant requirements. The important percentage of requirements closed or with low priority may be ascribed to different factors such as changes in the use cases objectives, evolution of the overall project understanding, non applicability of external requirements to SLA@SOI project and so on.
Figure 5 shows how many requirements correspond to the overall priorities “blocker”, “critical”, “major” and “minor”. Of the significant requirements, 116 are blocker, i.e. they are judged of high priority from at least one use case. Therefore these 116 requirements were identified as the most important candidates for v1 of the SLA@SOI framework.

Figure 6 shows how many requirements were required by each use case. It is evident that use cases B5 and B3 are the most demanding in terms of number of requirements, especially medium ones.

Despite the apparent homogeneity of these figures, there are actually important differences between the use cases. There are 59 requirements that have been assigned priority “high” by more than 2 use cases. Moreover amongst the “blocker” requirements there were 10 requirements that were required by just one use case.

It is also interesting to see the distribution of requirements with respect to Action Line A WPs. The next figure shows the percentage of requirements needed from each action line A WP by each use case. The figure refers just to the significant requirements.

Figure 5: Use cases distribution of requirements

<table>
<thead>
<tr>
<th>Use Case</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
</tr>
</thead>
<tbody>
<tr>
<td>B3</td>
<td>1.83%</td>
<td>5.11%</td>
<td>4.35%</td>
<td>2.97%</td>
<td>8.02%</td>
<td>5.02%</td>
</tr>
<tr>
<td>B4</td>
<td>1.83%</td>
<td>3.50%</td>
<td>4.68%</td>
<td>3.43%</td>
<td>8.25%</td>
<td>2.64%</td>
</tr>
<tr>
<td>B5</td>
<td>1.15%</td>
<td>5.38%</td>
<td>3.34%</td>
<td>1.83%</td>
<td>8.02%</td>
<td>7.92%</td>
</tr>
<tr>
<td>B6</td>
<td>2.29%</td>
<td>3.77%</td>
<td>3.01%</td>
<td>1.83%</td>
<td>8.25%</td>
<td>2.64%</td>
</tr>
</tbody>
</table>

Deleted: 6

Deleted: 85

Deleted: 91

Deleted: 85
All use cases are distributed on scientific WPs in a similar way and have the highest percentage of requirements on A5. From the picture there is evidence of some use cases peculiarities. B6 is the use case with fewest requirements impacting on A4; indeed most of the requirements associated to A4 are about infrastructure hardware management, while B6 is more concerned with human based infrastructure. Use cases B4 and B6 represents a little portion of the requirements on prediction, indeed most prediction requirements are about software services, which are less relevant in the aforementioned use cases.

It is evident that the most critical scientific WP, with respect to use cases requirements, is A5. This fact 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, that is impacted by the requirements.

The minor relevance of A1 with respect to use case requirements is also obvious, as at least at this stage the use cases are more concerned with functionalities offered by the SLA@SOI framework, rather than with architectural requirements or other non functional requirements. We have to consider that at this stage there are likely several technical requirements not explicitly expressed because they are implicit in the SLA@SOI assumptions, such as the development of an integrated framework.

**Consolidated Requirements**

In this section we describe the final formulation of the SLA@SOI framework requirements, coming from the consolidation process.

As of month 17, some aspects of the use cases were refined and the formulation of some requirements was outdated with respect to the current expectations of the B-Line for the A-Line. For this reason the consolidation process was preceded by a request to all the use cases to update the description of requirements and their priorities.

Afterwards, the collected requirements were analysed by WP B1 in order to remove redundant requirements and reduce the ambiguities in each requirement. To this end the requirements were split into four big groups according to 4 topics: business, provisioning-monitoring-adjustment, prediction, and integration. Each group of requirements was analysed by a person (called Topic Expert) belonging to B1 partners. Each topic expert closed duplicates of requirements, clarified the relationship between them, and aggregated the remaining requirements to a defined and specific functionality or characteristic of the framework, called “feature”. These features were then reorganized in categories corresponding mainly to service lifecycle phases and application development phases.

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 will be 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. In particular, the table shows the scientific work package associated to the feature and the use cases that will use them:

<table>
<thead>
<tr>
<th>WP</th>
<th>Feature</th>
<th>Use Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Framework Management</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1 Framework Configuration &amp; Setup</td>
<td></td>
<td>X X X X</td>
</tr>
<tr>
<td>A1 Framework Model Configuration</td>
<td></td>
<td>X X X</td>
</tr>
<tr>
<td>A1 Framework Operation</td>
<td></td>
<td>X X X</td>
</tr>
<tr>
<td>A1 Framework Access</td>
<td></td>
<td>X X X X</td>
</tr>
<tr>
<td>Design &amp; Development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A3 Reference Software Manageability Components and Configuration</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>A5 Service properties dependencies coding</td>
<td></td>
<td>X X X X</td>
</tr>
<tr>
<td>A6 Designing of QoS predictable systems</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>A6 Manageability Design</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>Pre-offering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2 Customer Registration</td>
<td></td>
<td>X X X</td>
</tr>
<tr>
<td>A3 3rd Parties Configuration</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>Service Offering</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A5 Discovery of Serv. and SLAT</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>A2 Product Management</td>
<td></td>
<td>X X X</td>
</tr>
<tr>
<td>A2 Product Discovery</td>
<td></td>
<td>X X</td>
</tr>
<tr>
<td>A2 Product Definition</td>
<td></td>
<td>X X X X</td>
</tr>
<tr>
<td>A5 SLAT Definition</td>
<td></td>
<td>X X X X</td>
</tr>
<tr>
<td>Service Negotiation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>A2 Business Negotiation</td>
<td></td>
<td>X X X X</td>
</tr>
<tr>
<td>A2 SLAT Customization</td>
<td></td>
<td>X X X 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>A2</td>
<td>Business SLA planning for negotiation</td>
<td>X</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>X</td>
</tr>
<tr>
<td></td>
<td>Service Provisioning</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>SLA enforcement</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>
<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>
<tr>
<td></td>
<td>Runtime Prediction</td>
<td></td>
</tr>
<tr>
<td>A6</td>
<td>Runtime infrastructure metrics prediction</td>
<td>X</td>
</tr>
<tr>
<td>A6</td>
<td>Runtime Software SLAs violations prediction</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Service Monitoring</td>
<td></td>
</tr>
<tr>
<td>A5</td>
<td>Monitoring coordination</td>
<td>X</td>
</tr>
<tr>
<td>A4</td>
<td>Virtual hardware infrastructure monitoring rules extraction</td>
<td>X</td>
</tr>
<tr>
<td>A4</td>
<td>Virtual hardware infrastructure observation and violation detection</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Software violation detection</td>
<td>X</td>
</tr>
<tr>
<td>A3</td>
<td>Monitoring rules extraction from SLA</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Service Reporting</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>SLA Management</td>
<td>X</td>
</tr>
<tr>
<td>A2</td>
<td>Push/Pull Business SLA Reporting</td>
<td>X</td>
</tr>
<tr>
<td>A5</td>
<td>SLA Status &amp; Resource Reporting</td>
<td>X</td>
</tr>
<tr>
<td>A4</td>
<td>Virtual hardware Infrastructure Reporting</td>
<td>X</td>
</tr>
</tbody>
</table>

**Table 1: Required Framework Features**

*Figure 9 and Figure 10 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.
Figure 7: 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 some kind of dependency between the implementations.
Figure 8: Runtime Features of the SLA@SOI Framework

In the following sections we describe each of the identified features. A-Line is expected to further enrich the definition of features during the design and development phases, to obtain a more general and integrated framework. Moreover the features might be enriched during the second half of Y2 based on new requirements coming from the experience of use of the framework.

Deleted: the
Deleted: one
Deleted: further
Deleted: 85
Deleted: 91
Deleted: 85
4.1.1 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.

4.1.2 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: A5  
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, especially with 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.

### 4.1.3 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:* A3  
*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.

### 4.1.4 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...
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 constrain 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 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.

### 4.1.5 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.
4.1.6 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
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.).

4.1.7 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.

### 4.1.8 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.

### 4.1.9 Service Monitoring

#### Monitoring Coordination

**Responsible WP: A5**  
**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.

4.1.10 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

-- Deleted: 85 --
-- Deleted: 91 --
-- Deleted: 85 --
**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.
5 Status of Implementation

In this section we present the current state of the implementation of the consolidated features described in section 4. For each WP of action line A, we describe the status of implementation of the features under responsibility of that WP.

We remember that action line A comprises the following work packages:

- **WP A1 Architecture & Integration** defines the overarching architecture of the SLA management framework, thus assuring an efficient integration of the respective work package results including the interlinking with the various use cases.
- **WP A2 Business Management** addresses the complete business-level interaction between a service provider and a customer including SLA specification & negotiation, business assessment and post sales management.
- **WP A3 Service Management** addresses the SLA-aware service management of a service provider, including service landscape modelling, monitoring, steering and provisioning.
- **WP A4 Infrastructure Management** addresses the SLA-aware infrastructure management of an infrastructure provider including harmonized management of different virtualization technologies, adaptive resource management and monitoring support.
- **WP A5 SLA Management & Foundations** delivers the solid foundation for SLA management at arbitrary layers including support for negotiation, brokering, translation, provisioning, planning, adjustment, and monitoring.
- **WP A6 Predictable Systems Engineering** addresses the engineering of predictable systems including software and hardware layers by advanced modelling and prediction techniques.

Overall, work packages A2-A4 are designed in a way that they cover the three main management areas of a service/infrastructure provider. Each area has quite distinct requirements and challenges and typically relates to a different department in a provider’s organization. SLA management as such will have many similarities across these areas. Therefore, work package A5 realizes the general foundations for SLA management. Depending on the needs of a specific provider, these foundations can be either fed into the respective management areas or can alternatively receive information from all areas and realize a holistic SLA management across all layers. Work package A6 contributes the perspective of a software provider and realized prediction models (captured at design-time) and tools for predicting service behaviour both at software and hardware level. The outcome of A6 can be flexibly used in any of the work packages A2-A6 according to the requirements of a specific provider. So far, the various work packages take quite specific perspectives on the overall problem of SLA management. Therefore, A1 is in charge of harmonizing these perspectives and specifying the overall system architecture as well as assuring the practical integration of different system components.
The research effort of the first year has not been explicitly driven by the industrial use case requirements\(^2\), but just by the B2 reference use case and by scientific research goals. Nevertheless we expect that some of this results already cover some of the features to be implemented in Y2 and Y3.

In the next subsections each scientific WP will describe the status of its implementation against the features. Each one of the following sections is divided in 3 subsections:

- "Current state", where the current results of the WP for each feature are described; when considered useful this section also include more technical details on the considered features;
- "Advances with respect to the state of the art", where the status of the current results of the WP are contrasted with the state of the art;
- "Lessons learned & next steps", which describes unforeseen difficulties or lessons learned and gives a perspectives on future works.

### 5.1 A1- Architecture & Integration

This section provides a summary and assessment of the main scientific results related to work package A1 Architecture & Integration.

#### 5.1.1 Current state

The WP A1’s main focus of activities was driven towards materializing the Y1 adhoc demonstrator and integrating all the different functionalities developed in the other work packages. A1’s implementation efforts with respect to each of the SLA@SOI framework features are described below:

\(^2\) The first year development is oriented to the realization of the B2 Ad-hoc demonstrator.
Feature: Framework Configuration & Setup

This feature requires for strong flexibility in the overall configuration and setup of the framework (for more details see Section 4). The year 1 implementation was completely monolithic and geared towards exactly one demonstrator. Plans and specifications for the refined architecture aim at a largely flexible setup where the framework can be used in three different flavours: as a complete framework, via a set of predefined manager modules or as a toolbox comprising all the various reuse components that the complete framework consists of. Furthermore, a common plugin mechanism shall support the easy extension or replacement of framework functionality.

Feature: Framework Model Configuration

This feature requires for strong flexibility in the overall configuration of the basic framework models (for more details see Section 4). The year 1 implementation was largely static in this regard apart from allowing the definition of new common metrics – however, this has not been used by the adhoc demonstrator. Plans and specifications for the refined architecture aim at a flexible setup where also the SLA models can be more easily extended with domain specific parameters.

