



*SLAs Empowering a Dependable Service Economy*



**Project no.** FP7- 216556

**Instrument:** Integrated Project (IP)

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

# Deliverable D.A1b Integrated Framework

**Keywords:**

Web Site, Service Level Agreement, Service-Oriented Infrastructure

**Due date of deliverable:** 31<sup>st</sup> July 2011

**Actual submission to EC date:** 29<sup>th</sup> July 2011

**Start date of project:** 1<sup>st</sup> June 2008

**Duration:** 38 months

**Lead contractor for this deliverable:** FBK

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	Yes



Document Status	
Deliverable Lead	Gabriele Zacco, FBK
Reviewer 1	Sam Guinea, PMI
Reviewer 2	Franz Brosch, FZI
PMT Reviewer	Wolfgang Theilmann
Complete version submitted to reviewers	June 20 <sup>th</sup> , 2011
Comments of reviewer 1 received	June 30 <sup>th</sup> , 2011
Comments of reviewer 2 received	July 18 <sup>th</sup> , 2011
Revised deliverable submitted to PMT	June 19 <sup>th</sup> , 2011
PMT Approval	July 25 <sup>th</sup> , 2011

Contributors	
Partner	Contributors
FBK	Gabriele Zacco, Nawaz Khurshid

Notices
<p>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 2010 by the SLA@SOI consortium.</p>
<p>* Other names and brands may be claimed as the property of others.</p>

This work is licensed under a [Creative Commons Attribution 3.0 License](https://creativecommons.org/licenses/by/3.0/).

Document History			
Version	Date	Author	Changes
0.1	20 June 2011	Gabriele Zacco	First version
0.2	15 July 2011	Gabriele Zacco	Implemented reviewer 1 comments
0.3	21 July 2011	Gabriele Zacco	Implemented reviewer 2 comments

0.4	27 July 2011	Gabriele Zacco	Implemented suggestions	PMT
-----	--------------	----------------	-------------------------	-----

## ***Executive Summary***

This document describes the implementation of the SLA management framework created by the European research project SLA@SOI. It serves as formal deliverable D.A1b for reporting the work progress of part of the work package A1 at project month 38. It complements deliverable D.A1a which provides the specification of the architecture of the SLA framework.

The goal of the implementation activities within the project consisted in the production of a multi-faceted software artefact. The first result of such activities has been the realization of a reference platform for the management of SLAs that is at the same time complete although generic, and customizable to be applicable to any possible domain. Another expected outcome was a finalized platform ready to be adopted in a reference use case like the Open Reference Case. Last but not least, the development activities should deliver a set of re-usable components, each handling a particular portion of the SLA management and service provisioning features that might be used in a standalone fashion or in conjunction with other components and/or within other frameworks.

This document provides an overview of the activities carried on in the last year of the project to cope with these (and many more) requirements and in particular to provide to the Use Cases of the B-line work package a customizable and configurable platform that could be flexibly adapted to the different scenarios and allows to be easily evaluated. It must be remarked that the evaluation of the features of the platform is not within the scope of this document, since it is handled by the D.B1a deliverable. Within this document, the framework is evaluated just with respect to its development and integration goals.

The document describes the technical background that was set up to complete the expected important development activities, the approach used for driving the implementation and the integration of the framework (both from a technical and from a functional point of view), and the way in which all this was made freely available to a public audience like the one of the SourceForge site.

# ***Table of Contents***

1	Introduction .....	8
1.1	Context and Scope .....	8
1.2	Document Overview .....	9
2	Framework Realization .....	9
2.1	Technical Environment .....	10
2.2	Development .....	11
2.3	Integration .....	12
2.3.1	Technical Integration .....	12
2.3.2	Functional Integration .....	12
2.4	Documentation .....	17
3	Framework usage .....	17
4	Conclusions .....	17
4.1	Summary .....	17
4.2	Outlook on Future Work.....	18
5	References .....	18
	Appendix A: Glossary .....	19
	Appendix B: Abbreviations .....	21

# List of Tables

Table 1: Document changes against previous version .....9

# 1 Introduction

## 1.1 Context and Scope

This document describes the implementation of the SLA management framework created by the European research project SLA@SOI. The implementation description includes both the development and the integration of the components that constitute the framework.

This document serves as formal deliverable D.A1b for reporting the work progress of work package A1 during the third phase of the project.

This document complements deliverable D.A1a, which describes the reference architecture of the framework and provides the guidelines for its adoption within a given domain. The document is complemented in turn by the SLA@SOI tutorial, contained within D.B2b, which supplies an example of how the steps of the adoption guidelines have been concretely executed for implementing the open reference demonstrator. Besides, D.B8a complements the information reported here about the release of the framework as an open source project on SourceForge. Last, the instructions for building the framework [4], for running it [5], and for executing the integration tests [6] can be definitely considered parts of the logical structure of this document.

