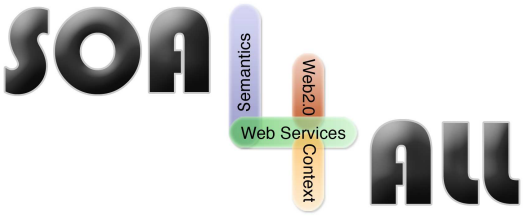

Project Number:	**215219**
Project Acronym:	**SOA4All**
Project Title:	**Service Oriented Architectures for All**
Instrument:	**Integrated Project**
Thematic Priority:	**Information and Communication Technologies**

D2.3.1. Service Monitoring and Management Tool Suite Design

Activity N:	Activity 1 – Fundamental and Integration Activities	
Work Package:	WP2 - SOA4All Studio	
Due Date:	M6 and M12	
Submission Date:	06/03/2009 Resubmission: 11/03/2009	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organization Responsible of Deliverable:	INRIA	
Revision:	2.0	
Authors:	Adrian Mos Carlos Pedrinaci Jose Manuel Gómez-Pérez Guillermo Álvaro Rey Christophe Hamerling Francoise Baude Cristian Ruz	INRIA OU ISOCO ISOCO EBM INRIA INRIA
Reviewers:	Lai Xu Tomas Pariente Lobo	SAP ATOS

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	20/01/2009	Initial revised structure, following M6 Interim Review. Initial high-level architecture. Also explained the overall alignment within SOA4All.	Adrian Mos Carlos Pedrinaci
0.2	25/01/2009	Minor update on the structure, moved architecture alignment to Intro.	Adrian Mos
0.3	26/01/2009	Added content in Requirements, improved overall architecture, initial description of BEP and MM, initial content for Provenance Analysis and the UI Consoles.	Adrian Mos Christophe Hamerling Jose Manuel Gómez-Pérez Guillermo Álvaro Rey
0.4	05/02/2009	Added Executive Summary, analysis use-cases, scalability, Proactive functionality, and improved other sections.	Adrian Mos Guillermo Álvaro Rey Francoise Baude Cristian Ruz
0.5	06/02/2009	Minor updates	Adrian Mos
0.6	23/02/2009	Added detailed API descriptions, remaining introduction sub-sections, and improvements to most sections.	Adrian Mos Guillermo Álvaro Rey Cristian Ruz
0.7	24/02/2009	Added section on Knowledge-Based Semantic Monitoring, and minor updates.	Carlos Pedrinaci Adrian Mos
1.0	05/03/2008	Updated doc following internal review	ALL
Final	11/03/2008	Overall format and quality revision	Malena Donato (ATOS)

Table of Contents

List of Figures

Glossary of Acronyms

Acronym	Definition
AP	Analysis Platform
API	Application Programming Interface
BAM	Business Activity Monitoring
BEP	Basic Event Processor
BP	Basic Profile
BPA	Business Process Analysis
BPEL	Business Process Execution Language
BPM	Business Process Monitoring
CML	Conceptual Modeling Language
COBRA	Core Ontology for Business pRocess Analysis
CPU	Central Processing Unit
D	Deliverable
EC	European Commission
ESB	Enterprise Service Bus
EVO	Events Ontology
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol Overview
IC2D	Interactive Control and Debugging of Distribution
JBI	Java Business Integration
JMS	Java Message System
KOPE	Knowledge-Oriented Provenance Environment
MM	Monitoring Mediator
MOWS	Management of Web Services
MUWS	Management Using Web Services
MXML	"Magic eXtensible Markup Language"
OCML	Operational Conceptual Modeling Language
PSM	Problem-Solving Method
QoS	Quality of Service
RIA	Rich Internet Application
SCA	Service Component Architecture
SENTINEL	SEmaNTic busINess procEsses monitoring tooL

SOA	Service-Oriented Architecture
SUPER	Semantics Utilized for Process Management within and between Enterprises
TMDA	Task Method Domain Application
UML	Unified Modeling Language
WP	Work Package
WS-I	Web Services Interoperability
WSA	Web Service Addressing
WSBN	Web Services Base Notification
WSDM	Web Services Distributed Management
WS-RF	Web Service Resource Framework
XML	Extensible Markup Language

Executive summary

The Analysis Platform aims at providing SOA4All users with information about services and processes in order to help them gain a better understanding of their use. Analysis information is also used by SOA4All infrastructure in order to drive adaptive behavior, however in the context of WP2, we focus on obtaining it and presenting it to the users. This document presents the initial design of the SOA4All Studio Analysis Platform. It also addresses the integration of the Analysis Platform in the overall Studio as well as within the overall project.

The design is the result of a requirements analysis integrating feedback from the use-case work packages and it is detailed enough to allow immediate code implementations while being generic enough for further evolutions and extensions. The document gives an initial overall view of the platform architecture and continues with detailed descriptions of individual modules. The sections on individual modules clearly identify existing work and how it will be reused as well as required future work for achieving the desired functionality.

The Analysis Platform is connected to the SOA4All infrastructure through a system of adaptors that facilitate the collection of monitoring events from several data sources, including the DSB itself, the underlying grid and the execution engine. It is also internally connected to the Studio using appropriate Studio services in order to benefit from repository integration and direct exchanges with the editors.

The Analysis Platform is composed of

- The Mediation and Basic Event Processing components: responsible for obtaining and normalizing events originating in data sources as well as performing basic processing for extracting derived analysis data. These components also contain the only entry points to the analysis repository and the runtime infrastructure, facilitating appropriate interactions with the other components in the Analysis Platform.
- The Provenance Analysis component will produce domain-oriented interpretations of past executions increasing the understanding about their results. The existing KOPE* tool suite will be extended to classify low level information from different process documentation infrastructures during process execution against a template repository in the form of reusable Problem-Solving Methods (PSM).
- The Knowledge-Based Semantic Monitoring component, based on existing work in the SENTINEL project, will detect deviations of the actual execution services from their specification; but more importantly, it will support the explanation of process execution with meaningful interpretations in a way closer to how users design and reason. This feature will allow the interpretation of the monitored events by non-expert users of the SOA4All through integration of appropriate information in the Studio editors.
- Monitoring consoles for graphically displaying basic performance and usage parameters to a variety of users ranging from end-users to technical administrators of SOA4All domains.

1. Introduction

1.1 Purpose and Scope

This deliverable provides the design and architecture of the Analysis Platform as well as its integration in the wider scope of the SOA4All Studio. Although some detailed aspects of the architecture and design might evolve as the project progresses, it is expected that the main components of the platform as well as their interactions will remain stable. Therefore this document will serve as a technical blueprint for the developments that are going to be undertaken in Task 2.3. In addition, the document gives functional descriptions and user interface requirements that will guide testing and development. Existing approaches that are going to serve as a foundation to the platform are also described in the deliverable with information about extending and integrating them in order to provide the complete required set of functionalities for the Analysis Platform.

1.2 Structure of the document

The document starts with the Introduction section, which mainly presents the Analysis Platform in the wider context of the SOA4All Studio. In Section 2 we present the requirements originating in the use-cases that have been identified by the partners with feedback from use-case work packages. In addition, a generic discussion of monitoring and management needs in SOA environments sets the stage for the understanding of some of the architectural decisions taken for the platform. Section 3 gives an overview of the architecture highlighting the main components of the Analysis Platform. The three sections that follow present different parts of this architecture. In Section 4 we present the lower-level blocks handling the collection of events and basic processing and management operations. Section 5 discusses techniques and an existing tool for provenance analysis, while Section 6 describes existing and future work related to knowledge-based monitoring. An overview of existing graphical consoles that can be used as starting points in the development of the Analysis Platform's UI is given in Section 7, followed by a presentation of the different types of views and UIs that will be developed in Task 2.3. Lastly, a short summary is given in Section **Error! Reference source not found.**.

1.3 Architecture Alignment

In this section we describe the alignment of the Service Analysis Platform with the SOA4All architecture currently being devised in WP1. Given that the Service Analysis Platform is a component of the SOA4All Studio, most of the details regarding its integration and alignment with the project vision are common to that of the SOA4All Studio. Still, where necessary we shall mention the specific details affecting the Service Analysis Platform. It is worth noting however, that at the time of this writing, the architecture of the SOA4All project is in an early stage and it is therefore not possible to provide a complete and thorough alignment.

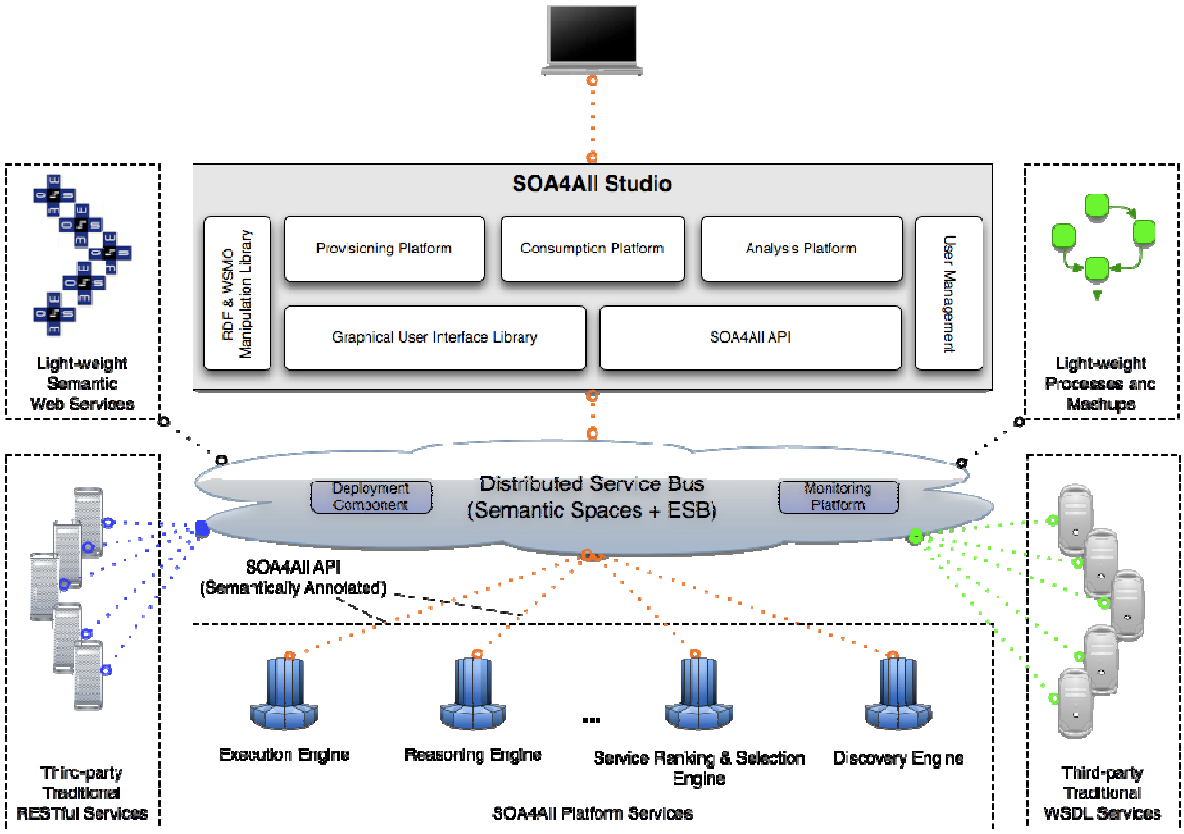

Figure 1. SOA4All Architecture.

Figure 1 provides a high-level view on the overall SOA4All architecture centered on the infrastructural components, the main artifacts manipulated and how the SOA4All Studio is integrated. Integration in SOA4All takes place through the so-called Distributed Service Bus (DSB) which integrates notions from Enterprise Service Buses and Semantic Spaces into a unified distributed infrastructure providing message-based and event-based communication, as well as a distributed RDF repository. From a client perspective, the DSB provides access to the platform services as well as the myriad of services provide by third parties. The services offered through the bus include, traditional third-party WSDL and RESTful services, light-weight Semantic Web Services based on annotations over traditional services, and infrastructural services supporting actions such that the discovery, ranking & selection and execution of services.

In a nutshell, the SOA4All Studio is the gateway for the user. It therefore provides a Web-based interface for creating or enhancing annotations, browsing them, discovering suitable services, invoking services, and finally analyzing their execution. These different functionalities are provided by three different platforms composing the Studio. In particular, the Service Provisioning Platform supports the user in providing annotations may them be, WSMO-Lite, MicroWSMO, tags or ratings. The Service Consumption Platform allows the user to browse, discover and invoke existing services. Finally the Service Analysis Platform provides the means for users to analyze the execution of services either at runtime or post-execution.

The SOA4All Studio, like every other component is integrated into the overall architecture through the Distributed Service Bus. It is worth noting however that as opposed to platform services that provide infrastructural services, the Studio is mainly a client for these different components, allowing the user to interact with them in a seamless and transparent way. To support this, there is an internal component within the Studio in charge of supporting the

interaction with the bus by making use of the appropriate messages and protocols. The concrete formats and protocols are currently being established within WP1 but they will be based on Web and WS-* standards.

The DSB also provides a scalable RDF storage & querying system. In particular, every annotation provided by the user through the Service Provisioning Platform will ultimately be stored for future use in the DSB. Similarly, raw monitoring data concerning the execution of services, like the message exchanges, will be stored in the DSB for further reference. Additionally, in order to support monitoring the execution of services—simple and composite—the DSB provides a notification mechanisms such that applications can register themselves as observers [1] for certain events of interest. Whenever any of these events occur, the DSB—thanks to the templates binding mechanism provided by Semantic Spaces—notifies any application interested.

The Service Analysis Platform will provide the means for analyzing the execution of services both at near real-time as well as post execution. To this end the platform will access the monitoring data gathered by the runtime infrastructure as it is pushed by the infrastructure or on demand depending the kinds of analysis being performed. The Service Analysis Platform contemplates three different modes for analyzing the data: real-time, periodic and on-demand. Real-time computation ensures that results are calculated as soon as the data is available. This mode is computationally expensive and will therefore only be applied to very concrete cases based on user requirements. Periodic computation of analysis results will be used for calculating certain data after some prefixed intervals of time. It is therefore less computationally expensive than the previous one although it will also provide results in a less frequent manner. Finally, on-demand analysis will be actively triggered by the user.

The Service Analysis Platform will support the users in defining the kinds of analysis they want to be performed (e.g., average execution time), the computation mode required, and finally presenting these results to the users using charts and other graphical representations so that humans can better understand the results obtained. Additionally, the results obtained at analysis time will be stored thanks to the DSB so that they can take part in future analysis or even support the adaptation of the execution of services. These results will be fed back to other parts of the SOA4All Studio so that when (re)designing processes/mashups or selecting which service to use, past behavior of services can allow users to take better informed decisions.

2. Requirements

2.1 Connection to Use-Cases

The design of the Analysis Platform is being driven by requirements originating in use-case analysis performed in connection with the work-packages WP7, WP8 and WP9. A summary of the use-cases identified for the Analysis Platform is illustrated in Figure 2 using UML.

