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

D2.3.3 Service Monitoring and Management Tool Suite Second Prototype

- Prototype Documentation -

Activity N:	Activity 1	
Work Package:	WP2 – SOA4All Studio T2.3 – SOA4All Analysis Platform	
Due Date:	M18	
Submission Date:	04/09/2009	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	INRIA	
Revision:	1.0	
Author(s):	Adrian Mos	INRIA
	Carlos Pedrinaci	OU
	Guillermo Álvaro Rey	ISOCO
	Iván Martínez	ISOCO
	Christophe Hamerling	EBM
	Dong Liu	OU
	Samuel Quaireau	INRIA
	Fy Ravoajanahary	INRIA
Internal Reviewers	Sven Abels	TIE
	Patrick Un	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	21.07.2010	Kick-Off Version	INRIA
0.2	03.08.2010	Added content from first drafts from INRIA, OU and ISOCO.	INRIA, OU, ISOCO
0.3	04.08.2010	Added info on Monitoring Configuration and basic Installation procedure	INRIA
0.4	10.08.2010	Knowledge Analytics section added	ISOCO
0.5	11.08.2010	DSB Monitoring Console section	EBM
0.6	13.08.2010	Overall cleanup and conclusions	INRIA
0.7	27.08.2010	Integrated Reviewers’ comments	INRIA

Table of Contents

List of Figures

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
REST	Representational State Transfer
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 D2.3.3 of the second prototype of the SOA4All Analysis Platform. It describes the functionality of the prototype and places the implementation in the architectural context described by the previous deliverables D2.3.1 and D2.3.2.

The main additions to the first prototype include updated user-interaction functionality, scalability improvements and better integration with other SOA4All infrastructure components. Long-term storage of analysis data using RDF is an important addition to the current set of functionalities and it enhances both scalability (through better data management strategies) and integration (through better data-usage opportunities by third parties).

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; and D2.3.2 which presented the functionality of the first prototype of the AP. Therefore, this document focuses on the new developments since D2.3.2 as well as changes to the architecture due in particular to scalability considerations. The document is to be used in conjunction with D2.4.3 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 while the actual deployment of the AP is distributed, with a common Studio core component and an AP-only component running on different sites.

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 prototype implementation in the context of the updated AP architecture. It discusses the scalability and data warehouse topics and presents a set of RESTful APIs made available to developers. However, Section 2 does not go into details about such APIs as such details can easily be obtained from associated documents that will be made available with the software packages.
- Section 3 provides a description of the functional improvements in the AP since the first prototype. Older functionality, which is still available in the new prototype, is not described and readers can refer to D2.3.2 for information on the variety of widgets and views available in the AP.
- Section 4 describes the changes to the installation and execution procedures in the new prototype. These changes are due to the fact that the Studio is now decoupled which assumes separate installations on remote machines of each of the Studio components. A separate, complete document will be made available containing installation instructions for the entire Studio, so this section simply highlights the main points related to the AP.

2. Architectural Considerations

As a natural evolution from the first Analysis Prototype, the second prototype has been enhanced with a focus on scalability and integration, in addition to new functionality. The additional functionality is presented in Section 3, while this section presents the main architectural differences.

Scalability requirements are addressed primarily by carefully choosing appropriate data-handling strategies to process data coming from the various data sources. The implementation of an Analysis Warehouse is of major importance in this regard. Integration refers mainly to the fact that, owing to the more advanced maturity of the various runtime components, we can now extract data being produced by remote DSB nodes and remote execution engines. So by integration here we mean better connection to the actual SOA4All runtime components.

2.1 Scalability Overview

This section briefly discusses scalability aspects that are being addressed in the current Analysis Platform prototype. The discussion is based on a simple example illustrated in Figure 1.

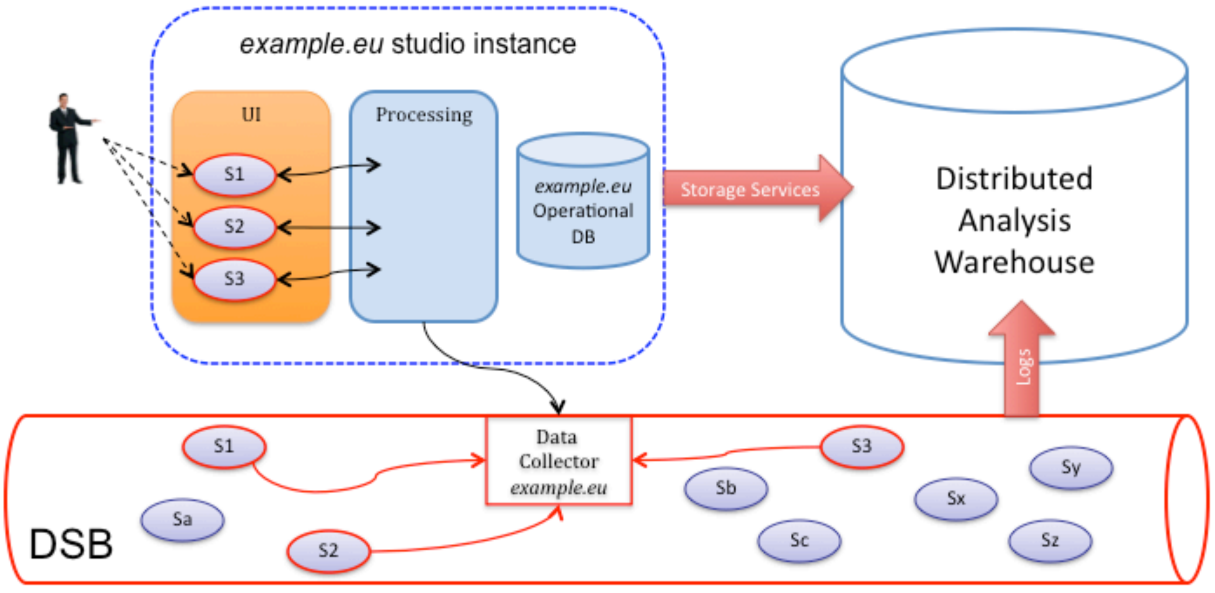

example.eu studio instance
UI
Processing
S1
S2
S3
example.eu Operational DB
Storage Services
Distributed Analysis Warehouse
Logs
Data Collector example.eu
S1
Sa
S2
S3
Sb
Sc
Sx
Sy
Sz
DSB

Figure 1. Scalability Considerations

In the scenarios envisaged by SOA4All, the Analysis Platform, together with other Studio components, will potentially be deployed on many nodes, each administered by an organisation that wants to provide SOA4All functionality to its users. However, the invisible SOA4All infrastructure based on the distributed service bus can be seen as transcending individual providers and offering a pervasive infrastructure base that is accesible from any SOA4All-enabled node. This is also true for the RDF storage support available as a pervasive service for SOA4All nodes. This pervasive aspect is realised through different federation techniques that are described in various deliverables of WP1.

This architecture however has important implications with regard to scalability of the Analysis Platform. In order to cope with the extremely large numbers of users envisaged by SOA4All as a whole, as well as potentially massive amounts of data, we leverage the natural distribution provided by the infrastructure and optimize data processing and storage to achieve a good compromise in terms of performance and scalability. Figure 1 shows a sample instance of the Studio called *example.eu*. This Studio instance runs its own individual Analysis Platform instance, separated from other instances available with other providers. Users connecting to *example.eu*'s Studio will access this version of the Analysis Platform and will be interested, as a group, in a relatively limited number of services and processes (that are relevant to *example.eu*'s business). This implies that all the processing that this instance of the Analysis Platform will perform will relate to these services (illustrated in the image as S1, S2 and S3). Its internal operational database will contain detailed analysis data for these items. This prevents this instance from processing data for services that are not of interest to its users (in effect, only services that at least one user of the domain is interested in, will be analyzed).

