

Project Number: **215219**
 Project Acronym: **SOA4All**
 Project Title: **Service Oriented Architectures for All**
 Instrument: **Integrated Project**
 Thematic Priority: **Information and Communication Technologies**

SOA4All Analysis Platform

D2.3.2 Service Monitoring and Management Tool Suite First Prototype

- Prototype Documentation -

Activity N:	Activity 1 – SOA4All Runtime	
Work Package:	WP2 – SOA4All Studio	
Due Date:	M18	
Submission Date:	10/09/2009	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	INRIA	
Revision:	1.0	
Authors:	Adrian Mos	INRIA
	Carlos Pedrinaci	OU
	Guillermo Álvaro Rey	ISOCO
	Iván Martínez	ISOCO
	Christophe Hamerling	EBM
	Guillaume Vaudaux-Ruth	INRIA
	Dong Liu	OU
	Samuel Quaireau	INRIA
Reviewers	Sven Abels	TIE
	Florian Schnabel	SAP

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	x
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	02.08.2009	Kick-Off Version	Adrian Mos INRIA
0.2	10.08.2009	Contribution on the status of KOPE integration in the overall architecture	Guillermo Álvaro Rey, Iván Martínez, SOCO
0.3	15.08.2009	Functional description of most of the elements of the AP as well as the architectural overlay and installation instructions	Guillaume Vaudaux-Ruth, INRIA
0.4	20.08.2009	Initial consolidated version	Guillaume Vaudaux-Ruth INRIA
0.5	21.08.2009	Descriptions of knowledge-level widgets	Guillermo Álvaro Rey, Iván Martínez, ISOCO
0.6	22.08.2009	Description of the time line widget and OWLIM installation instructions	Dong Liu, OU
0.7	23.08.2009	Added Executive Overview, Introduction and Summary; created a new consolidated version	Guillaume Vaudaux-Ruth, Adrian Mos, INRIA
0.8	31.08.2009	Revised document after internal review	Guillaume Vaudaux-Ruth, Adrian Mos, INRIA
1.0	04.09.2009	Minor corrections and changes	Guillaume Vaudaux-Ruth, Carlos Pedrinaci, Dong Liu, Adrian Mos, INRIA, OU
1.0	10.09.2009	Final Editing	Malena Donato, ATOS

Table of Contents

EXECUTIVE SUMMARY	6
1. INTRODUCTION	7
1.1 PURPOSE AND SCOPE	7
1.2 STRUCTURE OF THE DOCUMENT	7
2. ARCHITECTURAL CONSIDERATIONS	8
2.1 DATA SOURCES	8
2.2 BEP	9
2.2.1 <i>Service-oriented methods</i>	9
2.2.2 <i>Process-oriented methods</i>	9
2.3 UI WIDGETS	10
2.4 KOPE INTEGRATION	10
2.5 SENTINEL	11
3. PROTOTYPE FUNCTIONALITY IN THE STUDIO	13
3.1 OPENING THE ANALYSIS PLATFORM VIEWS FROM THE STUDIO	13
3.2 VIEWS AND ORGANIZATION OF ANALYSIS WIDGETS	14
3.3 SERVICES	14
3.4 PROCESSES	15
3.5 CUSTOMIZABLE MONITORING DASHBOARD	16
3.6 ANALYSIS WIDGETS	19
3.6.1 <i>Service Response Time History</i>	19
3.6.2 <i>Service List</i>	20
3.6.3 <i>Service Description</i>	20
3.6.4 <i>Real-Time Alert Manager</i>	20
3.6.5 <i>Time Window Selector</i>	21
3.6.6 <i>Process Overview</i>	22
3.6.7 <i>Process Event List</i>	23
3.6.8 <i>Process Event Detail</i>	23
3.6.9 <i>Process Time Line</i>	24
3.6.10 <i>Knowledge Analytics manager</i>	24
3.6.11 <i>Knowledge Analytics for a Service</i>	25
3.7 DEMO GENERATOR	26
4. INSTALLATION & SETUP	27
5. CONCLUSIONS AND NEXT STEPS	29

List of Figures

Figure 1. Overall Architecture of the Analysis Platform	8
Figure 2: SOA4ALL Studio entry point.....	13
Figure 3. The Analysis Platform Icon	13
Figure 4. Selecting a View	14
Figure 5. Service List.....	14
Figure 6. Predefined Service Detail View	15
Figure 7. Predefined Process View	16
Figure 8. Empty Customisable Analysis Page	17
Figure 9. Palette of Monitoring Widgets.....	17
Figure 10: Monitoring Dashboard – Example #1	18
Figure 11: Monitoring Dashboard - Example #2	18
Figure 12. Service Response Time History Widget.....	19
Figure 13. Service List Widget.....	20
Figure 14. Service Description Widget.....	20
Figure 15. Alert Manager Widget.....	20
Figure 16. Alert Popup.....	21
Figure 17. Time Window Selector Widget.....	21
Figure 18. Process Overview Widget.....	22
Figure 19. Process Overview Widget Legend.....	22
Figure 20. Process Event List Widget.....	23
Figure 21. Process Event Detail Widget	23
Figure 22. Process Time Line Widget.....	24
Figure 23. Knowledge Analytics Manager Widget.....	25
Figure 24. Knowledge Analytics (for a specific service) Widget	25
Figure 25. DEMO Generator Interface.....	26
Figure 26. Sesame Control Console.....	28

Glossary of Acronyms

Acronym	Definition
AP	Analysis Platform
API	Application Programming Interface
BEP	Basic Event Processor
D	Deliverable
DSB	Distributed Service Bus
EC	European Commission
EVO	Events Ontology
GUI	Graphical User Interface
KOPE	Knowledge-Oriented Provenance Environment
SENTINEL	SEmaNTic busINess procEsses monitoring tool
SOA	Service-Oriented Architecture
SUPER	Semantics Utilized for Process Management within and between Enterprises
WP	Work Package
WSDM	Web Services Distributed Management
XML	Extensible Markup Language

Executive summary

This document accompanies the software deliverable D2.3.2 of the first prototype of the SOA4All Analysis Platform. It describes the functionality of the prototype and places the implementation in the architectural context described by deliverable D2.3.1.

The document presents in detail the views and widgets implemented in the Analysis Platform, briefly presents programming interfaces for certain building blocks and includes specific installation instructions that are not already covered by the overall Studio prototype documentation presented in D2.4.2.

Please note that an “alpha” version of the Analysis Platform software, at the time loosely integrated in the SOA4All Studio, was presented at the M12 review to complement the architectural specifications in D2.3.1. It has since evolved significantly to include more functionality and to seamlessly integrate into the overall Studio prototype.

1. Introduction

1.1 Purpose and Scope

The SOA4All Analysis Platform (AP) aims to provide the SOA4All users with information that would help them understand the performance characteristics and usage patterns of the services and processes they use. Such information must be presented at different levels of abstraction in order to be adapted to the different stakeholders that may require analyzing processes and service executions, as well as to the different types of problems or opportunities that may appear. That is why the AP provides a wide array of widgets in its graphical views, organized according to their potential use. Furthermore, as presented in the document, the AP offers a completely customizable approach to data visualization so as to correspond precisely to the expectations and needs that more advanced users have.

This document complements the previous T2.3 deliverable, D2.3.1, which presented in detail the different building blocks of the AP, their integration into the overall architecture, as well as their underlining concepts. Therefore, it does not go into details on any of these subjects and stays focused on the current D2.3.2 prototype implementation. The document is to be used in conjunction with D2.4.2 as the Analysis Platform is implemented as part of the Studio. In fact the actual software of the prototype is part of the entire Studio codebase and its binaries are integrated into the overall Studio server-side component, the .WAR file that groups all Studio contributions (please refer to D2.4.2 for indications of how the .WAR file is structured).

1.2 Structure of the Document

The rest of the document is structured as follows:

- Section 2 presents some architectural considerations to better place the current implementation in the context of the overall AP architecture. It shows the entities that have been implemented and presents an overview of the APIs that are of interest to Studio developers. However, Section 2 does not go into details about such APIs as such details can easily be obtained from the javadoc descriptions. In addition, the scope of the document is the prototype description primarily at the user-level.
- Section 3 represents the main part of the document and provides a functional description of the AP by describing the user-level entities that it offers. This is where the views and widgets are presented together with their expected usage as well as their interrelations. This part can be seen as a user-guide for the software delivered in the D2.3.2 prototype.
- Section 4 describes how the prototype can be installed and run. Since the main installation instructions are presented at the Studio-level in D2.4.2, this section only covers those instructions that are complementary to the overall Studio installation.

2. Architectural Considerations

This section briefly presents the architectural considerations for the first prototype of the Analysis Platform. It can be seen as a complement to the D2.3.1 Deliverable, which described the architecture and expected functionality of the Analysis Platform. As such it does not cover in detail the different parts of the AP, it rather focuses on the concrete implementation available for the main parts specified in D2.3.1.

Figure 1 below is a version of the diagram presented in D2.3.1 as Figure 4 (in Section 3). It highlights the parts that have an implementation in this prototype and it shows the parts that have been omitted in this version (the entities that are grayed out).