The implementation of the SLA@SOI framework described in the following has achieved the result of making available three different kinds of software artefact. First of all, it has produced a set of reusable components that handles different aspects related to SLA-management and service provisioning from the business to the infrastructure level. Second, a generic framework, in which not all of the interfaces have been implemented, thus leaving the possibility of customizing the general solution to any domain and in particular to the B-line Use Cases. This version of the framework glues together the reusable components mentioned above to provide a consistent and complete support for the overall handling of SLA at the different levels. Third, a version of the framework in which all the component interfaces have been implemented in order to effectively adopt the solution within the Open Reference Demonstrator of the project.

### **Main achievement**

The main achievements of the A1 work package described in this document are the **implementation of the reference architecture described in D.A1a and the integration of the developed components into a comprehensive framework**. The SLA@SOI platform integrates the implementation of all the features that derives from the scientific results of the other work packages within the project. As demonstrated by the integration tests, by the Open Reference Demonstrator, and by the framework adoption from the B-line Use Cases, the integration of the functionalities is seamless and allows for a complete management of the SLA life-cycle.

The source code of the platform has been delivered as an open source project on the SourceForge incubator, along with many different sources of detailed documentation (tutorial, adoption guidelines, API) about how to exploit it.

The framework has been released in two flavours, both as a fully-fledged suite, in which the components can interact seamlessly, and as a toolbox, in which the singular components can be re-used as libraries.

## 1.2 Document Overview

The remainder of this document is structured as follows. Chapter 2 provides an update about the activities related to the development and integration that have taken place during the third year of the project. Chapter 3 provides updated information about the usage of the integrated framework. Chapter 4 concludes with a short summary, and an outlook on the future.

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

Section	Change overview
Chapter 1	Significant revision to reflect overall project's lifetime
Chapter 2	Merges Chapters 2 and 3
Chapter 3	Merges Chapters 4 and 5. Platform adoption moved to the document "Adoption Guide" [12]
Chapter 4	Revision including outlook on future

## 2 Framework Realization

The development of the framework during the third year of the project has taken advantage of the major shift represented by the release of the code onto the SourceForge open source project incubator [1]. This has meant a significant revision of many aspects of the development, integration and documentation of the code developed within the SLA@SOI framework, thus resulting into a big effort sustained by the developers and by the management team, but it also has represented the occasion to improve and engineer many aspects in the production and maintenance of the software results of the project. The target of delivering a final version of the platform that can be sustained in the years to come has produced many positive effects in several different aspects.

First of all, the technical environment and the infrastructure underlying the development has been enhanced to guarantee the possibility for future developers to take advantage of the most recent versions of the adopted tools. Then, all the aspects related to the development including the procedures documented in the guidelines, the testing activities, the quality management have been improved and intensified. Moreover, the integration approach, set up in the second year of the project, has been consolidated and tuned to better cope with the most recurring issues. Last, the documentation has been particularly taken into account, so to grant future users and developers of the platform a high degree of confidence in understanding both the features already implemented and the extension mechanisms of the source code.

The setup and administration of the SourceForge project has required considerable efforts and has been sustained mainly by Intel and, then, by FBK.

The details about these aspects can be found in the following subsections. Some more can be also found in D.B8a.

## 2.1 Technical Environment

The third year of the project has been used to consolidate the results achieved during the previous year, and to guarantee the Use Cases the possibility to use and evaluate an as stable as possible platform. In this perspective, only minor intervention in terms of the development and integration environment have been conducted. Nevertheless, some adjustments have been made both to fix the causes of recurring issues and to guarantee to the future developers a state-of-the-art development environment.

An upgrade of the main technologies used for building and executing the framework has been carried on. The most noticeable shift has been the porting from Java 5 to Java 6 of all the platform source code. Besides that, also the building tool, Maven, has been upgraded to version 3.0.2 from version 2.1.0. This has guaranteed a definite increase in the efficiency of the building of the platform that, considered the amount of code involved, is not to be underestimated. The other main upgrade has been the one related to Pax-runner, the tool used for the provisioning of the OSGi runtime. In this case, the adoption of version 1.7.0 has guaranteed speeding up the startup of the framework and the avoidance of some issues that were present in the previous versions that were adopted. It is useless to say that the adoption of such new technologies has not been painless, causing quite a big overhead to the developers and to the integration team. Overall, the effort was still balanced by the achievement of a good degree of stability and by reasonable performances in the compilation and execution of the platform.