Naturally, example.eu, as all the other SOA4All-enabled nodes running the Studio, will connect to the DSB and will leverage the highly-distributed RDF storage. This enables the collection of data for any service or process executing anywhere in the world, when such data is required. We envisage that even for services/processes that no user has expressed interest in yet (in any domain instance), the DSB will collect basic data such as moving averages for execution times and availability information, and store it in the RDF storage, in order to have minimum bootstrap information ready when users become interested in the particular service/process. As soon as they have become interested, data collection becomes much more significant as it is driven by the Analysis Platform instance, and detailed analysis data can be stored in the individual operational databases. In short, detailed analysis for selected entities is performed “locally” in the same domain as the user, and basic analysis and long-term storage is performed on the distributed infrastructure for all entities.

2.2 RESTful Services

In order to expose Analysis Platform functionality to any user, we have created a number of RESTful services that are annotated and registered in the iServe repository. This enables any interested party to invoke the functionality of the AP and obtain analysis information about different services used by SOA4All users.

The table below presents the list of RESTful APIs currently available.

Operation Name	Input	Output
AverageResponseTime Average Response Time	service name	Average Response Time
Availability Availability since the given	service name + start date	Availability since the given date

date		
Availability Last availability status	service name + LAST	Last availability status
Availability Availability between two dates	service name + start date and end date	Availability between two dates
Information about a service	service name	Information about a service
Availability Global availability	service name	Global availability

The RESTful invocation pattern is the following:

http://soa4all.inrialpes.fr/monitoring/services/**serviceName**/**OperationName**/**Parameters**

Ex1:

When calling just http://soa4all.inrialpes.fr/monitoring/services/**serviceName** we obtain basic information about the service.

Ex2:

When calling http://soa4all.inrialpes.fr/monitoring/services/**serviceName/Availability/20091013232354/2011.10.20** we obtain the availability between the 2 given dates expressed in 2 different formats.

2.3 Decoupling from Studio Core

The Analysis Platform has been completely decoupled from the main Studio code-base. This is part of an effort to distribute and manage the different components of the SOA4All Studio more effectively and the details of this approach and its advantages over the previous prototype are explained in detail in the Second Studio Prototype Deliverable D2.4.3. The implications of the decoupling primarily relate to the way the AP is installed, since it does not come as integral part of the Studio anymore. Functionality-wise, it has preserved all the characteristics of the previous prototype detailed in D2.3.2. Section 4 outlines some of the installation implications of this decoupled approach, but a full installation guide for the Studio will be made available by the SOA4All Consortium as a separate document.

2.4 Storage in Analysis Warehouse

As shown in Figure 2, the analysis warehouse is made up of two parts: analytical data storage and analytical data publication. The former offers a persistence mechanism for the data, events and logs that are collected from the DSB, whereas the latter publishes analysis results as Linked Data.

In short, what the storage component of analysis warehouse does is translating raw data into RDF triples conformant to the COBRA and EVO ontologies [1], and saving to the RDF repository. RDF2Go [2] is used to make it easy for analysis warehouse to access the RDF repository. Accordingly, RDFReactor [3], a new code generation tool, replaces Elmo, which is previously employed to integrate with SENTINEL. [D2.3.2]

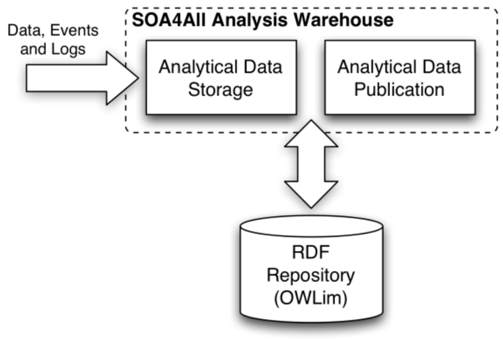


Figure 2. Overall Architecture of Analysis Warehouse

In the rest of this subsection, we focus on how to publish analysis results as Linked Data, i.e. the publication component of analysis warehouse. First, we describe the conceptual model and organization of analytical data, and then detail the architecture and implementation of analysis warehouse as well as its RESTful API.

2.4.1 SDMX-RDF

Statistical Data and Metadata eXchange (SDMX) [4] is an ISO standard for exchanging and sharing statistical data and metadata among organizations. It consists of an abstract information model (SDMX-IM) and concrete XML- and UN/EDIFACT- based syntaxes. SDMX was however not devised having the Web in mind and as a consequence concepts, datasets or observations cannot be uniquely identified on the Web through URIs nor can this data be easily manipulated by Web browsers. This fact prevents the discovery, combination and correlation of SDMX data over the Web in a convenient manner. These requirements are particularly important notably to our use cases and in inter-organisational scenarios such as those envisioned in SOA4All use cases.

SDMX-RDF [5] is an ongoing effort trying to adapt SDMX to publish statistical data and metadata in RDF following Linked Data principles. By publishing statistical data in RDF, SDMX-RDF provides URIs to the metadata and data hold within SDMX warehouses for their linking, combination but also more advanced analysis paving the way for the application of provenance analysis techniques.

SDMX-RDF employs a hypercube for statistical data representation. Essentially, statistical data provides a collection of observations made about certain aspects considered relevant and along a number of dimensions. Dimensions identify what the observations apply to (e.g., individual, geographical region, etc). Additionally metadata is provided to describe what has been observed (e.g., unemployment rate, economic growth, etc) as well as how it was measured (e.g., estimation, unit of measure used, etc).

On the basis of this hypercube, data analysts can slice and dice the information in a large number of ways allowing them to detect correlations, or better assess the situation at different levels of granularity across dimensions and groups of observations.

Although at a relatively early stage, SDMX-RDF already defines means to structure statistical data as a hypercube in RDF. In a nutshell, SDMX defines DataSets that hold Observations (see Figure 3). All these observations conform to the DataSet specification, a DataStructureDefinition defined in terms of Components. These components can be Dimensions (i.e., what the observations apply to), Measures (i.e., what has been observed), and Attributes (i.e., additional metadata about the observations like the method used).

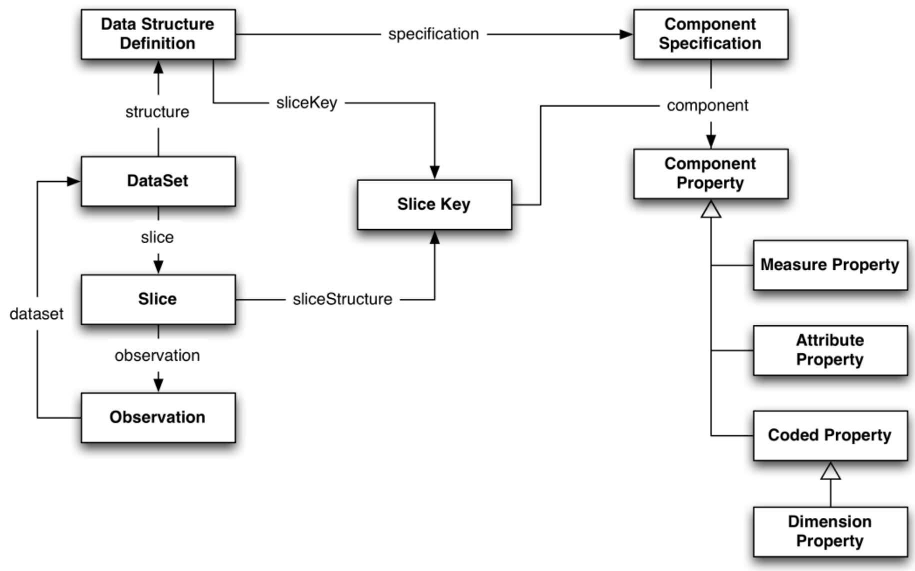


Figure 3. Key Concepts of SDMX-RDF

In order to dereference statistical data on the Web, a draft of URL structure is presented at [6]. Starting from it, we come up with several URL templates for retrieving analytical data stored in the warehouse, which are listed in the table below.