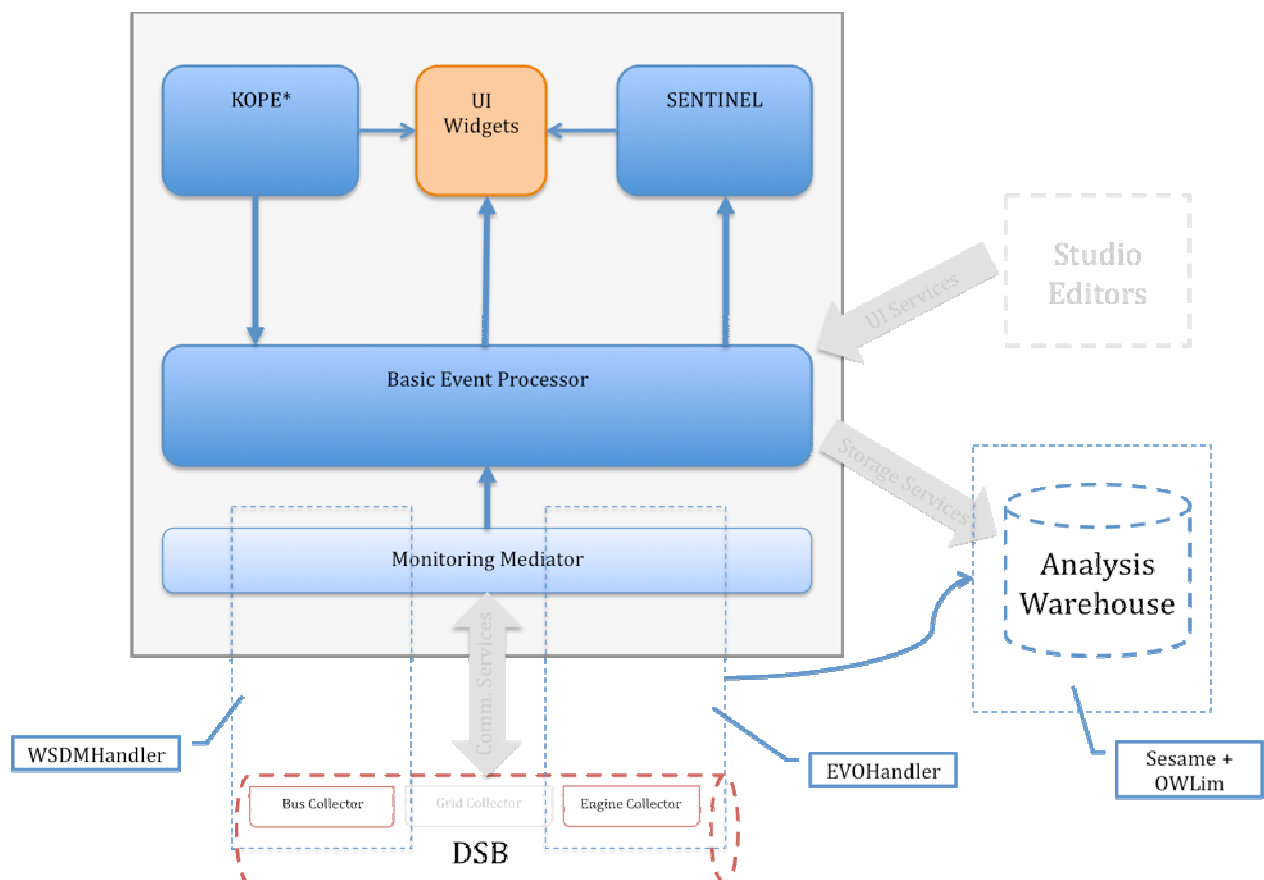


Figure 1. Overall Architecture of the Analysis Platform

It can be noted from the illustration that the connection to other editors in the Studio has been omitted in this version. It will be implemented in the next prototype.

2.1 Data Sources

We have implemented in this version two initial data sources, one corresponding to the Bus Collector and one corresponding to the Engine Collector in Figure 1. The data sources have been assimilated to the Monitoring Mediator rather than passing through the communication services of the Studio. A better separation will be implemented in the next phase.

The `WSDMHandler` entity available in this prototype is able to collect events from the WSDM¹

¹ Web Services Distributed Management (WSDM): <http://www.oasis-open.org/committees/wsdm/>

bus, convert them to the event hierarchy used by the BEP (as per Section 4.4.1 from D2.3.1) and send them to the BEP.

Similarly, the `EVOHandler` can obtain events from the execution engine expressed in terms of the analysis ontologies, namely EVO and COBRA (see D2.3.1), convert them to the appropriate hierarchy and send them to the BEP for processing. In addition, it stores the information from the events in the Analysis Warehouse. The current version relies on a standalone repository based on Sesame² and OWLim³. Future versions of the prototype will directly interact with the storage functionality provided by the Distributed Service Bus.

2.2 BEP

The Basic Event Processor implementation in this prototype performs several important functions. It obtains data from the data sources (implemented by the two handlers described above), performs basic processing and provides the data to consumers (the console widgets, the Knowledge Analytics component KOPE* and SENTINEL). For efficient local processing, the BEP currently employs a high-performance local database – hsqldb⁴.

The BEP offers an API that provides simplified access to its data. This API is used by its clients such as the UI widgets, KOPE or SENTINEL and is presented in a reduced form below. The full descriptions of the methods can be found in the javadoc of the monitoring module available in the SOA4All SVN repository under

```
trunk/soa4all-studio/soa4all-analysis-platform/monitoring-bep
```

A brief list of the main methods of this API is given in the next two sub-sections. Their purpose is self-explanatory.

2.2.1 Service-oriented methods

```
public ServiceInfo getServiceInfo(Date fromDate, String serviceName);
public ServiceInfo getLastServiceInfo(int nbInfo, String serviceName);
public List<? extends ServiceInfo> getServices(Date fromDate);
public List<? extends ServiceInfo> getServicesWithoutState();
public ServiceInfo getServiceInfo(String serviceName, Date occurredAfter,
    Date occurredBefore);
public List<? extends ServiceInfo> getServices(Date fromDate, IOperator
    operator);
```

2.2.2 Process-oriented methods

```
public List<ProcessInfo> retrieveCurrentWholeProcessTrees();
public List<ActivityInstance> getKnownActivitiesWithEvents(ProcessInstance
    processInstance);
public List<ActivityInstance> getKnownActivities(String processNamespace,
    String processName);
public EventUpdate getProcessAndActivityLastEventsSince(Date startDate);
public EventHistory getLastEventsSince(Date startDate);
```

² <http://www.openrdf.org/>

³ <http://www.ontotext.com/owlim/>

⁴ <http://hsqldb.org>

2.3 UI Widgets

A complete set of widgets grouped in pre-defined and customisable monitoring pages have been implemented. These widgets are presented in detail in Section 3. They obtain their data from the BEP using the BEP client interfaces discussed in Section 2.2.

2.4 KOPE integration

The first version of the Knowledge Analytics (KOPE*) component focuses on the (knowledge-level) analysis of services. We will expand the analysis to users and goals after M18, while also performing more advanced computations taking advantage of the concepts linked to the services used.

Currently, the component allows users to track the evolution of services over time, basing the analysis on three top-level concepts derived from lower-level data:

- **Frequency:** How much a service is used. Derived from:
 - Visibility frequency (how many times a service is opened)
 - Invocation frequency (how many times it is actually consumed)
- **Performance:** How well the service behaves. Derived from:
 - Time performance (how long the execution takes)
 - Reliability performance (if the execution finishes well)
- **User Perception:** What do the users think about the service. Derived from:
 - Ratings perception (average of the ratings)
 - Reviews perception (number of ratings, comments and tags)

Once tracking is set for a service, the batch-processing component begins to extract relevant information for that service each night. This component requires interactions with three sources to obtain data:

- From the **BEP**, which will be queried for the necessary information about executions of those services (important source for the calculations on Performance and Invocation Frequency)
- From the **Auditing Framework** (actions of the users within the platform -e.g., open a service, invoke a service-, stored in the Semantic Spaces via the Storage Services provided by T2.4)
- From the **Feedback Framework** (ratings, tags and comments by the users, also stored in the Semantic Spaces via the Storage Services, as explained in D2.1.3)

The results of the batch-processing computations (the analysis results) are stored in a new repository in the Semantic Spaces via the Storage Services again. Hence, from the whole bunch of execution logs, interaction logs, ratings, etc., we will have the relevant information for each service ready to be accessed in an intermediate storage.

This information already computed will permit us display the three relevant concepts on a selected time-frame, in order to see at a glimpse the evolution of these concepts over time. This functionality is illustrated graphically by Figure 24, while Sections 3.6.10 and 3.6.11 cover with more detail the functionality that can be accessed by the graphical interface of the Analysis Platform.

The functions that our component make available via its API to other components, and which we will also use for generating the graphs, are self-explanatory. They need a service identifier and a time frame (dateIni and dateEnd) to retrieve the information from the computations repository. These functions are:

```
getServiceVisibilityFrequency(serviceId, dateIni, dateEnd);
getServiceInvocationFrequency(serviceId, dateIni, dateEnd);
getServiceAggregatedFrequency(serviceId, dateIni, dateEnd);

getServiceTimePerformance(serviceId, dateIni, dateEnd);
getServiceReliabilityPerformance(serviceId, dateIni, dateEnd);
getServiceAggregatedPerformance(serviceId, dateIni, dateEnd);

getServiceRatingsPerception(serviceId, dateIni, dateEnd);
getServiceReviewsPerception(serviceId, dateIni, dateEnd);
getServiceAggregatedPerception(serviceId, dateIni, dateEnd);

getServiceAggregatedData(serviceId, dateIni, dateEnd);

startTrackingService(serviceId);
stopTrackingService(serviceId);
```

2.5 SENTINEL

SENTINEL⁵ makes use of semantic technologies to facilitate monitoring on business processes and activities, namely it not only keeps track of execution of business processes, but also derives business level knowledge from the low-level trails. On the other hand, both the components deployed on the DSB and the underlying infrastructure continually generate events, thus the monitoring tool relies on event processing. In short, the main functionalities of SENTINEL are as follows:

- Processing the events collected through the BEP to, for example, update the information about lifecycle of business processes, which is performed based on the rule engine Drools.
- Persisting data into the RDF repository, e.g. OWLIM.
- Computing metrics, e.g. the execution time of business processes.
- Visualizing raw events or results of metrics computation by different metaphors, e.g. table, time line, etc.