The move of the SLA@SOI framework to the SourceForge site has produced a major shift in the infrastructure underlying the development. The already wide perspective under which the development had been organized – due to high rate of distribution of the implementation – has been extended to take into account the community of developers that in the future will sustain the evolution of the framework. It has also represented the occasion to improve and make more efficient the tools and procedures adopted for the development. Porting the development into such a new setting has meant the re-creation of many supporting infrastructures onto the open source environment. The creation and the maintenance of the SourceForge project have been performed by Intel in collaboration with FBK and have covered many aspects including the management of the accounts and of the public mailing lists, the generation of reports and statistics about the production of source code. Some other important aspects follow.

First of all, a new public SVN repository [2] has been created and maintained. All the code being part of the core platform has been endowed with a relevant open source license (namely, BSD) and moved to the new repository. The former private SVN repository has been kept, of course, to host all the rest of the project documentation but also as a storage for all the software production not directly related to the core framework, e.g. the software for the Use Cases and all other code that was not meant to be published as open source.

Then, attached to the SVN repository a new instance of the Trac tool [3], the one plugged into the SourceForge project itself, has been fed. This instance has completely substituted the one used in the first two years. All the tickets and milestones have been moved to the new environment and have been thoroughly used to help and check the progression of the development. The ticketing system in particular has been very important in the latest phases of the integration to increase the collaboration between the integration team, discovering issues, and the developers, successfully resolving them.

The wiki [3] attached to the Trac system has been used to document the new, engineered, version of the framework. All the specifications related to the

architecture, to the implementation, to the integration and testing of the platform, to the adoption and usage of the framework have been reported to the wiki. The collaborative contribution of all the project partners has been decisive to achieve a thorough description of all the aspects related to the platform. This wiki can be considered an extension of the one used in the first two years of the project which, of course, similarly to the SVN, could not be dismissed, also because not all the specification and documentation could be disclosed to the world. The importance of making a wiki available to the public users/developers is witnessed by the fact that it is actually being already successfully exploited by external (with respect to the project) developers for understanding the behaviour of the platform and for making the first steps towards the extension of the code.

Allowing public users to build the platform has meant to create and maintain a relevant artefacts repository, where internal and external libraries could be hosted for easing the compiling process. This new repository has been setup on SourceForge infrastructures and has required a big revision of the build mechanism of the platform and of all the framework modules. An important issue to be faced has been the inclusion and distribution of dependencies (libraries) considering their license. In order not to invalidate the BSD license chosen by the SLA@SOI Consortium for the release of the code, none of such dependency should have carried this *viral* licenses like, e.g., GPL or similar. A comprehensive assessment has been carried on to avoid such an occurrence. Currently, all the code needed for running the integration tests of the SLA@SOI framework is available under BSD license on the SourceForge website.

## 2.2 Development

The biggest effort about the management of the development activities has been devoted to clarify the procedures for the development and the release of platform modules, so that they could be used both as standalone components and be seamlessly integrated into the overall framework. In this respect the guidelines for the configuration of modules, the development of source code, the building [4] and the execution [5] of the platform, its testing and the release of the code to the repository have been enhanced. Many of such aspects have been already addressed in the previous sub-section when describing the tools to which they are related (Maven for the build, SVN for the commit). About testing, the request forwarded to the developers in the latest stages of the implementation has been to conduce a very strict check of the conformity of the code, starting from a clean build up to the execution of the integration tests, before committing the code to the repository in order to minimize the possibility of breaking the code on the main trunk. Such guidelines are available on the SourceForge wiki [6].

The development has taken advantage of the usage of Continuum which has been maintained since the second year. The tool has been upgraded and its daily builds as served as a reference for the stability of the platform from the build point of view. The Continuum tool was not installed on the SourceForge site due to limitations of the SourceForge infrastructure for this kind of application. The Continuum instance was made available on FBK servers.

The quality management issues, the unit testing and coverage indicators and the coding style prescriptions have been monitored within the Maven generated site. The final results for all of these reports are available at [9].

The source and binary code of the framework, and of its component, are available for download at [10]. In order to fulfil the project goal of delivering the components of the framework also in the guise of a "toolbox", the latest release of the platform also includes – at the same location – the binary packages of the pieces of software that correspond to the top level component of the architecture.

Such downloadable bits witness the possibility of not only adopting the SLA@SOI framework as a whole but also to re-use singular or multiple parts of its complex structure.

## 2.3 Integration

### 2.3.1 Technical Integration

The integration efforts conducted by FBK in collaboration with UDO, TID, Intel and SAP have been focused in the direction of avoiding the occurrence of conflicts when wiring together the several components (almost 200 bundles) of the SLA@SOI framework, which has actually been the first cause of issues during the integration.

On the one hand, as mentioned above, the guidelines for enabling a seamless integration of the modules have been enriched, suggesting developers how to configure their modules so to avoid any possible clash with other components or at least to ease the detection of the possible issues. Discussions have taken place in the early stages of the third year, suggested by TID, whether it was the case to adopt new approaches like Virgo [7] or use different *bundle configuration files*. In the end it was considered more appropriate to improve the current approach and perform a thorough review of the Maven configuration files to have a better handling of the common libraries and to avoid conflicts at run-time. This was considered overall safer than switching to a new approach.