Feature: Framework Operation

This feature requires for framework operation under the common Unix, Linux and Windows operating systems (for more details see Section 4). Most parts of the year 1 implementation were independent of any specific operating system with one exception which was the prediction service (relying on a Windows platform). This issue shall be resolved for the refined architecture.

Feature: Framework Access

This feature requires common access methods (Java, WSDL, Browser) to framework functionality (for more details see Section 4). The year 1 implementation was completely realized via Java components. Remote WSDL/SOAP-based interfaces have been specified only for the interaction between the framework and service customers, although the chosen Spring framework already supports such remote interfaces in general. Plans and specifications for the refined architecture aim at supporting WSDL-based access also between different framework components. Furthermore, graphical user interfaces will be browser-based.

5.1.2 Advances with respect to the state of the art

To the best of our knowledge there is no SLA management framework available which has a comparable coverage of related issues such as eContracting, design-time prediction, run-time prediction, software management, infrastructure management. In particular, the consistent management of SLAs across layers is highly unique.

Therefore, the overall architecture is highly innovative against the known state of the art.

The most related work to this one are two commercial frameworks, namely the SOA Governance Interoperability Framework (GIF) from HP [6] and CentraSite.
from Fujitsu and Software AG [7]. Both of them focus on the management of SOA based applications and therefore have a much more narrow scope than SLA@SOI.

5.1.3 Lessons learned & next steps

Overall the results from the architectural work within the first project year have been largely successful. The resulting architecture successfully integrates a variety of different perspectives and domains which were considered quite separate in the beginning.

An issue that turned out to be more complex to obtain than expected is the proper separation of concerns. On the one hand we see the central role of SLAs where the related management activities tend to require complete knowledge of the overall business/IT landscape along the complete lifecycle. On the other hand there is the ambition to create some generic SLA foundations which could be applied to different domains independent of how actual artefact management in those domains is done. These two goals clearly conflict with each other. The current implementation does not completely resolve it as the notion of software components and the management of software-related SLAs is not completely separated.

Within the next phase of the project, the architecture shall be further developed along all the topics discussed above. In addition to the integration of new functionality, special focus will be put on a more strict separation of concerns where flexible configuration eventually allows supporting different setups with different concrete separation requirements. Furthermore, the next phase will start to tackle those topics which have not been addressed at all within the first phase, namely the integration and the configuration issues.

The roadmap of architecture activities in the next phase consists of the following steps:

- open discussion of conclusions from the ad-hoc demonstrator
- detailed gap analysis of ad-hoc framework against requirements from industrial use cases and external stakeholders
- high-level development of possible solution features to the identified gaps including assessment of expected complexity and effort as well as dependencies between features
- assessment of overall feature catalogue and selection of the top-ranked ones which can be realistically developed within the next project phase
- development of a detailed work plan according to selected features
- development of high-level architecture (lab architecture) for the second project phase
- validation of architecture with use cases, external requirements and scientific work packages (i.e. explicit cross-check to which extent requirements are met by the architecture design)
- refinement of the lab architecture
- early prototype validation against industrial use cases
- refinement of the lab architecture to its final version
- final validation of lab architecture against industrial use cases and external requirements

5.2 A2- Business Management
5.2.1 Current state

The main focus of A2 WP activities was driven towards materializing the Y1 adhoc demonstrator with special emphasis on eContracting management related artefacts. A2’s implementation efforts with respect to each of the SLA@SOI framework feature are described below:

Feature: Customer Registration

Feature details:
Various information is needed about the End Customer for the commercial interaction between End Customer and the Service Provider. This feature requires the availability of End Customers to define their information in order to be able to interact with the framework. Some of the details they must provide are contact details, billing information, some information about their preferences and reporting, etc. This information can be changed by customers and can be enhanced in order to negotiate new products.

Feature Implementation Status
This feature is scheduled to start in Year 2, therefore, implementation wasn’t started on this feature in Y1.

Feature: Product Discovery

Feature details:
This feature requires the capability for (human) End Customers and/or Providers to query for available products based on product description, categorization and other specific information from the SLAT. Basically this process entails two different tasks:
1) Maintaining and managing the Service Registry for each Service Provider in which the details about the different Services / Products are stored and
2) Exposing this information to end Customers as a result of their queries.

Feature Implementation Status
In year 1, an initial implementation of this feature was provided as part of the Product Discovery module located in eContracting module. The functionality that has been initially implemented is a keyword-based search tool that allows realizing a basic searching over the product catalog. For year 2, it is planned to investigate whether category searching techniques can be used in the improvement of the online searching, in order to generate more relevant results. Category searching will allow the customer to find products according to some of the specific characteristics by which they were classified.

Feature: Product Management

Feature details:
A Product will need to go through different stages during its lifecycle (such as creation, negotiation, provisioning, retirement, etc.). This feature requires the availability to manage the whole lifecycle of a product. During its lifecycle a
product is managed by different people in different departments, which have different roles in the commercial creation process:

- Product manager creates products associated to the SLATs available.
- Business manager defines business rules that will drive the negotiation process.
- Sales manager defines thresholds and all kind of business information related to the sale process according to business rules defined by business manager.
- Customer manager defines how to use the customer information profile to create rules, special offers and rewards (e.g. by segments). Finally, the customer manager leads the relationship with customers.

It is then necessary that a product follow different states based on its current status (available, expired, etc.).

**Feature Implementation Status**

In year 1, an initial implementation of this feature was included in the Business Web Tool. The functionality initially implemented is a graphic tool that is able to create and view some information related to a Product. The relationship between customers and contracted products is also available for viewing.

**Feature: Product Definition**

**Feature details:**

This feature offers the possibility of specifying all the information around the product that it is offered to the customers by a provider. The information that it is possible to create is the product description, offers and prices and SLA Templates. Also it is possible to associate some kind of policies to each product, in order to increase or decrease the price in the SLA assessment process.

To do that, this feature should also take into account Business Rules defined by the Business Department of the Service Provider.

**Feature Implementation Status**

In year 1, an initial implementation of this feature was included in the Business Web Tool. The functionality initially implemented in year 1 is a basic graphic tool where it is possible to create some information related to a Product - namely:

- product description,
- offers and prices,
- SLA Templates, and
- some kind of policies

**Feature: Business SLA Negotiation**

**Feature details:**

This feature requires the capability to support Product customization (based on some specific information about the Customer profile) and the negotiation process between Customer and Provider in order to achieve an SLA. During this process the Business Rules will have to be applied to get to the final agreed SLA and also it will be necessary take into account the negotiation template provided by the service provider.