The integration of SENTINEL is achieved by using the code generation tool of Elmo, which generates the interface of storing and retrieving instances of the concepts in COBRA and EVO ontologies from the RDF repository. In this way, the rule engine has access to the raw event data, and also can persist reasoning results as RDF statements. The functions defined in this component are:

⁵ Pedrinaci, C., Lambert, D., Branimir Wetzstein, Tammo van Lessen, Luchesar Cekov, and Marin Dimitrov (2008) **SENTINEL: A Semantic Business Process Monitoring Tool**, Workshop: Ontology-supported Business Intelligence (OBI2008) at 7th International Semantic Web Conference (ISWC2008), Karlsruhe, Germany

```
getEventRepositoryManager();  
getExecutionHistoryManager();  
getMetricsRepositoryManager();  
insertEvent(event);  
computeMetric(metricName);
```

3. Prototype Functionality in the Studio

This section presents the user functionality of the Analysis Platform showing the concrete interface offered by this prototype and how it can be leveraged to obtain information about services and processes. The functionality offered by the different parts of the platform is described together with the corresponding graphical widgets. We expect the actual users of the AP to be people that have varying skills and knowledge about the SOA4All infrastructure. They will range from people that have no knowledge of the platform to people that have certain notions of the infrastructure (e.g. they know that there is an underlying bus and that processes are executed in a process engine). The views and widgets of the AP cater to a variety of such users presenting information at different levels of abstraction.

3.1 Opening the Analysis Platform Views from the Studio

Users can access the Analysis Platform views by selecting it from the main Studio Dashboard view (illustrated in Figure 2). The icon corresponding to the AP is indicated in Figure 3.

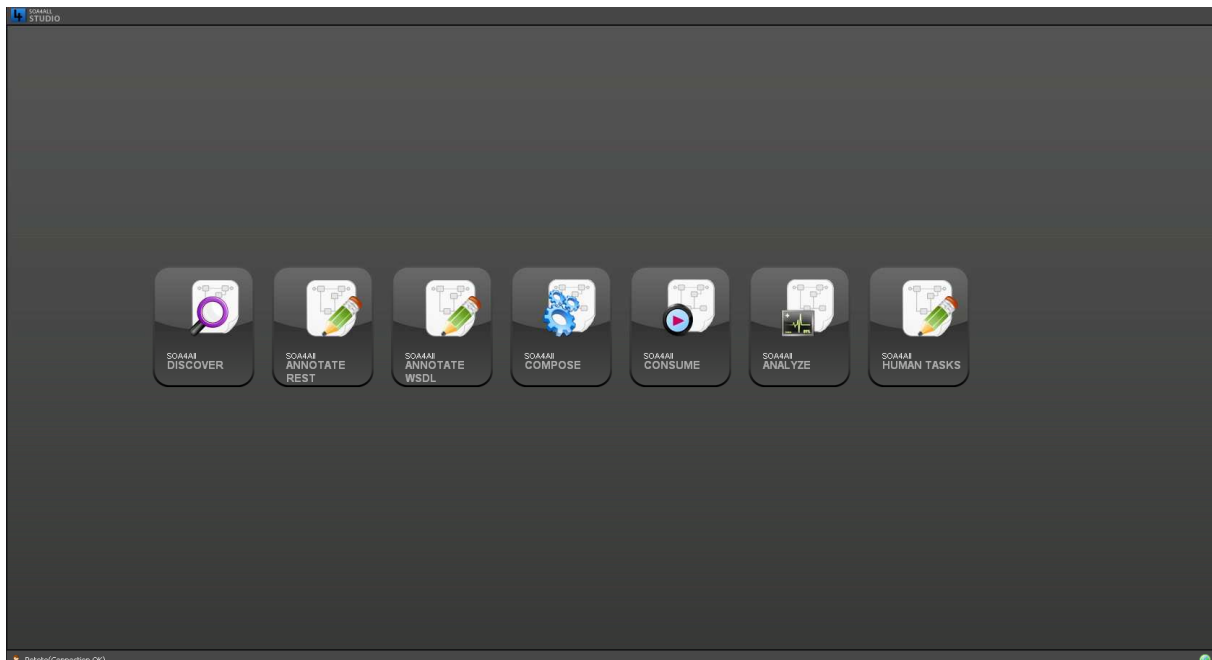


Figure 2: SOA4ALL Studio entry point



Figure 3. The Analysis Platform Icon

3.2 Views and Organization of Analysis Widgets

Currently, the AP has three main view sets. The selection of the view is done through the menu bar, which is found in the upper part of the screen. It is illustrated in Figure 4.

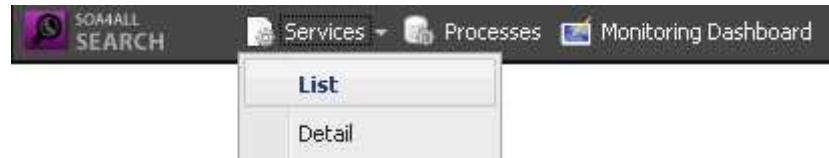


Figure 4. Selecting a View

The first two views sets have pre-defined content and target two main monitoring areas: services and processes. The third view set has user-definable content and provides mechanisms that users can leverage to compose their console using a combination of available widgets.

View sets can have multiple views and in the current implementation, the “Services” set has two views, the “List” and “Detail”. More may be added to the view sets in the future.

3.3 Services

There are two views associated with this set. The first view (illustrated in Figure 5) is intended to provide a way to display any list of services that might be of interest for the user currently logged in. For instance, this could represent the favorite services or the services most recently used. Each service has a link (its name) and an indicator that represents the overall availability of the service.

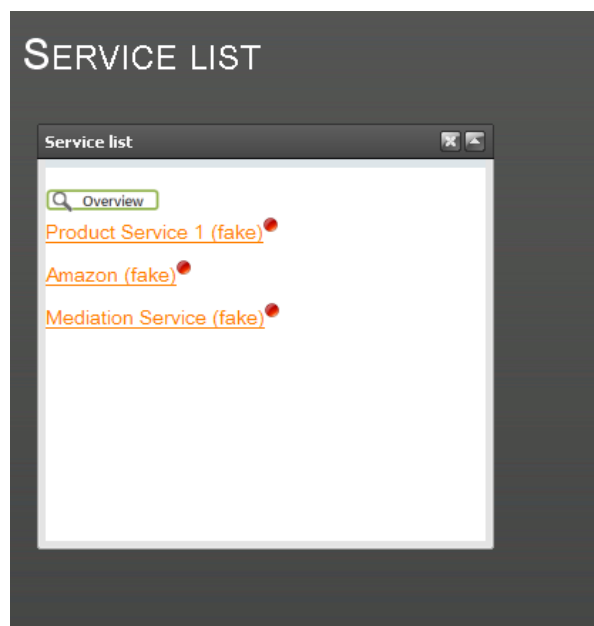


Figure 5. Service List

Clicking on the service name takes the user to the detail view for it, as detailed below and illustrated in Figure 6. The availability indicator can be red or green depending on whether the service is unavailable or available respectively.

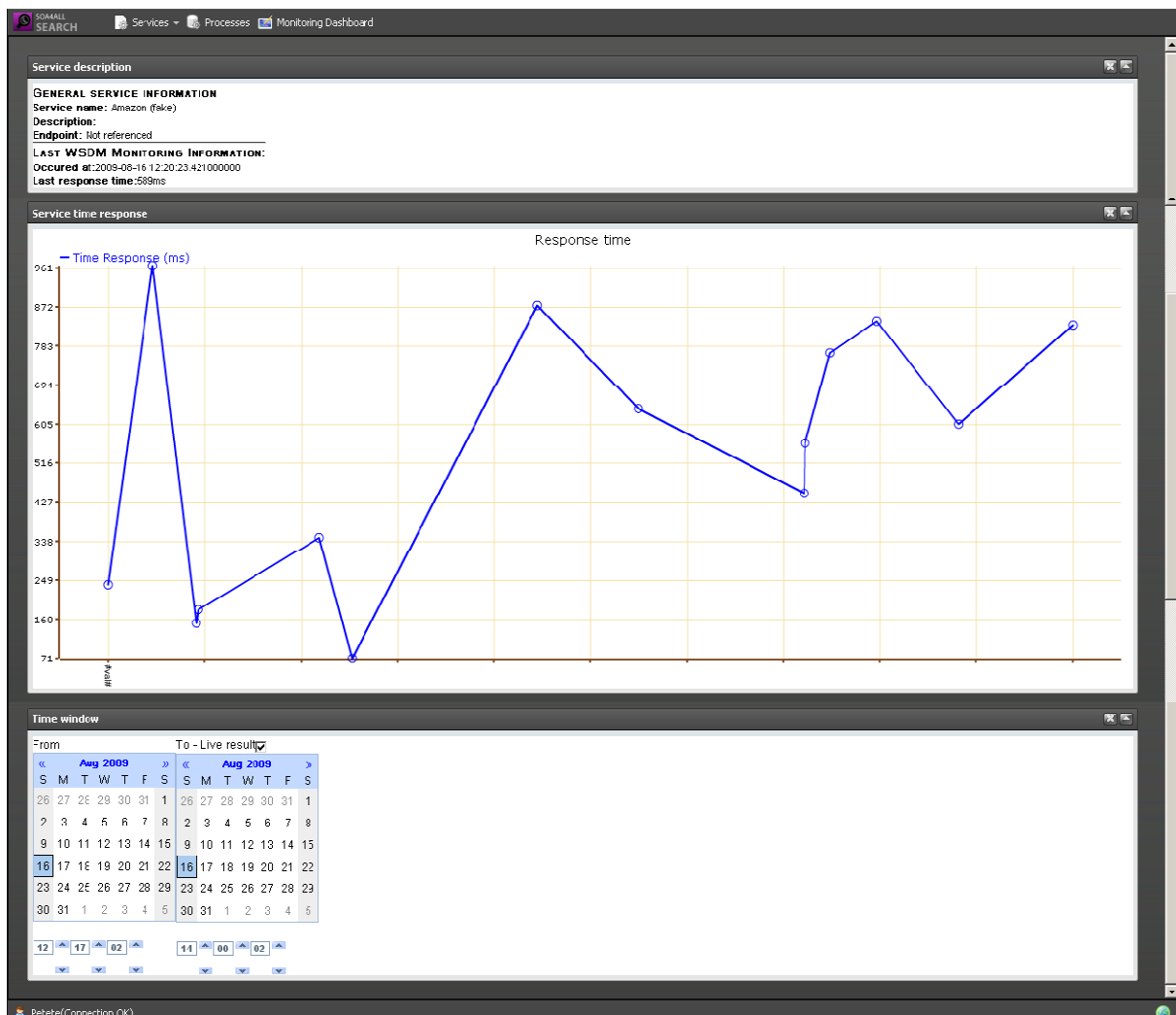


Figure 6. Predefined Service Detail View

The Service Detail View contains a predefined set of widgets: the Service Description, the Service Response Time History and the Time Window Selector. They are illustrated in more detail in the sections below so they are not described here. They all show information pertaining to a service that has been selected in the Service List page, if arriving at the detail page from the list page, or to the latest service selected in the Service List page, if arriving at the detail page directly from the menu bar.