URL Template	Description
/dsd/{id}	A Data Set Description
/dsd/{id}/dimension	Dimensions within a DSD
/dsd/{id}/measure	Measurements within a DSD
/dsd/{id}/dimension/{did}	A dimension in a DSD
/dsd/{id}/measure/{mid}	A measurement in a DSD
/dataset/{id}	A data flow or a dataset
/dataset/{id}/observation/{oid}	An observation of a data set

2.4.2 Analytical Data as SDMX-RDF

The analysis warehouse is defined based on SDMX-RDF in order to capture statistical information concerning services, processes and operations in a way that better supports its publication on the Web and its automated processing. For the current version we have defined a number of DataSets holding individual, daily, weekly and monthly Observations. The Observations are captured along two Dimensions: the service Operation concerned, and the temporal properties, i.e. the Time Instant at which it was observed or Time Interval it spans. Here, Time Instant and Time Interval are both concept defined by the OWL-Time ontology [7].

Depending on the DataSet two different groups of Measures are captured which are subsets of the typical measures used for service monitoring [8]. For the individual analysis DataSet, we capture the Response Time, the Invocation Response, and the Invocation Result for each invocation carried out. The Response Time measures the time elapsed during the execution of an operation. The Invocation Response captures whether the operation responded or not. The Invocation Result captures what the result of the invocation was in order to detect whether there is an error.

The other DataSets use the finer grain DataSet to derive average results for the following measures: Response Time, Availability and Reliability. The former has the exact same meaning as earlier although it is here an average value rather than a direct Observation. Availability and Reliability are understood as per the formulas below:

$$\text{Availability} = \frac{Sucessf\,ulInvocations}{TotalInvocation} * 100$$

$$\text{Reliability} = 1 - \frac{ErrorMessages}{TotalMessages} * 100$$

Over time the analysis warehouse will capture large amounts of data that will be very valuable for analyzing the behaviour of services, for better ranking services according to certain performance measures, etc. We must however maintain a certain control over the size of the warehouse to avoid an information overload while retaining the sufficient level of granularity desired.

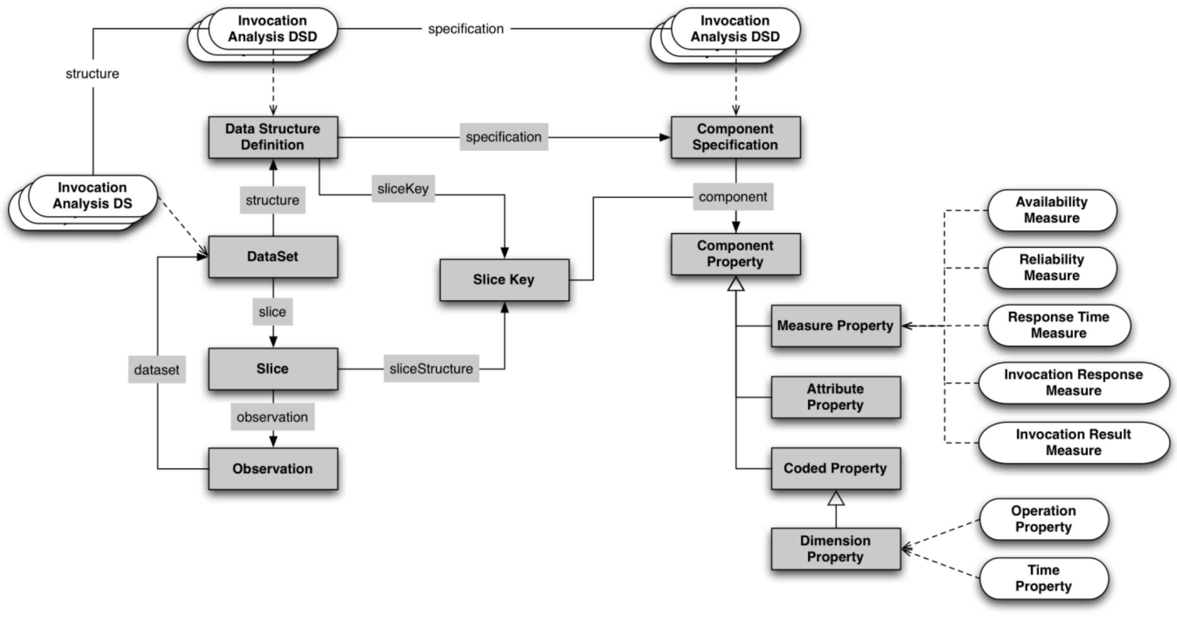


Figure 4. Dataset definition for Analysis Warehouse

The datasets defined for the analysis results are as follows (detailed overview in Figure 4):

- **Invocation Analysis DataSet:** This DataSet will hold Observations for every single invocation carried out over the last 2 days.
- **Daily Analysis DataSet:** This DataSet will hold daily aggregated data based on the Invocation Analysis DataSet. The data hold will be kept for the last 90 days.
- **Weekly Analysis DataSet:** This DataSet will hold weekly aggregated data based on the Daily Analysis DataSet for the last year, i.e., 52 weeks.
- **Monthly Analysis DataSet:** This DataSet will hold monthly aggregated data for the

last 5 years, i.e., 60 months.

2.4.3 Architecture and Implementation

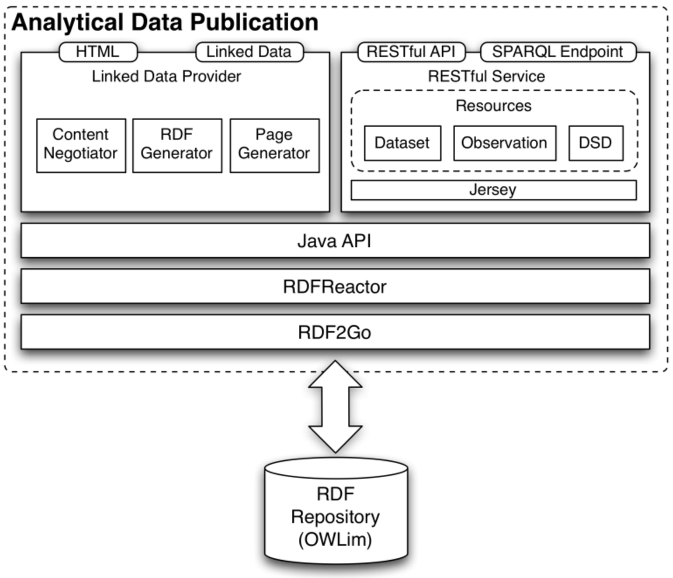


Figure 5. Architecture of Analytical Data Publication Component

Figure 5 depicts the overall architecture of the analysis warehouse. OWLim [9] serve as the repository for the RDF triples of analytical data. RDF2Go [2] provides an unified interface to various triple (and quad) stores. Here, RDFReactor [3] goes through it to get the access to the repository. RDFReactor performs two important functions: at design time, it automatically generates Java classes from RDFS ontololgies; at runtime, it bridges Java objects and RDF triples as well as literal values. On top of the Java API Linked Data is built upon the Java codes generated by RDFReactor. It serves as the provider and publisher of analytical data on the Web of Data. The RESTful service of analysis warehouse is realized under the framework of Jersey [10], which is a reference implementation of JSR-311 [11].