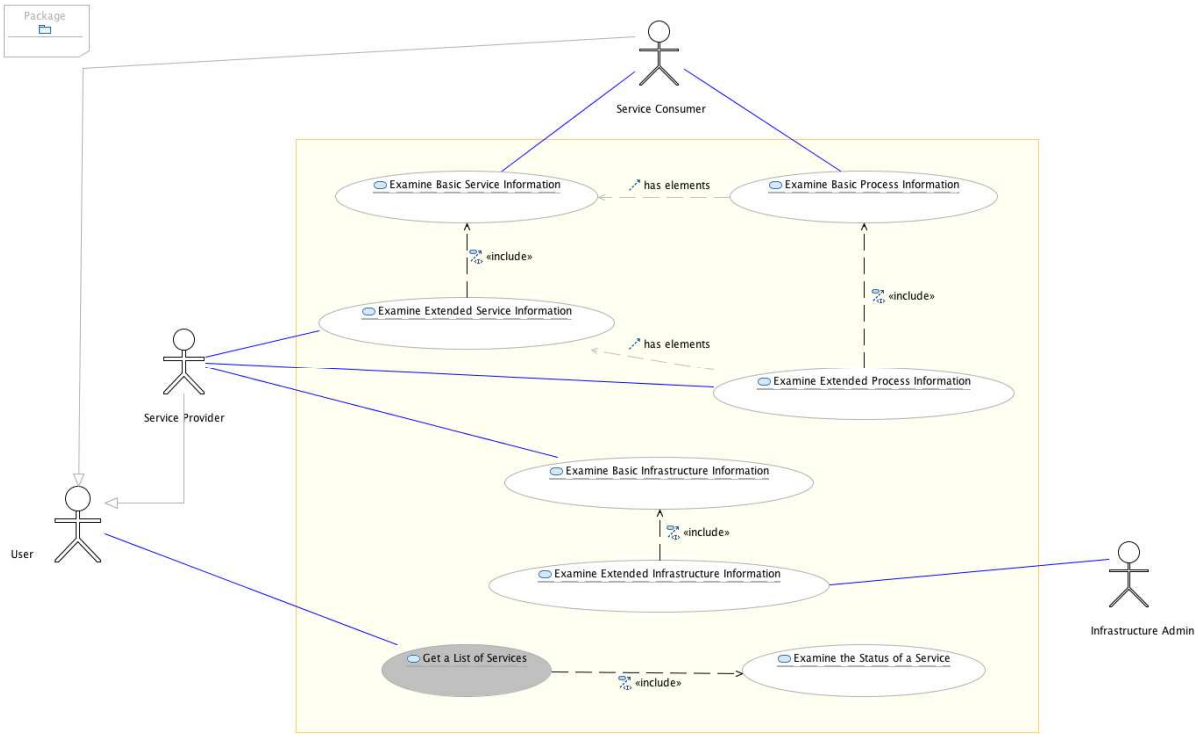

Figure 2. Summary of Analysis Platform Use-Cases

The following sub-sections present the actors and each of the use-cases.

2.1.1 Actors

- **User**: any user of the SOA4All platform, so our typical "non-technical" end-user
- **Service Consumer:** a User that is engaged in consumption of services, i.e. uses SOA4All services in one form or another. Note that a Service Consumer can also be a provider.
- **Service Provider:** a User that creates and provides services (basic or composed). Note that a Service Provider can also be a consumer, in particular when providing composite services.
- **Infrastructure Admin:** a role for people administering the technical infrastructure of a SOA4All domain / node.

2.1.2 Examine Basic Service Information (B_SERV)

A Service Consumer wants to find out about basic performance and usage information for a particular service. All the information is obtained for a particular time window that can be selected using a time slider widget (at least a classic date selector). He/she can obtain the following information:

- Performance

- avg. response time
- avg. throughput

- Status
  - available XX % of the time
  - NOT available
- Total number for requests

A UI mock-up for this use-case is presented in section 7.2.2.

2.1.3 Examine Extended Service Information (E_SERV)

A Service Provider needs to obtain information about a service. This information goes beyond and is more detailed than what is available in B_SERV, which this use-case includes. The extra information includes:

- User statistics for a service:
  - how many independent users
  - where do users come from
- Number of concurrent calls
  - min (with date)
  - max (with date)
- Which of the user's processes and compositions use this service

A UI mock-up for this use-case is presented in section 7.2.3.

2.1.4 Examine Basic Process Information (B_PROC)

A Service Consumer needs to find out basic information about processes/compositions that he/she creates through the consumption of services. This requires a high-level view of the process in the form of a simple one-level tree, in which the root represents the process name and the included services are represented as leaves. For the overall process, as well as for each of the included service, this use case will include the information specified in B_SERV.

A UI mock-up for this use-case is presented in section 7.2.5.

2.1.5 Examine Extended Process Information (E_PROC)

A Service Provider needs to obtain detailed information about process executions for the processes that are offered to the outside world. This use cases includes all the information in B_PROC and in addition it provides for each of the elements of the process tree (process and services) additional information as offered by E_SERV.

A UI mock-up for this use-case is presented in section 7.2.5.

2.1.6 Examine Basic Infrastructure Information (B_INFR)

A Service Provider needs to find out basic information about the infrastructure his/her services and processes are executing on. This is similar to information one can find out from a web-hosting platform when hosting web-pages. It is helpful to know what the platform can report regarding the usage of the resources the user has been allocated. The following information should be available:

- Availability of the infrastructure
  - % of time ON / OFF

- Capacity Information
    - o available bandwidth
    - o used bandwidth

2.1.7 Examine Extended Infrastructure Information (E_INFR)

An Infrastructure Administrator (such as an employee of a SOA4All infrastructure provider) needs to obtain detailed information about the domains that are being administered. In addition to the information presented in B_INFR, which this use-case includes, the following data must be available:

- Existing nodes in the platform
- Platform Information
    - o types of protocols and their usage
    - o usage information for binding components / service engines etc.
- Overview of low-level message exchanges
- Other DSB statistics

2.1.8 Examine the Status of a Service (STATUS)

Any SOA4All User using the search and discovery capabilities of SOA4All will eventually find a list of services that will be displayed. The user will need to have some elementary analysis information associated in a graphical way to the elements of the list. The type of information that can be displayed alongside listed elements includes:

- current availability (no time-window)
- average performance information (for the entire lifetime of just for the current day).
    - o response time
    - o throughput

A UI mock-up for this use-case is presented in section 7.2.1.

2.1.9 Examine the Goals that match a Process Template (GOALPROC)

A Service Consumer wants to obtain extra information about the different kinds of process templates that are available. The user will be able to explore the different Goals that trigger the kinds of executions defined by each process template. The logs of previous executions will permit to show, for each template, a list of Goals that match the particular template to certain degree:

- Name
- Description
- Percentage of matching (how much does that particular Goal match the selected template)

A UI mock-up for this use-case is presented in section 7.2.6.

2.2 Multi-Level Monitoring and Management

SOA is seen as an enabler for business agility through better adaptation of IT resource to business needs. This adaptation must be propagated throughout the information system, from the business layer through the IT design layer and to the infrastructure layer. The diagram in Figure 3 illustrates these three layers separated in two conceptual spaces: design

and runtime.

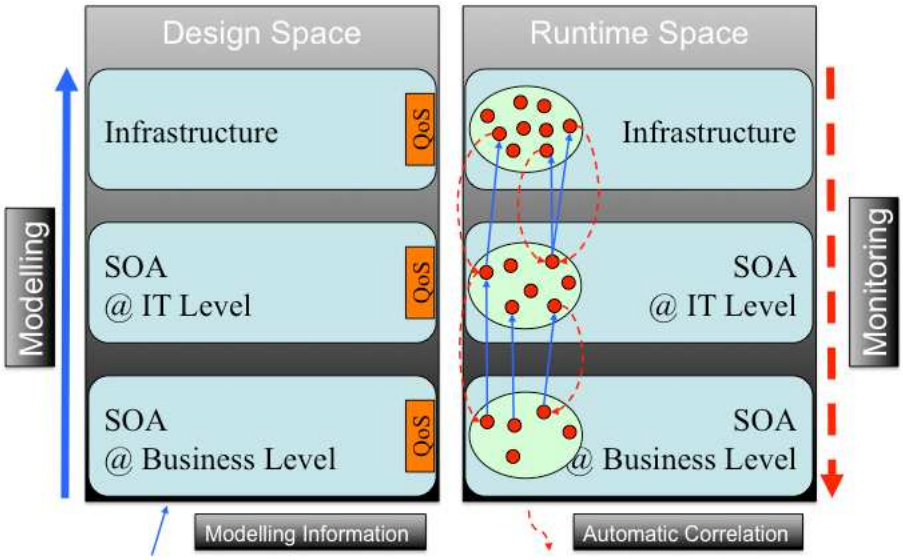

Figure 3 - Multi-Layer SOA Spaces.

2.2.1 Multiple Layers

The **Business Level** layer corresponds to the high-level, business-oriented definition of the business processes. Throughout this document, by "business" we refer to the core-application functionality and not necessarily to commercial activities. For instance, a government's business might include collecting taxes and providing healthcare, while a company's business might involve selling products. In this layer we typically see BPMN (or in the context of SOA4All – lightweight process description) diagrams describing processes that are very close to the actual business functionality of the organization. This challenge is addressed by the tasks in SOA4All that focus on the process composition language as well as user interaction.

Moving further on the modeling refinement axis, there is the **IT SOA Level**. The elements of this layer are SOA entities with a technical connotation. Services available in Service Registries and Repositories, executable orchestration processes defined with languages such as BPEL (to be refined in WP6) or component and composite definitions as defined by the Service Component Architecture (SCA) standard are all found at this level. Although SOA4All end-users will not see nor define SCA artifacts directly, they will be manipulated for providing adaptive behavior based on monitoring information, as architectural definitions on the DSB and the process execution engine.

The **Infrastructure Level** is concerned with entities that are related to the architectural definition of the SOA infrastructure. The infrastructure in SOA4All revolves around a Distributed Service Bus (ESB) that contains process-execution engines, SCA containers as well as lower-level grid-management components. As for the previous, IT SOA level, typical SOA4All users will not work at this level of technical detail. However it is important to be able to trace monitoring information to this level as well, as system administrators and infrastructure developers would need such information.

2.2.2 Monitoring and Analysis

The SOA vision revolves around agility at all levels in the IT stack. Changes in business requirements must be quickly resolved in the technical implementation of the system and changes in the IT system must be visible and comprehensible in the business layer. For instance, new services can be dynamically deployed and existing services can become

unavailable at runtime. In addition, QoS constraints must be observed and enforced at all levels such that the contractual obligations are met.

The design space involves the progressive specification, design and development of SOA concepts such as services and process by several different roles. This is achieved in SOA4All through the use of the SOA4All Studio.

The runtime space involves the layers of infrastructure required to support the execution of the artifacts manipulated in the design space explicitly, as well as those artifacts that are implicit (the invisible infrastructure such as JVMs or Oss). This is addressed in the SOA4All infrastructure components, most notably the Distributed Service Bus (DSB) and the process execution engine (WP1 and WP6 respectively).

The capability to monitor components at all levels (technological and business) is therefore essential to any technical stack supporting SOA. In order to facilitate the understanding of real-time events originating in the infrastructure components, it is essential that the events be conceptually mapped to the application-level entities defined in modeling layers. For this bottom-up transformation of technical events into meaningful application-level events to work, we need to inject specific information into components starting from the modeling layers (operation represented by the modeling arrow). As the application is defined, such "tracking" data can be attached to high-level components and successfully propagated through subsequent layers via transformations and generative techniques (for instance when creating deployment artifacts). Conversely, when events caught at runtime need to be propagated and understood by the upper, application-level layers, the "tracking" data can help in making correlations. In SOA4All such "tracking" information will be added at the level of the DSB (including grid-level) as well as at the level of the process-execution engine.

The multi-layer monitoring and management behavior of the SOA stack is represented in Figure 3 by the arrows traversing the layers. The solid arrows indicate the injection of "tracking" information in the same direction and sense as the modeling arrow from the design space. Elements (represented as circles) from higher-level layers are given a correspondence into less abstract layers by the specification of modeling intent (several iterations through Studio editors). This refinement operation is illustrated by the fact the one circle in the business layer can correspond to multiple circles in the IT SOA and infrastructure levels. The dotted arrows represent the monitoring event mapping operations, which are responsible for progressively adding semantics to extracted data while traversing the layers. The transformations are based on information previously injected in the design space. This approach is presented in more detail in [2].

2.3 Scalability

SOA4All main objective is to provide a framework and an infrastructure that help to realize a world where billions of parties are exposing and consuming services via advanced Web technology.

The current web only exposes around 28,000 traditional WS-based web services[1]; SOA is largely still an enterprise specific solution exploited by and located within large corporations used mainly for integration. Nevertheless, as mobile devices and more efficient wireless communications facilitate ubiquitous computing, and as optical and broadband communication infrastructures expand, we expect the number of Web services to grow exponentially in the next few years.

[1] According to seekda.com the number of WSDL services available online on March 04, 2009 was 27.813

This near-term scenario imposes great scalability requirements on the overall SOA4All infrastructure; and more especially on the Analysis Platform. For example, the monitoring and provenance tools should cope with the exponential growth of the number of message interchanges and the size of log files. The monitoring and management infrastructure should be either able to handle growing amounts of work in a graceful manner (definition of scalability given in [35]); or to be readily enlarged to cope with new workload on the fly (i.e. should be elastic).

It is worth noting that within the project, the SOA4All Runtime and, in particular, Task 1.6 (*Monitoring and Execution Context Management Infrastructure*) will work in the generic scalability issues regarding the gathering of large amounts of logs. Still, we address here some subjects that are particularly relevant for our Analysis Platform.

For instance, it is important to note that being the scenario envisaged by SOA4All one where a vast number of executions will take place, and so the data gathered, we cannot expect to perform an exhaustive computation on those logs. Instead, our knowledge-level analysis computations will be done in a "guided" way, as batch processes, on a smaller set of previously extracted information.

In Section 3, we will address the different components that are part of the Analysis Platform, which will help us differentiate the continuous (on the fly) collection of data from the Distributed Service Bus (DSB) of WP1 and the analysis made on top, which will happen as batch processes.

In particular, the knowledge-level environments that will be addressed in next section will operate on the relevant information previously extracted from the data level, thus leveraging the layered approach and the abstraction of information in a first step, and then operating just on a minor set.

3. Overall Architecture

This section gives a high-level overview of the Service Analysis Platform architecture. It is illustrated in Figure 4, which presents the main components of the platform and their connections to the external components. The elements within the light grey rectangle correspond to Analysis Platform components while the elements drawn with a dashed line represent external components. The thick arrows illustrate interactions between the Analysis Platform and the external components.

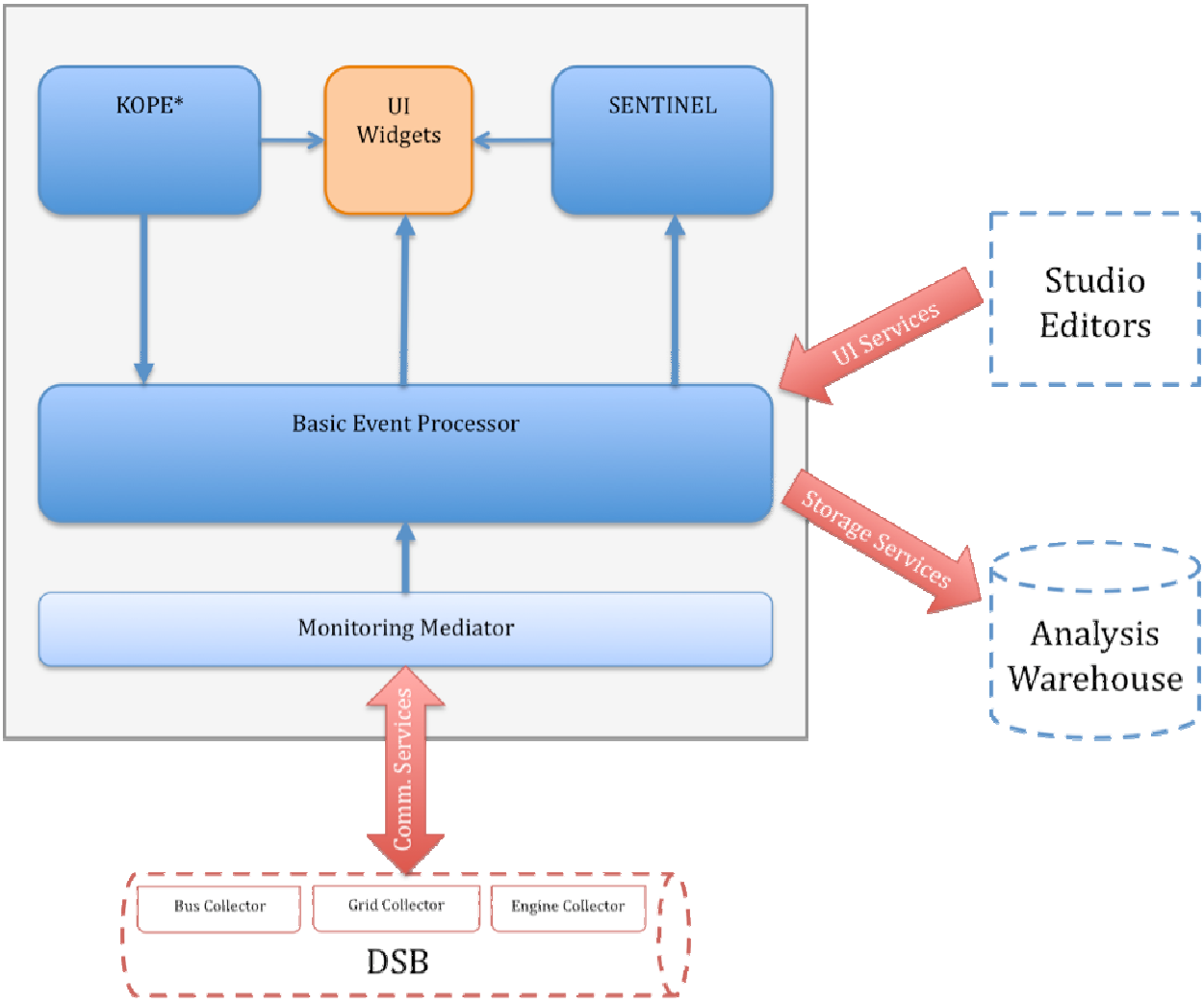

Figure 4. Overall Architecture for the Service Analysis Platform

The main source of data for the Analysis Platform is the *SOA4All Distributed Service Bus* (DSB), which is the backbone of the SOA4All runtime infrastructure. Several data collectors (bus, grid and engine collectors) available through the bus will feed relevant monitoring data to the AP, which in turn will be able to control and filter data collection through appropriate management operations. In addition, the AP will be able to send management commands to the DSB to instruct its components to perform a variety of infrastructural operations related to the customization or control of DSB components.