The customer profile will be used in order to select the negotiation strategy. The Business Rules are specific constraints related to commercial or business characteristics of the services offered by the service provider and that have

```
Implications (i.e., they have to be followed) in negotiation processes. The business rules are applied in an automatic way in the negotiation process—e.g., they can provoke that the negotiation is refused or finish.

If the negotiation is not possible in an automatic way, the manager will have a Graphical User Interface (GUI) to try to achieve in a manual way.

**Feature Implementation Status**

In year 1, an initial implementation of this feature was provided by the Business Negotiation module inside eContracting. It was implemented as a basic one round negotiation. During the negotiation, it was checked whether there was a valid product associated with the SLA offer given by the customer, and the different values of the guarantee terms were checked against the original templates in order to find differences and apply the corresponding policies to adjust the price.

**Feature: SLAT Customization**

**Feature details:**

This feature requires the capability to customize the SLA templates of a product offered to a specific customer. This is done by taking into account Customer requirements as well as specific information that the Service Provider has about the Customer. The customer profile has several different aspects that can be used by this task—e.g., customer register information and preferences, previous business relationships and behaviours, and information analysed from BSS (Business Support System). The outcome is a customized SLA template appropriate for the particular Customer requesting the Product.

**Feature Implementation Status**

The effort planned for this task in year 1 was moved to the negotiation task, that was not originally scheduled for year 1, but was necessary in order to be aligned to the overall project.

**Feature: Business SLA planning for negotiation**

**Feature details:**

This feature concerns the capabilities needed by a Business Manager to define the parameters to achieve an automatic negotiation. These parameters describe the conditions under which the provider is available to agree an SLA offer from the customer. For instance, they could describe the minimum range of benefits or the maximum number of consecutive offers, a customer could present in order to achieve an agreement.

If it is not possible in an automatic way, the manager will have a Graphical User Interface (GUI) to try to achieve a manual negotiation.

**Feature Implementation Status**

In year 1, this feature was implemented through the creation of Policies. These are specific business rules used to adjust the price of an SLA during the negotiation process. In subsequent periods, the business rules will be enhanced with the definition of new kinds of business rules that, for instance, specify the margin of benefits expected from a negotiation process.
Feature: Business Adjustment

**Feature details:**
- This feature requires capabilities for the management of SLA violations detected at software and infrastructure level, regarding their possible economical impact.
- The feature also comprises the investigation of the best appropriate solution to restore the normal situation from the business perspective, and the triggering of the suitable actions. Examples of these actions are the evaluation of penalties to be applied, the re-negotiation of a SLA or termination of the current agreement.

**Feature Implementation Status**
- In year 1, an initial implementation of this feature was provided through the Events Monitoring and Penalty Management module inside e-Contracting. This entailed the reception of business violations of the SLA detected in the software and infrastructure layers, as well as a basic calculus of the penalties to apply because of this violation. These penalties are based on the conditions signed inside the SLA.

Feature: SLA Management

**Feature details:**
- This feature requires the capability to examine and manage all information related to SLA documents. This will allow checking the conditions signed inside SLAs by customers and providers in order to avoid misunderstandings between them.

**Feature Implementation Status**
- In year 1, there was a partial implementation of this feature. When a Guarantee Term’s violation of an SLA was received, it was necessary to apply a penalty. The information about which penalty to apply was one of the conditions signed inside the SLA between the customer and provider. So, a basic functionality was implemented to access to that information and generate the concrete amount of the penalty that had arisen.

Feature: Push/Pull Business SLA Reporting

**Feature details:**
- This feature requires the capability to generate reports in two ways: push & pull. Under some specific conditions (for instance when a violation and penalty is generated) the system must be able to notify some kind of information to the customer (in an automated way). The second one is requested by customers (pull). The customers want to retrieve information about the consumption of their SLAs under demand. Through these interfaces, it is possible to communicate the information related to the behaviour of SLAs to the end-customer. The kind of information that will be reported to the end customer include, among others: SLA violations, billing for the use of products and SLAs status.

**Feature Implementation Status**
- This feature is scheduled to start in Year 2. A partial implementation was, however, provided in year 1, permitting the Ad hoc demonstrator to receive business violations, in order to advise it of the breach of the SLA signed.
5.2.2 Advances with respect to the state of the art

Limits of the studied models

Business management is in charge of handling the different business aspects in the relationships among customers and the different providers. In order to do so, different data models have been studied such as SID (Shared Information Data model from TMForum) and WS-Agreement. Below are presented the limits of the data model studied.

- SID model doesn’t support the full SLA perspective
  - It is necessary to do an adaptation of SID. As we exposed here, SID has some service level agreement aspects that are far off WS-Agreement standard and business terms.
  - The SID model can be enriched with aspects supported by WS-Agreement and use cases of SLA@SOI. Then the outcome will support a complete business framework.
- WS-Agreement doesn’t support a common standardized business model that manages business terms.
  - WS-Agreement is a very open standard, and it leaves some aspects without a concrete definition. For example, business terms about service support and limitations, economic information about prices and offers, etc. In a real business environment, these aspects have to be covered and well known with the parties. If we don’t do it, the business environment won’t be a real and operational environment and would be a proof of concept with a very limited value in the real world.

Innovative characteristics supported and planned results

- Business data model definition
  - The business model supports many business aspects that must be modelled for supporting SLA@SOI features. The business data model will be enriched with more requirements for other Work Packages, mainly B-line work packages, that supports very different use cases.
- SID contribution to enlarge and support concepts and business view obtained in SLA@SOI with the business data model view
  - SID model is well defined in terms of product offers and prices, although we will enrich it with business service level agreement terms, oriented to support the WS-Agreement standard and SLA framework defined in Work Package 5.
- WS-Agreement contribution to support business parameters, terms, and offer and prices information
  - WS-Agreement has not defined business parameters and in some aspects is open and under-defined. We have defined a way to specify the business terms, offers and prices.

These above extensions over the data models defined (SID, WS-Agreement) will allow the whole definition and management of products and SLAs. The aspects covered are, for instance, the prices or the penalties to apply when one of the conditions signed inside the SLA is not fulfilled. The features that are based on this enhanced model are product definition, product management or product discovery which searching parameters are described in the model.
• Business engine core for defining and supporting dynamic negotiation agreements
  o The negotiation phase can be developed in a complex way, with two rounds cycle, in order to support an automated mechanism for dynamic negotiation.

The features that will cover these aspects are Business SLA Negotiation in order to manage the business selling process and Business SLA planning for negotiation that will provide the business negotiation process the objectives to fulfil (for example the price expected) in order to achieve an SLA.

• Business engine for supporting the customization and personalization of offers
  o The business layer has to support the customization offers for customers. There are a lot ways to make the customization and we have to study the issues in depth. This is a value added feature.

The feature that will cover this aspect is SLAT customization, taking into account the specific profile (obtained from Business Support System) and characteristics of each customer.

5.2.3 Lessons learned & next steps

Business and commercial creation

In the first year there was limited work on commercial creation, since the focus was instead on building the overall SLA@SOI framework. We expect to study in depth the commercial creation aspects in year 2.

Negotiation

It is necessary to create a set of core of negotiation capabilities in SLA@SOI. There are two main packages involved in this task: Work Package 5 and Work Package 2.

New negotiation scenarios coming from revised use cases will be taken into account. These scenarios will impact upon other parts of the SLA@SOI framework. The impact of these changes on the different components will be investigated. Developments will be made with the view to evolve the negotiation component into an automated component.

Customer profile

We have to define the different aspects and terms of the customer profiles and how they can be used at the business level to customize the services and also how can be used on the overall framework.

Infrastructure layer

In year 1, the business, software and infrastructure layers were organised in an hierarchical way, with business separated from infrastructure by the software layer. It is worth noting that it could be very useful to modify the approach for supporting the manageability of the infrastructure layer from business layer...
directly. This is due as the trend of today is for managing and selling infrastructure capacities (without software services). This tendency could be reflected in our model.

**Business Adjustment**

In year 1, the business adjustment capabilities were designed and implemented through the Events Monitoring and Penalty Management module. This module dealt with the reception of SLA Violations from lower level layers (software and infrastructure) and implemented a basic method of calculation of the penalties derived from a given agreement un-fulfilment. This simplistic approach, while covering the scenarios of the year 1 demo, showed the complexity behind the adjustment functionality and gave some possible directions to further improve it. Apart from the penalty calculation, business adjustment should identify the root problem of the violations and trigger specific actions to solve the issue, in those cases where lower level adjustments can not take the right decisions. Examples of such actions are the re-negotiation of the SLA due to economical reasons, the termination of the agreement, or, in a multi-provider environment, the decision to change a service provided by one 3rd party to another service being offered by another service provider. All these actions are important not only to guarantee best customer experience, but also to take into account the economical impact of the violations and the actions to recover from them. Adjustment modules in other layers do not care about the business implications of the SLA enforcement.

5.3 **A3- Service Management**

5.3.1 **Current state**

The WP A3’s main focus of activities was driven towards materializing the Y1 adhoc demonstrator with special emphasis on service management related artefacts. A3’s implementation efforts with respect to each of the SLA@SOI framework feature are described below:

**Feature: Software Landscape**

*Feature details:*

This feature requires the availability of software components which can be used to store and manage the landscape meta-models. These meta-models are used to represent the software elements required for the provisioning of the services as well as their dependencies. Additionally, the software landscape will be used for capturing the packaging and deployment related information.

*Feature Implementation Status*

An initial landscape meta-model was designed to provide for the needs and requirements of the adhoc demonstrator. This meta-model enabled the service providers to represent and describe software elements along with their inter-dependencies. Additionally, the landscape meta-model provided for capturing the deployment and packaging artefacts, such as virtual appliances and virtual machine images, which can be used during the course of service provisioning process. As per requirement of this feature, a software landscape component was designed and implemented which can be used to store, update and query for the software artefacts required for the effective provisioning of a service.
Feature: Software Provisioning

Feature details:
As part of the overall SLA provisioning, software artefacts of the service landscapes need to be deployed / installed and configured according to the SLAs agreed with the customers. This feature relates to the interfaces, mechanisms and techniques which the framework will support to facilitate the provisioning and configuration of the software artefacts.

Feature Implementation Status
The feature will be implemented in Y2. As part of the realization of this feature, progress will be made in two directions; 1) definition of interfaces which SLA managers can use to trigger the provisioning and configuration of the software artefacts during SLA provisioning process and 2) realization of generic, customisable and extensible mechanisms which can facilitate the provisioning and configuration of a variety of software artefacts for heterogeneous service landscapes.

Feature: Dynamic Binding Setting

Feature details
This feature requires the availability of software components implementing techniques and mechanisms which facilitate the possibility to manage (add/replace/remove) the binding rules for an invoke activity of a deployed WS BPEL process. In detail a binding rule is composed of a condition and of binding information. The condition (a set of process variables and their values) allows to identify the process instances category which will use the specific binding information for that invoke activity.

Feature Implementation Status
This feature is realized by extending the administration interface of the (WSBPEL) standard ActiveBpel process execution engine. In order to support the requirements imposed by the feature the engine has been modified to implement such extended administration interface developing a first prototype. A basic scenario of using such feature foresees the deployment of a WSBPEL process before the provisioning and then, once SLA negotiations have successfully been completed for that process, specific binding information (for the category of process instances created for the same customer) can be dynamically added through the extended administrative interface.

Feature: Automatic Binding

Feature details
This feature requires the capabilities of abstract process execution. This capability provides the possibility to execute a WS-BPEL process having some service invocation not yet bound.

Feature Implementation Status
In order to support the requirements imposed by this feature, a methodology to deploy abstract processes has been defined. This methodology proposes to use a better separation of concerns, with respect to other approaches, by using the standard WS-BPEL language for process description and a decoupled SLAT based representation for describing the component services. Implementation activity to
adapt the run time platform to execute such kind of abstract processes has not yet started.

**Feature: Reference Software Manageability Components and Configuration**

*Feature details*