On the other hand, another big effort was spent to reach a very high dependency convergence rate about the dependencies (93%). This also contributed definitively to diminish the risk of library clashes during the execution of the framework. The reports about dependency convergence are available in the Maven generated website [9].

Last, in order to achieve more control on the code execution and to improve the detection of the issues the Pax-runner profile repository for common libraries was substituted with our own profile repository.

### 2.3.2 Functional Integration

During the third year of the project we have also consolidated the functional integration results achieved in the two previous years.

The integration of the framework component from a functional point of view had been already achieved in the second year of the project. It was based on the execution of a set of integration tests aiming at showing how the architecture modules could really play together as expected. Such tests consist of complete workflows through the whole framework. The testing of the integrated platform is based on three of such scenarios that are extracted from the Open Reference Case. By putting together some of the multiple features of the individual components of the framework, these three scenarios demonstrate the capability that the individual platform modules have to cooperate in order to achieve high level goals. These goals correspond to the main activities that the platform is expected to carry on, namely negotiating, provisioning, and monitoring a SLA.

The first integration test is about the negotiation of a SLA. The sequence of interactions that are necessary between the components of the platform in order to reach the agreement on the service level, is showed in Figure 1.

The second integration test, that depends upon the previous one, is the provisioning scenario in which the services and resources, that were agreed

within the negotiated SLA, gets provisioned. This is represented by the sequence diagram in Figure 2.

Finally, the third scenario, that depends upon the two previous ones, is called the runtime scenario. It deals with the monitoring of the services and resources that have been provisioned at the previous step. It also consider reactions to possible violations of the provisioned SLA. Its representation, in the form of a sequence diagram, is presented in Figure 3.

Of course, these three tests do not cover the entire spectrum of capabilities of the platform both from a high level perspective and also in terms of functionalities that the single components provide. Nevertheless, on the one hand they cover a significant portion of the framework features, while, on the other hand, they have played a central role in assessing the successful integration of the components of the platform into a unique framework for the overall management of SLAs.

The Y3 consolidation of these integration tests consisted in the following aspects. First of all, the integration tests have been used in the form of non-regression tests to verify that the enhancements of the platform component, occurred during the third year, had preserved the correct execution of these global scenarios. In this respect, the integration team with the collaboration of all the developers, have performed intense activities of bug hunting and debugging to discover and fix all the issues that were preventing the right outcomes of the tests. The log of the execution of the integration tests have been continuously updated and published in the internal project wiki and when necessary also on the public wiki. The issues discovered during the execution of the tests were immediately reported onto the Trac system. Secondly, the execution of the integration tests themselves was improved from a usability point of view, both for the entire scenarios and for their subparts. Last, the documentation of these scenarios and of how it is possible to execute them has been considerably extended and is available on the SourceForge site in a dedicated section [6].

With all these activities we were able to demonstrate that the components of the framework really play together as they should and that the services of the framework provide sensible results, not only when operated individually as standalone, reusable components, within their unit tests, but also when they are requested to cooperate to fulfil higher level goals.