We distinguish between two types of monitoring events that can be received from the DSB: *infrastructure events and application events*. These will be detailed in section 4.

The following are the components of the Analysis Platform. They are briefly presented in this section and a detailed description can be found in the following sections of this deliverable.

- **Monitoring Mediator (MM):** Obtains Data from the DSB and the execution engines using the Studio APIs. It is both a listener and a proactive entity: its main role is to interface with the infrastructure developed in WP1 (it receives events and can also filter and control the event sources); it can also talk to the management/monitoring APIs in WP1
- **Basic Event Processor (BEP)**: performs pre-processing of monitoring events from the MM, including computing basic averages and statistics; provides data to KOPE* and SENTINEL for knowledge extraction; feeds data to basic UI widgets; uses the analysis warehouse to store derived information and basic computation results; the BEP acts as a single point of entry to the analysis warehouse thus interfacing with the other internal components of the analysis platform as well as with the Studio editors requesting analysis information to display on diagrams.
- **KOPE*:** this environment will make use of PSM to interpret the environment provenance information. It will produce domain-oriented interpretations of process executions to increase user understanding of such executions, which will most likely be potentially very large and complex. Since KOPE performs post-mortem analysis, it will query the BEP for monitoring information when necessary.
- **UI Widgets:** basic graphical representation (as seen in mockups, such as average response time for a service, availability etc.);
- **SENTINEL:** this environment will provide knowledge-based techniques in order to detect and diagnose process deviations based on monitoring information and informed by context data.

4. Mediation and Basic Event Processing

This section presents the mechanisms used by the Analysis Platform to integrate with the Studio editors and the runtime part of SOA4All as well as the event processing functionalities that are going to be offered by the BEP module.

4.1 Overview

The focus of this section, as illustrated in Figure 5, is to describe the Monitoring Mediator and the Basic Event Processor.

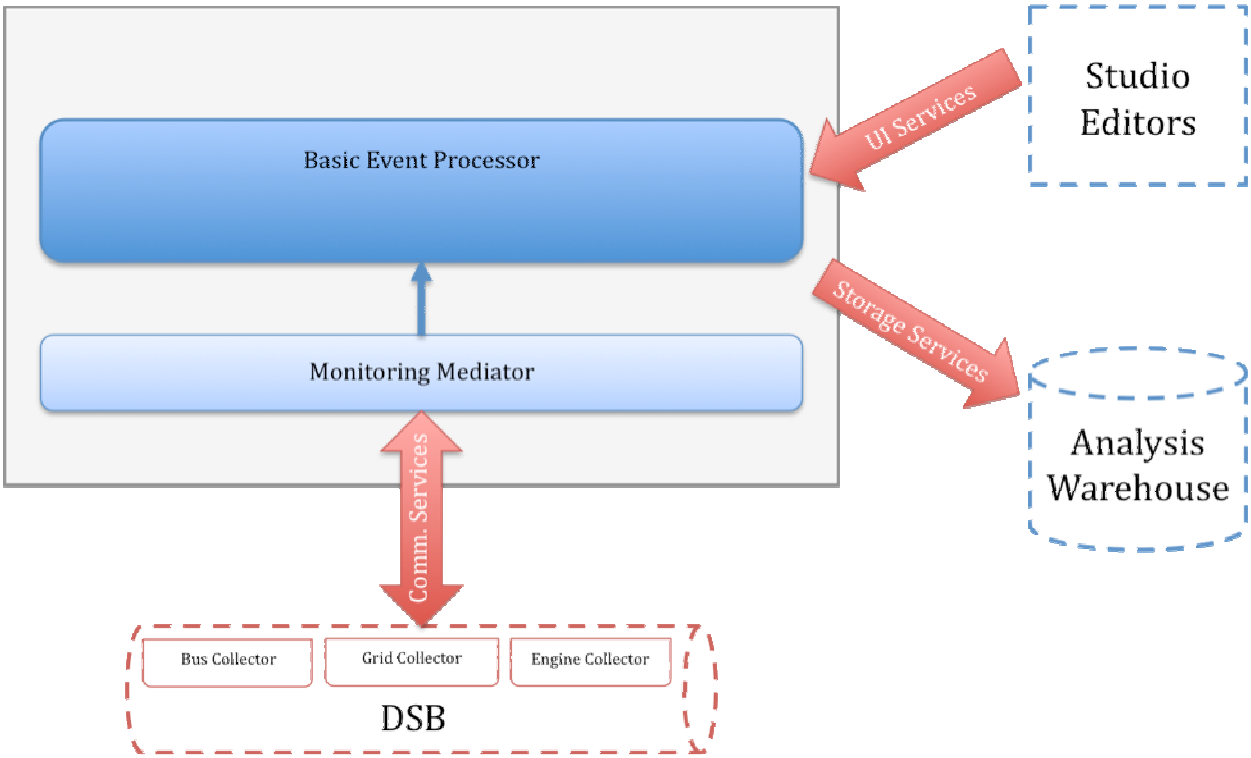

Figure 5. Mediation and Basic Event Processing Overview

The **Monitoring Mediator (MM)** will be the interface of the Analysis module to the runtime platforms developed in WP1 and WP6. It will connect to the aggregated runtime data collectors and will transform the events received into API calls to the Basic Event Processor. In addition, the Monitoring Mediator will be used to relay management operations originating in the Analysis Platform through to the runtime environments.

The Monitoring Mediator will use mechanisms complying with the WSDM standard to communicate to the data collectors.

The Basic Event Processor (BEP) is responsible for

- Parsing the monitoring events received from the MM and to perform data processing in order to extracts derived information (such as computing averages and basic aggregated events)
- Communicating raw and derived to upper-level processing entities, the KOPE* and SENTINEL engines.
- Updating the Ext-GWT data structures used by the basic UI widgets.
- Storing derived events into the analysis warehouse using the Studio Storage Services. These derived events can originate in the BEP itself or indeed can arrive

from KOPE* or SENTINEL.

4.2 Existing Functionality

There are several existing data collectors for different runtime systems, including Petals ESB, Proactive/GCM as well as execution engines such as SUPER SSB.

We will use existing standards and ontologies for monitoring, as outlines in the following subsections.

4.2.1 WSDM

WSDM uses Web services as a platform to provide essential distributed computing functionality, interoperability, loose coupling, and implementation independence. The OASIS WSDM working group has defined two specifications which we will use to provide efficient monitoring and management APIs, namely MUWS (Management Using Web services) and MOWS (Management of Web services).

- MUWS [26] defines how to represent and access the manageability interfaces of resources as Web services. Standard manageable resource definitions create an integration layer between managers and the different management protocols used to instrument resources.
- MOWS [27] defines how to manage Web services as resources and how to describe and access that manageability using MUWS. It provides mechanisms and methodologies that enable manageable Web services applications to interoperate across enterprise and organizational boundaries. The MOWS specification allows integration of management with Web services-based business applications and processes.

WSDM rests over several Web services specifications that have been standardized by different organizations. They include:

- WS-I Basic Profile (BP) [28] WS-I BP consist on a set of non-proprietary Web services specifications, along with clarifications to (and amplifications of) those specifications in order to promote interoperability.
- WS-Resource Framework (WS-RF) [29] WS-RF offers a standardized way to express the relationship between stateful resources and Web services. Therefore it provides mechanisms to access and manipulate the state of values that persist across, and evolve as a result of, Web service interactions.
- WS-Based Notification (WSBN) [30] WSBN is a group of specifications related to WSRF that permits event driven communication between Web services.
- WS-Addressing (WSA) [31] WSA provides a standard representation for services references. WSA provides also transport-neutral mechanisms to address Web services and messages.

4.2.2 COBRA/EVO

- for process-level monitoring: see Section 6 for details on these existing ontologies.

4.2.3 JMX / WSDM

- for infrastructure-level monitoring: The DSB infrastructure components (including the PEtALS ESB and the Proactive Framework) are mainly implemented using the Java technology and so use the Java Management eXtensions (JMX) for management. Since the JMX technology is quite simple to use and extend, components developers have extended their initial management goal to provide monitoring functionality.

However, JMX is a Java based technology used by Java developers for Java developers. In order to expose this functionality in a more generic way, components will also provide Web service compliant monitoring and management features using the WSDM specification by wrapping the JMX API.

4.2.4 ProActive / GCM

- ProActive/GCM allows to raise JMX events from the grid infrastructure-level. These events include the transmission of requests between components, and also lower level details like process load in the involved machines, and CPU/memory usage. Any client application can subscribe to these events in order to collect monitoring data from the infrastructure, and present it or wrap it for further analysis. For achieving scalability, a specialized ProActive-based JMX connector is used, which takes advantage of ProActive communication features in the transport of JMX messages.

4.3 Missing Functionality

- **The Monitoring Mediator**
- **The Basic Event Processor**
- **Analysis Warehouse.** This element will be developed as an interface to the Semantic Space. For the purpose of this document, its functionality is represented as a set of classes.
- **A Monitoring Event Structure.** The MM needs to interface with the runtime platforms and obtain monitoring events. The MM will therefore receive the runtime-generated events from data collectors and will convert them to events corresponding to the event-structure defined by the Analysis Platform. This structure will embed in a unified manner the different events structure from the different monitoring targets. In particular it will embed business activity monitoring events (COBRA/EVO), service-level events (WSDM) and infrastructure events (JMX / WSMO).
- **Integration with Basic Widgets**: the BEP will provide data for basic Ext-GWT widgets that will display simple monitoring information (availability for a service, status of a process execution, performance data for a service and a process).

4.4 APIs

This section describes the main interfaces of the different components involved in the Analysis Platform architecture.

4.4.1 Monitoring Events

The Analysis Platform considers two main types of events:

- **Infrastructure Events**: dealing with technical information originating in the SOA4All Infrastructure: the DSB, the Process Engine and the lower-level layers including the grid.
- **Application Events**: dealing with information originating in the application-level components executing on top of the SOA4All infrastructure. These are further separated into **Service Events** and **Business Activity Monitoring (BAM) Events**. The former correspond to individual services being executed in SOA4All domains or indeed outside SOA4All-managed infrastructure (in which case the information is simply collected using WSDM interfaces when available). The latter correspond to processes executed by the SOA4All infrastructure. For these events we are using the EVO ontology (see section 6) since it already provides a useful event classification for

business processes and activities.

Figure 6 shows a simplified view of the monitoring events that will be considered by the AP. It does not show the entire EVO hierarchy of events as that is referred to in more detail in section 6.

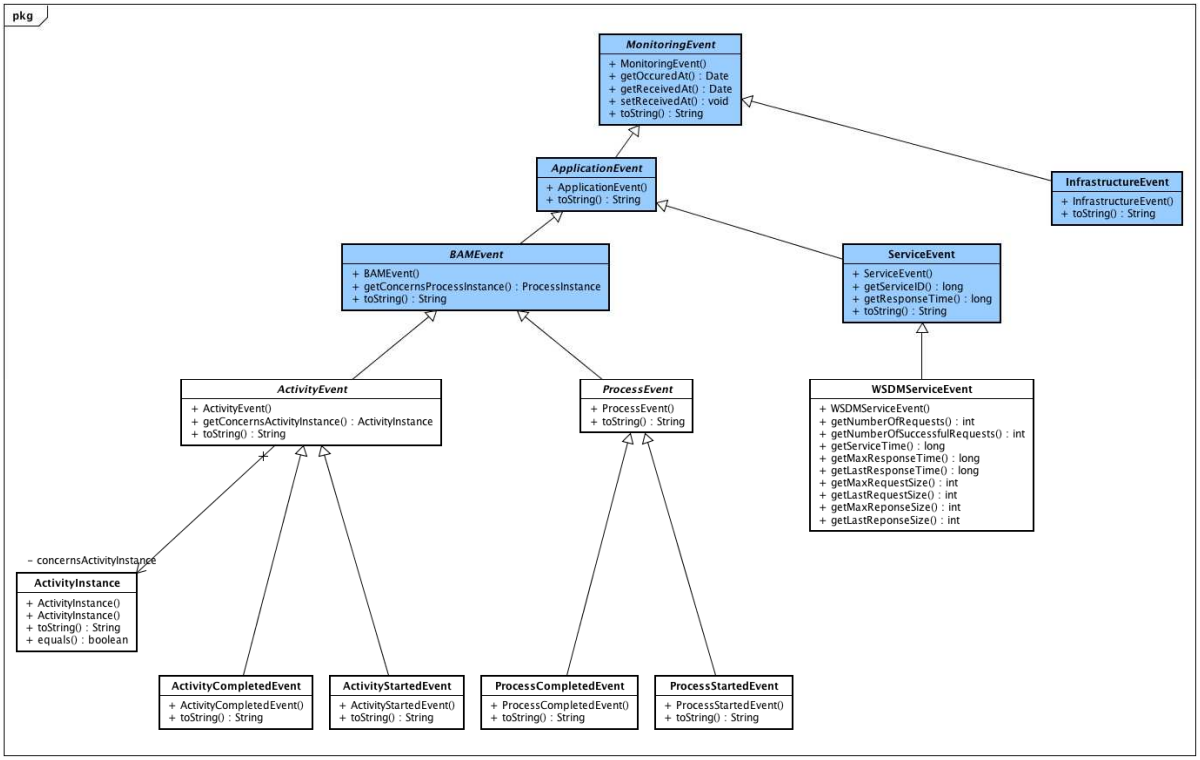

Figure 6. Analysis Platform Events (simplified view)

The main classes involved are:

`MonitoringEvent`: abstract, top-level event class containing basic getters and setters for properties related to execution time and reception time. This class is at the top of the hierarchy.

`InfrastructureEvent`: top-level infrastrcture event class to serve as hierarchy root for all events originating in the execution platforms.

`ApplicationEvent`: the top-level application event class.

`BAMEvent`: the top-level business process event class, in effect the bridge between the AP event hierarchy and the EVO ontology. This class is the parent of all the process and activity related events.

`ServiceEvent`: the parent for all the service-level events. We target in particular WSDM-type monitoring information originating in the execution platforms.

4.4.2 Mediation and Basic Event Processing

The Basic Event Processor (BEP) provides a single point of entry in the Analysis Platform for the events arriving from the monitoring sources as well as for the interactions between other modules in the AP and the basic collected data. Figure 7 illustrates the interactions between

the Monitoring Mediator and the BEP as well as the relationship with the collected basic monitoring data.