3.4 Processes

The process view shown in Figure 7 contains a predefined set of widgets pertaining to processes that are being executed by the platform and which are relevant to the user logged in⁶. This page contains the Process Overview, Process Event List and Process Event Detail widgets. All of these widgets are detailed in the sections below.

⁶ In this first prototype we do not differentiate between users. Support for this will be implemented in future versions through the overall user-support capabilities of the Studio.

Processes overview

http://www.ip-super.org/ontologies/execution-history#Fulfillment

- pi31 (COMPLETED)
- pi32 (STARTED)

Process event list

Event Type	Generated By	Creation Timestamp
ActivityCompleted	SBPELEE	2008-08-24 06:17:55.984
ActivityStarted	SBPELEE	2008-08-24 06:17:55.905
ActivityStarted	SBPELEE	2008-08-24 06:17:55.905
ActivityCompleted	SBPELEE	2008-08-24 06:17:55.905
ActivityCompleted	SBPELEE	2008-08-24 06:17:55.155
ActivityStarted	SBPELEE	2008-08-24 06:17:53.14
ActivityStarted	SBPELEE	2008-08-24 06:17:53.14
ActivityStarted	SBPELEE	2008-08-24 06:17:53.125
ActivityStarted	SBPELEE	2008-08-24 06:17:53.125
ActivityStarted	SBPELEE	2008-08-24 06:17:53.125

Process event detail

Event Id:
 Event Type: *STARTED*
 Generated By: *SBPELEE*
 Creation Timestamp: *2008-08-24 06:17:53.14*

Event Occurred During Process Execution: Type: http://www.ip-super.org/ontologies/execution-history#Fulfillment ID: pi32

Event Occurred During Activity Execution: Type: http://www.ip-super.org/ontologies/execution-history#InvokeSIPService ID: ai26

Figure 7. Predefined Process View

3.5 Customizable Monitoring Dashboard

This page allows the user to create their own console, as they require. User identification will be performed through the user profile and login functionality of the Studio, using OpenID (please see D2.4.2 for more information on user-management). The custom monitoring dashboard is similar in principle to the iGoogle⁷ pages where a user that has logged in can have their own Google start page with different widgets. However in addition to the placement functionality, we also aim to have appropriate synchronisation between the widgets. An example would be when selecting a service from a service selection widget to update the different service information widgets to actually point to the selected service.

When first loading this page, the customisable dashboard is empty and the user has the option of choosing widgets from a palette on the left side of the screen, as illustrated in Figure 8. By dragging widgets from the palette, the user can place them on the screen in the desired place. The widgets can also be repositioned after they have been placed in the console.

⁷ <http://www.google.com/ig>

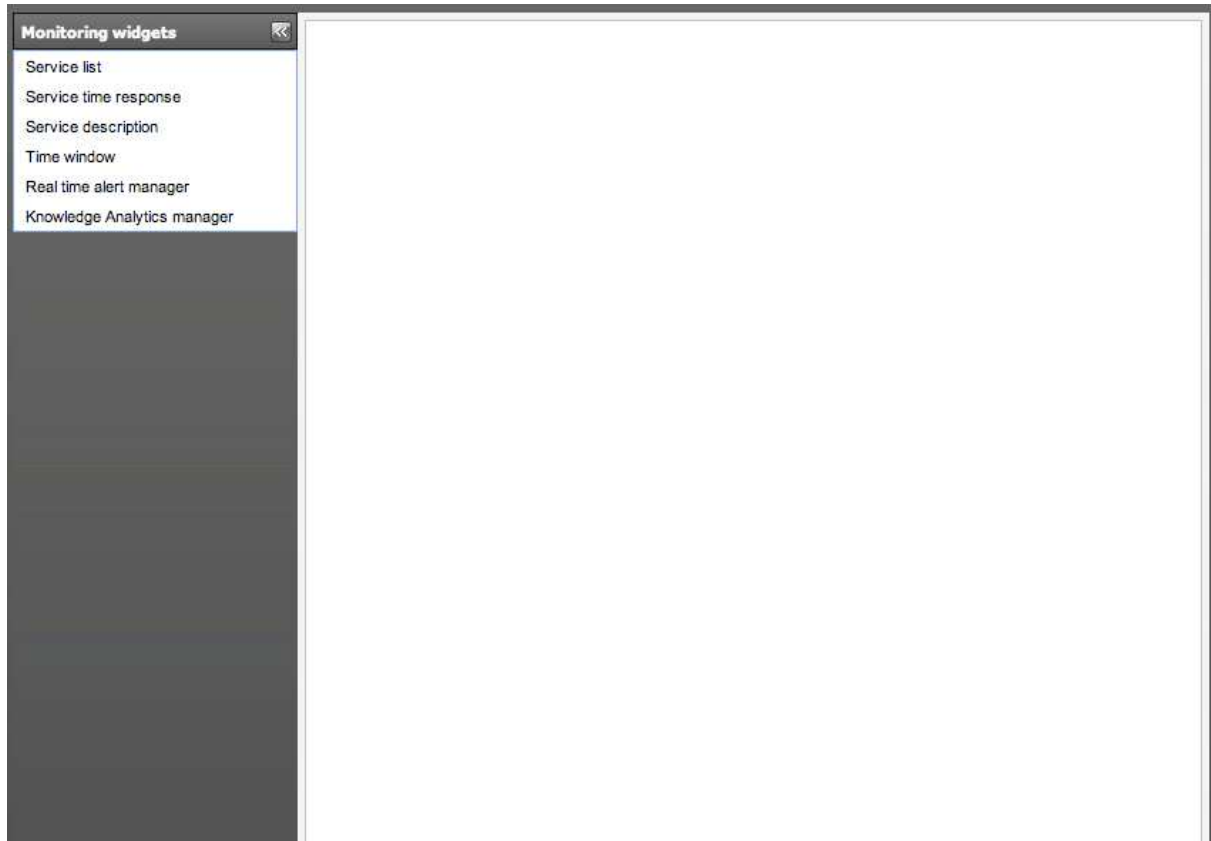


Figure 8. Empty Customisable Analysis Page

The palette is shown in Figure 9.



Figure 9. Palette of Monitoring Widgets

To illustrate the customisation options for this page, Figure 10 and Figure 11 show screenshots of two different versions that could be created within this layout.