This feature eases configuration/implementation of monitoring and control capabilities for software components (as required for SLA management) through offering a set of unified interfaces, reference management information models and reference captor/manageability provider implementations for common SOA software resources. This includes the implementation specific resource instrumentations (i.e. probes) for common SOA software resources such as ActiveBPEL engine and Axis WS instrumentation, which can be used as a blueprint for designing and implementing probes for other technologies.

*Feature Implementation Status*
In order to support the requirements imposed by this feature, a unified software manageability infrastructure was designed and implemented for supporting the unified manageability aspects of the adhoc demonstrator. This manageability infrastructure is composed of three major components: Service Instrumentation, Management Agent and Manageability Interface. In order to realize this unified manageability infrastructure, the business components of the service have to provide an additional management service interface. This interface is implemented by the Instrumentation and is used by the Management Agent which implements the Manageability Interface. This Manageability Interface is then used by the management applications or clients.

The instrumentation component of the unified manageability infrastructure was designed following the Active Component design pattern. The Manageability Interface required the specification of a Manageability Data Model which was partly derived from the Configuration meta-model of WP A6 and was influenced by the DMTF CIM standard. Specifically, the CIM Metric Model was extended to satisfy the requirements imposed by this feature. Details on this can be found in the WP A3 deliverable.

**Feature: Software Monitoring Rules Extraction from SLAs**

*Feature details*
This feature requires the extraction and formulation of rules which are required to be configured during the software monitoring configuration process. These rules need to be extracted or derived from the SLAs.

*Feature Implementation Status*
The implementation of this feature progressed in two activities, first expressing the rules for atomic services, and then expressions of the rules for processes. The details for each are given below:

The solution for atomic services was proposed and implemented by CITY. This approach adopted the Event Calculus (EC) for expressing the monitoring rules. EC allows the expression of generic monitoring properties that can be applied for the monitoring of behavioural and non-functional properties of several kinds of dynamic software systems. Details of the Event Calculus and the monitoring engine architecture can be found in the WP A3 deliverable.
The process monitoring was implemented by FBK. Within this monitoring framework the property to monitor must be specified using the Runtime Monitor Specification Language (RTML). RTML is very expressive as it allows specifying Instance Monitors as well as Class Monitors; moreover, it allows for specifying Boolean properties related to the execution of processes, as well as statistic properties and time-related properties. Eventually the framework is able to translate automatically RTML monitor specifications into the Java code that implements the monitors.

**Feature: Software Violation Detection**

**Feature details**

This feature requires the availability of real-time alerting of violations of configured conditions on provisioned software.

**Feature Implementation Status**

This feature leverages the features described above. Monitoring rules extracted and specified using the techniques described above are continuously evaluated by the monitoring engines to identify potential software violations, which are communicated to the respective consumer software components through the event bus for adjustment actions. Detailed information on the monitoring engine implementation can be found in the WP A3 deliverable.

**Feature: Software Adjustment**

**Feature details**

This feature requires the capabilities to apply runtime reconfiguration of software resources to correct situations where SLA violations have occurred and restore the normal functioning of the system.

**Feature Implementation Status**

The software adjustment capabilities were realized through the design and implementation of an autonomic adjustment module. This module is responsible for investigating the problem, identifying the root cause and resolving the issue. In case of an SLA violation, the Adjustment module can trigger re-planning, re-configuration and/or alerting to higher-level SLA. These capabilities are considered to be important in order to guarantee best user perception preserving underlining resources. Further details about this feature implementation can be found in the WP A3 deliverable.

**Feature: 3rd Party Configuration**

**Feature details**

This feature requires the capability to set 3rd party service providers to use as candidates for manual or automatic service binding. Each 3rd party provider can have already an associated SLA or not.

**Feature Implementation Status**

This feature relates to the task A3.6 of WP A3 which is scheduled to start in Y2, therefore, implementation wasn’t started on this feature in Y1.
5.3.2 Advances with respect to the state of the art

Compared to the existing approaches being used for SLA management generally and SLA-aware service management specifically, A3 activities have the following innovative advances and outcomes:

Current specification of Service Component Architecture (SCA) doesn’t address the service component behaviour related aspects which are critical in designing predictable systems for large scale complex SOA landscapes. Additionally, non-functional properties are not captured for the service components. Our modelling activities worked on these shortcomings and extended the SCA to take into account these factors.

Dynamic QoS aware binding and composition of services. We adopted an event-based approach to service monitoring, in which monitoring properties, derived from SLAs, are checked against events, e.g. service calls and responses, describing the evolution of the service execution environment. Therefore, besides monitoring engines (i.e. reasoning tools to check the satisfaction of monitoring properties), service monitoring also requires the instrumentation of the service execution environment. Such instrumentation should be able to capture the events relevant for monitoring and to make them available to the core reasoning engines of the service monitoring framework.

A unified manageability infrastructure for SLA-aware service management of SOA based application landscapes. The manageability infrastructure comprises of a unified manageability interface for getting monitoring information from various heterogeneous sources as well addressing the control aspects of these artefacts.

5.3.3 Lessons learned & next steps

The activities within WP A3 were driven by the requirements of the ORC based ad-hoc demonstrator. This demonstrator is a hosted solution for retail shops. This is a rather simplistic scenario; however, it proved valuable to validate the ideas and approaches devised to address various research topics. Starting from Y2, we have to shift our focus to various industrial use cases which come with a large set of heterogeneous requirements. We would apply these approaches to these use cases and would identify these shortcomings in the design choices we have made; however, we are confident that our approaches will prove to be suitable for them. An important aspect of our future work would be the service management of more dynamic scenarios involving multi-providers / multi-domains which might bring out additional research topics to be tackled for WP A3 activities.

5.4 A4- Infrastructure Management

This section provides a summary and assessment of the main scientific results related to work package A4 Infrastructure Management.

5.4.1 Current state

In the first twelve months of the project WP A4 has focused on researching, architecting and implementing a demonstration provisioning system to support the core infrastructure requirements of the use cases in general, and the year 1 adhoc demonstrator in particular. A4’s implementation efforts with respect to each of the SLA@SOI framework infrastructure management features are described below:
Feature: Virtual Hardware Infrastructure Provisioning

A demonstration virtual hardware infrastructure provisioning system supporting the complete life-cycle of an infrastructure request has been implemented. It has been designed to be extensible and flexible, adopts a scalable agent-based architecture, and includes support for both the KVM and Xen hypervisors (virtual machine providers).

Infrastructure offerings can be described via SLA Templates. Arbitrary combinations of arbitrary virtual machines (VMs) can be requested via a Provisioning Request. Suitable nodes to host these VMs are located, and the required VMs are provisioned if possible. The provisioned virtual infrastructure is monitored, with exceptions of desired state reported up the stack for potential adjustment and re-provisioning, or escalation back to the customer. Customer VMs can be torn down when no longer needed.

Feature: Virtual Hardware Infrastructure Monitoring Rules Extraction

The automated extraction and configuration of monitoring rules directly from SLAs is not yet implemented. Rules are manually configured in the current implementation based on the chosen SLA Template.

Feature: Virtual Hardware Infrastructure Observation and Violation Detection

The infrastructure provisioning system includes an advanced infrastructure monitoring sub-system built on Ganglia. This open-source monitoring platform supports the instrumentation of arbitrary low-level physical and virtual infrastructure properties. Instrumentation data is translated into higher level infrastructure metrics, and ultimately used to identify SLA violations and critical states that require infrastructure adjustment. The monitoring is built on top of a flexible publish-subscribe communications infrastructure, currently XMPP based, to support scalability. Arbitrary interested parties can subscribe to relevant monitoring feeds.

Feature: Virtual Hardware Infrastructure Adjustment

Initial support for adjustment of virtual hardware infrastructure has been implemented. When a provisioning request is associated with a previous provisioning request, the system identifies the differences and re-provisions the necessary infrastructure as appropriate. This adjustment is dependent on the host hypervisor adjustment capabilities. For example, adjustments to CPU can be made at runtime by many hypervisors, but adjustments to memory can often only be made offline. None of the hypervisors currently integrated support live-migration, hence virtual machines need to be restarted for some readjustments.

Feature: Virtual Hardware Infrastructure Reporting

The infrastructure monitoring sub-system can inform arbitrary interested parties of relevant events, new measurements and detected violations. However, a monitoring database and historical reporting system has not yet been implemented.
5.4.2 Advances with respect to the state of the art

Infrastructure management has made the following advances beyond the previous state of the art.

Regarding Infrastructure Provisioning, a lightweight technology-independent infrastructure model has been defined, implemented and deployed internally within Infrastructure Management. Although the focus has been on supporting key functional parameters at this stage in the project, the model has been designed to accommodate non-functional parameters also.

Key abstract methods for the interface into Infrastructure Management have been defined, supporting the provisioning, re-provisioning and management of virtualised infrastructure. Although the current implementation manipulates local Xen and KVM hypervisors, the prototype has been architected to allow manipulation of arbitrary internal hypervisors as well as 3rd party cloud infrastructure. SLA@SOI Infrastructure Management is helping to drive the industrial state of the art by being a founding member of the new OGF Open Cloud Computing Interface (OCCI) working group, and has helped draft and progress the definition of a cloud-computing API. At the time of writing, a draft of this standard has already been published for comment.

Regarding Infrastructure Monitoring and Adjustment, a powerful, scalable monitoring infrastructure has been architected and demonstrated. It supports the instrumentation of arbitrary physical (and virtual) hardware parameters, translates low level instrumentation data into higher level infrastructure and then SLA level events, and ultimately identifies SLA violations and undesired states enabling appropriate disposition. Monitoring is built on top of an abstracted communications layer, currently XMPP-based, ensuring scalability, extensibility and making it implementation agnostic.

5.4.3 Lessons learned & next steps

SLA@SOI Infrastructure Management has learned various lessons, not least of which was witnessing at first hand the broad nature and very fast pace of change in this key field. The OGF’s OCCI working group, for example, went from concept to publishing a draft standard within a few short months. It was a privilege for SLA@SOI to be invited to co-chair it.

With the rapid progress that key open-source as well as commercial infrastructure management vendors are making, it is also now clear that SLA@SOI Infrastructure Management cannot compete with the low-level technical developments in these projects, but instead can add most value by supporting and complementing them in its SLA-aware implementations.

SLA@SOI Infrastructure Management is unique in the industry giving the SLA-awareness which is at its core. In particular, as the industry matures, SLA-aware monitoring and management of infrastructure will become an interesting and active topic that SLA@SOI will be well placed to influence and perhaps lead.

To this end, next steps include not just incrementally improving and extending the existing architecture to include non-functional parameter support, real-time prediction and adjustment, and demonstration of third party infrastructure provisioning through collaboration with fellow travellers, but also increasing our engagement with key players in the community such as RESERVOIR (driving the
open source OpenNebula hybrid cloud provider) to help embed the learning’s of SLA@SOI into the broader Cloud Computing movement.

## 5.5 A5 - SLA Management & Foundations

### 5.5.1 Current state

The goal of WP A5 during Y1 was to materialize concepts and features using requirements from the ad-hoc demonstrator. In what follows, we elaborate on the distinct features, and discuss about their current implementation status.

**Feature: Service properties dependencies coding**

**Feature details**

There must be a way to define for different services how they depend on each other. For workflow-encoded composite services, this is usually provided by the workflow itself, but for other services it is not clear how this happens. A proper notation, and perhaps language (schema) is required.

**Feature implementation status**

This is a very complex feature which is the subject of ongoing research. An example of currently produced results is the paper submitted to the NFPSLAM’09 workshop (co-located with ICSOC/ServiceWave’09).

**Feature: SLA definition**

**Feature details**

This most important feature is related to the ability to define SLA templates and respective SLAs, from a modelling point of view, with all those constructs that are required for most of the use cases. Such constructs include, for example, KPIs such as operation invocation completion time, or non-performance-related guarantees such as exclusion terms. The feature is therefore about a complete and coherent model that pre-defines a number of terms, but allows its extension as per the requirements of specific use cases.

**Feature implementation status**

The project has tackled this requirement from the very beginning, in the following ways:

- Definition of the core meta-model (in WP A1) which provides an extensible foundation for term definition
- Definition of the A5 SLA model which provides a complete, abstract (and therefore, extensible) framework where core meta-model terms, business values and constraints may fit in
- Rigorous definition of an extensive set of agreement terms, which are commonly referred to in literature and related work.

The SLA model plays a central role throughout the project, as it will be used internally for all operations, independent of the external protocols and data models used for purposes of interoperability (e.g. WS-Agreement).
Feature: Discovery of Services and Templates

Feature details
It was a design decision from the very beginning, and a requirement from the use cases eventually, to perform service discovery and SLA template discovery in a single step. This feature ensures a consistent mechanism that binds services with their capabilities for guaranteed consumption and as such results in a coherent methodology for customers to discover services based not only on their functional, but also on their non-functional properties.