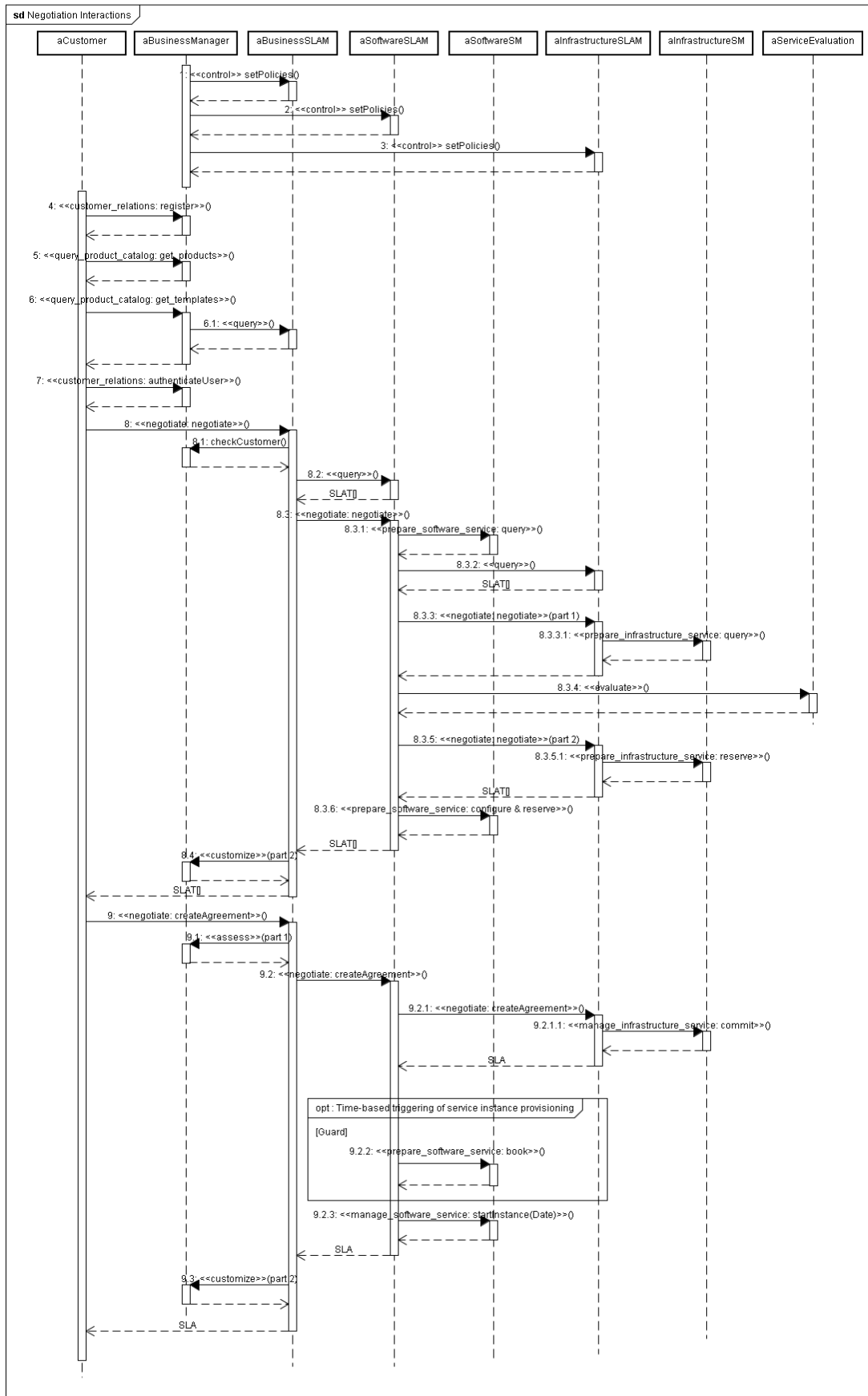


Figure 1: Negotiation scenario

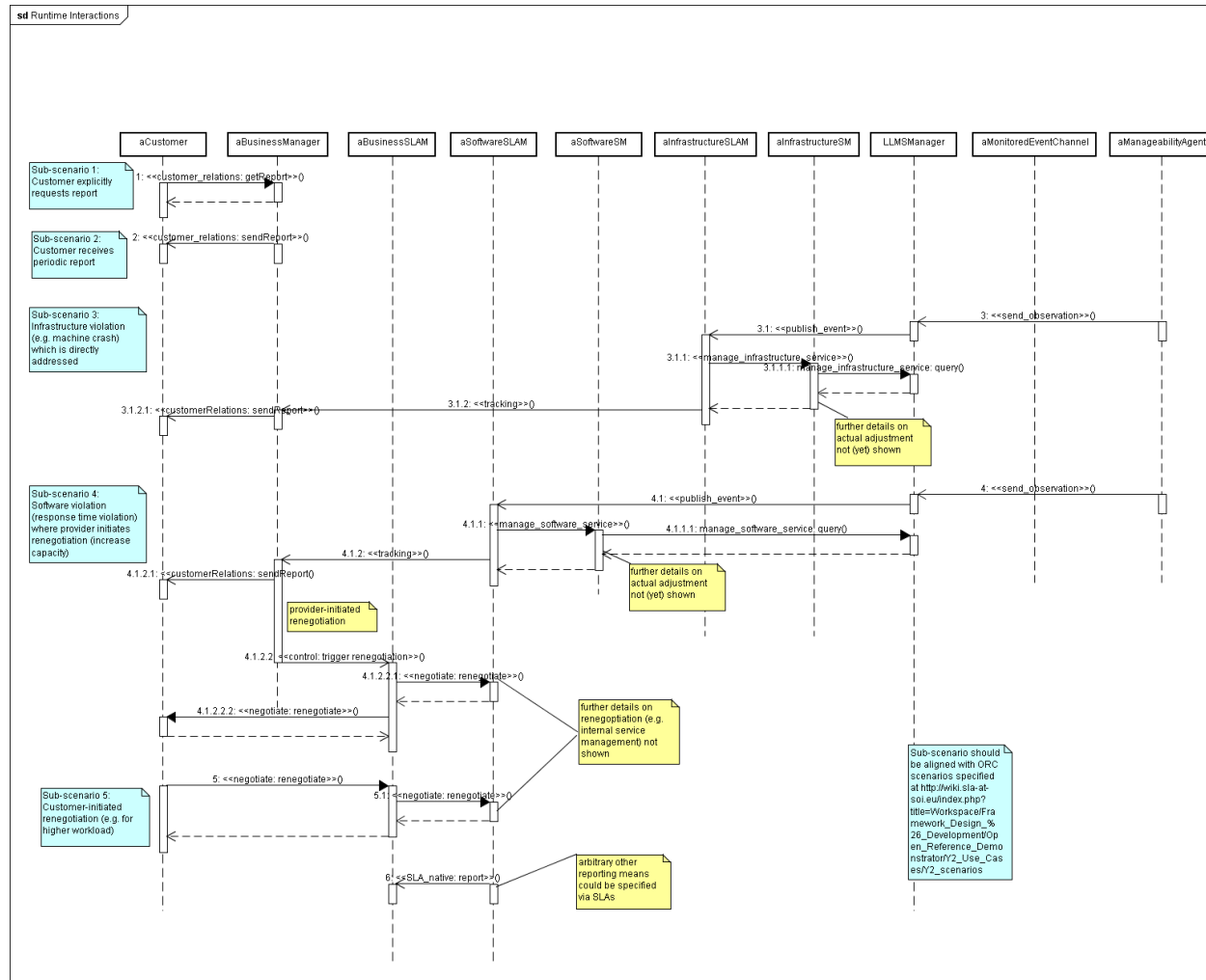


Figure 2: Provisioning scenario

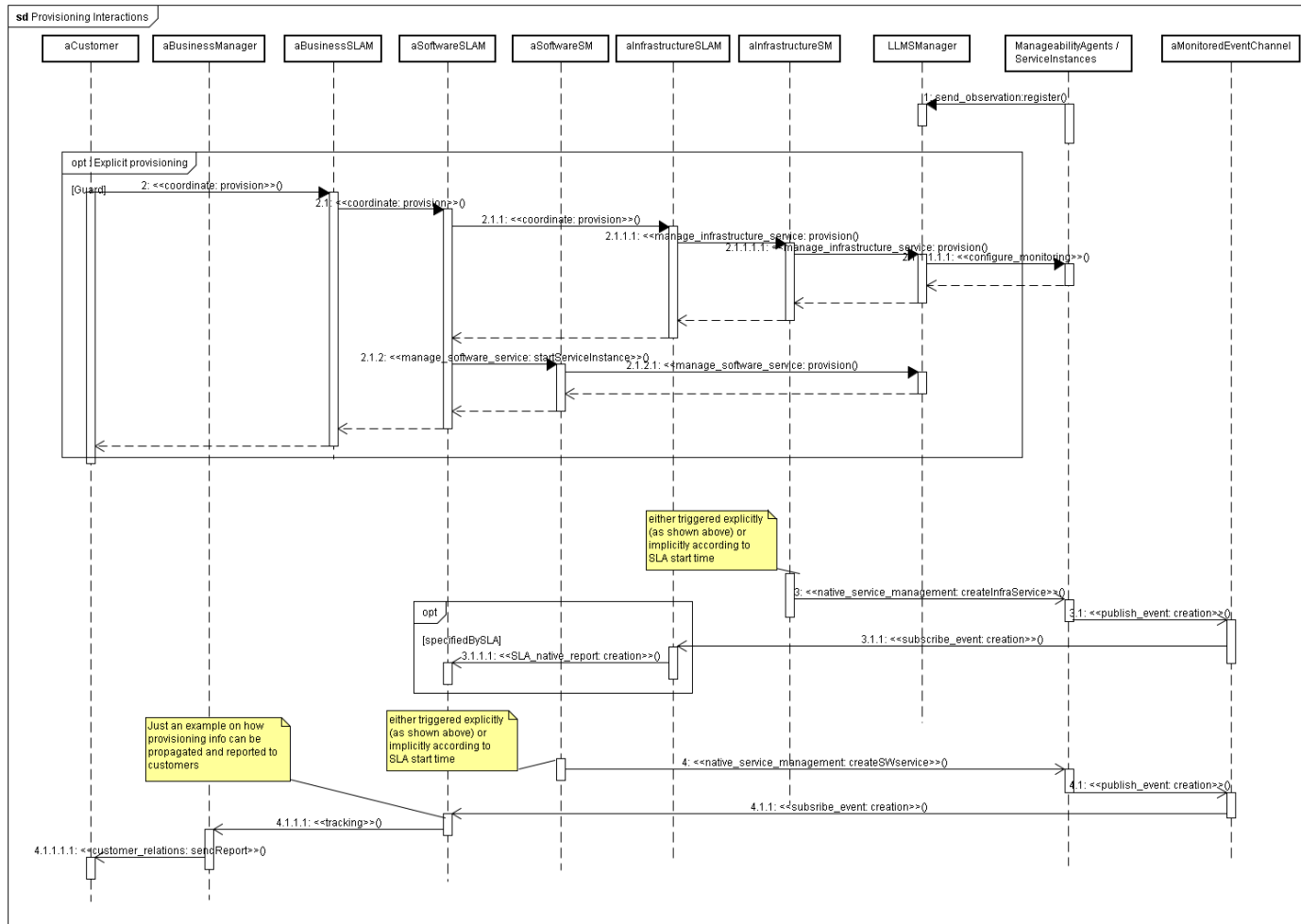


Figure 3: Runtime scenario

## 2.4 Documentation

The documentation of the results achieved with such an implementation and integration effort is an aspect that has been strongly pushed in this last year of the project. Both internally within code and externally with opportune documents the implementation of the features, the development and the usage of the platform have been clarified to their top.

The production of Java documentation is witnessed by the reports available both within the Maven generated website [9] and as a standalone *apidocs* archive [10].

Moreover, a lot of resources about the development and integration of the framework are available on the SourceForge site [1].

## 3 Framework usage

The possibility of guaranteeing an effective usage of the framework has been one of the main concerns of the third year activities within A1. It is due to this reason that it was decided to let the open source distribution of the code be driven by the integration tests. The requirement has been that public developers/users should be enabled to execute the integration tests. Thus, all the resources needed to accomplish this task have been released.

A possible usage of the framework is then thoroughly described, along with its build procedure, in the SourceForge wiki [6]. The information there contained allows any user to execute the three integration test scenarios that collect the most important features of the SLA@SOI framework (negotiation, provisioning and run-time monitoring)