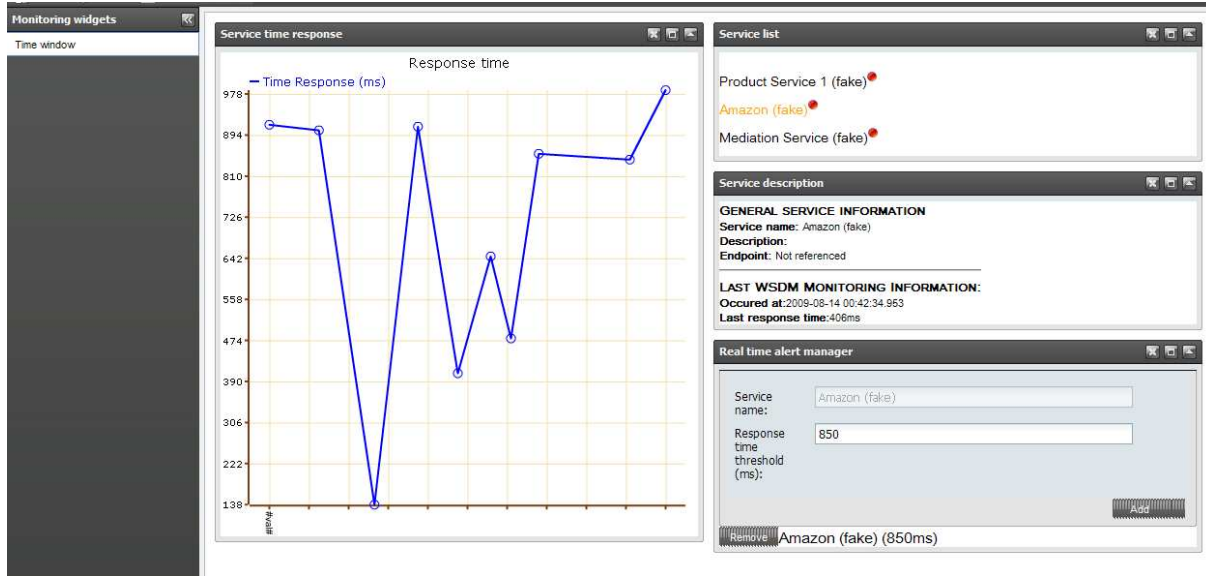


Figure 10: Monitoring Dashboard – Example #1

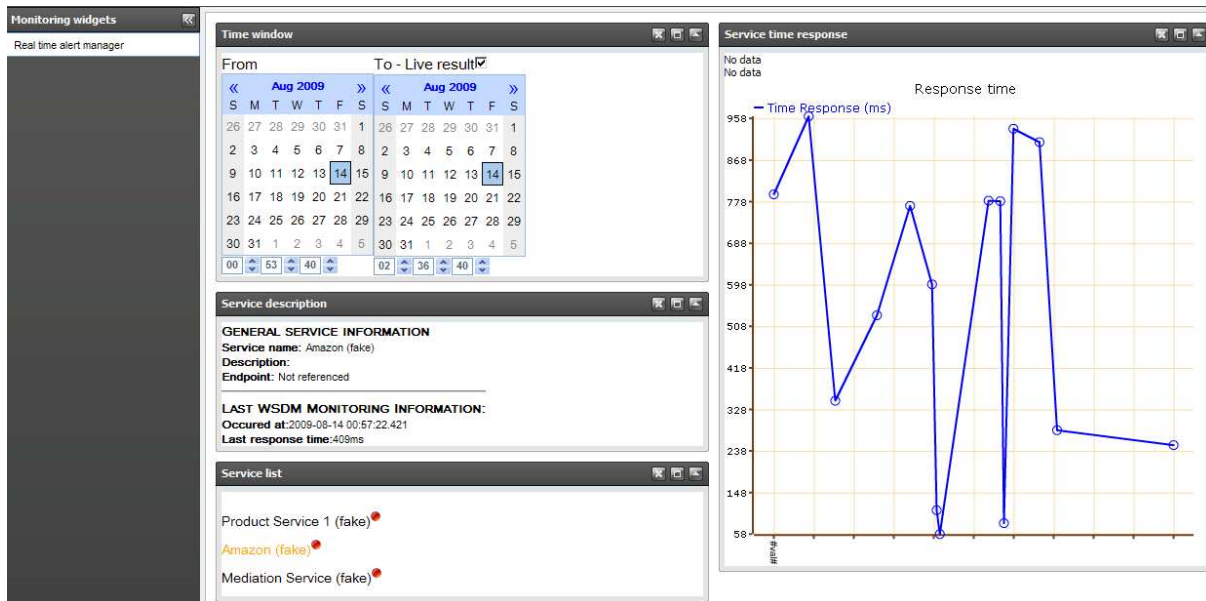


Figure 11: Monitoring Dashboard - Example #2

3.6 Analysis Widgets

The following subsections present each individual widget available in the current version of the Analysis Platform Prototype.

3.6.1 Service Response Time History

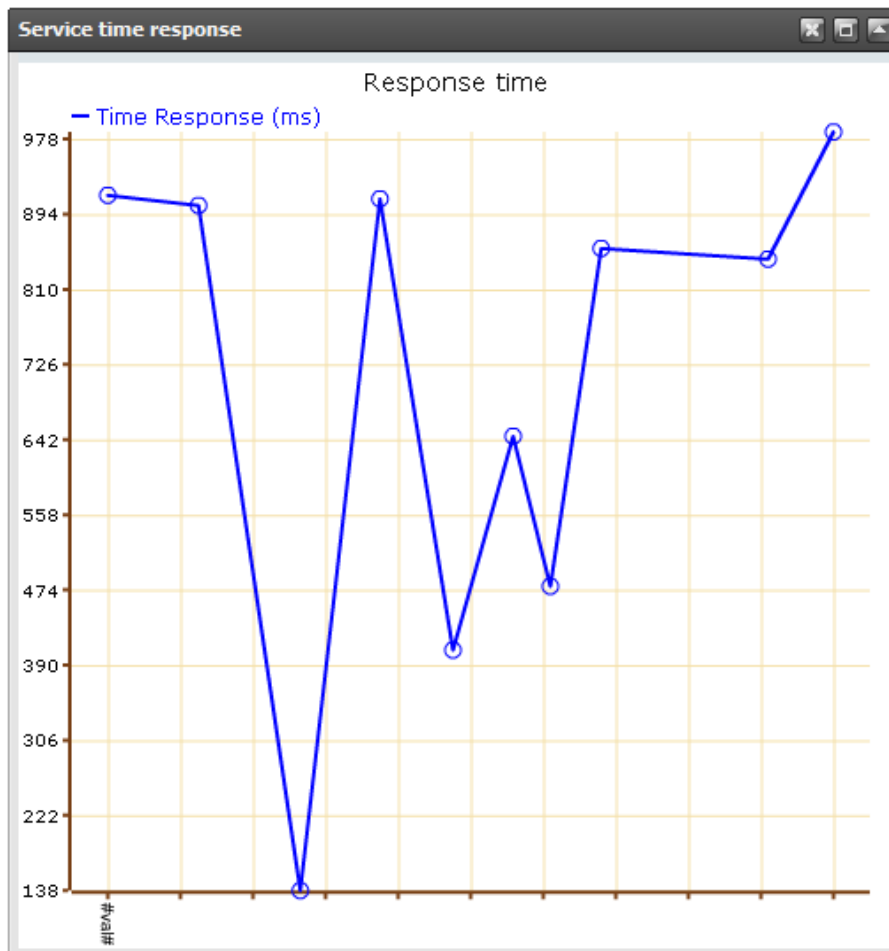


Figure 12. Service Response Time History Widget

This widget illustrated in Figure 12 shows the evolution of the response time of a service and can be automatically updated when a new event is received, using the Comet⁸ approach.

⁸ http://en.wikipedia.org/wiki/Comet_%28programming%29

3.6.2 Service List

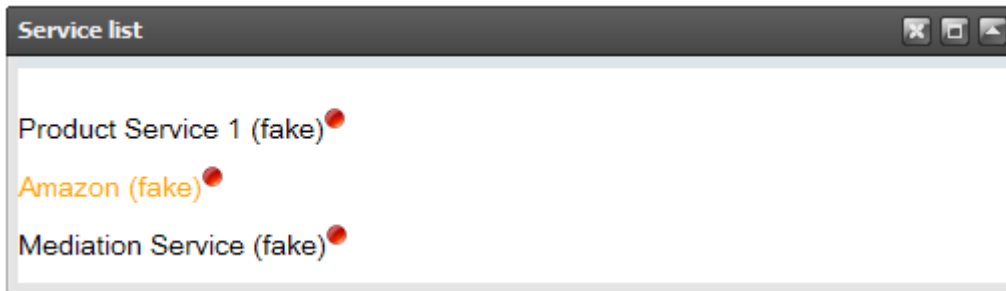


Figure 13. Service List Widget

This widget illustrated in Figure 13 lists the services relevant to the user (e.g. favorites or recently used). Clicking on the service name takes the user to the detail view for it, as detailed in Section 3.3. The availability indicator can be red or green depending on whether the service is unavailable or available respectively.

3.6.3 Service Description

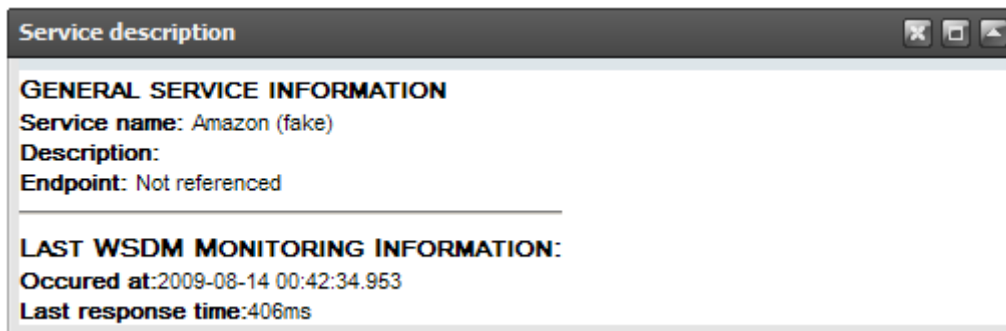


Figure 14. Service Description Widget

This widget shown in Figure 14 provides a textual description of a service. It displays performance information received using the WSDM standard. Note that Figure 14 does not show the entire range of information displayed by this widget.

3.6.4 Real-Time Alert Manager

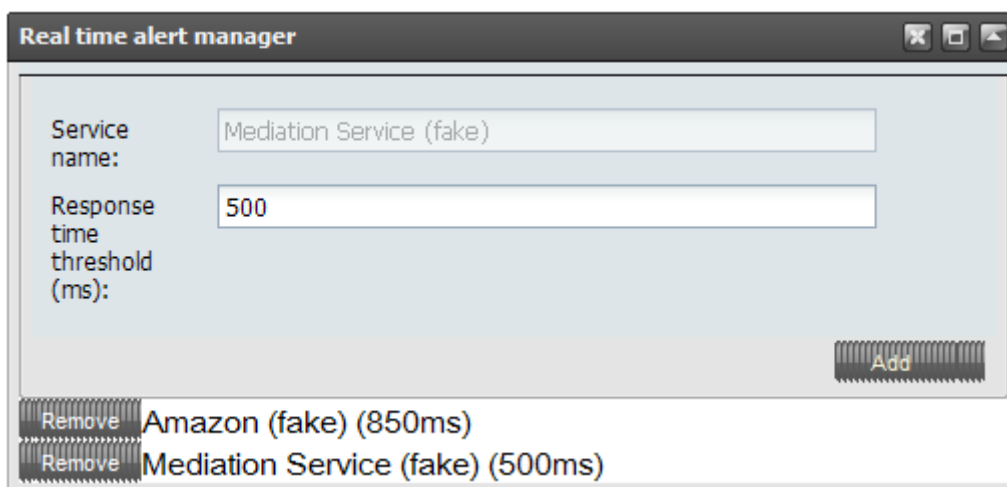


Figure 15. Alert Manager Widget

This widget shown in Figure 15 allows the definition of simple thresholds for service execution times. When the execution time of the service exceeds the specified threshold, the user is notified by a popup like the one in Figure 16 that an alert was raised.