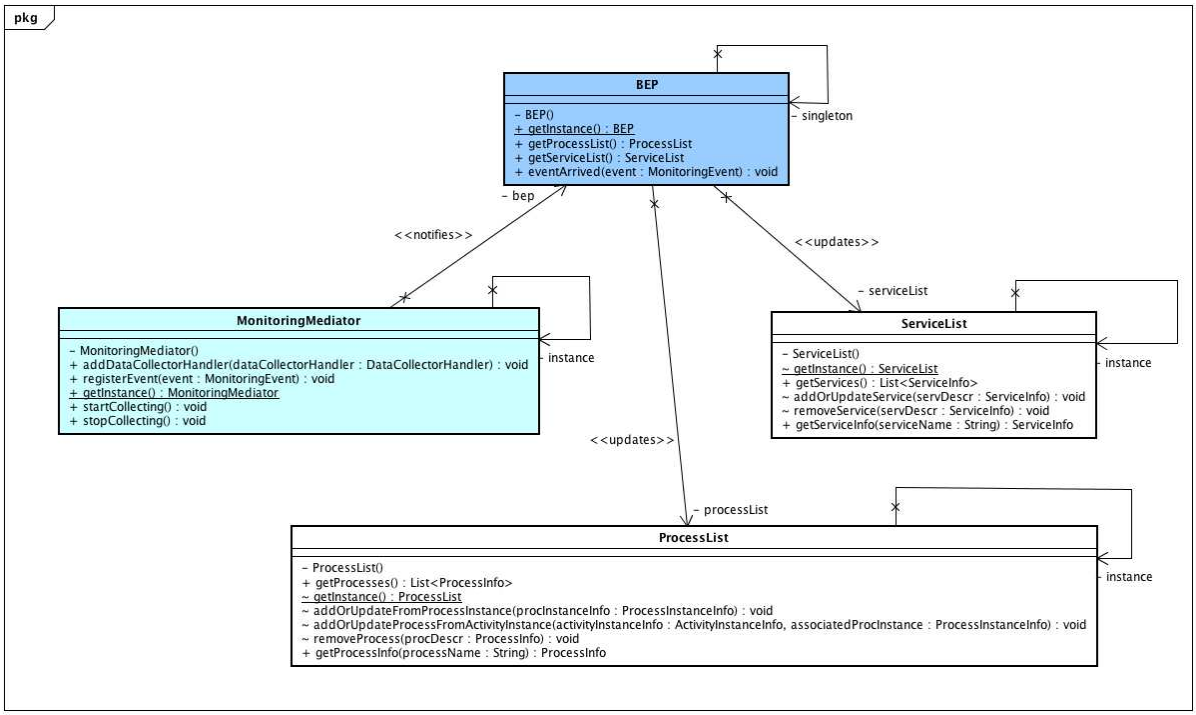

Figure 7. Interactions between the BEP and the Monitoring Mediator

The Monitoring Mediator notifies the BEP of any new monitoring events received from the data collectors, using BEP's `eventArrived()` method. It passes as a parameter the event instance corresponding to the hierarchy defined in Section 4.4.1. The BEP also provides methods for other components of the AP to use when requesting the list of existing services or processes. These lists are represented here as classes, as this is useful for understanding the kind of information they will contain as well as in the construction of a very first demo for M12. However, these classes will be replaced by the Analysis Warehouse, which will provide persistent support for monitoring information. It is important to note that the BEP also provides information for the basic UI widgets that directly display monitoring information (see section 7.2 for mock-ups related to the display of monitoring information).

The Monitoring Mediator and its role in interfacing with the data collectors are illustrated in Figure 8. In essence, the MM is a collection of handlers for different types of data collectors (as illustrated in the figure through different classes for an EVO Handler, a WSDM Handler and a Grid Handler respectively). Each handler can be instructed to start and stop listening for events (perhaps other management operations will be added to this interface in the future).

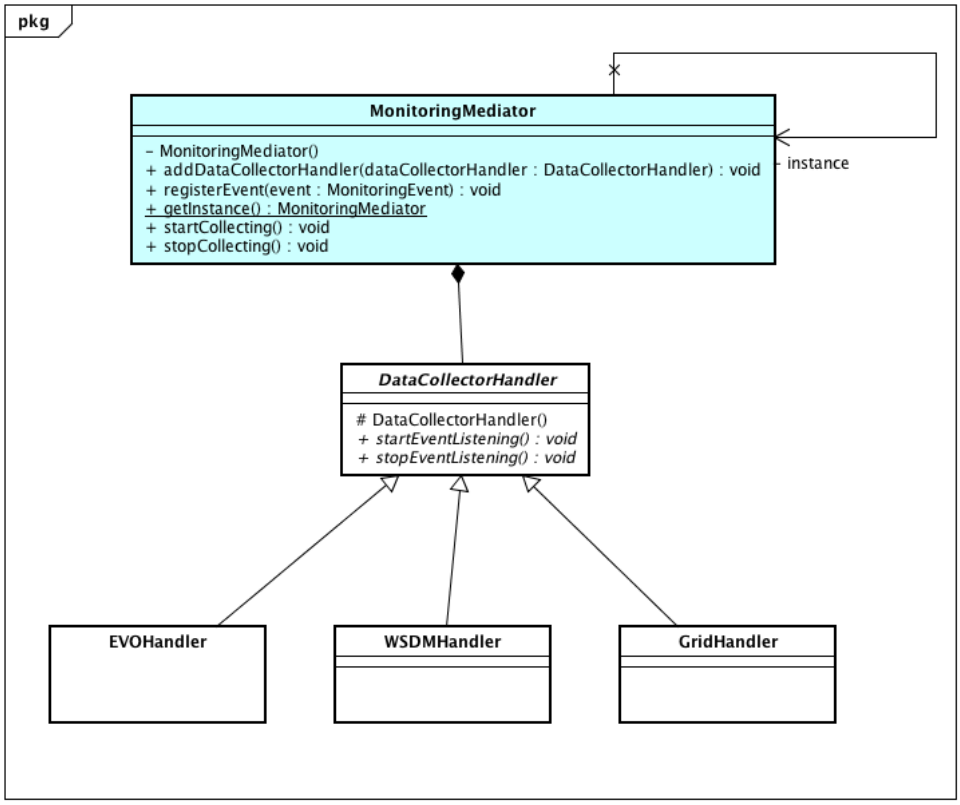

Figure 8. The Monitoring Mediator Interfacing with the Data Collectors

4.4.3 Integration with UI Widgets

For integration with basic UI Widgets, the COMET push-based programming model [32] will be used, as mandated by the SOA4All design strategies, defined in D2.4.1 through its GWT implementation, the rocket-gwt library [33]. In a nutshell, this technology overcomes the limitations of HTTP where the client drives the communication, by maintaining instead an open communication channel with clients so that if necessary the server can notify them about relevant information. This model implies that the BEP will push data to the ext-gwt widgets, which will get updated as required whenever events of interest arrive, or when new derived monitoring data is generated (i.e. by SENTINEL or KOPE*). Several classes will be used corresponding to the data required by each of the widgets. They will be update by the BEP and in turn will invoke the COMET-style updating method on the corresponding widgets.

5. Provenance Analysis

5.1 Overview

Within the Analysis Platform, the post-mortem analysis of executions will be very important in order to extract valuable information that will enhance subsequent executions. In particular, Provenance deals with tracing the origin of data, and we will extend previous work, adapting it to the needs of the project.

In SOA4All, and in particular within this work package, we have stressed the importance of finding Goals in order to discover the right services. We believe that we can extract very important conclusions by analyzing the executions that different Goals trigger, and the knowledge we infer from this analysis will enrich service consumption, that we will also address.

More precisely, we would like to use the information of previous executions in order to find similarities between Goals in terms of the (composed) executions they result in (if they are composed). We believe this is interesting, for we could find similarities not based on the semantics of the Goals, but on the executions they would imply.

Then, the information we will be able to extract about the similarities of the Goals will be relevant for the Recommendation System (to be developed in T2.7), as we will able to categorize Goals, and that system will be able to use it to suggest similar Goals. Eventually, the Recommendation System will be able to deliver those recommendations to the Service Consumption Platform, closing the loop between consumption and analysis, enriching the subsequent interactions with the platform, thus fulfilling our needs.

Additionally, it will be possible to use the information about similar Goals for reverse engineering of Goals and processes.

5.1.1 Similarities between Goals

First, to achieve this, we will find patterns in the executions thanks to process templates. As to the templates, we will use explicitly defined specifications of processes, as provided by the process editor (T2.6), as templates that define well-known execution patterns. These templates will allow us to discover occurrences of such patterns amongst the logs produced by the execution of a composition of services previously triggered by a Goal, where such composition is not the result of the enactment of a process model but of the execution of a Goal, instead. In other words, we will use explicitly defined knowledge (process models) to classify implicitly defined processes (summarized as the Goal triggering the invocation).

Thus, the similarities between Goals are obtained as a consequence of classifying the executions of the service compositions they trigger with respect to the available templates. So, if goal G1 triggers an execution compliant with template T in a ratio of X% and goal G2 is compliant with T in a Y%, G1 and G2 will be similar in a value returned by a function operating on X and Y, which we will investigate, (e.g. their average).

5.1.2 Obtaining the logs

This component will retrieve the logs directly from the Basic Event Processor, as depicted in Figure 4. Being a post-mortem tool, instead of receiving constant events from the stream, it will pull the information from it.

We will align the information about templates that we need to spot with the BEP, in order to be able to discover occurrences of those templates, and relate them to the Goals that have triggered those executions.

5.1.3 Graphical view

The most important outcome of our post-mortem analysis will be fed back into the service consumption lifecycle through the Recommendation System, as explained. However, the Analysis Platform user interface will also include a widget for observing the Goal similarities as spotted with this component. We refer here to the UI Mockup depicted in 7.2.6.

5.1.4 Conclusion

Thus, by and large, the objective is to find, for each Goal, similar Goals (in terms of the execution patterns they result in). This information will be displayed in the platform, but more importantly, it will also be able to enrich the Recommendation System (T2.7), already providing relevant information of similar Goals, so when a user invokes a Goal, the relevant ones can be presented to him as an option in the future. This is interesting to further close the loop between Consumption, Analysis and Recommendation.

5.2 Existing Functionality

5.2.1 KOPE

The approach of Knowledge Oriented Provenance Environment (KOPE, [36]) is to provide users with meaningful interpretations of process executions, explaining provenance in a way closer to how domain experts reason on a given problem, and facilitating their comprehension, using problem-solving methods (PSM). We will reuse and extend KOPE in order to obtain meaningful interpretations about service consumption.

5.2.2 PRoM

In addition to the process models (templates) that will come from the process editor, as previously explained, we will also be able to perform analysis based on previous executions with the help of PRoM [34], the Process Mining toolkit. The scenario will be that also PRoM will populate the template libraries with new templates that can be later reused for finding similar Goals as addressed before.

5.3 Missing Functionality

The missing functionality within our component will be covered by the KOPE* extensions, and we can summarize them here:

- For a given Goal, find out to which degree does it match each of the available execution templates.
- For each of the execution templates, find out which are the Goals that match them more.
- For any two given Goals, find the degree of similarity between them, depending on the execution they trigger.

5.4 APIs

We will enable a way of finding the degree of similarity of a certain Goal with the given patterns. The result will be a list with the different templates and how much they match with the executions of a Goal:

```
getGoalExecutionPatterns(Goal): pattern x -  degree of matching []
```

Additionally, we will be able to find out: which are the most relevant Goals for a given pattern, and to which degree:

```
getExecutionPatternGoals(pattern): Goal x -  degree of matching  []
```

We will also be able to find out the most similar Goals to a given one, and to which degree:

```
getSimilarGoals (Goal): Goal x -  degree of matching []
```

6. Knowledge-Based Semantic Monitoring

The vision pursued by SOA4All whereby billions of services would be provided, adapted, enhanced, created, and consumed on a world-wide scale poses important technical challenges. Among the challenges it poses this deliverable is particularly concerned with those related to the monitoring of services. In this section, as opposed to current practices within the state of the art in Web services monitoring, we focus on raising the level of abstraction within monitoring technologies in order to better support the interpretation of monitoring information by humans and machines.

We base our techniques on previous research as carried within SUPER[2] EU project (FP6-026850) in the context of Business Process Management (BPM). The reason for this is two-fold. On the one hand, BPM is undoubtedly one of the most demanding fields with respect to the support for advanced monitoring of processes and services given that enterprises' subsistence directly depends on them. On the other hand, achieving the SOA4All vision, in addition to obtaining an appealing infrastructure and toolset able to support the proper execution of services, requires also an appropriate support for carrying businesses over the Web. After all, the possibility to generate benefits out of technologies is most often the main aspect to achieve real impact at a large scale.

6.1 Overview

BPM intends to support "*business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information*" [4]. BPM acknowledges and aims to support the complete life-cycle of business processes which undoubtedly involves the analysis and reengineering of process models. However, BPM has made more evident the difficulties for obtaining automated solutions from high-level business models, and for analyzing the execution of processes from both a technical and a business perspective [5]. The fundamental problem is that moving between the Business Level and the IT Level is hardly automated. Deriving an IT implementation from a business model is particularly challenging and it requires an important and ephemeral human effort, which is expensive and prone to errors. Conversely analyzing automated processes from a business perspective, e.g., calculating the economic impact of a process or the performance of departments within an organization, is again an expensive and difficult procedure, which typically requires a human in the loop.

One of the distinguishing characteristics of BPM solutions with respect to traditional Workflow Management Systems is commonly referred to as Business Process Analysis (BPA) [4]. The main goals pursued by BPA are on the one hand the verification or validation of the execution with respect to prescribed or expected processes, and on the other hand the identification of potential improvements of business processes. The knowledge gained in this phase is thus employed for reengineering and fine tuning existing process definitions. This area therefore comprises a wide-range of fields such as Business Activity Monitoring, Business Intelligence, Business Process Mining and Reverse Business Engineering. The importance of BPA is widely acknowledged and in fact all the main vendors provide their own solutions [6]. The quality and level of automation provided by these tools are rather similar and not surprisingly major efforts are devoted to presenting the information in a simple yet meaningful way better supporting humans in the analysis phase.

We refer to Business Activity Monitoring (BAM) as the technology in charge of "*providing real-time access to critical business performance indicators to improve the speed and*

[2] http://www.ip-super.org/

effectiveness of business operations" [4]. For its very nature, within BAM the previously mentioned difficulties are even more outstanding. We have previously argued for the use of semantic technologies, namely ontologies and Problem-Solving Methods, as a means to enhance the state of the art in BPA [7]. In the light of this vision, we have defined Core Ontology for Business pRocess Analysis (COBRA) [8], which provides a core terminology where business practitioners can map domain-specific knowledge in order to analyze their business processes. We have also defined additional extensions for capturing semantically the logs produced by IT systems and for deriving knowledge in terms of COBRA. In the remainder of this section we shall present our existing technology and will introduce where appropriate the necessary extensions or adaptations to deal with the monitoring of services within SOA4All.

6.2 Existing Functionality

COBRA, depicted in [8], provides a core terminology for supporting BPA where analysts can map knowledge about some particular domain of interest in order to carry out their analyses. It is worth noting that COBRA does not aim to provide a fully-comprehensive conceptualization for supporting each and every kind of analysis since the scope would simply be too big to be tackled appropriately in one ontology. Instead COBRA provides a pluggable framework based on the core conceptualizations required for supporting BPA and defines the appropriate hooks for further extensions in order to cope with the wide-range of aspects involved in analyzing business processes. COBRA has been developed using the Operational Conceptual Modelling Language (OCML) [9], which provides support for executing the definitions in the ontology as well as export mechanisms to other representations including OWL and WSML. COBRA builds upon two ontologies, namely Base Ontology and Time Ontology, and is currently enhanced with Events Ontology for capturing audit trails, and Events Analysis Ontology which provides a set of generic reusable rules and relations[3]. Base Ontology provides the definitions for basic modeling concepts such as tasks, relations, functions, roles, numbers, etc. The interested reader is referred to [9] for further information. The other ontologies will be briefly described in the remainder of this section.

Although fully describing COBRA is outside of the scope of this document we introduce in this section those aspects that are necessary for understanding the rest of the document. COBRA provides a pluggable framework based on the core conceptualizations required for supporting BPA and defines the appropriate hooks for further extensions in order to cope with the wide-range of aspects involved in analyzing business processes. COBRA divides the world into *Temporal Entities* and *Persistent Entities* whereby the former are entities that have a temporal extent whereas the latter are essentially independent of time. Time Ontology provides a temporal reference by means of which one can determine temporal relations between Temporal entities based on a slightly extended version of Allen's interval relations [10], see Figure 10.