Feature implementation status
The feature has been implemented to a large degree. A SLA Template registry already exists, and can return templates that also include service description constructs. A Publish/Subscribe system for templates, to be designed and implemented in Year 2, will complement this registry with the necessary functionality for fully distributed operation.

Feature: Automated Orchestration of Custom (Re-)Negotiation Protocols

Feature details
In order to achieve a framework as generic as possible, it must be independent from the various message-sequencing needs of different use cases. As such, it must be able to implement different negotiation protocols, such as single-shot ("take it or leave it"), bilateral negotiations, English/reverse-English auctions, simple tenders, etc. For those, we need concrete descriptions that can be arbitrarily executed by the framework depending on the use case requirements.

Feature implementation status
During Y1 we only implemented a specific single-shot protocol as foreseen by WS-Agreement. The architecture for Y2 already foresees an engine to execute such arbitrary protocols, and a format to describe them is being worked upon.

Feature: Out-of-band SLA registration

Feature details
For certain use cases (e.g. some scenarios within the e-Government use case) there is no negotiation, rather a contract is already given and must be monitored / enforced. For those cases, we need the contracts to be inserted in the registry "out-of-band", that is, without prior negotiation, and the system should be aware and support their further management.

Feature implementation status
This feature is already implemented in a simple manner, by means of emitting an event during the insertion of a SLA into the registry. Using this event, the framework becomes aware of the new SLA (independent of the fact it was not negotiated before), and monitors its implementation.

Feature: Coordination of Provisioning

Feature details
Given a SLA, it may be the case that the required service(s) start immediately, or they may start at some point in the future. The exact consumption period is defined in the SLA. Additionally, a service may depend on other services, that may in turn be foreseen in the SLA, or not. Certain other physical resources are also needed to provision services. The exact knowledge of these requirements, and the temporal constraints/interdependencies, is the subject of this feature. Using such knowledge, the provisioning mechanism will be able to perform provisioning of different required objects in the right order, to ensure success during this process.

**Feature implementation status**

The Provisioning module from Y1 took into account the requirement, nevertheless it assumed knowledge about temporal requirements for making available each separate part of a larger provisioning process (e.g. a software component, a physical resource, etc). In Y2 and beyond we assume that this is not realistic, given the vast amount of possible hardware configuration and external factors that come into play and affect these times. Therefore, we will provide a much more deterministic process for the provisioning phase, which assumes an initial triggering, and knowledge only about the specific series for provisioning (i.e. what part of the system needs to be made available before another one).

**Feature: On-the-fly Deployment of Monitoring**

**Feature details**

A generic monitoring system for various different services/processes assumes the capability to adapt itself to these different requirements, at the very least in the lower levels of raw message capturing. This feature refers to this capability, where based on pre-specified rules and policies it becomes possible to dynamically compute and deploy the monitoring framework required to successfully monitor a service, taking into account its type, location of execution, etc.

**Feature implementation status**

The SLA4M Monitoring Manager has already been defined as an architectural artefact, and work is ongoing for enabling it with the computation of the required monitoring framework and the coordination of its deployment.

**Feature: Automated SLA Enforcement Coordination**

**Feature details**

A SLA management framework requires an entity addressing the need for taking automatically some (customisable) action when the SLA is violated. This feature refers exactly to this requirement. A typical example is the case where some software or hardware reconfiguration is necessary (service re-provisioning), while further actions may include re-negotiation of a service.

**Feature implementation status**

The Adjustment module is by definition looking into this issue. Re-provisioning services is tightly coupled with re-negotiation (as a fallback measure), and therefore it will be addressed in Year 2. In Year 1, adjustment is taking place mostly by means of software re-configuration. In general we assume an autonomic set-up, where independent agents process monitoring data and take fast decisions on the course of action.
Feature: Monitoring Coordination

Feature details
Following the deployment of monitoring infrastructure as part of the provisioning cycle, and given an executing service, there must be some coordination of monitoring activities for collecting, aggregating and consolidating monitoring information in a manner that makes it useful to final recipients (e.g. Adjustment module). This also includes all those mechanisms and the necessary engineering for achieving higher-level requirements, e.g. authorization methods for access to monitoring information by customers and 3rd parties.

Feature implementation status
The feature has been implemented to a limited extent in Y1, taking into account the requirements from the ad-hoc demonstrator. Messages received were processed to identify possible violations, and then forwarded to the demonstrator for purposes of visualization. Follow-up iterations of the framework will extend the approach to produce a more generic monitoring framework.

Feature: SLA Status & Resource Reporting

Feature details
During the execution of a SLA, but also after the end of its validity, the service providers (and possibly the customers too) need to know various statistics about it. Examples of such statistics are the numbers of times it was in violation or warning states, the utility (e.g. profit) produced, etc. Additionally, the provider typically needs to see how the SLA affected its resource consumption over the period it was applicable, usually in order to do some resource planning for the future. This feature refers to these requirements to extract useful statistical information from the SLA history of a provider or a single service.

Feature implementation status
This feature was not implemented in Y1 in a coherent way, rather as part of the ad hoc demonstrator and its real-time monitoring dialogs. In Y2, the SLA management console (foreseen to start design and implementation on M12) will take these requirements into account and produce necessary software artefacts.

1.1.1 Advances with respect to the state of the art
During Y1 we performed an exploration of the problem space, by means of using the ad-hoc demonstrator scenarios. Although the mechanisms and methods introduced were fairly simple due to the simplicity of the example, there were still significant contributions to the state of the art, the most important of which is the SLA model of task TA5.1. More specifically, this model started from the conceptual requirements and brought together ideas from multiple areas and prior art (e.g. WS-Agreement). To our best knowledge, this is the most elaborate, detailed and complete data model for describing SLAs. Our ongoing work in the area concerns producing an abstract syntax starting from the conceptual model, that would allow multiple renderings for XML, Java, or other representations.

Additional smaller contributions include the template registry implementing highly complex discovery of templates, the porting of a WS-Agreement implementation into a non-Web Services environment for testing purposes, and preliminary research results in the area of SLA translation and planning.
1.1.2 Lessons learned & next steps

The adhoc framework (and the ORC/adhoc-demonstrator) made much clearer the complexity of the problem, the different facets and the gray areas which are understood in much different ways by different groups and disciplines. The results of this industrial use case analysis have been important in understanding what the users expect from a framework such as the one that SLA@SOI is building, and we will use them as guidance for the upcoming architectural update, as well as implementation in Year 2. Certain requirements (such as having monitoring defined in the agreement as a contractual term) had escaped our radar, while in other cases we foresaw certain needs and have already provided solid footing for complete solutions (e.g. with SLA-template-based service discovery).

In Year 2, as efficient negotiation schemes will be implemented, and with many of the conceptual obstacles having been resolved, it will be possible to research further into tightly coupled issues such as planning and optimisation, autonomic management for adjustment/re-provisioning, etc.

5.6 A6- Predictable Systems Engineering

5.6.1 Current state

Feature: Design of QoS Predictable Systems

Feature details

Provides a way to predict performance and reliability of a software architecture. An architectural modelling language enables the specification of (software) service components, along with their interfaces, composition, usage, and deployment to underlying infrastructure. As the approach builds upon a model of the architecture, it does not depend on the implementation itself, nor on the actual execution of services. Thus, it is applicable already in early phases of the development process, namely design-time of services and systems. The approach may be used by software and service providers to determine feasible service offerings in terms of performance and reliability, as well as support of negotiation between a service provider and a customer.

Feature implementation status

In year 1, a QoS meta-model for service components has been developed, which is focused on software performance (response time, throughput, and resource utilisation) and – as a distinct feature - parameterised over the usage profile (input data and workloads). The QoS meta-model supports abstraction from unknown parts of the architecture through stochastic expressions (requirement #204) and prediction of QoS attributes for composite services (requirement #149). Software Developers are supported in creating the QoS-enabled architecture models through a comprehensive graphical editor based on Eclipse Rich Client Platform. To predict performance, we generate custom simulation code, which is fed into an event-based simulation framework for performance prediction based on the SSJ framework (Stochastic Simulations for Java) (requirement #182). We developed this framework as part of the Eclipse-based graphical editor, so it is fully integrated in the development environment. To demonstrate the approach, we created QoS-enabled architecture models for the ORC and simulating them, supporting two deployment options.
Moreover, we focused on the design and implementation of a prediction service, which allows to use the (generated) prediction models through a standard-based interface based on web services technologies. This included the detailed design of interfaces, data models, interactions, data flow) and their alignment with overall SLA@SOI architecture (in particular the negotiation module).

**Feature: Runtime Infrastructure Metrics Prediction**

**Feature details**
Provides a means to forecast infrastructure resource demands at system runtime. Infrastructure providers may use this information to (i) anticipate potential infrastructure SLA violations, to (ii) avoid such violations by taking countermeasures (re-provisioning of resources), and to (iii) save costs by temporarily suspending resources in case of low resource utilization.

**Feature implementation status**
In year 1, a matrix has been generated comparing algorithms based on a review and analysis of about 20 relevant technical papers and a PhD thesis. On the grounds of these comparisons and the requirements for run-time prediction, many improvements were made to three existing algorithms, each of which is suitable for a specific usage scenario. The improvements targeted two drawbacks of the existing algorithms, low prediction accuracy and inability to perform long duration step prediction. The experimental results on datasets from the SLA@SOI ORC and an internal Intel Batch Job pool demonstrated the effectiveness of the improvements and the capability of relatively long duration prediction of the three improved algorithms. This work has helped address the runtime analysis needs of requirement #182, as well as delivering the optimised prediction algorithms necessary to fulfil requirement #204.

**Feature: Runtime Software SLA Violations Prediction**

**Feature details and implementation status**
Regarding run-time prediction at service level, we focused on designing a monitoring framework for collecting information about the functional and non-functional behaviour of the respective business services (i.e. Quality of Service, QoS) and their implementing components. This information is used for predicting the services' non-functional properties in the near future. Moreover, we have been studying techniques, approaches and methodologies so far developed. This allowed us to have a clear state of the art as starting point for our investigations.

**Feature: Manageability Design**

**Feature details and implementation status**
A manageability metamodel is available, which supports the configuration of monitoring capabilities for systems designed on basis of the SCA assembly model. The metamodel is powerful enough to express monitoring capabilities required for validating predicted completion times for the whole assembly or single parts of it in relation to the usage profile (requirement #182).
5.6.2 **Advances with respect to the state of the art**

**Feature: Design of QoS Predictable Systems**

The possibility to compose QoS (prediction) models for composite services / applications in combination with the usage profile decomposition supported by the prediction service / the underlying event-based simulation framework represents a clear advance with respect to the state of the art.

**Feature: Runtime Infrastructure Metrics Prediction**

Regarding run-time infrastructure level prediction, the three improved algorithms achieved significant advances in terms of prediction accuracy and long duration step prediction. Compared with the three unimproved counterparts, the three improved algorithms respectively achieved 14.0%, 94.8% and 7.0% prediction error reduction on average on ten experimental datasets for one time step prediction. In terms of long duration step prediction, two improved algorithms made predictions for 10 time steps and achieved respectively 28.41% and 23.42% prediction error on average on three datasets; the third algorithm made predictions for 100 time steps and got 19.57% prediction error on average on two datasets.

All the improved algorithms show the effectiveness compared with their counterparts on most datasets. For short duration prediction, improved MPI would be a good choice. For long duration range prediction, improved PP would be the best option when the resource usages have periodic property. When more workload attributes can be used as inputs, it can be expected that improved FLC will achieve better performance.

**Feature: Runtime Software SLA Violations Prediction**

Work on this topic mainly starts in year 2. The envisioned solution will contain the following features, which reach beyond the state of the art (as investigated in year 1):

- Combines observation about system parameters and system interactions. At run-time, a system control flow depends on values exchanged during the system interactions. We want to capture this interplay in a model that combines time, values, and interactions.
- Does not require a training set. The model is computed on the fly and it is refined at run-time.
- Introduces the notion of genuineness of an observed event. We start from the point that not all we observe is genuine. In the context of distributed systems we cannot assume a message sent over the network truly comes from an expected sender. It could also be generated from a malicious intruder (man in the middle) or it might be altered.

**Feature: Manageability Design**

Regarding the manageability design, a couple of generic models like the Metrics part of the Common Information Model and the Application Response Measurement Framework exist. All of them would be applicable for the scenario we consider. However, due to the generic nature of these models a developer has little guidance on how to create meaningful information models. A meta-model
that is tailored to monitoring SCA-based applications would lead to a reduction of complexity for the developer. This, however, was missing so far.