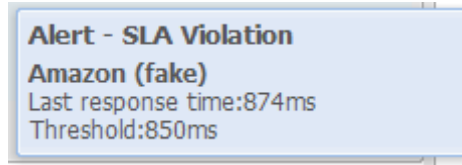


Figure 16. Alert Popup

3.6.5 Time Window Selector

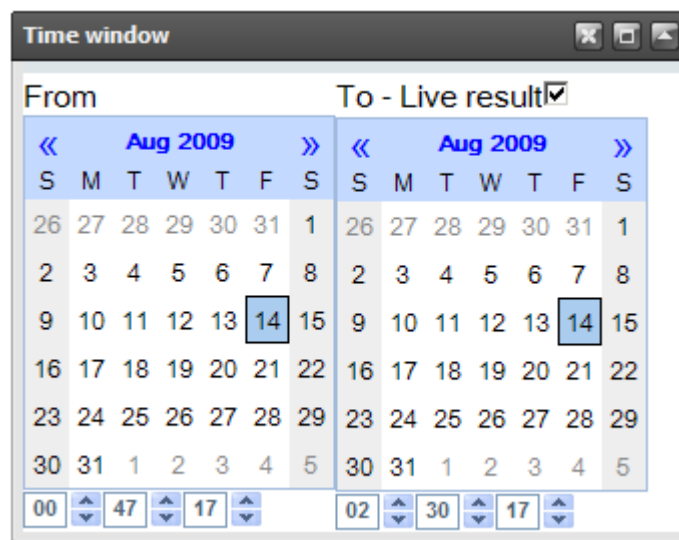


Figure 17. Time Window Selector Widget

This widget shown in Figure 17 allows the precise definition of the period of interest for historical data (such as for the Service Response Time History). It is used to select a start date and an end date as well as the time of the day for both dates with second-precision. Widgets that need intervals to display or compute data take this interval into account in their display. The "Live Result" tick-box available above the end-date selector is used to request that values be updated in "real-time" as they arrive, when they are not already available in the Analysis Warehouse.

3.6.6 Process Overview

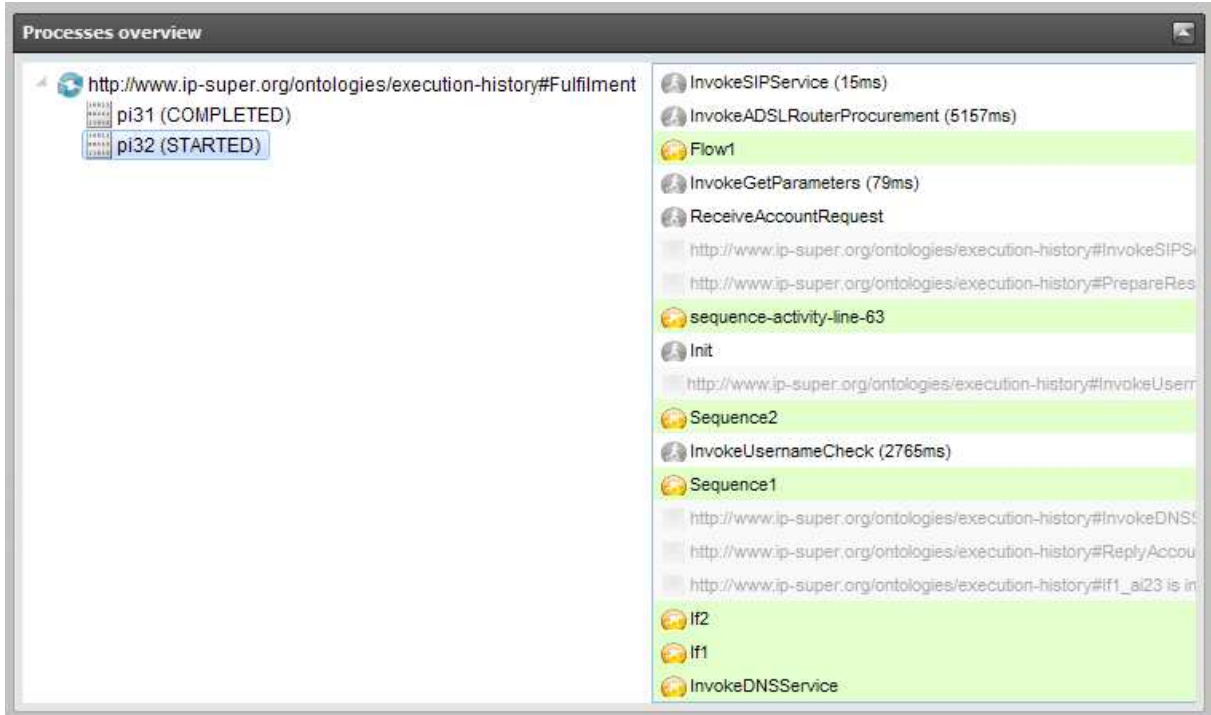


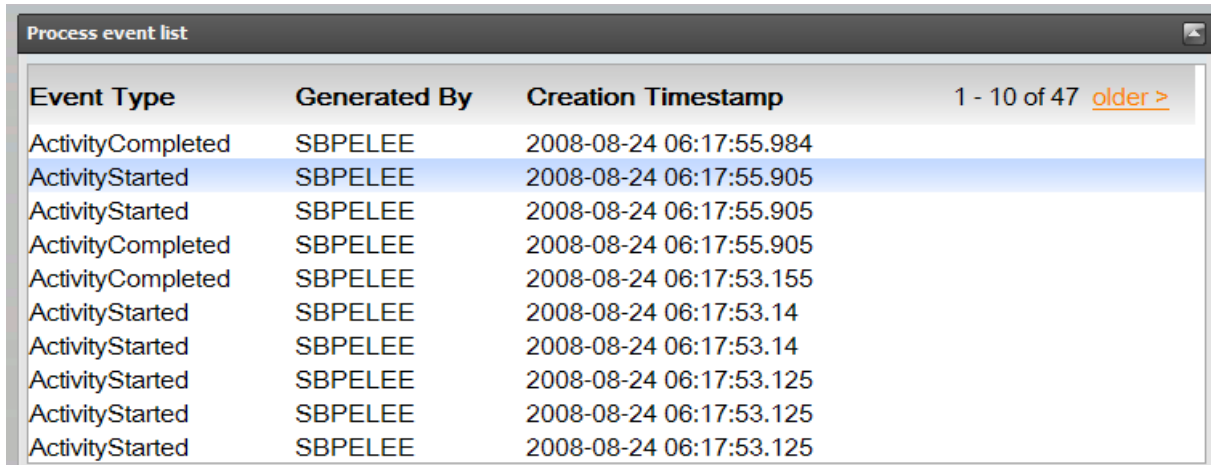
Figure 18. Process Overview Widget

The Process Overview Widget shown in Figure 18 displays a “real-time” overview of process execution. It uses a tree-based hierarchy to illustrate processes and their instances that are being executed, as well as those that have finished execution. For each process instance the user can observe the evolution of its activities as they happen.

	Indicates a Process
	Indicates a Process Instance
	Denotes an Activity Instance which is never called by the current process instance
	Indicates that the Activity is ongoing
	Indicates that the Activity has finished execution

Figure 19. Process Overview Widget Legend

3.6.7 Process Event List

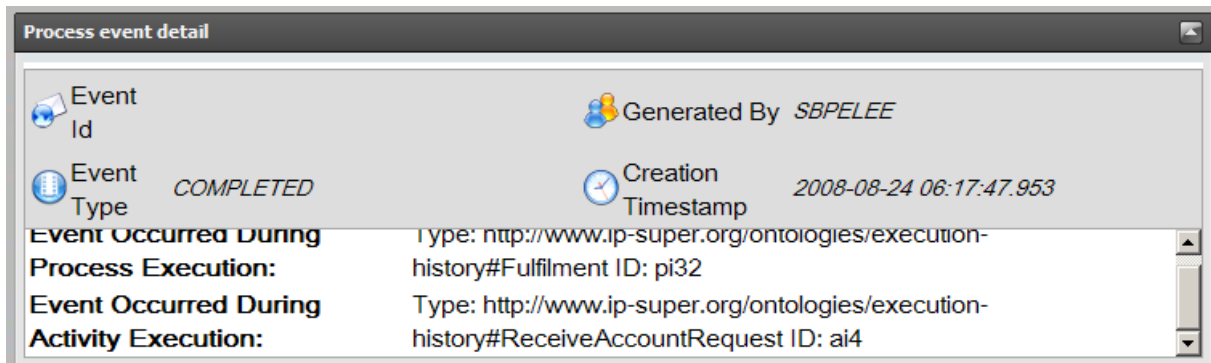


Event Type	Generated By	Creation Timestamp	1 - 10 of 47 older >
ActivityCompleted	SBPELEE	2008-08-24 06:17:55.984	
ActivityStarted	SBPELEE	2008-08-24 06:17:55.905	
ActivityStarted	SBPELEE	2008-08-24 06:17:55.905	
ActivityCompleted	SBPELEE	2008-08-24 06:17:55.905	
ActivityCompleted	SBPELEE	2008-08-24 06:17:53.155	
ActivityStarted	SBPELEE	2008-08-24 06:17:53.14	
ActivityStarted	SBPELEE	2008-08-24 06:17:53.14	
ActivityStarted	SBPELEE	2008-08-24 06:17:53.125	
ActivityStarted	SBPELEE	2008-08-24 06:17:53.125	
ActivityStarted	SBPELEE	2008-08-24 06:17:53.125	

Figure 20. Process Event List Widget

This widget illustrated in Figure 20 displays in table format the headers of raw monitoring events received from the process engine. When clicking on an event from the list, the Process Event Detail Widget described below is updated to correspond with the selected event.