[3] The ontologies can be found at http://kmi.open.ac.uk/people/carlos

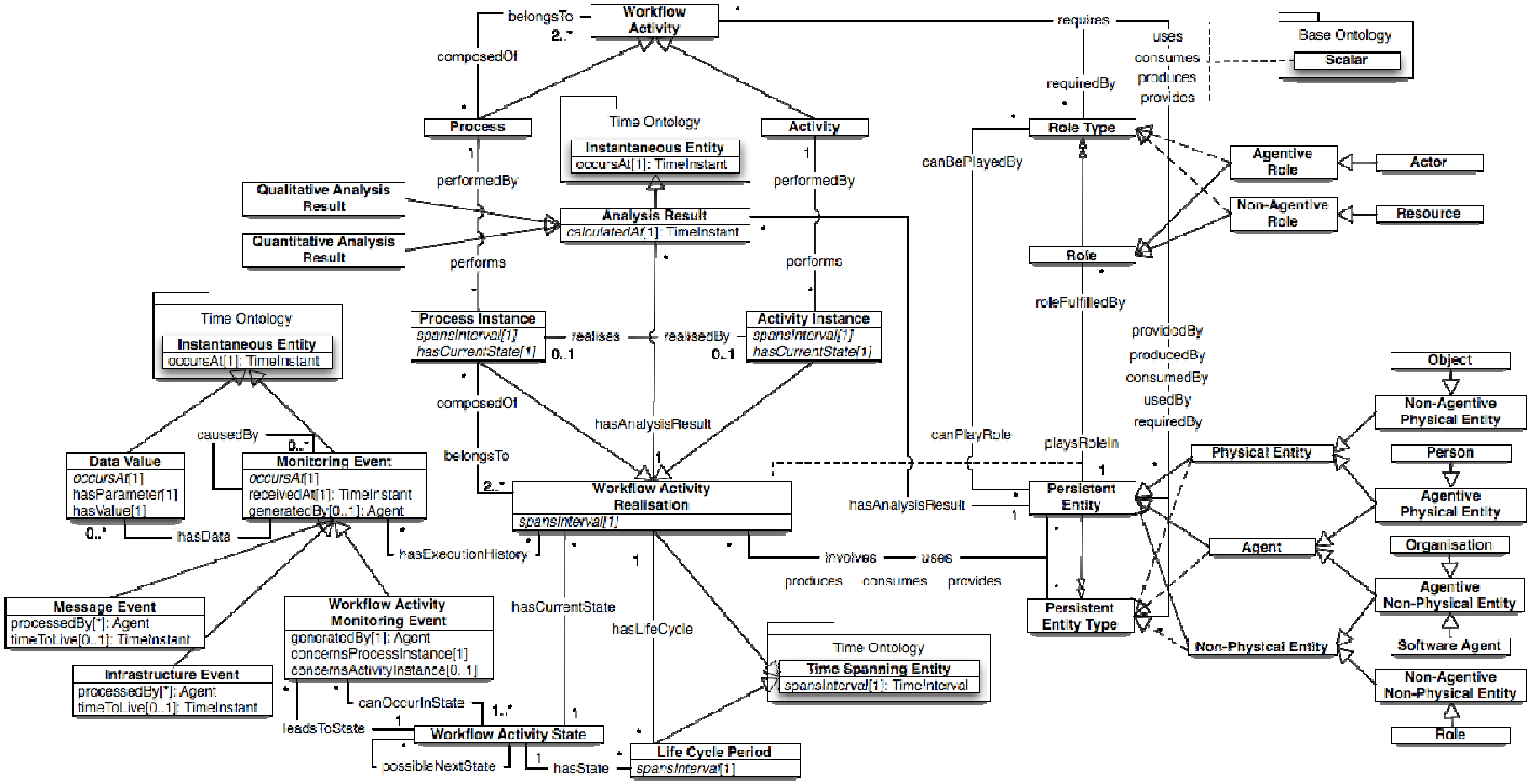

Figure 9. Core Ontology for Business pRocess Analysis.

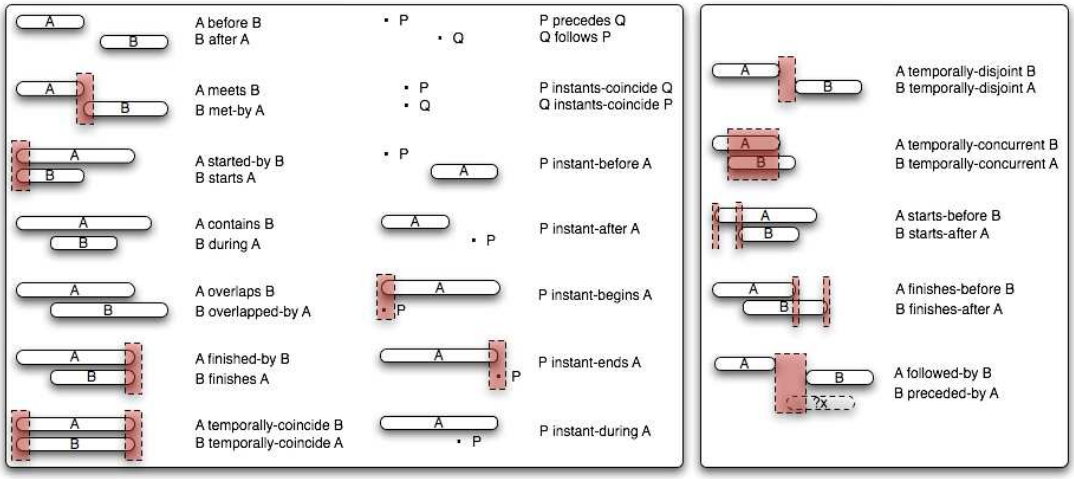

Figure 10. Temporal relations implemented in Time Ontology.

Core concepts in COBRA are *Business Activity* and *Business Activity Realisation*. Business Activity represents the specification of a business activity at a high-level where aspects such as the control flow are abstracted away. There are two kinds of Business Activities, namely *Process* and *Activity*. Activity represents atomic Business Activities whereas Processes are *composedOf* at least two Business Activities. Business Activity Realisations are Time Spanning Entities which represent the actual execution of Business Activities. Mirroring Business Activities, Process Instance and Activity Instance are the two kinds of Business Activity Realizations considered. Despite their name, which originates from BPM literature [4], both are concepts which represent the actual executions of Processes and Activities respectively. In the current version of COBRA that we will use within SOA4All, Business Activity and Business Activity Realizations have been renamed to *Workflow Activity* and *Workflow Activity Realization* respectively. The reason for this is mainly to accommodate the wider range of situations we will encounter within the project which is not uniquely limited to business processes but rather to any Workflow Activity, may this be a BPEL workflow, a Web service or even a Semantic Web service.

COBRA uses this high-level categorization as a foundational basis but it does not go however much further in the reuse of existing foundational ontologies for it aims at supporting analysis of processes and a complete grounding into this kind of ontologies would carry an important computational overhead. Instead, we provide a simple categorization of Persistent Entities specifically tailored to our needs, though informed by DOLCE, whereby we contemplate Physical and Non-Physical Entities which are disjoint. Physical entities are those that have a mass.

Physical and Non-physical Entities are further refined into *Agentive* and *Non-Agentive*. The distinction between these classes which are obviously disjoint, is that Agentive Entities are those that can take an active part within some specific activity. Finally, we define *Agent* as the union of both Physical and Non-Physical Agentive Entities. We include for reference and self-containment a few concepts widely used within BPM. For instance, we include *Object*, *Person*, *Organization*, *Software Agent*, and *Role*. COBRA, for its purpose is to provide core definitions for supporting workflow analysis, does not refine these classes any further. Instead they serve as placeholders for including additional conceptualizations as defined within SUPER, or other approaches like the Enterprise Ontology [11] or TOVE [12]. By doing so we aim at reducing the ontological commitment, while we support the seamless integration of further specific conceptualizations. Finally, since sometimes one needs not specify a concrete instance but rather the type, e.g. ``you require a computer'', we have

defined the meta-class Persistent Entity Type such that all the sub-classes of Persistent Entity are instances of Persistent Entity Type. This is depicted in Figure 9 by means of a double-headed arrow.

Workflow Activity Realizations are the bridge between the high-level conceptualization of the BPM domain and the low-level monitoring information captured at runtime by the IT infrastructure. Thus, Workflow Activity Realizations are further characterized by an *execution history*, a *life-cycle*, and the *current state* of the execution. The execution history is a set of *Monitoring Events* relevant for monitoring the life-cycle of a Workflow Activity, see Figure 9. Monitoring Events are Instantaneous Entities generated by Agents. They are characterized by a reception timestamp which is to be filled by the logging infrastructure upon reception of an event. The main goal of this attribute is to support monitoring even in environments where clock synchronization mechanisms are hardly applicable. Additionally, Monitoring Events can have a causality vector, i.e., the set of Monitoring Events that caused that particular event. This supports capturing the actual derivation of events by the monitoring infrastructure as necessary for Complex Event Processing. Finally, Monitoring Events might be characterised by additional associated data, which is expressed as Data Value instances. These instances identify a particular parameter and the value associated to it.

Monitoring Events are further refined into *Message Events* and *Workflow Activity Monitoring Events* (renamed from Workflow Activity Monitoring Event). The former accommodates Event-Based environments so that their execution can also be traced. The latter supports monitoring the life-cycle of Workflow Activity Realizations in Process-Aware Information Systems. Workflow Activity Monitoring Events therefore concern a specific Process Instance and, depending on the granularity of the event occurred, may also concern an Activity Instance. Similarly to the proposals in [13][14], Workflow Activity Monitoring Events are centered around the notion of state model. Every event identifies a particular transition within the state model, the transition being indicated by means of the *leadsToState* attribute. Conversely the *canOccurInState* attribute allows to ensure that the transitions are consistent with the prescribed state model or to detect anomalies within the execution history possibly due to missing events.

COBRA supports the definition of specific state models is a simple ontological form by means of the *Workflow Activity State* concept which has a set of *possibleNextStates*. Workflow Activity States are used to further characterize Workflow Activity Realisations with the *hasLifeCycle* and *hasCurrentState* slots. The former captures the overall life-cycle of Workflow Activity Realizations as a set of Life-Cycle Periods which are Time Spanning Entities whereby the executed Workflow activity was in a particular state. The latter is a shortcut for avoiding heavy usage of temporal reasoning in order to obtain the current state. On the basis of these Life-Cycle Periods it is possible to revisit the complete life-cycle of a Workflow Activity Realization in a suitable manner for interval-based temporal reasoning. Instead of prescribing a particular state model and the corresponding events COBRA remains agnostic from the domain-specific details. Still, we provide an extension, i.e., Events Analysis Ontology, with a set of generic event processing forward-chaining rules that can derive information based Workflow Activity Monitoring Events. These rules will be detailed in the next section.

Finally, given that COBRA aims to support Business Process Analysis, both Persistent Entities and Workflow Activity Realizations are characterized by a set of Analysis Results. Analysis Results are Instantaneous Entities of a *Quantitative* or *Qualitative* nature[4]. Being

[4] Note that we have used slot renaming for occursAt.

part of the core ontology for analyzing business process, this allows us to reuse results across different types of analysis, which paves the way for enhancing current techniques [13]. For instance, metrics computed at runtime can be reused when performing Reverse Business Engineering, mining results can be applied during monitoring, etc.

6.2.1 Events Ontology

COBRA has been extended with a reference Events Ontology (EVO) [8] that provides a set of definitions suitable to capture monitoring logs from a large variety of systems and ready to be integrated within our core ontology for analysing business processes. EVO is a semantic monitoring format employed by the SUPER execution infrastructure which includes Workflow Engines [15] (e.g., the Apache ODE BPEL execution engine) and Semantic Execution Environments [16] such as WSMX [17] and IRS-III [18]) that supporting the execution of Semantic Web services. Therefore the monitoring format has shown its applicability to the purposes of SOA4All. Nevertheless, it is based on existing syntactic formats, e.g., MXML [19] or the Audit Trail Format by the Workflow Management Coalition [14] which therefore confers it the ability to capture logs generated by a plethora of systems as those one could envision in SOA4All. The ontology is depicted in Figure 11.

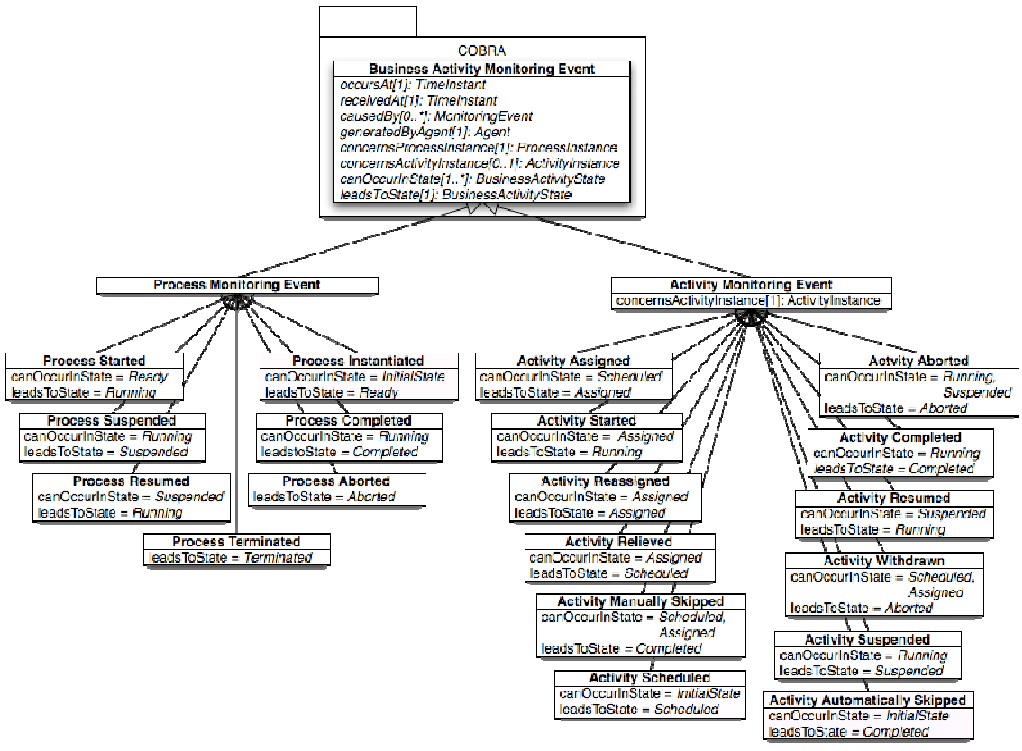

Figure 11. Events Ontology.

As prescribed by COBRA, EVO is centered around a state model that accounts for the status of processes and activities, see Figure 12. The state model has been captured ontologically and enhanced with additional relations. For instance it is possible to determine whether an Activity Instance has been allocated-- *isAllocated*--which is true for those that are either in state Running, Suspended, or Assigned. It is also possible to determine whether a Process is active-- *isActive*--which is equivalent to Running, or inactive--

isInactive--which is true for the rest of the states, etc.

The state model does not distinguish between Process Instances and Activity Instance. The reason for this is mainly to simplify some tasks, e.g. monitoring of active Workflow Activity Realisations. Still, this necessary distinction is preserved within the logs by means of the Workflow Activity Monitoring Events defined, see Figure 11. EVO includes two subclasses, namely *Process Monitoring Event* and *Activity Monitoring Event*. EVO currently captures seven Process Monitoring Events and twelve Activity Monitoring Events based on the state model in Figure 12. Process Monitoring Events capture the different transitions which are possible for Process Instances. A Process Instance can be *Instantiated*, *Started*, *Suspended*, *Resumed*, *Completed*, Aborted and *Terminated*. Activity Monitoring Events, in addition to the typical execution events, contemplate the distribution of work to Agents. Thus, there are events that capture the scheduling of activities, the *Assignment*, *ReAssignment*, or *Relief* of activities to specific agents. Additionally like MXML, EVO contemplates the possibility for skipping activities either manually or automatically, which lead to a correct completion. Finally, EVO captures the abortion of activities by means of two events *Activity Aborted* and *Activity Withdrawn*. The distinction between the two lays in the fact that only started activities can be aborted.