### 5.6.3 Lessons learned & next steps

By the end of the first project year, it was not fully clear if the prediction services developed by the A6 work package would fit to the requirements of industrial use cases. As a consequence, discussion with A-line work package leads (technical consumers of prediction services), as well as B-line industrial use cases (conceptual users of prediction results) was intensified, and within the revision of the overall requirements gathering process, a much better coverage of A6 features by use case requirements could be achieved.

Another issue that became obvious during year 1, was that the description of work of the A6 work package and its tasks is not fully in line with the project needs. For example, the creation of a holistic prediction service as suggested by task A6.4 turned out to be not feasible and does not add value compared to the realization of individual prediction services, as all of them have different goals, are applied in different stages of the software life-cycle, and used by different stakeholders. Instead, a uniform view onto the A6 contributions will be achieved by elaborating common concepts that underlie all prediction services. A corresponding update of the description of work will be submitted as part of a currently prepared DoW amendment.

The following paragraphs describe planned next steps per feature.

**Feature: Design of QoS Predictable Systems**

The design-time prediction service so far is limited to performance-related QoS properties for (service) components. Regarding these properties, a system’s behaviour can be accurately predicted. As a prerequisite, the QoS model for the single atomic components/services have to be well calibrated on basis of runtime measurements, while the composition of these atomic components and the support for varying usage profiles can be accomplished without further measurements. The required calibration methodology will be refined and documented in year 2.

Moreover, we are planning to extend the approach by further QoS properties, namely availability and reliability. From initial research and results on this issue we learned that discrete time Markov chains are well suited as underlying prediction models. Moreover, we verified that existing implementation and conceptual work available for the performance prediction can be reused to a large degree.

**Feature: Runtime Infrastructure Metrics Prediction**

Regarding run-time infrastructure level prediction, it was verified by experiments that machine learning algorithms can be adopted. These algorithms train and learn predictive models using historical monitoring data, and then use the model for prediction of future resource usage. Currently we only use individual metrics to build predictive models. In the future we will take into account more useful metrics, such as workload and job attributes and design time information to learn more accurate models. Besides, now we can make 10 time step prediction on most datasets and 100 time step prediction on some datasets having periodic property with relatively high accuracy. Next we will continue exploring solutions for long duration step prediction on common datasets. And also, we will
investigate what accuracy and how many time steps is needed for run-time prediction at infrastructure level. Finally, we are also planning to integrate run time prediction with other prediction services, in particular design time prediction.

**Feature: Runtime Software SLA Violations Prediction**

Currently, run-time prediction at service level does not take into account the interplays that take place between time, values and interactions. Thus, a novel model and approach is needed in order to overcome the current limitations. Moreover, since SLA@SOI deals with distributed systems key issues regarding the trustworthiness of exchanged events and the management of possible delays in their transfers should be taken into account.

Next steps include actual implementation of the approach and the creation of necessary data models, after the SotA analysis has been performed in year 1.

**Feature: Manageability Design**

Manageability design so far is limited to monitoring capabilities and has to be extended in a way that control capabilities can be specified or configured as well. Moreover, the configuration models so far have to be translated manually into corresponding implementations, including manageability infrastructure and instrumentation. This is a very complex and time-consuming task. Thus, the automation of this task (as far as possible) is a major concern for the remaining time of the project.
6 Requirement Evaluation

Here we present our plan for the evaluation of the project results.

The evaluation process has the goal 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) how the demonstrators developed with the developed framework provide business value to the committing stakeholders (both industries and non-profit organizations).

In describing this evaluation plan we first outline the process by which a requirement gets turned into software artefacts. This process has been described in detail in section 3, but we present a summary of the characteristics necessary for the establishment of metrics for requirements implementation. SLA@SOI will follow this implementation plan twice, once for the year two iteration of the implementation, and once for the year three iteration. We will revise this requirement evaluation process as necessary for its second iteration based on the experience of the first iteration.

In the next section we describe the evaluation process and in the following sections the metrics for the technical evaluation and the business evaluation.

6.1 Evaluation Process

Here we explain the process by which requirements are evaluated with respect to the requirements lifecycle. We divide the requirements lifecycle into four sequential phases which we denote as Specification, Design, Implementation, and Integration. We depict this process in Figure 10: Requirement Evaluation Process on page 71.

During the specification phase the requirements are derived from the use case scenario.

The design phase occurs after a number of use case requirements have been successfully communicated from the B-line use cases to the A-line workpackages.
The preconditions of a successful communication of an individual requirement are:

1. a shared understanding of the requirement by all B-line industrial use cases in the form of written requirements within a single TRAC ticket
2. a prioritisation of the requirement by each B-line industrial use case as to the relative necessity of the fulfilment of the requirement for the development of their respective prototype
3. a specific A-line work-package has been assigned responsibility for the fulfilment of this requirement
4. this specific A-line work-package has acknowledged that it has a sufficient understanding to begin design of the software necessitated by the requirement based on the written description associated with the TRAC ticket

With these pre-conditions satisfied, the A-line work-package begins the process of designing the software by identifying one or more features that will fulfil the requirement. Since requirements are not considered in isolation but as a group there will probably be a necessarily complex inter-relationship between the set of features that satisfy the set of requirements. Accordingly, we initially assumed a many-to-many ("m:n") relationship between requirements and features. After reviewing initial plans for this proposal, however, we felt that this many-to-many relationship would be too complicated to track and would provide ambiguous information. Instead, therefore, we introduced an additional step within the consolidation phase whereby requirements would be further refined and split so that each requirement would be associated with precisely one feature. At the end of the design phase the following preconditions for the implementation phase will have been met:

1. each requirement will be identified as corresponding to one feature
2. an implementation plan for the features will exist with a planned timeline and the necessary resources
3. design artefacts exist which communicate the plan for turning the requirements into software
4. a notion of how the testability of the features should occur is specified

After these preconditions have been met, the implementation phase of the requirements occurs. Of importance to the evaluation of the requirement implementation is the testability of the features delivered for each requirement. For each feature one or more tests is created to provide a measure of the success of the software in its implementation. These tests are constructed as unit tests which mean they are amenable to inclusion in an automated suite of tests. The initial source of the tests comes from the implementing A-line work-package which specifies the functional aspects of the software. After the implementation has proceeded far enough for the architectural APIs of the software to be stable, we anticipate that the B-line use cases needing the feature will also provide unit-tests. All tests will be included in the shared source tree.

At some point in the implementation process, the feature will be declared ready for its integration into the requesting use-cases. After an suitable initial period of integration effort by the use case partner (currently thought to be about two weeks), we will ask each of the use-cases to fill out a short survey to provide a point to collect information on the evaluation of the requirement implementation that need subjective feedback.

We expect that at each phase change (i.e. from Specification to Design, or from Implementation to Integration), there will not be such a neat and cleanly definable transition as presented in the diagram. We anticipate that there will be a fair amount of iteration between the two states, as the requirement or its implementation is presented, evaluated by the receiving party, and then the reworked to satisfy the revised information.
6.2 Scientific Metrics Evaluation

Following the Goals, Questions, Metrics methodology [1], we now present the derivation of the metrics that will be used in the evaluation of these requirements from the scientific work-packages. We start by listing the high level goals we have defined for the evaluation process in Table 2: Requirements Evaluation Goals on page 73.

<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 2: Requirements Evaluation Goals

For each of these goals we now provide questions whose answers provide information as to the satisfaction of the goal. For each question, we provide one or more metrics to provide answers for these questions. The metrics fall into one of three categories

1. Metrics whose measurement is embedded in the requirements process based on the customized TRAC-based we call Process-based metrics. The formula for calculation is given in the text below.
2. Metrics whose measurement is Testing-based on the performance of software features we detail in 6.2.1 below.
3. Metrics whose measurement is the result of the short set of questions each use case answers after a period of integration are known as Survey-based we provide forward references to the survey. We present the survey in 6.2.2 below.

G1: Intelligibility

By the goal of intelligibility we denote the need that requirements be understood by all parties involved in the process of their specification and fulfilment. Without understanding it is quite likely that requirements will not be implemented in a manner useful to the requesting party.

Q1.1: Is the requirement intelligible to all of the B-line use cases?
The B-line use cases are firmly placed in differing application domains, yet are expected to provide joint requirements. Since the requirements need to be identified as shared between use cases where possible, we evaluate the intelligibility by other use case partners.

M1.1.1: We use the assignation of per use case priorities as an indication of the intelligibility of the requirement, since this cannot be set if a given use case has not understood the requirement enough to produce a ranking.
Per requirement #(use cases that have set a priority) / 5. This rational value ranges from 0 (bad) to 1 (good).

Q1.2: Is the A-line responsibility for the implementation of the requirement understood?
M1.2.1: We use the assignment of a responsibility of per A-line work-package as the metric for whether the responsibility has been understood, as the acceptance of a requirement by an A-line work-package is a volitional step indicating that the requirement has been understood.
It has is a Boolean value, with true indicating that such an assignment has been made, false indicating that it has not been made.

Q1.3 Does that A-line work-package understand the requirement enough to implement it?
M1.3.1: When the requirement has been assigned to an A-line work-package, and that work-package provides identification with a feature, this indicates an understanding of the requirement.

G2: Monitoring

The progress of the implementation of a requirement must be able to be continuously monitored. This is an explicit requirement in SLA@SOI description of work. By requiring this goal out of our evaluation process, we gain a tool that can assess the progress of our shared work that can assist in the identification of problems early enough for corrective action to be undertaken.

Q2.1 How does the code associated with the feature stem from the requirement perform with respect to the unit tests?
M2.1.1: Measured by Testing-based

Q2.2 At which stage of the implementation life-cycle is the feature which satisfies the requirement?
M2.2.1 Each requirement will be positioned at one of the four steps in the lifecycle indicating the progress towards implementation.
We assign an integer for each requirement: 1 for specification, 2 for design, 3 for implementation, 4 for integration and 5 for completion.

G3: Traceability

The requirements should be traceable during the Specification, Design, Implementation, and Integration phases with the artefacts associated with their fulfilment. The goal of traceability assists in the comprehension of the resulting software architecture. If trade-offs between implementation choices need to be made, precisely knowing the requirements comprising the feature allows the of framework of impact of these trade-offs to be assessed.

Q3.1: Is the requirement traceable to a feature?
M3.1.1: For each requirement, the result of the consolidation portion of the design phase will result in a unique feature.
This process-based metric shall consist of a Boolean value as to whether the requirement has an associated feature.

Q3.2: Which software components are necessary to implement each feature?
M3.2.1: For each feature, one or more software components will be associated with the implementation of that feature. In order to maintain the traceability back to the original requirement we require the association to be maintained.
For every identified software component we will keep track of the feature and its associated requirements.

**G4: Implementation**

The goal of a requirement is the successful implementation as a software component that satisfies the requirement in the requesting application context.

Q4.1: Does the implementation function as was expected in the requirement?
- M4.1.1 Measured by survey question 2.
- M4.1.2 Measured by success of tests.

Q4.2: Does the implementation introduce unintended complications arising from its implementation choice when integrated with the use case?
- M4.2.1: Measured by survey question 3.

### 6.2.1 Testing

Following the contemporary discipline of automated testing for software development, unit tests will be used during the software implementation process. Initially the tests will be provided by the A-line developers. As the software matures in the course of its implementation, it should be possible for the eventual B-line recipients to contribute tests that provide some sort of domain-specific constraint on the performance of the delivery of the software.

The tests will be able to be run from the source code tree via a regularly occurring automated procedure ("batch job") whose reports can be retained for subsequent analysis. Since the tests are under the same revision control as the source code, it will be possible to reproduce the results for any given historical snapshot from the source tree if necessary (i.e. if the automatic procedure fails for some reason.)

This testing procedure forms a part of the overall quality control procedures enacted as part of the A1 work-package, so are not imposing a heavy-weight evaluation procedure here. The additional work that needs to be done is that each test needs to be associated with at least one feature of the software. If a test has a plausible interpretation as testing more than one feature of the software, we will allow multiple associations. This assignment will be recorded by some sort of test source-code level annotation convention which will be extractable by the reporting mechanisms present in the UNIX shell3. A suitable feedback mechanism for these assignments will be added to the requirements process by which the interested parties can verify this designation.