3.6.8 Process Event Detail



Event Id	Generated By SBPELEE
Event Type <i>COMPLETED</i>	Creation Timestamp 2008-08-24 06:17:47.953
Event Occurred During Process Execution:	Type: http://www.ip-super.org/ontologies/execution-history#Fulfilment ID: pi32
Event Occurred During Activity Execution:	Type: http://www.ip-super.org/ontologies/execution-history#ReceiveAccountRequest ID: ai4

Figure 21. Process Event Detail Widget

This widget illustrated in Figure 21 displays the details of a raw event selected in the Process Event Detail widget. The properties type, origin, creation time and concerned instances of business processes or activities, can be found in this widget.

3.6.9 Process Time Line

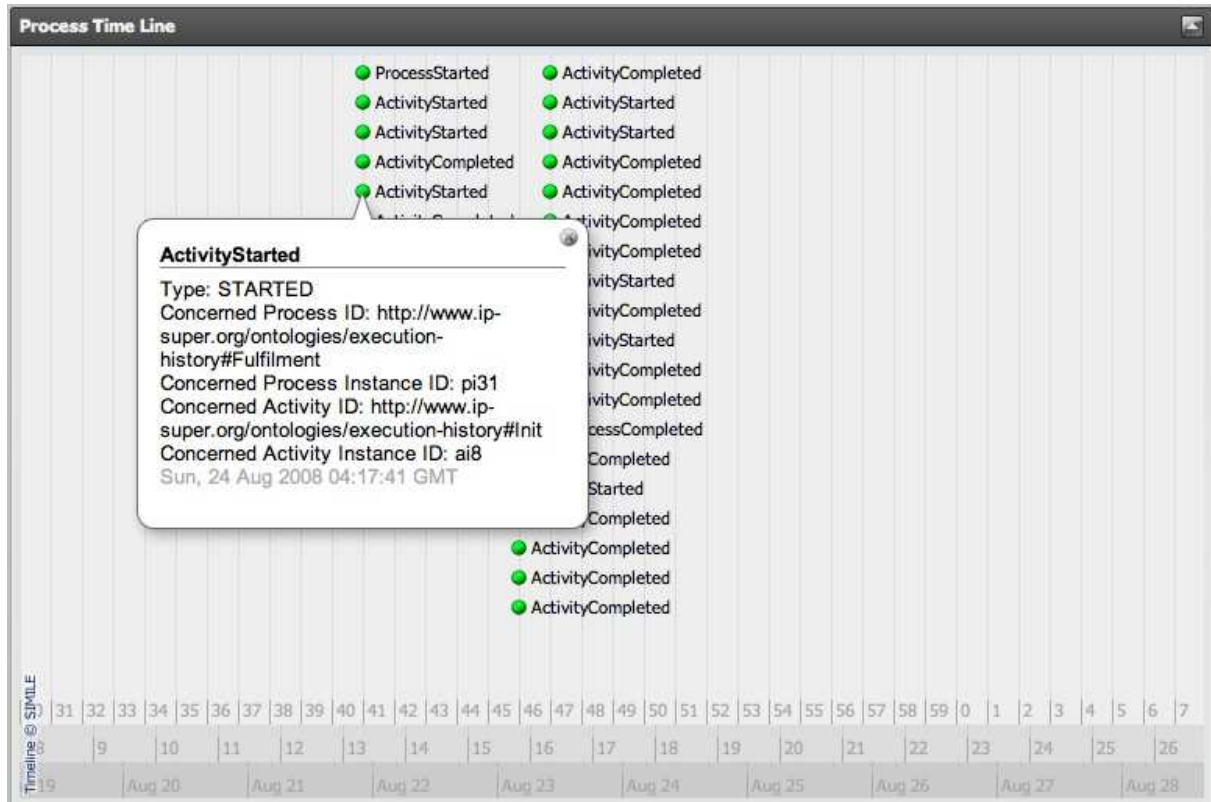


Figure 22. Process Time Line Widget

As illustrated by Figure 22, the Process Time Line widget visualizes both the raw events and the instances of business processes or activities through the metaphor of time line. Every small circle in Figure 22 represents a monitoring event, whilst every bar symbolizes an instance of process or activity. When clicking on an event or an instance, more details appear in the pop-up window. This widget is implemented based on the SIMILE Time Line⁹, which supports visualization of both instantaneous and continuous artefacts.

3.6.10 Knowledge Analytics manager

The Knowledge Analytics (KOPE*) component can be configured to track the information of a service over time, in a way that the necessary computations take place in a regular basis from the moment of its configuration.

This widget permits configuring new tracks on services, and then opening an additional widget (explained next) for each of the services being tracked. Figure 23 depicts the setting of a new track on a service, and the possibility to select one of the three already configured by clicking on the “Open” button.

⁹ <http://www.simile-widgets.org/timeline/>

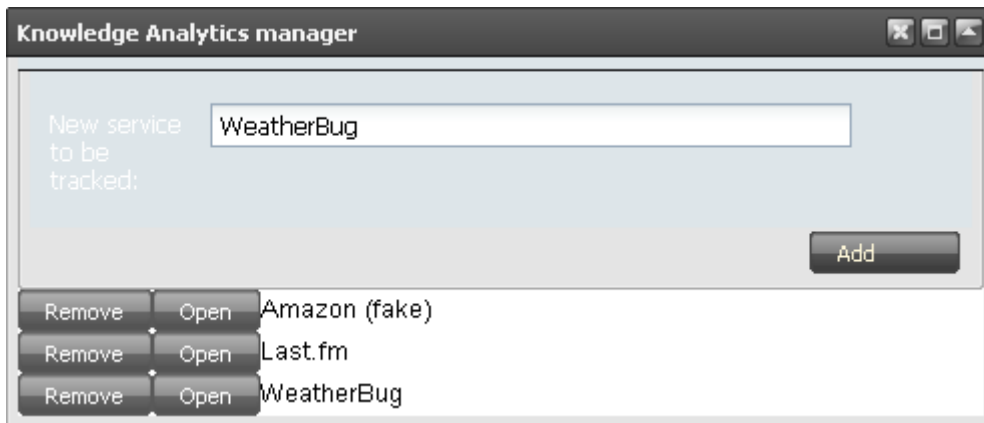


Figure 23. Knowledge Analytics Manager Widget

3.6.11 Knowledge Analytics for a Service

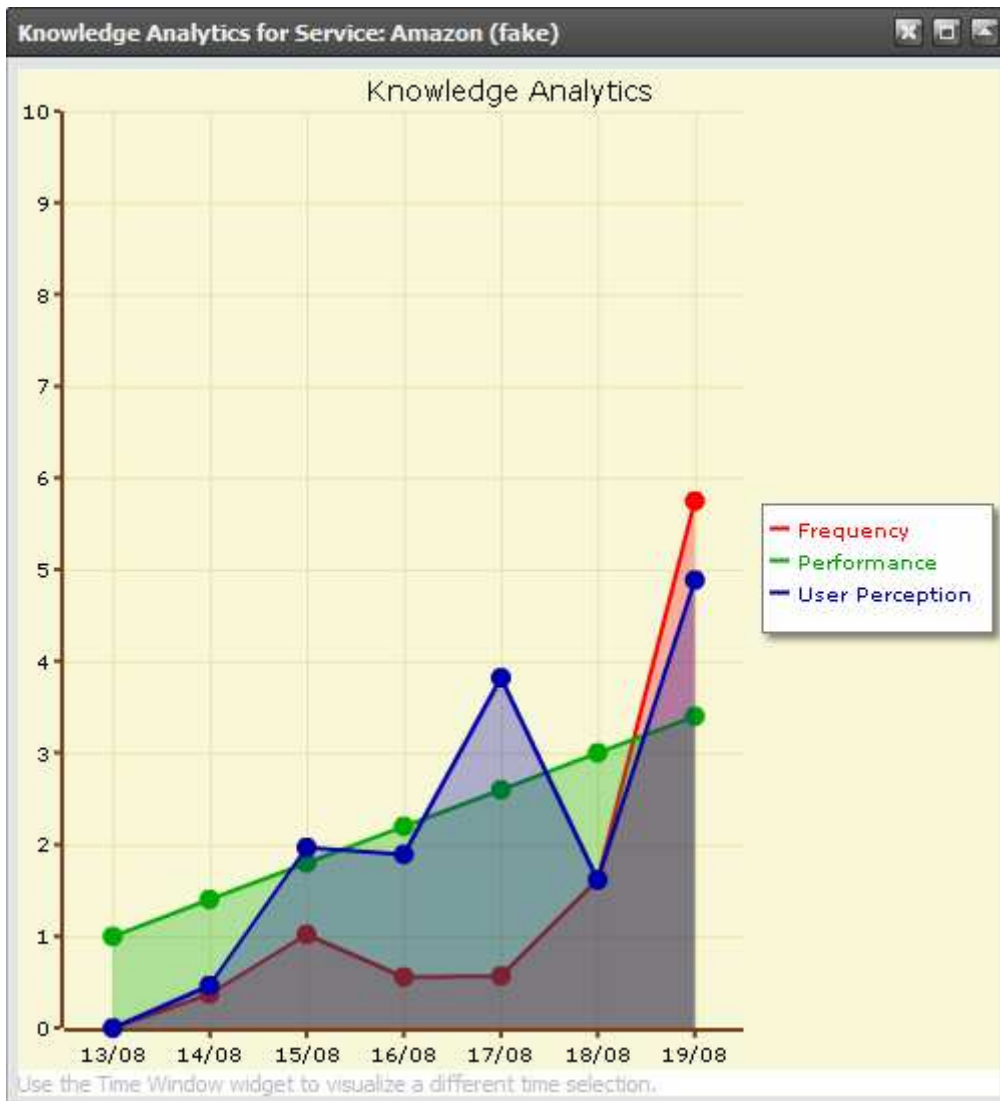


Figure 24. Knowledge Analytics (for a specific service) Widget

This widget displays the Knowledge Analytics for a selected service that has been previously

set to be tracked by the previous widget. It displays the three main concepts (frequency, performance and user perception) already computed by the component using a time scale. Figure 24 depicts this graph where those concepts can be compared over time.