2.4.4 RESTful API

In this subsection, we document the RESTful interface of the analysis warehouse, which allows manipulating data sets and observations. In principal, Data Structure Definitions (DSDs) have already been well defined (refer to section 2.4.2), before the running of analysis warehouse. On the other hand, only system administrators can update the DSDs whenever necessary. Therefore, to ensure the consistency of analytical data at runtime, the analysis warehouse does not expose API for adding, modifying or deleting DSDs.

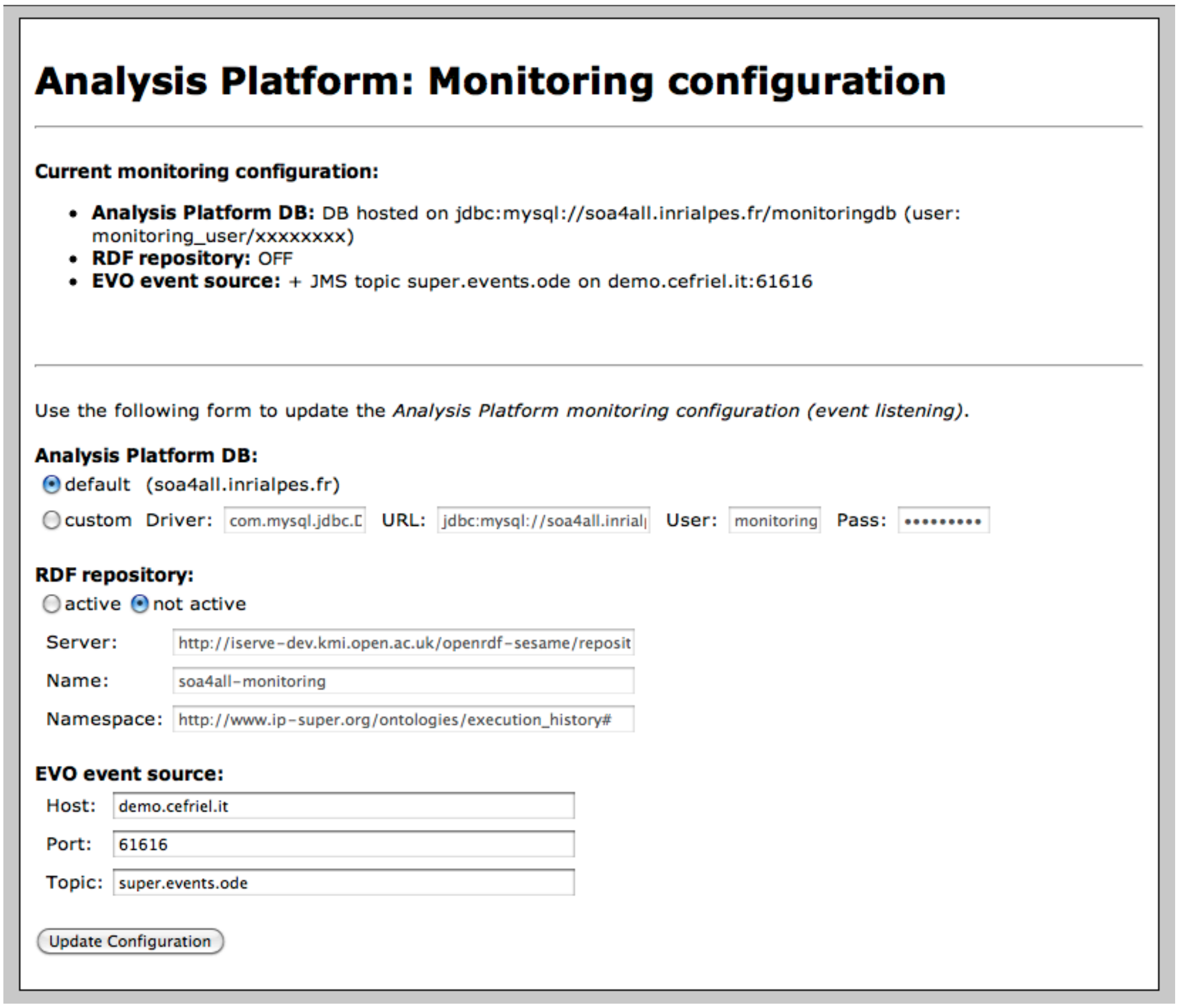
URL	HTTP Method	Parameter	Description
/dataset	POST	{dsd}: Mandatory. The URI of Data Structure Definition	Add a dataset
/dataset/{id}	DELETE	None	Delete a dataset
/dataset/{id}/observation	POST	{value} Mandatory. The observed value. {dataTypeURI}: Optional. The type	Add an observation to a dataset

		of observed value, by default xsd:decimal. {measure}: Mandatory. The URI of measurement. {dimensionURI}: At least one. The parameter name specifying the URI of dimension, is variable. {at} or {begin} and {end}: Time point or time period. A request must contain either an {at} parameter or both the {begin} and {end} parameters. To avoid time format issues, the time value must be in milliseconds.	
/dataset/{id}/observation/{oid}	DELETE	None	Delete an observation

3. New Prototype Functionality in the Studio

3.1 Monitoring Configuration Page

In order to better control the external data sources as well as the internal and external storage for monitoring data, a configuration page has been created. It is illustrated in



Analysis Platform: Monitoring configuration

Current monitoring configuration:

- **Analysis Platform DB:** DB hosted on jdbc:mysql://soa4all.inrialpes.fr/monitoringdb (user: monitoring_user/xxxxxxxx)
- **RDF repository:** OFF
- **EVO event source:** + JMS topic super.events.ode on demo.cefriel.it:61616

Use the following form to update the *Analysis Platform monitoring configuration (event listening).*

Analysis Platform DB:

[✓ default] (soa4all.inrialpes.fr)

custom Driver: com.mysql.jdbc. URL: jdbc:mysql://soa4all.inrial User: monitoring Pass: •••••••••

RDF repository:

active [✓ not active]

Server:	http://iserve-dev.kmi.open.ac.uk/openrdf-sesame/reposit
Name:	soa4all-monitoring
Namespace:	http://www.ip-super.org/ontologies/execution_history#

EVO event source:

Host:	demo.cefriel.it
Port:	61616
Topic:	super.events.ode

Figure 6. Monitoring Configuration Page

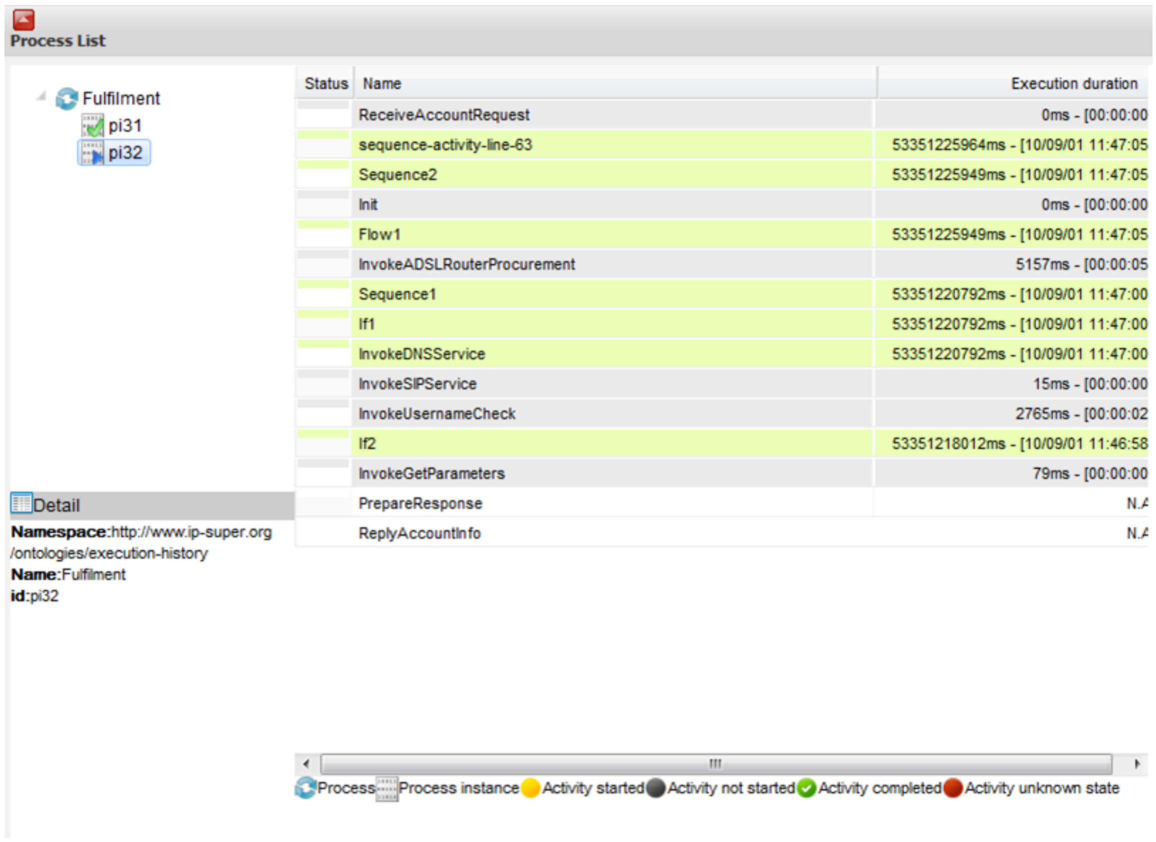
This page is currently accessible at:

http://soa4all.inrialpes.fr/monitoring-event-listener/config.jsp

However, for any new installation of the Analysis Platform instance, there will be such a corresponding page (see Section 4).

3.2 Updated Process Overview Widget

The Process Overview Widget shown in Figure 7 has been completely rewritten and improved from the previous version. The new version uses a table-view that is much clearer from a visual point of view and which can be ordered by different headers. The new version also has much better performance and uses less bandwidth as it significantly optimizes the amount of data being transferred via COMET from the BEP.


Process List

- Fulfilment
  - pi31
  - pi32

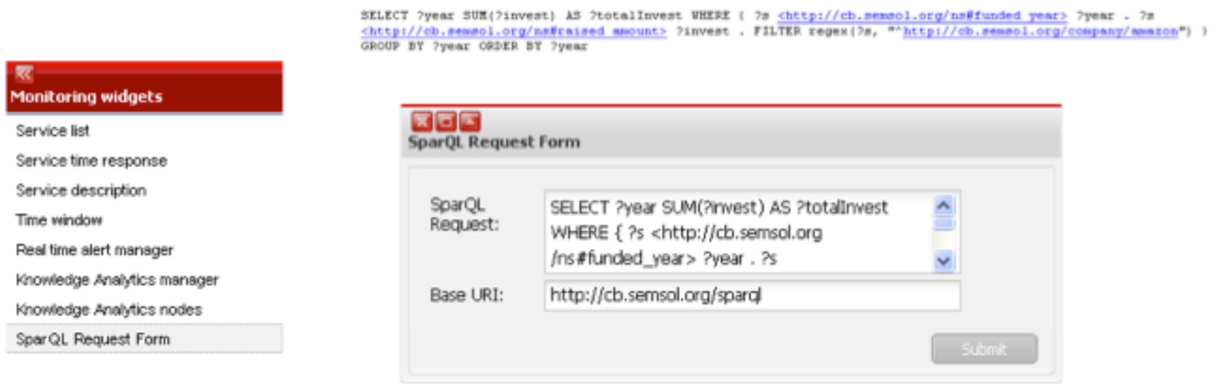
Status	Name	Execution duration
	ReceiveAccountRequest	0ms - [00:00:00
	sequence-activity-line-63	53351225964ms - [10/09/01 11:47:05
	Sequence2	53351225949ms - [10/09/01 11:47:05
	Init	0ms - [00:00:00
	Flow1	53351225949ms - [10/09/01 11:47:05
	InvokeADSLRouterProcurement	5157ms - [00:00:05
	Sequence1	53351220792ms - [10/09/01 11:47:00
	If1	53351220792ms - [10/09/01 11:47:00
	InvokeDNSService	53351220792ms - [10/09/01 11:47:00
	InvokeSIPService	15ms - [00:00:00
	InvokeUsernameCheck	2765ms - [00:00:02
	If2	53351218012ms - [10/09/01 11:46:58
	InvokeGetParameters	79ms - [00:00:00
	PrepareResponse	N.A
	ReplyAccountInfo	N.A

Detail
Namespace:http://www.ip-super.org/ontologies/execution-history
Name:Fulfilment
id:pi32

Figure 7. Updated Process Overview Widget

3.3 SPARQL Queries Widgets

We have developed widgets for invoking and displaying results of SPARQL Queries, in order to enable an “arbitrary” integration with data sources. The goal is to allow Analysis Platform users to connect to analysis information sources (or other data sources) and display different views. Figure 8 shows the Query Widget that can be used to write any SPARQL query and send it to the appropriate endpoint. The result is displayed in a new widget made available when the result has been retrieved and which is shown in Figure 9.


```
SELECT ?year SUM(?invest) AS ?totalInvest WHERE { ?s <http://cb.semsol.org/ns#funded_year> ?year . ?s
<http://cb.semsol.org/ns#raised_amount> ?invest . FILTER regex(?s, "^http://cb.semsol.org/company/amazon") }
GROUP BY ?year ORDER BY ?year
```

Monitoring widgets
- Service list
- Service time response
- Service description
- Time window
- Real time alert manager
- Knowledge Analytics manager
- Knowledge Analytics nodes
- SparQL Request Form

SparQL Request Form

SparQL Request:	SELECT ?year SUM(?invest) AS ?totalInvest WHERE { ?s <http://cb.semsol.org/ns#funded_year> ?year . ?s
Base URI:	http://cb.semsol.org/sparql

Figure 8. Basic SPARQL Query Widget

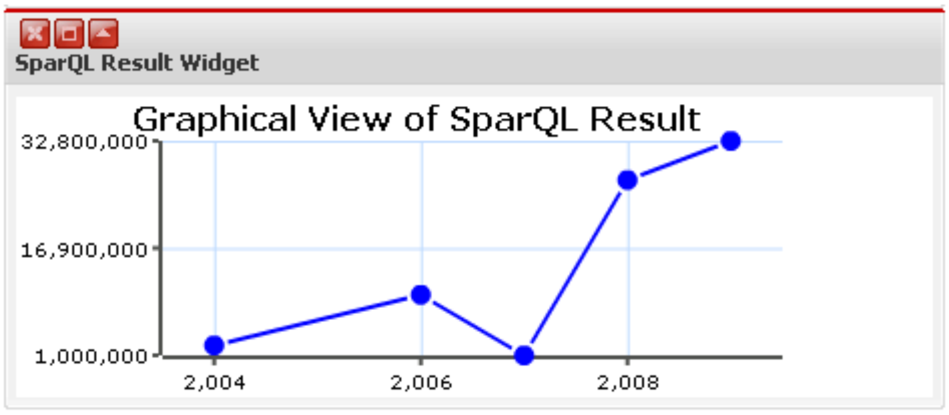


Figure 9. Displaying Results of a SPARQL Query

3.4 New Functionality: Knowledge Analytics Visualisation

The previous version of the Knowledge Analytics (k-Analytics, formerly known as KOPE*) module of the Analysis Platform, described in D2.3.2, dealt with the visualization of three different high-level metrics (Frequency, Performance and User Perception) over time, by making use of a visualization widget with the time in the x-axis and the values of each metric in the y-axis.