At a weekly interval, the tests shall be run, and the percentage of passing/failing tests per feature recorded. The number of tests will not be constant during the implementation process. New tests will be introduced to further constrain the behaviour of the component as more experience is gained in implementation. Old tests may be removed as no longer relevant in face of this experience. Consequently percentage of per-feature passing tests will not be strictly increasing, but the overall trend will provide information on the progress towards implementation.

---

3 That is by shell scripts using `grep(1)`, `awk(1)`, `sed(1)` and so-forth.
6.2.2 Survey

We now provide the current version of the survey to be completed by each use-case. The intention is to provide a light-weight mechanism to collect information for evaluation of the requirement. We expect to implement the survey via an online mechanism using a free tool such as SurveyMonkey4.

Each statement about the provided feature shall be commented on by replying with an integer from 1 to 5 with 1 indicating “Complete disagreement”, 2 indicating “Somewhat disagree”, 3 indicating “Neither agree nor disagree”, 4 indicating “Somewhat agree”, and 5 indicating “Complete agreement”.

<table>
<thead>
<tr>
<th>#</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The documentation accompanying the feature was sufficient to integrate it into your use-case.</td>
</tr>
<tr>
<td>2</td>
<td>The feature implements the requirement functionality.</td>
</tr>
<tr>
<td>3</td>
<td>The feature was easy to integrate introducing little or no unexpected complexity.</td>
</tr>
</tbody>
</table>

Table 3: Evaluation Survey

6.3 Business Metric Evaluation

As part of their specification, each of the Industrial Use Cases has detailed the areas of business value contribution expected of the SLA Management Framework. A common failure in measuring the business value of IT enabled solutions, is the inability to uniquely correlate measurable low-level benefits such as response time improvements, availability, etc., with higher level business objectives which have a more obvious relationship with higher level objectives such as profitability, competitiveness, etc. To avoid this issue, the Industrial Use Cases have followed a consistent approach, identifying specific and measurable business metrics which evaluate performance against specific business value dials, which in turn align directly with formal objectives of key actors in the use case scenarios.

The objectives (mainly business objectives) and metrics listed are not necessarily the full list of objectives and metrics of the actors involved, but are selected based on their likelihood of demonstrating positive impact based on the introduction of SLA management framework features.

This use of metrics which align with formal objectives, reinforces the credibility of a business value impact claimed as part of the evaluation.

As the Use Cases progress to design and implementation stage, it will become clear which business metrics can be evaluated objectively through unit tests, and which will require an expert judgement based on performance against (indirectly) related tests. It is assumed that contribution to all of the business metrics can be meaningfully judged through the use case evaluation.

The following tables contain these business metrics as an extension of the business objectives already listed in the use case specification documents.

4 http://www.surveymonkey.com/
As an example, in the case of the ERP Hosting Use Case (WPB3), one high level objective relates to Dynamic Service Provisioning. Within this objective, three specific Value Dials have been identified: Ease of Service Consumption, Dependability, and Flexibility. In turn, each Value Dial has been assigned measurable indicators or metrics which are listed also. By evaluating positive impact on these metrics, a claim for business value is credible because the metrics clearly align to the objectives of the actor in the use case scenario.

**Use Case B3 : ERP Hosting.**

<table>
<thead>
<tr>
<th>Business Objectives</th>
<th>Value Dials</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynamic Service Provisioning</td>
<td>Ease of Service Consumption</td>
<td>Reduced Time to Market</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduction in human assistance for configuration required to consume service if manual</td>
</tr>
<tr>
<td></td>
<td>Dependability</td>
<td>Availability</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mean time to recover</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Resilience</td>
</tr>
<tr>
<td></td>
<td>Flexibility</td>
<td>reduction in customisation and intervention required to adapt services, interfaces and to facilitate consumption</td>
</tr>
<tr>
<td>Efficiency</td>
<td>Energy Efficiency</td>
<td>Reduction in energy related costs incurred: direct energy costs and utility costs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduced Cost per unit of work</td>
</tr>
<tr>
<td></td>
<td>Technical Efficiency</td>
<td>Reduction in technical costs incurred: licensing, engineering and configuration overhead of applications</td>
</tr>
<tr>
<td></td>
<td>Operational Efficiency</td>
<td>Reduction in operational overhead: configuration, problem resolution, general management.</td>
</tr>
<tr>
<td>Transparent Service Management</td>
<td>End2End manageability</td>
<td>increased visibility - scope and precision - and control (or decreased cost of control) for full scope of services from consumed to provided.</td>
</tr>
<tr>
<td></td>
<td>Service Governance</td>
<td>improved robustness and decreased cost of governance objectives: change management, auditable logging, access control.</td>
</tr>
<tr>
<td></td>
<td>Agility</td>
<td>Reduction in time taken / effort required to reconfigure services and service lines in response to customer demand trends.</td>
</tr>
</tbody>
</table>

**Table 4: Business Metrics of Use Case B3**

**Use Case B4 : Enterprise IT**

<table>
<thead>
<tr>
<th>Business Objectives</th>
<th>Value Dials</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>IT Enabling the Enterprise</td>
<td>Agility</td>
<td>reduction in time taken / effort required to reconfigure services and service lines in response to customer demand trends.</td>
</tr>
<tr>
<td></td>
<td>Dependability</td>
<td>Service availability, mean time to recover, resilience, privacy and data security.</td>
</tr>
<tr>
<td>Automated Response</td>
<td>Reduction in issues not requiring escalation (automated) reduction in effort to finally resolve escalations (semi or non automated)</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>IT Efficiency</td>
<td><strong>Agility</strong> reduction in time taken / effort required to reconfigure services and service lines in response to customer demand trends.</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Energy Efficiency</strong> reduction in energy related costs incurred : direct energy costs and utility costs</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Utilisation Efficiency</strong> average utilisation across live server capacity / network average deviation from targeted optimum utilisation</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Cost effectiveness</strong> total spend per unit revenue</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Operational costs</strong> reduction in operational overhead : customisation, configuration, problem resolution, general management.</td>
<td></td>
</tr>
<tr>
<td>Governance - investment and technology adoption</td>
<td><strong>Control of resources</strong> improved robustness and decreased cost of governance objectives : change management, auditable logging, access control.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fine grained investment planning for optimal business result Precise alignment of infrastructure investment profile to revenue generating business processes. Precise correlation of investment requirements relative to achievement of business objectives</td>
<td></td>
</tr>
</tbody>
</table>

**Table 5: Business Metrics of Use Case B4**

**Use Case B5: Service Aggregator (1)**

<table>
<thead>
<tr>
<th>Business Objectives</th>
<th>Value Dials</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Infrastructure</td>
<td>Market Opportunity</td>
<td>% confidence in extrapolating trial results for new services, in determining demand and profitability for new services.</td>
</tr>
<tr>
<td></td>
<td>Efficiency</td>
<td>% reduction in engineering, operational, technical and energy costs</td>
</tr>
<tr>
<td></td>
<td>Scalability</td>
<td>% reduced cost and effort to scale out services ability to scale out services dynamically</td>
</tr>
<tr>
<td></td>
<td>Systematic Planning</td>
<td>judged improvement in delivery / reduction in cost of service delivery as a result of systematic planning</td>
</tr>
<tr>
<td></td>
<td>Dynamic Adjustment</td>
<td>reduction in time taken / effort required to reconfigure services and service lines in response to customer demand trends.</td>
</tr>
<tr>
<td>Service Creation</td>
<td>Market Opportunity</td>
<td>% confidence in extrapolating trial results for new services, in determining demand and profitability for new services.</td>
</tr>
<tr>
<td></td>
<td>Agility</td>
<td>reduction in time and effort and other costs to deliver new services to market, to make adjustments e.g. pricing, to enhance or to discontinue services</td>
</tr>
</tbody>
</table>
Industry Standards | reduction in engineering (integration) and operational overhead resulting in standardised interfaces
---|---
Service Management | Multi-party monitoring | judged advantage gained through monitoring of third party services - contractual leverage, confidence of delivery of end customer service, etc.
| Multi-party Aggregate reporting | judged advantage gained through reporting of third party services - contractual leverage, confidence of delivery of end customer service, etc.
| Construction/enforcement of multi-provider SLAs | judged advantage gained through holistic management of multi-party SLAs with providers of third party services - contractual leverage, confidence of delivery of end customer service, etc.

**Table 6: Business Metrics of Use Case B5 (1)**

**Use Case B5 : Service Aggregator (2)**

<table>
<thead>
<tr>
<th>Business Objectives</th>
<th>Value Dials</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Management</td>
<td>Business control (rules and policies definition)</td>
<td>judged advantage from definition of business rules and policies, in context of holistic SLA management framework.</td>
</tr>
<tr>
<td>Fast decision making</td>
<td></td>
<td>reduction in time and effort and other costs to make adjustments e.g. pricing, to enhance or to discontinue services</td>
</tr>
<tr>
<td>Time to market reduction</td>
<td></td>
<td>reduction in time and effort and other costs to deliver new services to market</td>
</tr>
<tr>
<td>Services control based on the monitoring of Telco and Third Party services</td>
<td></td>
<td>judged advantage gained through monitoring of third party services - contractual leverage, confidence of delivery of end customer service, etc.</td>
</tr>
</tbody>
</table>

| Customer Satisfaction | Confidence in the performance of consumed services | judged or sampled confidence of end users in delivery and performance of services delivered to end users |
| Perceived difference in quality (customer views) | | judged or sampled confidence of end users in delivery and performance of services delivered to end users |
| Delivery of products that have guarantees because we have an SLA model behind, and we cannot fail the customer | | % reduction (elimination to zero) of undetected SLA violations. judged improvement in confidence of service delivery based on contribution of SLA framework. |

| New markets and environments | Differentiation from competition | judged total competitive advantage (time to market, efficiency, quality) based on performance against all metrics combined |
| Reach new market niches and the long tail | | % confidence in extrapolating trial results for new services, in determining demand and profitability for new services. |
| taking advantage of the collective intelligence because Telefonica sells third party services | | judged advantage from intelligence and expertise gained from sale of third party services |
New b2b environment to do business with third parties | efficiency gains and any other judged advantages (ease of engaging additional business, etc) from the use of the SLA management framework as an e-contracting mechanism

| efficiency and optimisation | internal resources optimisation based on customer demand | % reduction in engineering, operational, technical and energy costs due to demand-based provisioning

| on demand provisioning of services and infrastructure resources (energy and operational) | % reduction in engineering, operational, technical and energy costs due to demand-based provisioning

**Table 7: Business Metrics of Use Case B5 (2)**

**Use Case B6: e-Government**

<table>
<thead>
<tr>
<th>Business Objectives</th>
<th>Value Dials</th>
<th>Metrics</th>
</tr>
</thead>
</table>
| citizen             | service dependability | Perceived: % of improvement of sampled evaluation of dependability of delivered service from the point of view of the citizen, gathered by repeated surveys both online and offline (e.g. Evaluation against Likert scale).
|                     |             | Objective: % of improvement of measured performance of service relative to minimal requirements in SLA |
| rights knowledge    |             | Objective: % of citizens aware of their right about service quality measured by the number of citizens accessing the SLAs. |
| service quality transparency |             | Objective: % of citizens aware of the objective quality of the services they are using, measured by the number of citizen accessing the SLA status report. |
| user feedbacks leveraging |             | Objective: % of citizens reusing the same service, that improve their feedback with respect to the previous use. Perceived: Judged (by governance) usefulness of these feedbacks in managing performance and selection of service providers, measured by periodic survey. |
| user preference matching |             | Objective: % of service requests containing a preference selection Objective: % of resulting services delivered, fulfilling this selection request. |
| Integrated offer of services |             | Perceived: judged time saving and efficiency delivered to the citizen, gathered by survey both online and offline (evaluation against Likert scale). |
| government          | service chain transparency and bottlenecks awareness | Objective: % Number of SLAs of the composite service shown in the report for the governance / total number of SLAs of the composite service |

*Deleted: 85*  
*Deleted: 91*  
*Deleted: 85*
<table>
<thead>
<tr>
<th>Service Supplier</th>
<th>Business Metrics of Use Case B6</th>
</tr>
</thead>
<tbody>
<tr>
<td>resource allocation efficiency</td>
<td>Objective: % of improvement in utilisation of resources (directly managed and contracted) achieved through the automated aggregation of resources;</td>
</tr>
<tr>
<td></td>
<td>Objective: % of improvement of reallocation of resources, described by the reduction in the average time required for the reallocation of resources.</td>
</tr>
<tr>
<td>cost effectiveness</td>
<td>Objective: % of expenditures tracked by KPIs;</td>
</tr>
<tr>
<td></td>
<td>Objective: % of reduction of cost per instance of service provided.</td>
</tr>
<tr>
<td>citizen needs awareness</td>
<td>Objective: % of delivered services followed up with a feedback evaluation.</td>
</tr>
<tr>
<td></td>
<td>Objective: % of requests with positive feedback from those in which the citizens preferences have been satisfied.</td>
</tr>
<tr>
<td>service supplier</td>
<td>SLA compliance and performance awareness</td>
</tr>
<tr>
<td></td>
<td>Objective: % of supplier KPIs published automatically, % reduction on time for publication.</td>
</tr>
<tr>
<td></td>
<td>Perceived: benefit to prioritization of services from supplier perspective.</td>
</tr>
<tr>
<td>resource allocation efficiency</td>
<td>Objective: % of improvement in utilisation of resources (directly managed and contracted) achieved through the automated aggregation of resources;</td>
</tr>
<tr>
<td></td>
<td>Objective: % of improvement of reallocation of resources, described by the reduction in the average time required for the reallocation of resources.</td>
</tr>
</tbody>
</table>

**Table 8: Business Metrics of Use Case B6**

Combining this structured approach with the diversity of our Industrial Use Case scenarios, we expect to precisely and credibly evaluate the potential industrial impact of the SLA Management Framework.
7 Conclusions

7.1 Summary