It is also possible to expand each of those concepts in order to see with more detail the evolution of the related lower-level concepts over time.

It is worth noting that the time scale used by this widget can be modified by using the Time Window Selector widget addressed in Section 3.6.5.

3.7 DEMO Generator

This section refers to functionality that is not directly part of the prototype but which is used to illustrate the functionality of the prototype. The Analysis Platform prototype needs data obtain through its data sources in order to be able to display analysis information through its widgets.

We have built a Demo Generator application that is responsible for generating data to be used when “real” data arriving from the SOA4All Infrastructure is not available. This application generates data that corresponds to the WSDM and EVO event sources and it presents a simple user-interface that can be used to control the generation independently of the Studio pages of the Analysis Platform. This interface, illustrated in Figure 25, is available through a web page found at this relative address: [http://\[SOA4All Studio Address\]/DemoHelper/](http://[SOA4All Studio Address]/DemoHelper/).

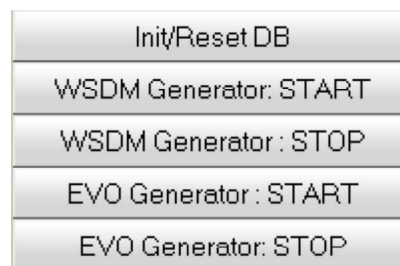


Figure 25. DEMO Generator Interface

The first button is used to clear the internal DB used by the BEP, in order to start with fresh data. The other buttons marked with START and STOP for WSDM and EVO control the event flow for these two sources.

4. Installation & Setup

In order to test the functionality of the Analysis Platform prototype, the SOA4All Studio needs to be installed. The procedure for the installation and setup of the Studio is described in Section 4 of the D2.4.2 deliverable. Once the Studio is running, the Analysis Platform can be accessed through the Dashboard as indicated in Section 3.1.

Note that the Analysis Platform prototype can currently be tested by simply pointing your browser to

<http://coconut.tie.nl:8080/soa4all>

where the Studio currently runs.

When installing on a new machine, apart from the indicated procedure for installing the Studio, an extra step that is required for the first prototype of the Analysis Platform to run is the installation of a local DB software that is used internally by the BEP to buffer monitoring events. This software, called “hsqldb” is available for download at <http://hsqldb.org>. We have tested the prototype with version 1.8.0 of hsqldb, which needs to be downloaded and installed with all its default values on the system that runs the Studio. The installation involves unpacking the downloaded archive. Once the software is installed, it needs to be started with the following command:

```
"java -cp hsqldb.jar org.hsqldb.Server -database.0 file:monitoringdb
-database.0 monitoringdb"
```

Once the hsqldb server is started, the generation of events can be controlled through the prototype DEMO generator as described in Section 3.7 and results observed in the appropriate pages as described throughout Section 3.

SwiftOWLIM serves as the RDF repository for the demonstration of the monitoring and management tool. In order to install SwiftOWLIM, the following pre-requisites should be installed and available:

- Java Virtual Machine 1.5 or later
- Tomcat 6.0 or later

Before installing SwiftOWLIM, Sesame should be installed on Tomcat by deploying the 'openrdf-sesame.war' and 'openrdf-workbench.war', which can be found in the binary distribution of Sesame. As a plug-in for Sesame, OWLIM is installed under Sesame by simply copying 'owlim.jar' and 'tree.jar', which can be found in the lib directory of the SwiftOWLIM distribution, to the directory 'openrdf-sesame/WEB-INF/lib'. In this way, SwiftOWLIM can run in the remote access mode.

The installation of SwiftOWLIM is accomplished by taking the above steps. Here, we explain how to create the semantic repository:

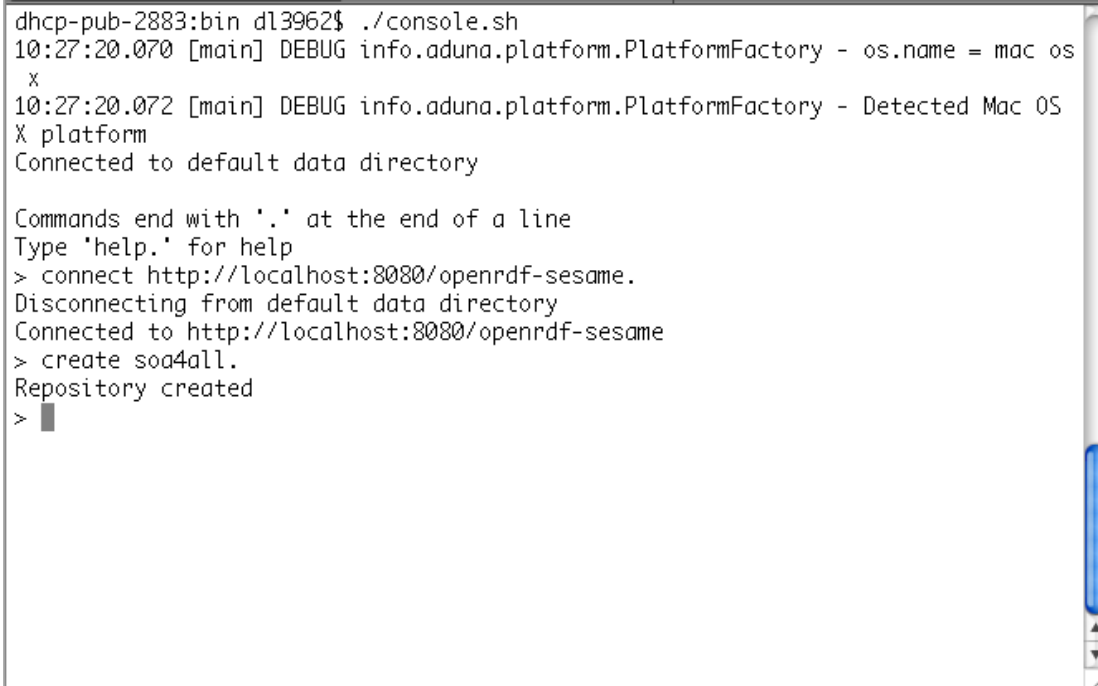
- Browse the data directory of OWLIM, and create a new directory 'templates' there.
- Create a file named 'soa4all.ttl' under the 'templates' directory, and add the following contents to the file:

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix rep: <http://www.openrdf.org/config/repository#>.
@prefix sr: <http://www.openrdf.org/config/repository/sail#>.
@prefix sail: <http://www.openrdf.org/config/sail#>.
@prefix owl: <http://www.ontotext.com/tree/owl#>.
```

```
[ ] a rep:Repository ;
    rep:repositoryID "soa4all" ;
```

```
rdfs:label "SOA4All Monitoring Data Repository" ;
rep:repositoryImpl [
  rep:repositoryType "openrdf:SailRepository" ;
  sr:sailImpl [
    sail:sailType "swiftowlim:Sail" ;
    owl:ruleset "owl-max" ;
    owl:partialRDFS "true" ;
    owl:noPersist "true" ;
    owl:storage-folder "monitoring-data-storage" ;
    owl:base-URL "http://www.ip-super.org/ontologies/execution-
history#" ;
    owl:new-triples-file "new-triples-file.nt" ;
    owl:entity-index-size "200000" ;
    owl:jobsize "200" ;
    owl:repository-type "in-memory-repository" ;
    owl:defaultNS "http://www.ip-super.org/ontologies/execution-
history#"
  ]
].
```

- Copy the 'templates' directory to 'OpenRDF Sesame console' directory, which is automatically created by running the command-line based Sesame console at the first time.
- Run Sesame console, connect to the repository, and create the soa4all repository by command 'create soa4all.' as shown in Figure 26.



```
dhcp-pub-2883:bin dl3962$ ./console.sh
10:27:20.070 [main] DEBUG info.aduna.platform.PlatformFactory - os.name = mac os
x
10:27:20.072 [main] DEBUG info.aduna.platform.PlatformFactory - Detected Mac OS
X platform
Connected to default data directory

Commands end with '.' at the end of a line
Type 'help.' for help
> connect http://localhost:8080/openrdf-sesame.
Disconnecting from default data directory
Connected to http://localhost:8080/openrdf-sesame
> create soa4all.
Repository created
>
```

Figure 26. Sesame Control Console

5. Conclusions and Next Steps

This document presented the software deliverable of the first prototype of the Analysis Platform. It can be noted that significant functionality is available in this implementation, demonstrating successful integration within the overall SOA4All Studio as well as feasible targets for the different components of the AP.

There are however important remaining developments for the AP. Different editors, such as the process editor from T2.6 need to use information generated by the AP to augment their graphical displays. The APIs that are available and that were presented in Section 2 may need to be refined to correspond to this need. The AP also needs to properly handle user identity in order for the presentation of information to correspond to each user's services and processes as well as to the customized dashboard.

The AP needs to consolidate the usage of the semantic storage in conjunction with the needs of other tasks and WPs. More information is potentially required to be placed in this storage; it may also need to collect more information from the storage to compute more complex metrics.

One of the main remaining challenges however refers to the scalability ambitions of SOA4All, which requires the AP to scale accordingly. The current prototype already supports a distributed environment and it will need to be tested and potentially refactored to correspond to the wide distribution of its data sources both from functional as well as non-functional points of view (i.e. dealing with large amounts of data in information presentation as well as ensuring that the AP infrastructure can cope with the demands).