The evolution of the k-Analytics module into the second prototype described in this document stresses the importance of the semantic relations of services and concepts, combining them with the previously addressed metrics. The component is able to represent, through a node-based graph visualization, the main characteristics of any service (the concepts related to the service, its operations, and the input and output messages for each of those) stored in iServe [12], for it makes use of its SPARQL endpoint to retrieve the desired information.

Therefore, what the k-Analytics component is primarily providing now is an efficient visualization of Linked Data services by making use of its internal semantic relations, which permits end-users better understand by a simple glimpse the annotations of a service.

In the next subsections, more details are given about the data sources used (3.4.1), the way the desired information is retrieved (3.4.2), as well as examples of use of the module (3.4.3), public online prototype information (3.4.4) and the current limitations of the component (3.4.5).

3.4.1 K-Analytics Data Sources

As advanced, this version of the k-Analytics module combines information from two different sources:

- **iServe**: The repository of semantic annotations on services provides the necessary information about the relations amongst any service, its operations, and the input and output messages related to those.
- The SOA4All Studio **Storage Services**: The aggregated metrics are available as RDF and linked to the different services as well as to the concepts related through the sawsdl:modelReferences.

It is worth noting that thanks to the Linked Data access through the SPARQL endpoint of iServe, the k-Analytics component can be continuously updated with any new service annotations (i.e., when a service is annotated, it is automatically made available in this module for its visualization), at least in terms of the semantic relations.

3.4.2 K-Analytics Access to the Sources by SPARQL

Both the Storage Services and iServe expose its data through a SPARQL endpoint, which makes it easy for the k-Analytics component retrieve the desired information from them. We illustrate below the kind of information retrieved from the repositories with a query performed to each of those.

Example 1. Query to iServe: Operations related to a particular Service

The following listing depicts a query to iServe (endpoint at `http://iserve.kmi.open.ac.uk/data/execute-query`) used to obtain the operations that belong to a given service (in the example, a particular Last.fm service with its unique URI), and the label associated of the operation, if any:

```
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX msm:<http://cms-wg.sti2.org/ns/minimal-service-model#>
SELECT DISTINCT ?op ?labelOp
WHERE {
  <http://iserve.kmi.open.ac.uk/resource/services/ebed628d-ce25-46dd-a690-
96ee4003bf2c#LastFmEvents> msm:hasOperation ?op .
  OPTIONAL { ?op rdfs:label ?labelOp. }
}
```

Example 2. Query to the Storage Services: Aggregated metrics for a particular Service

The following listing contains a query to the Storage Services (endpoint at `http://coconut.tie.nl:8080/storage/repositories/KAnalyticsRepository`) to retrieve the different aggregated metrics (as explained in the previous deliverable, two for Frequency, two for Performance, and two for Perception) for a particular concept:

```
PREFIX an:<http://www.soa4all.eu/analysis#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT ?visFreq ?invFreq ?timPerf ?relPerf ?ratPerc ?revPerc
WHERE {
  ?analysis rdf:type
             <http://www.soa4all.eu/analysis#conceptAggregatedAnalysis> .
  ?analysis an:hasConcept
             <http://www.service-finder.eu/ontologies/ServiceOntology#Free> .
  ?analysis an:hasVisibilityFrequency ?visFreq .
  ?analysis an:hasInvocationFrequency ?invFreq .
  ?analysis an:hasTimePerformance ?timPerf .
  ?analysis an:hasReliabilityPerformance ?relPerf .
  ?analysis an:hasRatingsPerception?ratPerc .
  ?analysis an:hasReviewsPerception ?revPerc .
}
```

3.4.3 K-Analytics Examples of Use

To use the k-Analytics component, an end-user can search for iServe services through the input form located at the upper left corner. Once the list of relevant services is retrieved, the user can select one of them and it will be displayed in the main dashboard area.

The service annotations will be used to represent the relations amongst the service and its related concepts (via sawsdl:modelReference in the annotations of the service), along with its operations, and the input and output messages that each of those have.

In this line, we first illustrate the display of a service in the node-based graph without mixing the semantic relations with the metrics (which we will show in the next example). Figure 10 depicts the display of the annotations of a "Multimap" service (S), which has six related concepts (C), as well as an operation (O), which in turns has associated input and output messages (M), with four and two input and output concepts (C), respectively.

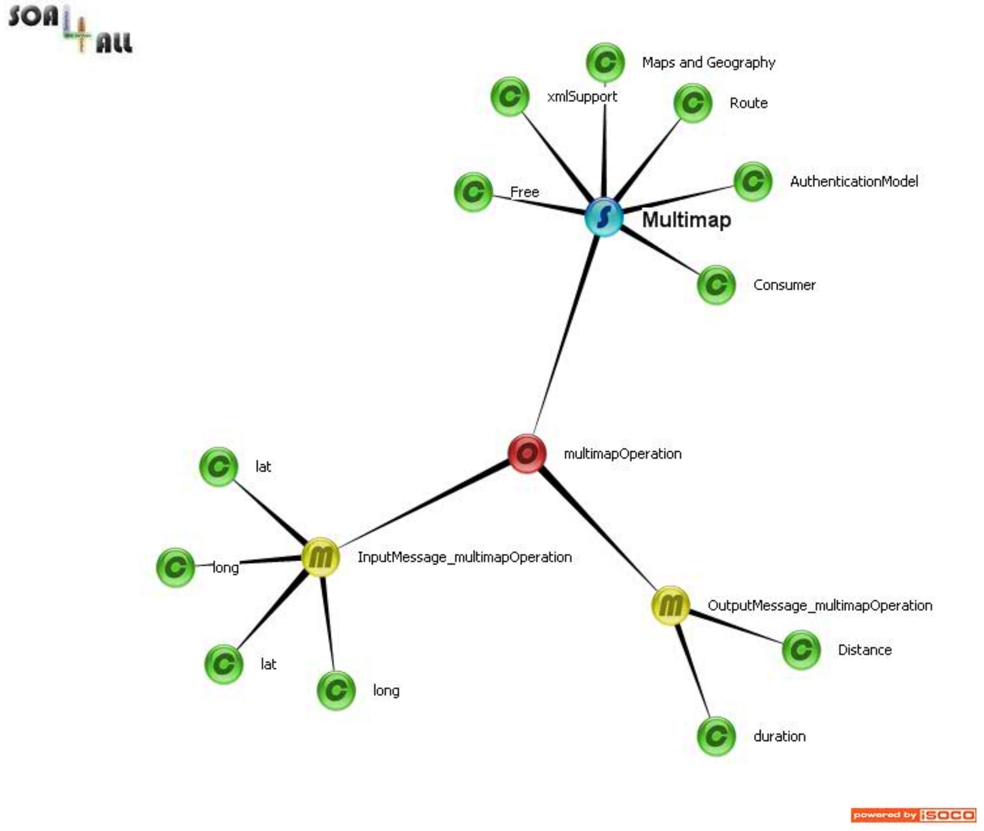


Figure 10: k-Analytics example without metrics

Of course, as we have explained, in addition to the semantic relations retrieved from the service repository, the metrics for each service and aggregated for each concept are shown in the same graph. This is done by giving different sizes to each relevant node according to the metric selected (Frequency, Performance and User Perception) on the bottom left drop-down menu.