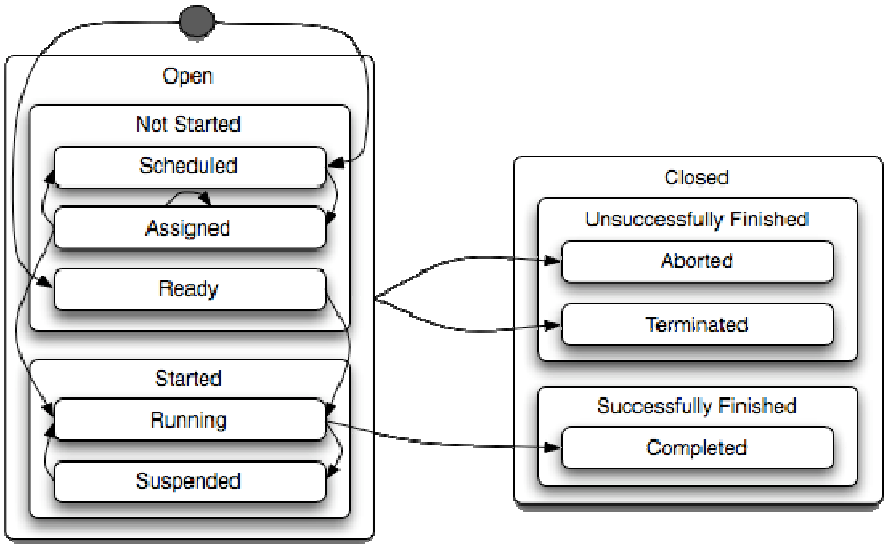

Figure 12. Events Ontology State Model.

6.3 SENTINEL

In order to achieve the level of automation and genericity required by businesses we advocate for an extensive use of semantic technologies. In the remainder of this section we will describe SENTINEL (SEmaNTic busINess procEsses monitoring tooL), a tool that advances the state of the art in BAM by making an extensive use of semantic technologies in order to support the integration and derivation of business level knowledge out of low-level audit trails generated by IT systems. Within SOA4All we will build upon our initial work on SENTINEL and we will extend it towards supporting more advanced monitoring functionalities on top of the project infrastructure.

6.3.1 Overall Approach

BAM is the meeting point between Data Warehousing, Business Intelligence, Process Monitoring and Process Mining. It is therefore based on the integration and application of

diverse technologies, which are already challenging on their own. The quality and level of monitoring provided by state-of-the-art monitoring tools are rather similar and not surprisingly major efforts are devoted to presenting the information in a simple yet meaningful way better supporting humans in the interpretation of monitoring information [15].

Often companies invest in very expensive customized solutions that integrate domain-specific details in order to increase the level of automation. After long periods of consultancy and development, quite advanced solutions can be obtained but their benefit in mid and long term is not so clear. The business world is characterized by ever changing conditions, and one key to success is precisely the capacity to adapt and react to these changes. Customized applications make typically certain assumptions that after a while do not hold anymore. As a consequence, companies need to engage into expensive development processes in order to readapt the software. What is needed instead are general purpose solutions that can handle heterogeneity and evolution and still support advanced BAM facilities.

Despite the advances so far, there is still a long way to go to achieve the level of adaptability in process-aware systems that current businesses require. The reason for this is mainly that the semantics of the data manipulated concerning some specific business domain, are only present in the head of the business analyst and are not available for automated processing by machines [5]. Automating these tasks in a domain-independent manner requires capturing both static knowledge, like for example a company's processes and organizational structure, and procedural knowledge such as how to detect that a process will miss a deadline.

Conceptualizing static knowledge in a way that can support automated reasoning by machines is well supported by means of ontologies [20]. On the other hand, research in Knowledge Engineering has shown that Problem-Solving Methods (PSM) are an appropriate way for encapsulating procedural knowledge in a reusable way [21][22]. PSM are reusable knowledge-based components able to support the development of highly complex systems by integrating diverse task-specific but domain-independent expertise for solving knowledge intensive tasks using ontologies as the lingua franca [21][22]. Their genericity stems from the formalization of the relevant concepts for performing a specific task in an ontological form constructing in this way a formal interface to the task-specific expertise. This interface can then be used for applying the problem-solving expertise over domain specific data by defining mappings that bridge the gap between both conceptualizations. Additionally there is often a conceptual separation between the task to solve and the method used which supports the application of diverse techniques on a per case basis. Research in Knowledge Engineering has shown that PSM are an appropriate means for abstracting away the complexity of Knowledge-Based Systems leading to modular solutions that support the maintenance and evolution of the systems [21][22].

SENTINEL therefore aims to support advanced monitoring techniques by making use of extensive conceptualizations of businesses and Workflow Activities as presented in the previous section, together with PSM able to provide domain-independent expertise for analyzing workflow executions automatically.

6.3.2 Architecture

Figure 13 presents the overall architecture of SENTINEL including external components that interact with the tool represented as computers or repositories. These components represent infrastructural services from SOA4All (e.g., Semantic Execution Environment, Provisioning Platform, etc) as well as third-party servers providing services over the Web. The monitoring tool will connect to these external components through the SOA4All communication infrastructure, i.e., the Distributed Service Bus that uses Enterprise Service

Bus [23] implementations as well as the Triple Spaces infrastructure [24]. In addition to message routing and endpoint virtualization life-cycle management facilities will allow for starting, stopping, removing components and/or artifacts that are deployed to them.

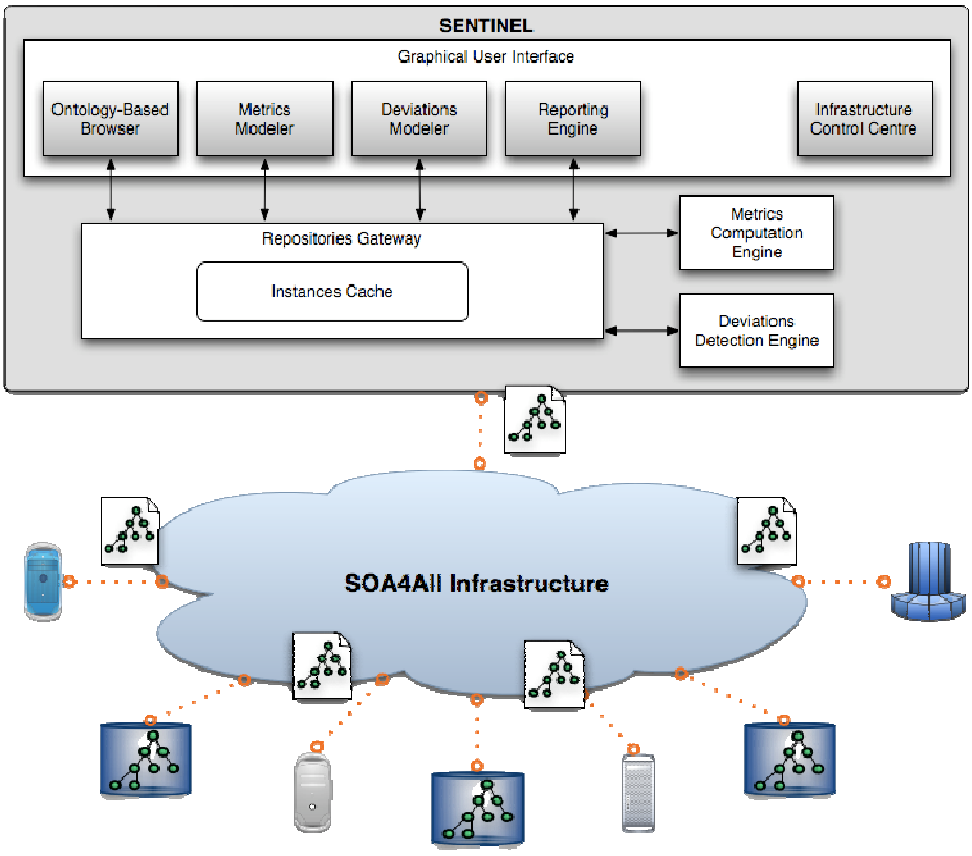

Figure 13. Architecture of SENTINEL.

The architecture of the monitoring tool is basically decomposed into three main parts. At the lowest level there is the support for accessing repositories capturing knowledge about domains, users, processes, services, previous executions, or even metrics and deviations. This knowledge will be used by the other two components, namely the analysis engines (i.e., Metrics Computation Engine and Deviations Detection Engine) and the advanced graphical user interface.

In order to support effectively monitoring services, the monitoring tool will be connected to the SOA4All monitoring platform in order to process monitoring messages populated by the execution infrastructure. This notification machinery is expected to be provided in an efficient yet distributed way by the Triple Space infrastructure developed in WP1. Upon reception, events will be processed by the tool to populate the user interface, to derive new knowledge, and to trigger additional computations in a similar vein to that of Complex Event Processing [25]. This will include rules that detect certain conditions (e.g., the instantiation of a process, its completion, etc) and derive additional knowledge to be stored in the repositories. For example, any time a new event is received, the status of the process instance should be updated including the different states of the process instance during its life-cycle, the agents (systems or humans) involved, etc.

The main underlying characteristic of SENTINEL is the use of semantic technologies as the

key pillar for achieving a domain-independent yet highly automated and advanced monitoring tool. The tool will be supported by a set of spaces capturing information about processes, agents, etc as illustrated in Figure 13. This information will be captured in terms of the ontologies previously introduced, namely COBRA and EVO, as well as additional domain-specific models.

6.4 Missing Functionality

The current status of SENTINEL based on the research carried out in SUPER already highlighted some of the main benefits that can be achieved in monitoring by using semantic technologies. However, most of the work was necessarily devoted to establishing a solid basis for developing new processing techniques. In SOA4All we will build upon the results obtained in order to better profit form the use of semantic technologies for the monitoring of services and service-based applications. In particular we shall expand the gathering of information to contemplate the infrastructure and interactions with external services. We shall expand our software towards the generation of further information in ways that can better support the analysis of services. This information will be the basis for supporting the developing of advanced processing algorithms to detect process deviations. And finally, we will indeed open up our software to the Web, offering a dynamic and Web-based user interface able to bring both detailed and high-level information as required by different users. In the remainder of this section we shall cover these aspects in more detail.

6.4.1 Information Gathering

As we previously introduced, the events that will be communicated will expressed in terms of COBRA and EVO. For the purposes of SOA4All, the ontological model has been extended with support for capturing infrastructure messages in order to monitor the SOA4All infrastructure, and variable changes in order to get a detailed view on process executions. The former will be supported by means of the concept *Infrastructure Message* in COBRA, whereas the latter will be captured using *Variable Changed Event* instances, see Figure 14.

On the basis of these new events, SENTINEL will support monitoring the infrastructure tracking things like the deployment of services, and it will also support the monitoring of any communication with external services. These interactions will be the main means for SOA4All software to gather knowledge about services owned by third-parties, paving the way for determining overall performance figures about them, or whether the correct Message Exchange Pattern was used.

The use of an ontological representation of the monitoring information, will allow us for instance to display, query and filter the information in generic terms or to access additional information captured within the different repositories. For instance, if an event refers to a concrete Process Instance we can access the Process Instance definition and visualise graphically its definition using a graphical representation. Additionally we will include statistical information concerning processes and their executions. This view will be populated with metrics and so-called Key Performance Indicators concerning workflow activities and their execution. This information will be backed by Semantic Spaces as a distributed repository which will deal with the technical intricacies for manipulating large amounts of interconnected data efficiently.

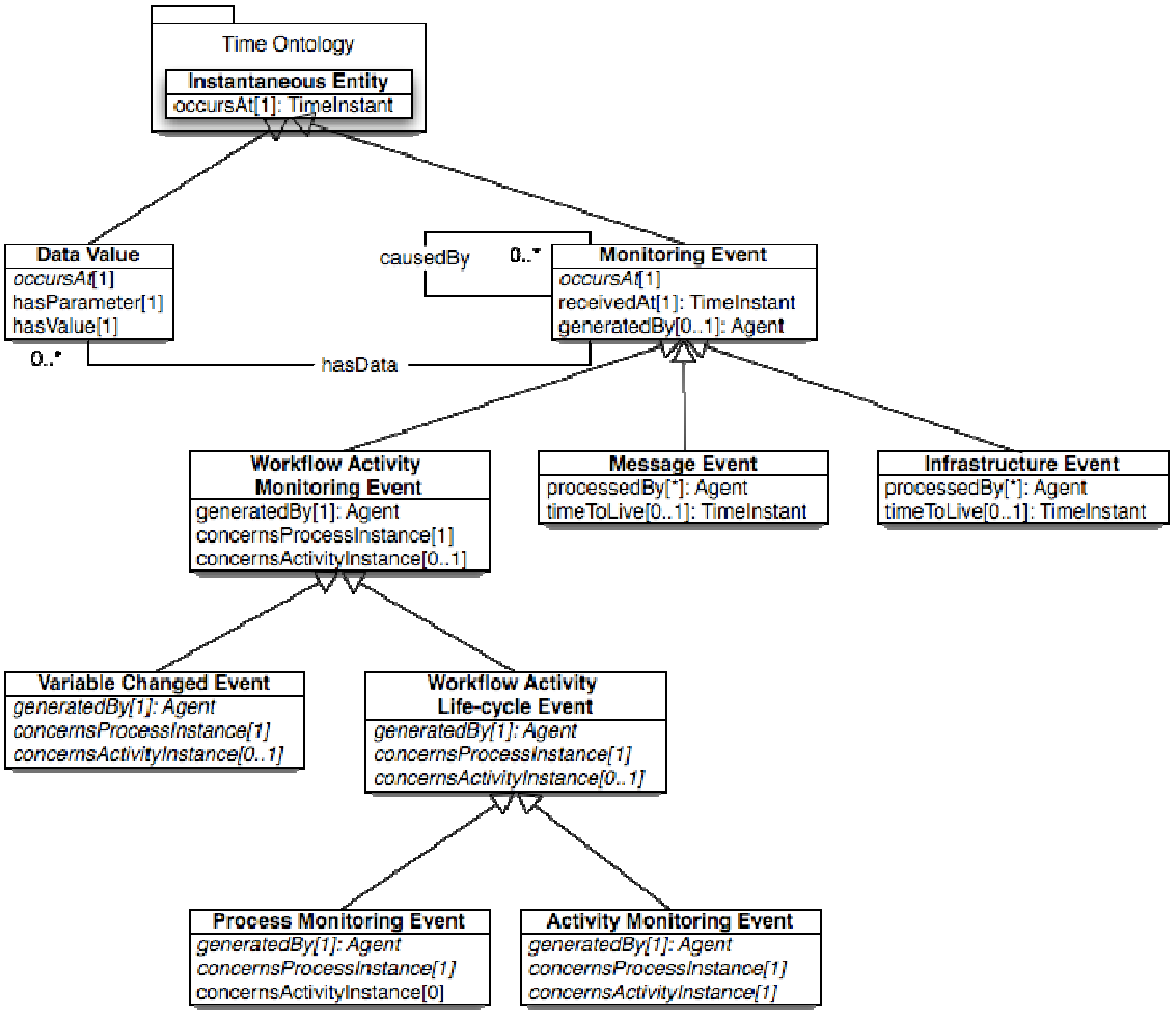

Figure 14. Excerpt of the ontological model illustrating the different kinds of events.

6.4.2 Information Generation

Raw monitoring information is important in that it allows to derive additional information about the execution of some software in sufficiently abstract terms allowing us to determine whether the execution was correct, whether it can be improved, etc. Some of this information, e.g., process execution time, can be computed automatically independently from the domain and have therefore been included by default in the tool. However, most of the data that can be derived and that can be of interest to the users when analysing processes, is domain dependent. For instance, the computation of the Quality of Service depends on the set of parameters taken into account and the priorities given to them.

The current version of SENTINEL supports the derivation of additional knowledge based on the raw events captured by the monitoring infrastructure in a rather ad-hoc form [37]. In particular, general-purpose metrics about the execution time, or the number of failures are embedded within the software and are computed continuously. Additional metrics can be added by hand and some work was devoted to supporting the definition of metrics by users. However, metrics definition remains a complex process and their integration within the overall monitoring process needs to be improved.

6.4.2.1 Metrics Definition

We will enhance SENTINEL with better support for defining and computing domain-specific metrics. Our approach will be based on previous research on Problem-Solving Methods. In particular, we build upon the Task Method Domain Application (or as we previously stated TMDA) framework [9] for Knowledge-Based Systems reuse and development. In a nutshell, TMDA prescribes constructing Knowledge-Based Systems based on the definition of task ontologies that define classes of applications (e.g., diagnosis, classification), method ontologies that capture the knowledge requirements for specific methods (e.g., heuristic classification), domain ontologies which provide reusable task-independent models, and application ontologies for integrating domain models with domain-independent problem solvers.

Metric computation will be thus defined within the TMDA framework as a kind of task that takes a Metric definition as input and returns a Quantitative Analysis Result with the actual value for the Metric at that particular point in time, see Figure 15. A key aspect in order to support metrics computation concerns the support included for defining the metrics themselves. In this respect we will use Metrics Ontology that will provide us with the capacity for specifying and computing metrics, as necessary for analysing and managing business processes and services, in a domain-independent way.