More instructions about how to configure and start the framework after it has been customized for a given domain can be found in the last chapter of the adoption guide [12].

Moreover, the SLA@SOI tutorial [13] completes the picture in terms of usage providing the step-by-step instructions about how to use the framework within the open reference demonstrator use case.

Last, instructions about the adoption and usage of the framework in the other scenarios devised in the SLA@SOI projects can be found in the deliverable for work packages from B3 to B6.

The binaries for the execution of the framework are available for download at [10].

## 4 Conclusions

### 4.1 Summary

We have described the activities conducted in the third year of the SLA@SOI project to release the expected software artefacts, that is to say a set of re-usable components for handling separate aspects of SLA management and service provisioning, a general though complete platform assembling the mentioned components into an overall platform that could be exploited by the B-line work packages for their demonstrators and a complete instance of the

framework capable of addressing the needs of the ORC open reference demonstration scenario.

We have explained the technical background that has made this possible, the development, integration and documentation activities that were carried on and described in short the usage that has been made of the platform itself.

## 4.2 Outlook on Future Work

The future sustainability of the efforts that have been produced for developing and making publicly available the SLA@SOI framework are formally described in details in D.B8a.

Besides that, we can say that the framework has raised the interest of other recently started projects (e.g., CONTRAIL [8]) that have showed their will to take up the challenge of re-using the results of SLA@SOI. The possibility that the platform development may continue is also indicated by the increasing number of people that, following the recent publication of the source code, are already getting in touch with the SLA@SOI project development team, through the public SourceForge mailing lists, requesting details about how to execute the integration tests and asking details about the configuration of the single modules. We take this as a sign that a community is ready to start growing.

# 5 References

- [1] "SLA@SOI SourceForge website", <http://sourceforge.net/projects/sla-at-soi/>
- [2] "Public SVN repository", <https://sla-at-soi.svn.sourceforge.net/svnroot/sla-at-soi>
- [3] "Public TRAC", <http://sourceforge.net/apps/trac/sla-at-soi/>
- [4] "Build instructions", <http://sourceforge.net/apps/trac/sla-at-soi/wiki/BuildPlatform>
- [5] "Execution instructions", <http://sourceforge.net/apps/trac/sla-at-soi/wiki/ExecutePlatform>
- [6] "Integration tests documentation", <http://sourceforge.net/apps/trac/sla-at-soi/wiki/DocumentationIntegrationTest>
- [7] "Virgo Web server", <http://www.eclipse.org/virgo/>
- [8] "CONTRAIL EU project", <http://contrail-project.eu/>
- [9] "Maven generated website", <http://sla-at-soi.sourceforge.net/doc/slasoi-framework/>
- [10] "Download page", <https://sourceforge.net/projects/sla-at-soi/files/v2/>
- [11] "SLA@SOI Ohloh website", <http://www.ohloh.net/p/sla-at-soi>
- [12] "SLA@SOI Adoption Guide", [https://sla-at-soi.svn.sourceforge.net/svnroot/sla-at-soi/platform/trunk/doc/SLA@SOI-Adoption\\_Guide.doc](https://sla-at-soi.svn.sourceforge.net/svnroot/sla-at-soi/platform/trunk/doc/SLA@SOI-Adoption_Guide.doc)
- [13] "SLA@SOI Tutorial", <https://sla-at-soi.svn.sourceforge.net/svnroot/sla-at-soi/platform/trunk/doc/SLA@SOI-tutorial.doc>