In the following example, depicted in Figure 11, annotations of a Last.fm events service are displayed in the usual node-based graph (the service with five related concepts, one operation, the two input/output messages, etc.), combining the results for Performance in the sizes of each node. In this case, an end-user is able to inspect further characteristics of a service by comparing the sizes of the nodes of each of the related concepts. For instance, because of the small size of the "Free" node, one could argue that services that are related to the "Free" concept usually have a lower Performance.

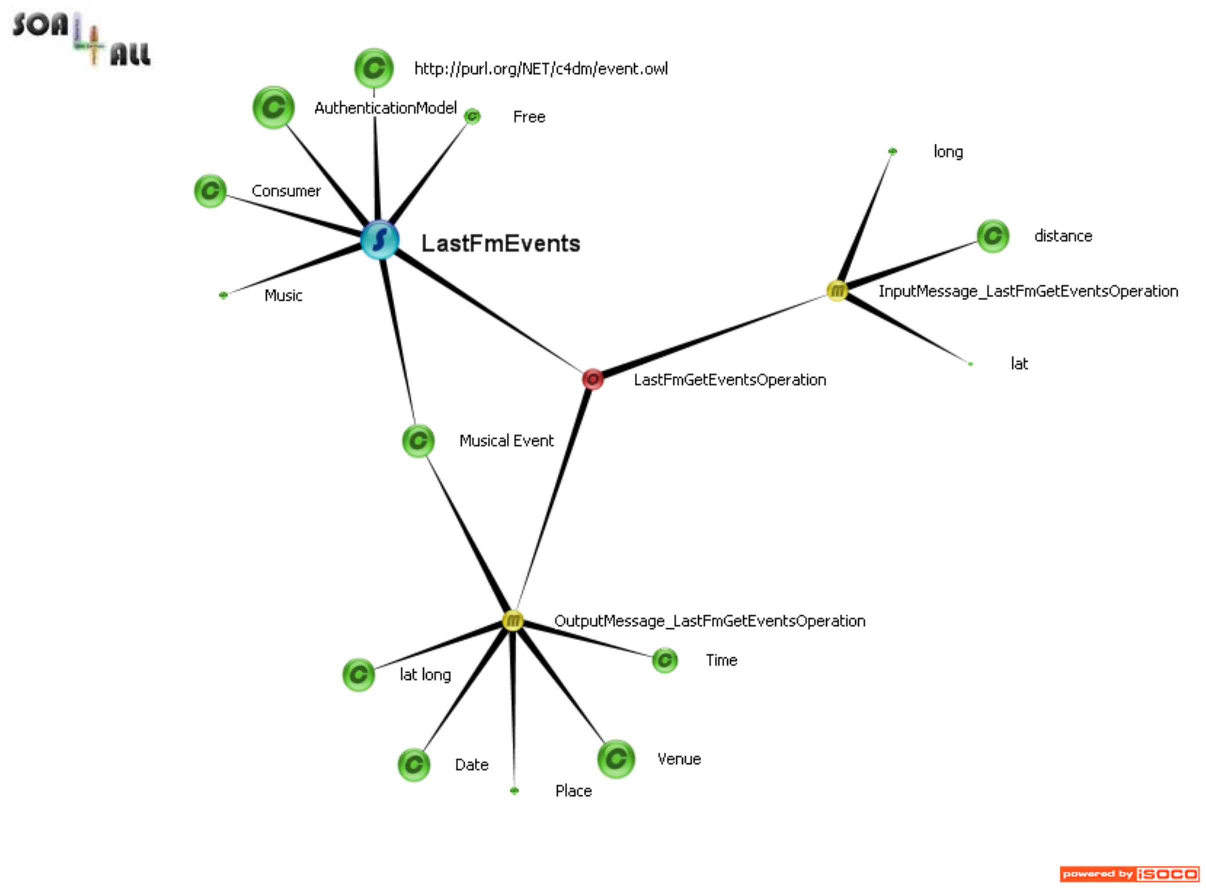


Figure 11: k-Analytics example showcasing metrics

This last example also showcases other characteristic of the node-based visualization, which is the ability to visually discover shared relations in the annotations. In this case, both the service itself and one of the outputs are related to the "MusicalEvent" concept. It can be argued that an end-user would be able to find this fact by inspecting the RDF annotations, but obviously this component gives a much more direct way of doing so with a simple glimpse to the visualization.

3.4.4 K-Analytics Public Deployed Version

Currently, there is an online version of the k-Analytics second prototype running at the following location: `http://soa4all.isoco.net/kAnalytics`, as a completely decoupled module, even when it can be also accessed through the complete Analysis Platform bundle.

Along with the online running prototype, there is a public description of the component and its main characteristics, which can be accessed at: `http://soa4all.isoco.net/kAnalytics/about`.

3.4.5 K-Analytics Limitations

Due to the visualization libraries used to create the draggable node-based graphs, and the fact that the module is bundled into an applet, it is only guaranteed that this prototype version will be accessible with the Windows operating system and a Java version higher than 1.6.0_20. It is expected that this limitation will be lifted as soon as the different JVMs running on different platforms and browsers address the library elements in a uniform manner.

3.5 DSB Monitoring Console

The DSB monitoring console is used to display monitoring data produced by message exchanges between service consumers and providers. As described in various deliverable of WP1 and WP2, the DSB monitoring architecture is based on state of the art monitoring specifications such as WS-Notification exchange data encapsulated within WSDM format. Figure 12 gives a high level view of the DSB monitoring architecture.