On the basis of our conceptualization we can capture kinds of metrics, e.g., “process instance execution time”, as well as specific metrics to be computed, e.g., “process instance X execution time”. The former are defined as concepts, whereas the latter are modeled as instances. In this way we can provide libraries of metrics such as general purpose ones, or specific for some domain like Supply-Chain, and at analysis time the analyst can specify which of these metrics should be computed over which entities by instantiating them. This provides a convenient way for organizing metric definitions and seamlessly supports the comparison of results by kind of metric, e.g., “which is the process which takes longer”, as well as it allows tracking their evolution over time.

Central to Metrics Ontology is the concept Metric which is defined as a Quantitative Analysis. Metrics are specified by a set of input roles that point to domain-specific knowledge [9]. We refine Metrics into two disjoint kinds, Function Metrics and Aggregation Metrics. A Function Metric is a metric that can be evaluated over a fixed number of inputs. For example, the Metric Process Instance Execution Time is a Function Metric which takes as input one Process Instance. Conversely, Aggregation Metrics (e.g., “average process execution time”) take an arbitrary number of individuals of the same kind (e.g., a set of Process Instances) as input. Therefore, Aggregation Metrics are computed over a population in order to obtain an overall perception of some aspect of interest such as the average execution time of some particular process. The population to be processed can be defined intensionally as an (domain-specific) ontological query so that the metric computation can focus on certain processes, or resources of interest. In this respect the use of semantic technologies play a key role towards supporting business analysts in the analysis of processes, allowing them to use their domain-specific terminology and still use a generic machinery to process the information in a seamless way.

The current state of the tool support the automated computation of metrics defined as specified earlier. However, defining these metrics by hand is not a trivial task given the required conceptualisation work to be performed as well due to the particular syntax for representing them. In SOA4All we will therefore devote efforts to supporting users in defining these metrics in more convenient ways. To this end we will provide user interface support that will represent a simple entry point for users to define metrics and will internally generate the required ontology elements for further processing.

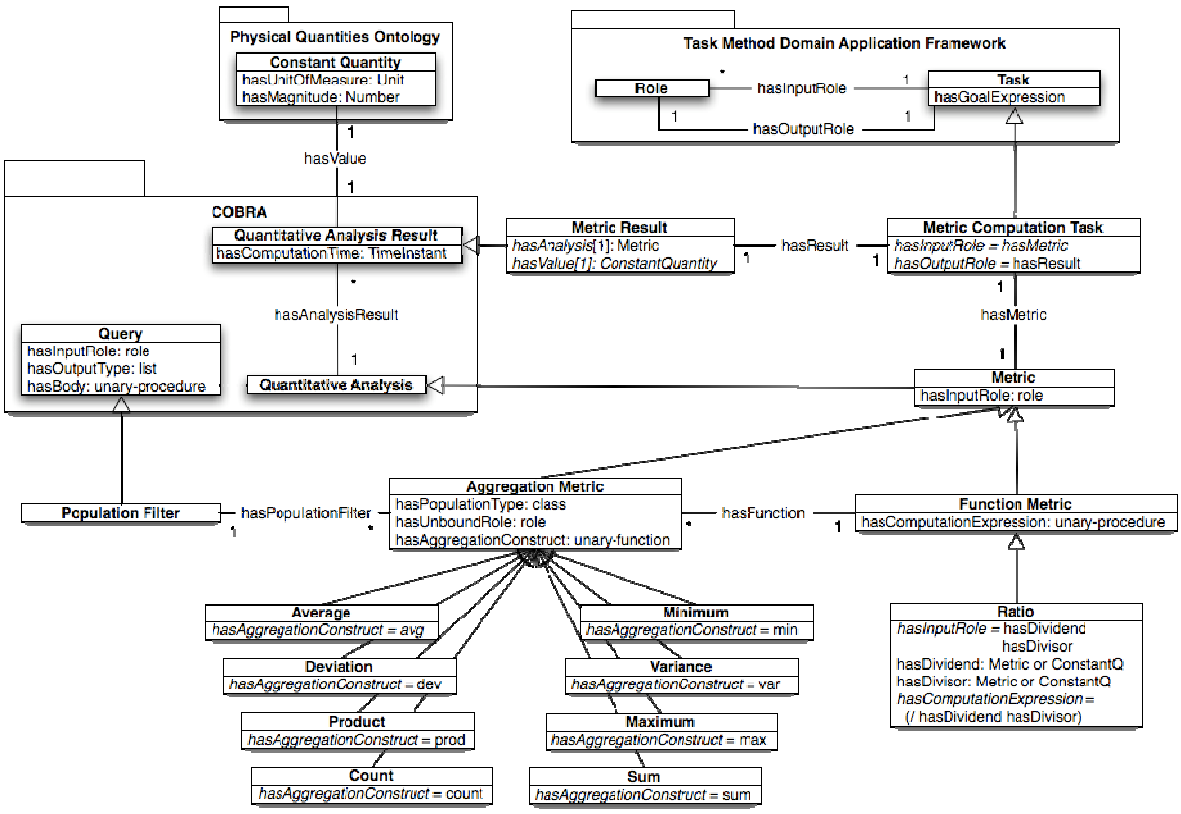

Figure 15. Metrics Ontology.

6.4.2.2 Metrics Computation

In order to support the automated computation of metrics, which is indeed metric dependent, each metric has a computation expression which is defined as a unary procedure. In this respect it is worth noting that the language used to define the metrics themselves as well as to develop Metrics Computation Engine is Operational Conceptual Modelling Language (OCML) [9]. OCML seamlessly supports the integration of static and dynamic knowledge paving the way for a rapid prototyping of a fully operational solution. It is worth noting however that OCML provides support for importing and exporting data represented in other languages such as OWL and WSML and therefore allows the wider application of our techniques over data represented in Semantic Web and semantic Web services formalisms.

Currently, general purpose metrics are computed continuously and user-defined metrics are computed on demand and based on the whole body of data. Indeed, these restrictions are too important. Not every event during the lifecycle of processes or services will affect all the metrics, users may want to compute their metrics periodically or at concrete moments, and the computation of metrics may often benefit from previously available results.

Metrics computation will therefore be extended in order to cover for these requirements. In particular we shall contemplate 3 main modes of computation: on demand, periodic, continuous (possibly based on some condition. The first mode, which is already available, will allow users or other applications to trigger the computation of a given metric on demand. The second mode, will allow the specification of a time interval upon which a metric will be computed. In this way users can save computational resources by indicating that particular metrics should only be computed after a certain amount of time has elapsed. Finally, the third mode will allow the specification of conditions which, when met, will trigger the computation of metrics.

These three modes will provide users with the possibility to specify when metrics should be computed thus paving the way for saving computational resources, which in monitoring tools is an important requirement. Additionally, we shall explore means for computing certain metrics incrementally so that previous calculations can reduce the amount of work to be performed. This is foreseen to have a great performance impact on the computation of what we previously referred to as aggregation metrics given that only a very reduced portion of data will have to be taken into account for computing new metric values.

6.4.3 Information Processing

Deriving information from raw events is necessary for properly analysing processes and services. This is, indeed, a basic requirement whether the final analysis will be performed by users or by some software. However, the amount of data that needs to be generated and processed poses important problems. On the one hand, although higher-level derived information is more scarce and useful than raw data, how the information is represented (internally and visually) determines to an important extent how easily it can be analysed. On the other hand, processing the data is time consuming and how fast we obtain some information may determine which actions can be taken, if any.

Analysing the information is typically a human task, therefore most efforts are often devoted to presenting the data visually in ways that can better support humans in analysing monitoring information. Although we mention these aspects (see Section 7.1.3 for more details), having information represented semantically paves the way for the application of knowledge-based techniques for their analysis. Our work in this respect will focus on two main issues. On the one hand we shall explore how we can guide the exploration and derivation of monitoring data using high-level strategic knowledge. On the other hand we will work further on supporting the application of knowledge-based software for processing the monitoring data.

Guiding the exploration and derivation of monitoring data aims to better support the processing of information by focusing the efforts on those aspects that are expected to be of most relevance. The main idea is that, by knowing the potential impact of certain aspect we can devote more efforts to computing things that will shed more light on the current status of services or processes. To this end we plan to develop the ideas previously described in [39] which is based on the notion of high-level goals and objectives and maps them down to operational concerns that can be measured. In this way, by knowing which are our users main goals or concerns, we can decide which metrics are relevant and compute only these on demand, saving resources and producing data which is mostly driven by users' interests.

Finally, the automated processing of monitoring data will based on the application of PSM ideas in order to provide a generically applicable engine as well as a set of interchangeable methods that can be selected and applied at runtime depending on their suitability. The main idea is to accommodate the large variety or services, users, and domains by means of a set of generically applicable expert modules. Initial work has already been devoted to applying heuristic classification for determining trusted services [40]. In the project we shall expand this work to better link to the monitoring machinery and information. We will therefore explore the application of heuristic classification as the main epistemological approach to diagnosing process and services by allowing us to classify them with respect to predefined classes.

7. User Interface

The widgets, consoles and screens that will be used by the Analysis Platform will be accessible in a unified and integrated manner through the SOA4All Studio. The point of entry to the Analysis Platform, from a user point of view, will be the SOA4All Studio Dashboard (described in D2.4.1), as illustrated in Figure 16. By clicking the "Analysis" button, the user will have access to the different views and widgets described in this section.

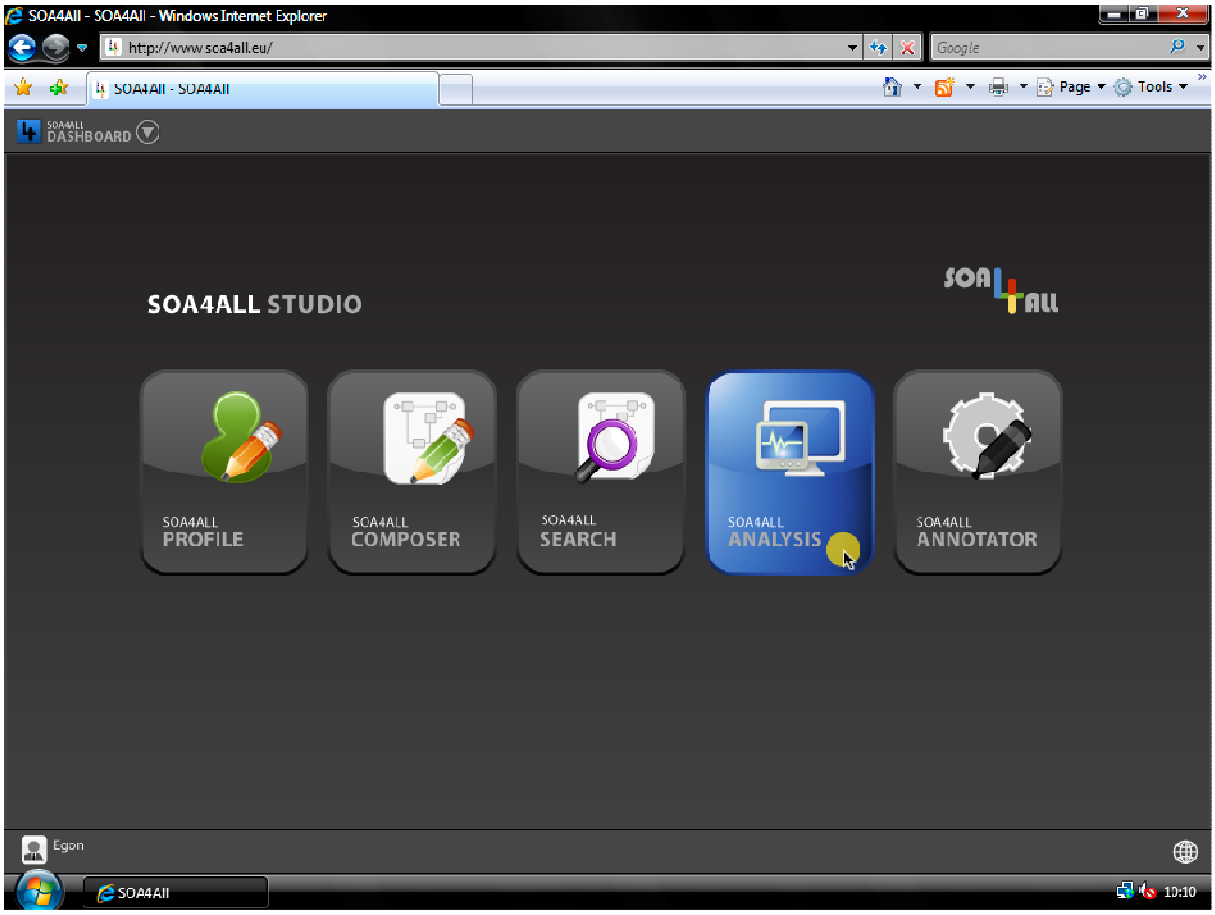

Figure 16. Accessing the Analysis Platform User Interface through the Dashboard

Monitoring information is often structured around three different views [14]: (i) the Process View which is concerned with key performance indicators of processes and services; (ii) the Resource View centered around the resources, human or mechanized, required for executing processes; and (iii) the Object View which focuses on business objects such as inquiries, orders or claims. These three views are populated with statistical information such as the minimum, the maximum, the average or the deviation of some parameter of interest.

These views are of major relevance to analysis and management, and as a consequence they are typically supported by monitoring tools such as Business Activity Monitoring solutions. However, different users have different roles, interests, access rights, and preferences and these vary depending on the specific scenario, the focus of their analysis, etc. The user interface of a fully-fledged general purpose solution must therefore be characterized by its flexibility [14]. This includes for instance support querying, filtering and ordering the information according to user defined specifications [38]. Indeed, given the kinds of users addressed, the specification of these queries and filters should be supported in a simple way so that humans can browse the existing execution information effectively.

Similarly, different domains exhibit particular characteristics, which impede a “one size fits all” approach. The monitoring tool should therefore support users in defining their own visualization templates to be populated with relevant monitoring information. The visualization framework should be supported by a wide range of graphical representations such bar charts, line charts, pie charts, time series charts, etc. Additionally, the visualisation framework should support the presentation of user-defined information combining diverse statistical information about processes, etc.

The use of knowledge-based technologies will play a major role in bringing flexibility to the analysis tool suite to be able to adapt to a myriad of users and services in a seamless way. On the one hand the use of a formal conceptual model closer to human understanding than low-level syntactic representations will bring the body of knowledge to a higher level of abstraction more suitable for human interpretation. On the basis of this conceptual model we shall support humans in defining queries or navigating through the data by simply following the conceptual schema and generating the appropriate ontological queries transparently. This will allow, among other things, to seamlessly navigate across the three layers previously introduced (Infrastructure Level, SOA Level, Business Level). Additionally, we will envisage the use high-level strategic models such as the one presented in [39] in order to guide the presentation of analysis information driven by the importance and impact data can have on underlying services and their related interdependencies.

7.1 Existing Consoles and Widgets

This section presents existing UI elements that are relevant in the context of the Analysis Platform. Even though these consoles focus on lower-level technical monitoring information, they can serve as good starting points for further development in a more user-oriented direction. In fact, some elements of these consoles will be migrated to GWT and integrated in a consistent way into the wider Analysis Platform UI.

7.1.1 PEtALS WebConsole

The PEtALS WebConsole illustrated in Figure 17 is a WEB based GUI which provides distributed monitoring and management features for the PEtALS Distributed Enterprise Service Bus.

The JBI specification PEtALS is based on defines a JMX management API that has been extended to provide advanced management and monitoring features.

This console uses the JMX API to:

- Collect data from all the containers of the management domain, processes this data and display JBI message content and useful statistics like service response time, service and container load, message queue sizes and message repartition.
- Manage the JBI artifacts. It is possible to start, stop, deploy and un-deploy artifacts from the web browser and check their states.

The WebConsole also provides:

- an embedded JBI client used to invoke hosted JBI services directly from the web browser without any additional deployment or configuration.
- A module to configure JBI services, add endpoints, and create service proxies without any generation on the client side.