## Appendix A: Glossary

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

Agreement Initiator	An agreement initiator is a party to a <i>service level agreement</i> . The initiator creates and manages an agreement on the availability of a service on behalf of either the service customer or service provider, depending on the domain-specific signalling requirements.
Agreement Offer	An offer is the description of the agreement relationship that is sent from <i>agreement initiator</i> to <i>agreement responder</i> during agreement creation, indicating the relationship which the initiator would like to form.
Agreement Responder	The agreement responder is a party to a <i>service level agreement</i> . The responder implements and exposes an agreement on behalf of either the service provider or service customer, depending on the domain-specific signalling requirements.
Agreement Template	An agreement template is an XML document used by the <i>agreement responder</i> to advertise the types of offers it is willing to accept.
Agreement Term	Agreement terms define the content of a <i>service level agreement</i> .
Business Service	A business service is exposed/invoked via at least some non IT elements.
Business Manager	A specialization of <i>service provider</i> : person that defines the SLATs of products and joins available services in a product.
External Service	External services are exposed across the boundaries of an organization, i.e. across at least two administrative domains.
Framework Administrator	A specialization of <i>service provider</i> : person that configures/adapts the SLA@SOI framework for a specific application.
Guarantee Term	Guarantee terms define the assurance on service quality associated with the service described by the service definition terms. They refer to the service description that is the subject of the agreement and define service level objectives, qualifying conditions and business value expressing the importance of the service level objectives.
Hybrid Service	A hybrid service is a set or bundle of other services where all these services are exposed to the customer but have different service interface types (e.g. an IT service and a business service).
Infrastructure Manager	A specialization of <i>infrastructure provider</i> : person/system that is interested to measure and control infrastructure properties.
Infrastructure Provider	A specific kind of service provider that focuses on the provisioning of <i>infrastructure services</i> .

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

availability.	Entities can dynamically establish and manage agreements via Web service interfaces.
Service Level Objective	Service Level Objective represents the quality of service aspect of the <i>agreement</i> . Syntactically, it is an assertion over the agreement <i>terms</i> of the agreement as well as such qualities as date and time.
Service Provider	An organization supplying services to one or more internal customers or external customers.
SLA Manager	A specialization of <i>service provider</i> : person/system that is responsible for managing SLATs and SLA relationships.
Software Designer	A specialization of <i>software provider</i> : person that designs/develops the architecture and components of a specific SLA based application.
Software Manager	A specialization of <i>service provider</i> : person that defines software-based services, takes care of their management and supports the SLA manager in creating appropriate SLA templates.
Software Provider	An organization producing <i>software components</i> which might be used by a <i>service provider</i> to assemble actual <i>services</i> .
Software Service	A software service is a specific <i>IT service</i> which is exposed/invoked by means of software entities such as Web services, user interfaces, or software-based business processes.
Software Component	Software components are the entities produced at design-time by a <i>software provider</i> .
Service Type	A service type (or abstract service) specifies the external interface of a service possibly including non-functional aspects. It does not specify any means (components, resources) which are needed for the actual provisioning of that service.

## Appendix B: Abbreviations

AOP	Aspect Oriented Programming
API	Application Program Interface
BM	Business Manager
BSD	Berkeley Software Distribution
B-SLAM	Business SLA Manager
EMF	Eclipse Modelling Framework
ERP	Enterprise Resource Planning
IE	Interaction Event
FCR	Finite capacity regions
GPL	General Public License
ISLAM	Infrastructure SLA Manager
ISM	Infrastructure Service Manager
IoC	Inversion of Control
KPI	Key Performance Indicator
LLMS	Low Level Monitoring System
LQN	Layered Queueing Networks
MA	Manageability Agent
MRE	Monitoring Result Event
MVC	Model View Controller
NFP	Non-functional property

ORC	Open Reference Case
OSGi	Open Service Gateway initiative
OVF	Open Virtualization Format
QoS	Quality of Service
QPN	Queueing Petri Nets
PAC	Provisioning and Adjustment Component
POC	Planning and Optimization Component
POJO	Plain Old Java Objects
SaaS	Software as a Service
SE	Service Evaluation
SLA	Service Level Agreement
SLAM	SLA Manager
SLAT	Service Level Agreement Template
SM	Service Manager
SME	Small and Medium-sized Enterprise
SOA	Service Oriented Architecture
SVN	Subversion
SW-SLAM	Software SLA Manager
SW-SM	Software Service Manager
TCO	Total Cost of Ownership
TOGAF	The Open Group Architecture Framework