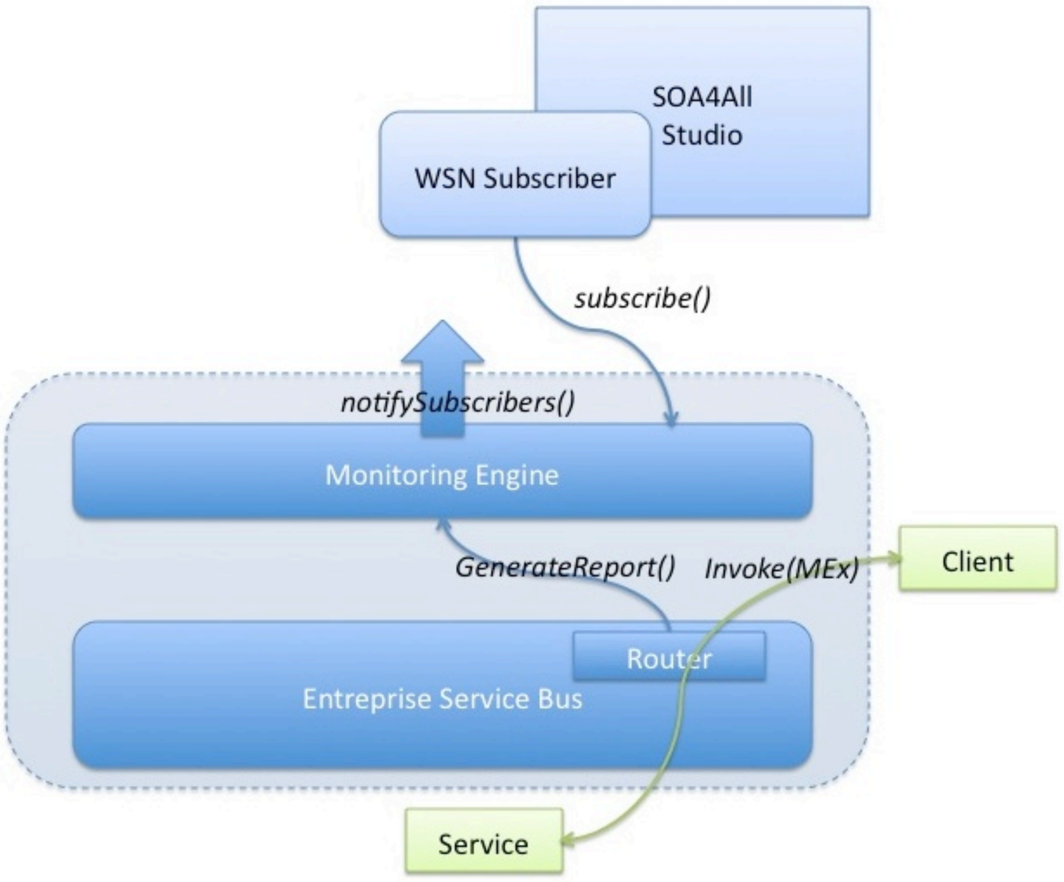
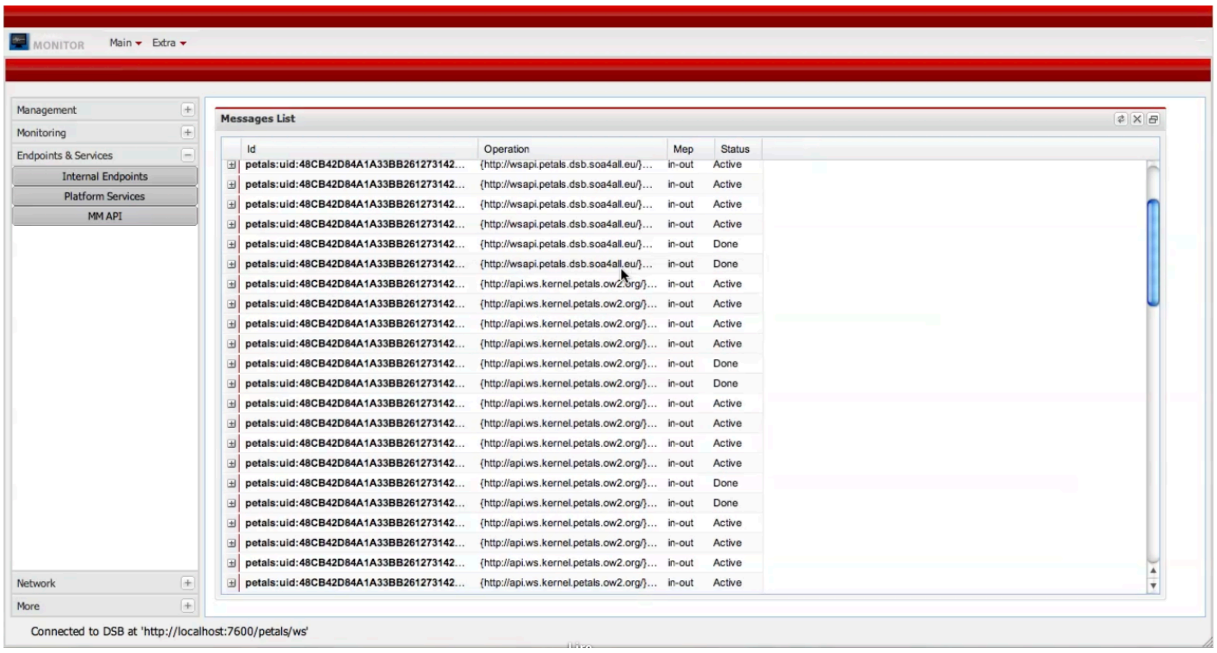


Figure 12. High-Level DSB Monitoring Architecture

In Figure 12, the WSN Subscriber module is able to store data received from the DSB monitoring layer into the 'data-collector' introduced in 2.1. The SOA4All DSB monitoring and management console is able to retrieve and display technical monitoring data as shown in Figure 13 or to display this monitoring data into the Analysis Platform interface.



Messages List

Id	Operation	Mep	Status
petals:uid:48CB42D84A1A33BB261273142...	{http://wsapi.petals.dsb.soa4all.eu/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://wsapi.petals.dsb.soa4all.eu/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://wsapi.petals.dsb.soa4all.eu/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://wsapi.petals.dsb.soa4all.eu/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://wsapi.petals.dsb.soa4all.eu/}...	in-out	Done
petals:uid:48CB42D84A1A33BB261273142...	{http://wsapi.petals.dsb.soa4all.eu/}...	in-out	Done
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Done
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Done
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Done
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Done
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active
petals:uid:48CB42D84A1A33BB261273142...	{http://api.ws.kernel.petals.ow2.org/}...	in-out	Active

Connected to DSB at 'http://localhost:7600/petals/ws'

Figure 13. Raw Message Exchanges List

3.6 Persistent Custom Dashboards

The Analysis Platform offers a view that can be customized to contain different widgets according to user needs (see Section 3.5 in D2.3.2). The second prototype of the AP now enables the storage of individual user configurations using the identifier of the user currently logged-on. Upon subsequent connections of the same user, the appropriate monitoring dashboard configuration will be automatically restored.

4. Installation & Setup

Each Analysis Platform instance is composed of 5 components:

- A local database (HSQL or MySQL) storing monitoring events
- A monitoring console (integrated into the dashboard), reading events form the DB in "real-time" – in *soa4all-monitoring.war*
- A webapp to expose monitoring services through a REST API - in *monitoring.war*
- A webapp to set monitoring configuration parameters (DB connection, EVO server/topic, RDF storage) – in *monitoring-event-listener.war*
- A webapp to expose the k-Analytics module – in *kAnalytics.war*

To install an Analysis platform instance, one has to deploy the local DB and the 4 WARs on a Tomcat server).

For "clients" of monitoring data (console and REST API, reading data from the DB), the DB connection parameters are set through a "connection.properties" file included in WAR files before deployment (in an internal JAR called *monitoring-bep-client*).

For "servers" generating monitoring data (EVO and WSDM handlers writing data into the DB), the DB connection parameters are set through the third webapp after deployment.

A detailed installation procedure for the entire Studio will be submitted by the Consortium at a later date.

5. Conclusions

This document presented the software deliverable of the second prototype of the Analysis Platform.

The main challenges addressed by the second prototype are related to scalability, integration and improvements in user interaction. By better tackling data storage through long-term and short-term storage strategies, we have achieved improved scalability and usability of monitoring data for third parties.

We believe that in its current version, the Analysis Platform prototype provides a significant set of functionalities that demonstrate how large scale monitoring data can be obtained and used in the scenarios envisaged by SOA4All.

6. References

[1] Pedrinaci, C., Domingue, J., and Medeiros, A. (2008) A Core Ontology for Business Process Analysis, 5th European Semantic Web Conference 2008, Tenerife, Spain, eds. Sean Bechhofer, Manfred Hauswirth, Joerg Hoffmann, Manolis Koubarakis

[2] RDF2Go. http://rdf2go.semweb4j.org

[3] RDFReactor. http://rdfreactor.semweb4j.org

[4] Statistical Data and Metadata eXchange (SDMX): http://sdmx.org/

[5] RDF vocabulary, specs and design patterns for publishing open linked statistics using RDF. http://publishing-statistical-data.googlecode.com

[6] Draft URL Structure for Accessing SDMX Repository: http://groups.google.com/group/publishing-statistical-data/web/draft-url-structure

[7] Time Ontology in OWL. http://www.w3.org/TR/owl-time/

[9] OWLim. http://www.ontotext.com/owlim/

[10] Jersey. https://jersey.dev.java.net/

[11] JSR-311. https://jsr311.dev.java.net/nonav/releases/1.1/index.html

[12] Pedrinaci, C., Liu, D., Maleshkova, M., Lambert, D., Kopecky, J., and Domingue, J. (2010) iServe: a Linked Services Publishing Platform, Workshop: Ontology Repositories and Editors for the Semantic Web at 7th Extended Semantic Web Conference