After introducing the process followed to collect SLA@SOI framework requirements, we described the result of such a process. The requirements have been organized in 42 features, classified according to different criteria, including the phase of the service lifecycle in which they are considered, the effected layer, the involved user type, and the main scientific work package that will take care of their implementation.

Each scientific work-package has reported the current status of implementation of each requested features and the related scientific result in terms of advancement with respect the state of the art.

We discovered that in spite of the fact that the first year of development was driven by the needs of the ORC, several feature are already partially implemented, therefore the Y1 results are a good starting point to satisfy the needs of use cases to be implemented in Y2.

Finally we described the evaluation process that will be used in Y2 to assess the results of the project form the technical and business point of view.

7.2 Outlook on Future Work

In the next months the A-Line will release the definitive design of the framework architecture for Y2 and then the corresponding implementation. During this months it will be important to continue the discussion between A-Line and B-Line on the design of the components and interfaces implementing the identified features.

As described in the section on the evaluation process the next important step is the definition of the automatic software tests needed for the automatic evaluation of the software results with respect to the technical requirements.

Another fundamental step will be the organization and allocation of all the resources needed to perform the surveys for the assessment performed by human experts both from the technical and business point of view.

The results of the evaluation of the Y2 framework and the lessons learned during the development of the applications will be the base for updating the requirements to consider in Y3.
8 References


Appendix A: Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>NGOSS Shared Information and Data model</td>
<td>A set of comprehensive standardized information definitions, developed by TMF, acting as the common language for building easy to integrate OSS (Operational Support System) and BSS (Business Support System) solutions.</td>
</tr>
<tr>
<td>TeleManagement Forum</td>
<td>An international non-profit industry association focused on improving business effectiveness for service providers and their suppliers. The forum provides practical solutions, guidance and leadership to improve the way that digital services are created, delivered and charged.</td>
</tr>
<tr>
<td>WS-Agreement</td>
<td>A standardised specification for the establishment of SLAs between initiators and responders.</td>
</tr>
<tr>
<td>Requirement</td>
<td>Functional or not functional characteristics of the SLA@SOI Framework desired from potential adopter (service providers).</td>
</tr>
<tr>
<td>Feature</td>
<td>Functionality that a specific software component of the SLA@SOI framework should implement to satisfy one or more requirements.</td>
</tr>
<tr>
<td>Research Topic</td>
<td>Subject of investigation of a scientific work package (action line A) that should produce specific project results.</td>
</tr>
</tbody>
</table>

Appendix B: Abbreviations

- SaaS: Software as a Service
- SID: NGOSS Shared Information and Data Model
- SLA: Service Level Agreement
- SOA: Service Oriented Architecture
- SotA: State of the Art
- TMF: TeleManagement Forum
- QoS: Quality of Service
- WP: Work Package

Appendix C: Status Definitions used in TRAC

The following status values have been used in Trac, which captures the current state of each requirement:

- **proposed**: In this state are all requirements, when they are put into the TRAC-system at the first time or when a change to them is proposed. (initial status set by the responsible B-line WP partner if a requirement is new or changed).
• **issues:** In this state there is more information needed for A-line to continue the validation process. If an extensive reconciliation between A-line and B-line will be necessary this should be done via wiki. In this case a remark in the TRAC (e.g. comments field) must note that there are additional statements in wiki. This remark in the TRAC have to be done by the responsible A-line. (→ status set by A-line).

• **rejected:** After evaluation from A-line the requirement is set to "rejected" if an implementation is not possible or the realisation makes no sense from the perspective of the A-line. It could be also set to rejected if the addressed A-line WP is not responsible for it (the requirement has no influence to the addressed WP). (→ set by A-line).

• **accepted:** Status "accepted": A-Line accepted the requirement proposed by B-Line. (→ set by A-line).

• **checked:** After examination of the requirement from B-Line the priorities of it have to be set and the status has to be changed to "checked". The status "checked" indicates that the requirement is incorporated in a feature and ready to be analysed by A-Line. This status will be set by B1 (TA) based on the TRAC report results. (→ set by B1 TA).

• **designed:** A-Line has defined components & interfaces that implement the requirement. After acceptance of the proposed architecture by B-line, the status of the requirement is set to designed. (→ set by A-line)

• **implemented:** A-line has completed the implementation of the requirement (→ set by A-line).

• **cancelled:** This could be, if a requirement has been rejected from A-line and the action lines agree not to implement it, or B-line renounced to ask the requirement (all priorities set to zero) (→ set by B1 TA after check with B1 leader)

• **tested:** A-line has completed the tests of the feature associated to the requirement (→ set by A-line).

• **approved:** After finalized tests of the implementation of a requirement from A-line the result has to be proved from B-line. If the result is accepted from B-line the status will be set to “approved” (→ set by B-line).

• **failed:** After finalized tests of the implementation of a requirement from A-line the result has to be proved from B-line. If the result is not accepted from B-line the status will be set to “failed” (→ set by B-line).
## Review results

<table>
<thead>
<tr>
<th>Reviewer name</th>
<th>George</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the deliverable comply with the common deliverable structure (glossary, etc.)?</td>
<td>YES</td>
</tr>
<tr>
<td>Is the formatting/layout acceptable</td>
<td>(please fill in your answer)</td>
</tr>
<tr>
<td>Are there any major inconsistencies in regards to other deliverables that should be addressed (if yes, please inform PMT)?</td>
<td>(please fill in your answer)</td>
</tr>
<tr>
<td>Are there any major flaws with the deliverable that require further tracking (if yes, please inform PMT)?</td>
<td>(please fill in your answer)</td>
</tr>
<tr>
<td><strong>Free comments:</strong></td>
<td></td>
</tr>
<tr>
<td>(please fill in your comments; additional comments may be made throughout the deliverable document and MUST be marked by tracked changes!)</td>
<td></td>
</tr>
</tbody>
</table>

SEE COMMENTS and TRACKED AMENDMENTS
THE DOCUMENT MUST BE SPELL_CHECKED AS IT CONTAINS MANY GRAMMATICAL ERRORS.

---

<table>
<thead>
<tr>
<th>Reviewer name</th>
<th>Ramin Yahyapour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the deliverable comply with the common deliverable structure (glossary, etc.)?</td>
<td>yes</td>
</tr>
<tr>
<td>Is the formatting/layout acceptable</td>
<td>yes</td>
</tr>
<tr>
<td>Are there any major inconsistencies in regards to other deliverables that should be addressed (if yes, please inform PMT)?</td>
<td>no</td>
</tr>
<tr>
<td>Are there any major flaws with the deliverable that require further tracking (if yes, please inform PMT)?</td>
<td>no</td>
</tr>
<tr>
<td><strong>Free comments:</strong></td>
<td></td>
</tr>
<tr>
<td>Overall the deliverable is okay.</td>
<td></td>
</tr>
<tr>
<td>I could imagine that a reviewer is threatened by the amount of output created by the process to identify requirements. In comparison the actual requirement result is almost lost, or better said, difficult to digest. Some simpler, human readable summary at the end would have been beneficial to bridge the details into a bigger</td>
<td></td>
</tr>
</tbody>
</table>
picture. However, I do not see how we can improve this on short notice.

I removed some English and formatting issues. There were many, I assume that I did not catch all issues. Some further proof-reading by a native speaker would help.

We do not provide some analysis whether the use-cases really fit to highlight the whole framework. Some critical remarks that we analyze and also try to adapt the use-cases to fully exploit the potential of the framework might be useful from a political perspective.

Anyway, I think that the deliverable is in good enough shape for submission.

---

**Reviewer name**

Joe Butler

**Does the deliverable comply with the common deliverable structure (glossary, etc.)?**

The list of abbreviations seems quite short: is this complete?

**Is the formatting/layout acceptable**

There are various tables and figures in the document: are these tagged and should we have a list of tables and figures at the start of the document?

**Are there any major inconsistencies in regards to other deliverables that should be addressed (if yes, please inform PMT)?**

no.

**Are there any major flaws with the deliverable that require further tracking (if yes, please inform PMT)?**

No.

**Free comments:**

I have commented in the form of embedded comments in the document. Comments include specific notes on level of detail and use of text.

I have not made any changes.

There are grammatical and spelling errors which MS-Word has highlighted, which should be addressed.

*(please fill in your comments; additional comments may be made throughout the deliverable document and MUST be marked by tracked changes!)*
2 State of the Art Analysis

3 Requirements Process

3.1 Requirements Collection

3.2 Requirements Validation

3.3 Requirements Consolidation

3.4 Requirements Implementation

3.5 Industrial use case overview

3.5.1 Use case B3: ERP Hosting (SAP)

3.5.2 Use case B4: Enterprise IT (Intel)

3.5.3 Use case B5: Service Aggregator (Telekom Austria)

3.5.4 Use case B6: E-Government (ENG)

3.6 Overview of External Sources

3.7 Requirements Analysis

4 Consolidated Requirements

4.1 Framework Management

4.1.2 Design & Development

4.1.3 Pre-offering

4.1.4 Service Offering

4.1.5 Service Negotiation

4.1.6 Service Provisioning

4.1.7 SLA Enforcement

4.1.8 Runtime Prediction

4.1.9 Service Monitoring

4.1.10 Service Reporting

5 Status of Implementation

5.1 A1: Architecture & Integration

5.1.1 Current state

5.1.2 Advances with respect to the state of the art

5.1.3 Lessons learned & next steps

5.2 A2: Business Management

5.2.1 Current state

5.2.2 Advances with respect to the state of the art

5.2.3 Lessons learned & next steps

5.3 A3: Service Management

5.3.1 Current state

5.3.2 Advances with respect to the state of the art

5.3.3 Lessons learned & next steps

5.4 A4: Infrastructure Management

5.4.1 Current state

5.4.2 Advances with respect to the state of the art

5.4.3 Lessons learned & next steps

5.5 A5: SLA Management & Foundations

5.5.1 Current state

1.1.1 Advances with respect to the state of the art

1.1.2 Lessons learned & next steps

5.6 A6: Predictable Systems Engineering

5.6.1 Current state

5.6.2 Advances with respect to the state of the art

5.6.3 Lessons learned & next steps

6 Requirement Evaluation

6.1 Evaluation Process

6.2 Scientific Metrics Evaluation

6.2.1 Testing
The features have been grouped in feature-classes. The topics are consisting of grouped feature-classes.

As an example the procedure in case of requirement #235:

Requirement [#235]: “Design time prediction must support shared service components”
The priorities have been set by the B-line WPs to a defined value (high, medium, low, zero). For this requirement the priorities have been set to:

B3: Medium
B4: Low
B5: High
B6: Low

The result was an overall priority “blocker” (minimum one high priority) for this requirement. The status has been changed to “checked” from B1 after all priorities have been set.

Thereafter the topic experts assigned the requirement #235 to the feature-class “Design and Development” and to the feature “Static Software Quality Prediction”.

and set a milestone for its implementation

, specifies related features, sets the status to "accepted"

functionality specified by the requirement

(only for functional requirements)

If

involved and considered from its conception

At the end of the consolidation process, we counted a total of 242 requirements. Fifteen percent (15%) of these requirements came from external sources (collected through direct interaction with external parties or by analysis of the literature
Identified as part of the state of the art review. Figure 3 shows the percentage of requirements identified through the different types of targeted resources.

![Pie chart showing percentage of requirements by source]

Figure 2: Percentage of requirements by source

During the revision steps, 50 requirements were closed as they were considered out of scope after further analysis. If all
of a requirement had been set to “zero” by all B-line WPs, the status of the requirement had been set to “closed” after a check from B1-WP-lead and A-line.

We defined as “significant” the requirements whose priority had been set to “high” or “medium” by the B-line.
Of the remaining 192 requirements that had a defined priority, 81 were asked by all the use cases (although with different priority), and 48 were considered useful by at least 3 use cases.

An overall priority was assigned to each requirement. Possible values for the “overall priorities” were:
- Blocker: at least one use case assigned high priority
- Critical: at least two use cases assigned medium priority
- Major: at least one use case assigned medium priority
- Low: at least one use case assigned minor priority

We defined as “significant” the requirements whose priority is higher than Low.

If we consider the priorities, a
21 requirements are considered of

can say we currently have

collected 171 significant requirements. The important percentage of requirements closed or with low priority may be ascribed to different factors such as changes in the use cases objectives, evolution of the overall project understanding, non applicability of external requirements to SLA@SOI project and so on.
Possible values for the "overall priorities" were:

- **Blocker**: at least one use case assigned high priority
- **Critical**: at least two use cases assigned medium priority
- **Major**: at least one use case assigned medium priority
- **Low**: at least one use case assigned minor priority

**Figure 5 shows how many requirements**

Of the significant requirements, **103 are blocker**, i.e. they are judged of high priority from
at least one use case. Therefore these 103 requirements were identified as the most important candidates for v1 of the SLA@SOI framework.

Figure 6 shows how many requirements were required by each use case. It is evident that use cases B5 and B3 are the most demanding in terms of number of requirements, especially medium and high ones.

![Figure 5: Use cases distribution of requirements](image)

Despite the apparent homogeneity of these figures, there are actually important differences between the use cases. There are 44 requirements that have been assigned priority "high" by more than 2 use cases. Moreover amongst the "blocker" requirements there were 9 requirements that were required by just one use case.

**It is also interesting to see the distribution of requirements with respect to Action Line A**
WPs. The next figure shows the percentage of requirements needed from each action line A

WP by each use case. The figure refers just to the significant requirements having a "high" or "medium" priority for the specific use case
Figure 6: Percentage of high and medium requirements by Action Line WPs

All use cases are distributed on scientific WPs in a similar way and have the highest percentage of requirements on A5.
B4 and B6 instead have very few requirements on prediction, while most requirements are on foundation.

It is evident that the most critical scientific WP, with respect to use cases requirements,

This fact 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, that is impacted by the requirements.

B5 has a very high number of requirements for
A6 due to the need for prediction on demand.

The minor relevance of A1 with respect to use case requirements is also obvious, as at least at this stage the use cases are more concerned with functionalities offered by the SLA@SOI framework, rather than with architectural requirements or other non functional requirements. We have to consider that at this stage there are likely several technical requirements not explicitly expressed because they are implicit in the SLA@SOI assumptions, such as the development of an integrated framework.