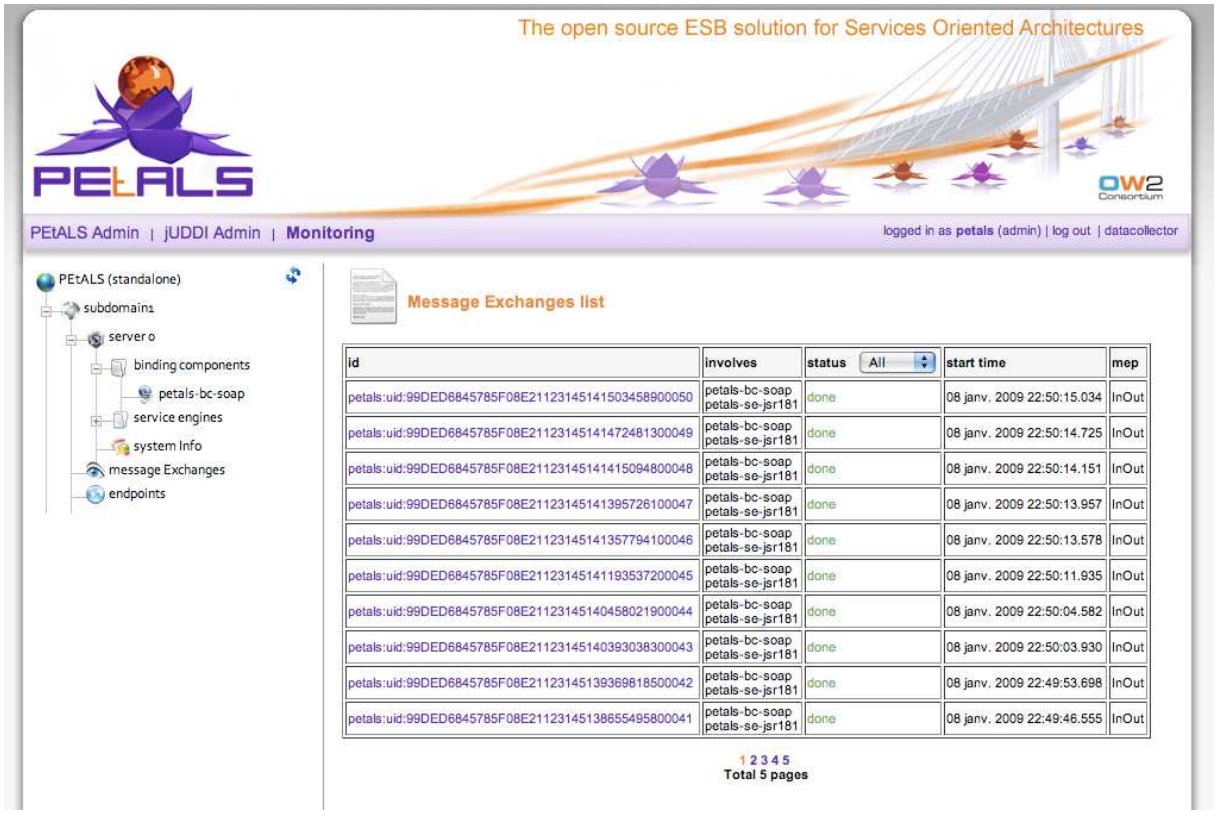

Figure 17. Monitoring Message Exchanges

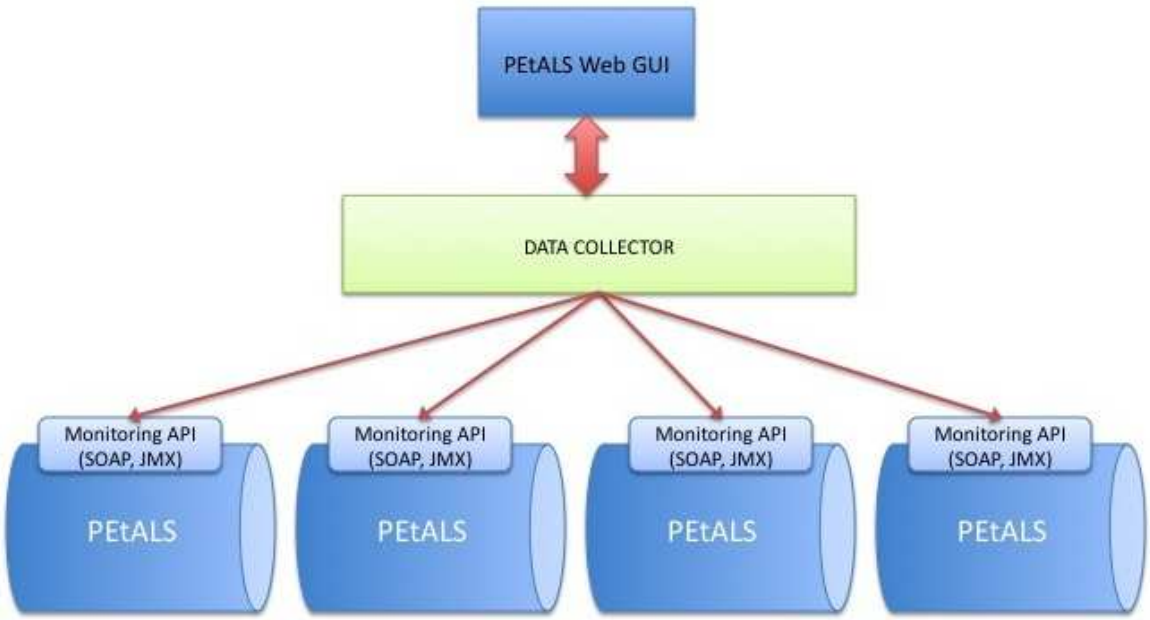

Figure 18. PEtALS High Level Monitoring and Management Architecture

The high-level architecture of the WebConsole illustrated in Figure 18 provides a single access point to manage containers distributed over several nodes and domains. The web application hosted on any Web Application server is connected to an intermediate layer (the data collector) which is in charge of:

- Establishing connections to the monitoring and management PEtALS API. These connections are possible since each PEtALS node have a complete knowledge of the network topology. It means that at initialization time, the data collector get all the connection information from its single access point and is able to create all the distant connections with remote containers. With this approach a command (JMX call) sent to a container from the client (the web application) transits through the data collector layer and is 'routed' to the right container.
- Aggregating data from containers. The data collector can subscribe to JMX events raised by remote containers and aggregate this data which is then exposed to external clients (the WebConsole for example). A example of aggregated data is a message exchanged between containers. To be able to follow this message and to display all the steps, the message is time-stamped on each exchange partner and a global message exchange is exposed on data collector API.

7.1.2 IC2D Monitoring

IC2D is a GUI based on Eclipse RCP (illustrated in Figure 19), which allows to monitor the infrastructure of distributed and grid applications (developed using ProActive/GCM), and deployed using ProActive/GCM deployment model. IC2D subscribes to JMX events raised by ProActive and displays the active objects, JVMs, and ProActive nodes involved in the application, the communication relationships among them, the time spent in communication, along with infrastructure details like the machine load, CPU usage, and memory consumption. It also allows to trigger actions on the infrastructure, like migration of active objects between different nodes.

IC2D acts as a grid collector that subscribes to JMX events from all nodes hosting the grid application. For scalability purposes, collection of distributed events, and distributed triggering of JMX (MBeans) actions are supported by a new JMX connector: instead of relying upon the classical RMI connector for JMX, we rely on a ProActive-based JMX connector that takes advantages of all ProActive communication features (asynchrony, parallelism, security) in order to transport JMX management messages [3].

Although IC2D is not necessarily a monitoring and analysis application/console suitable for the SOA4All end-user, it could be useful for a SOA4All infrastructure provider administrator, in order to analyze parameters of the grid infrastructure. Other metrics can be plugged in if required. However, any client application can subscribe to the ProActive provided JMX events to get the same monitoring information. So, some relevant and aggregated events could be usefully propagated up to the SOA4All analysis platform designed in the current T2.3 and highlighted in this document. To achieve this, it would only be a matter of defining a Grid collector as shown in Figure 5, that would gather, filter and aggregate JMX ProActive events, so making them available for further collection by the analysis platform.

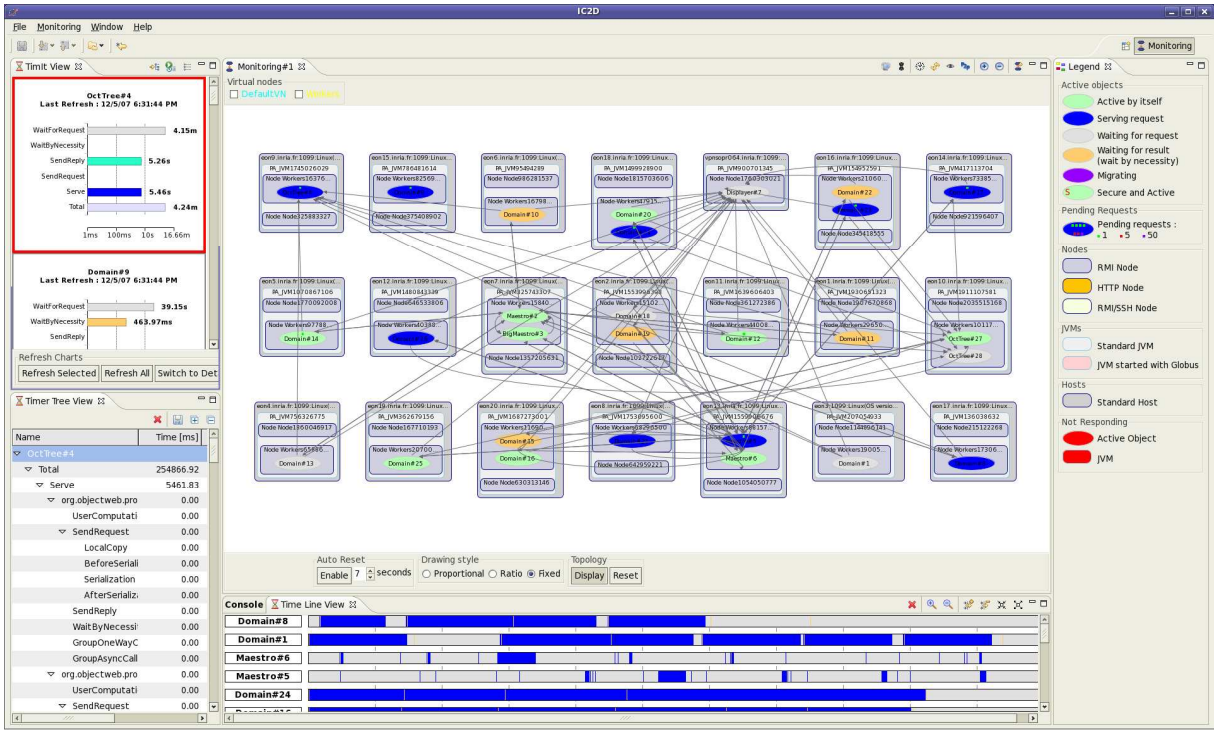

Figure 19. IC2D Monitoring and Analysis tool, for Grid applications and infrastructure

7.1.3 SUPER / SENTINEL UI

The current version of SENTINEL as developed within SUPER makes use of a traditional desktop-based user interface (as illustrated by the screenshot in Figure 20). In the context of SOA4All, the **Graphical User Interface of SENTINEL** will be built on top of the monitoring machinery by making use of the graphical user interface libraries been developed as part of the SOA4All Studio, see D2.4.1. As a consequence any user with a Web browser will benefit from a fully-fledged monitoring interface with an extensive support for the visualization and manipulation of ontologies, the representation of processes, and the access to underlying infrastructural components. Indeed, the runtime infrastructure will be adapted accordingly, see D1.4.1 for additional details.

The user interface will include real-time visualization of monitoring events as they are populated by the execution infrastructure. In order to do so, we shall use the Comet implementation being integrated within the SOA4All Studio infrastructure, which will allow the server to actively push information, as it is available. Further details on how this is achieved can be found in D2.4.1.

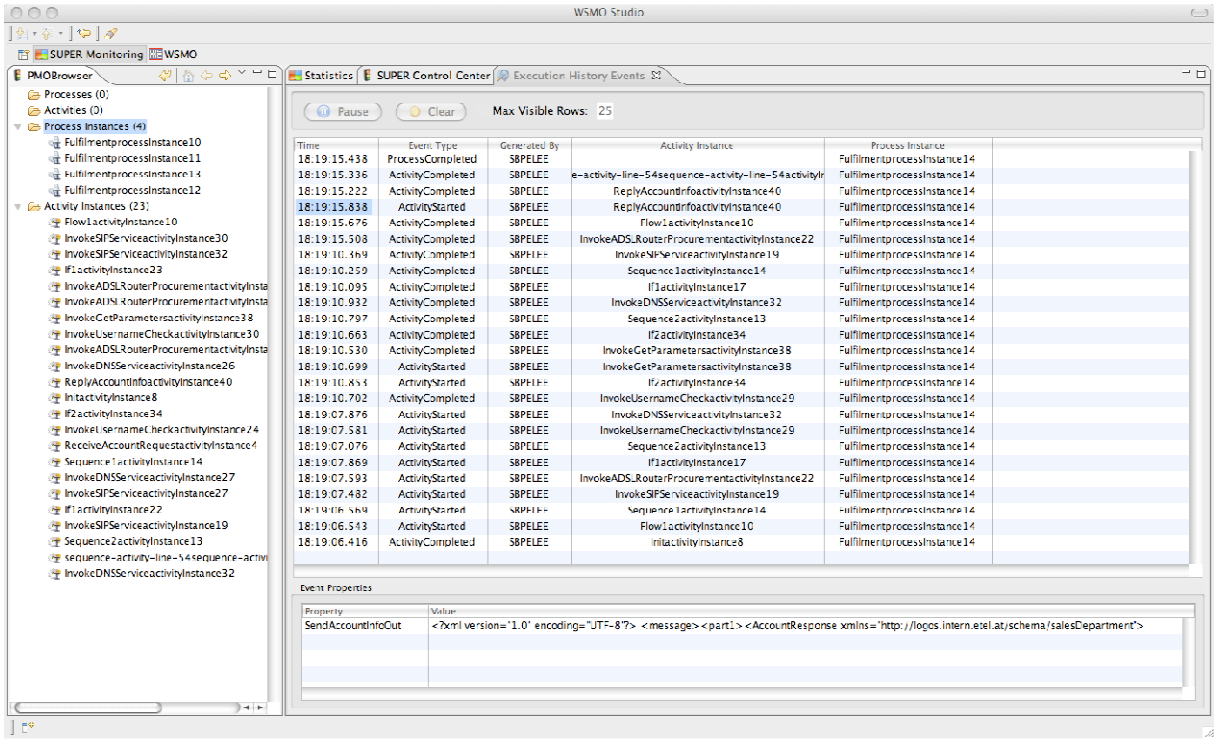

Figure 20. Existing SUPER Monitoring Console

Finally, given the vision of the project that contemplates a large number of users, services and domains, we need to accommodate a large variety of requirements. We shall leverage our ontology-based representation of monitoring data and derived information in order to accommodate these different scenarios in a seamless way. In particular, in addition to the typical monitoring dashboard facilities we shall provide an ontology-based query assistant that will support users in the generation of queries over the analysis data. This user interface will be informed by the ontologies previously described and will allow users to explore the information derived in an intuitive way through the simple generation of ontological queries through a point & click interface. This interface will allow the selection of instances of certain concepts based on a set of conditions over the attributes. As a result of the user interaction, an ontological query in SPARQL will be generated and sent to the Analysis Data Warehouse for its interpretation.

7.2 Missing Functionality (UI mockups)

The mockups presented in this section illustrate the UI functionality that will be developed as part of the Analysis Platform. The entire set of widgets that are illustrated here will be provided by the Studio as components and will fit visually and functionally into the Studio interface (see D2.4.1). They will be programmed and configured by the Analysis Platform in order to drive the display of relevant data in the relevant context.

7.2.1 Search / Favorites View

When users search for services or when they sign on to their session on the Studio, they could get a list of services. A simple monitoring visual clue can be attached to the services in the form of an ON / OFF indicator (green / red), as illustrated in Figure 21. This could signal the availability of the service in the moment of the search. The user can quickly check the same information over a given duration (in which case a percentage can be associated with the icon to indicate percentage of availability). The user can also click on Monitoring Details to quickly go a detailed page for monitoring data (see below). In the case of composite services, there will also be a Service Composition button that will take the

user to the composition / process editor WP2 / WP6 (see below) where monitoring information can be displayed in context.

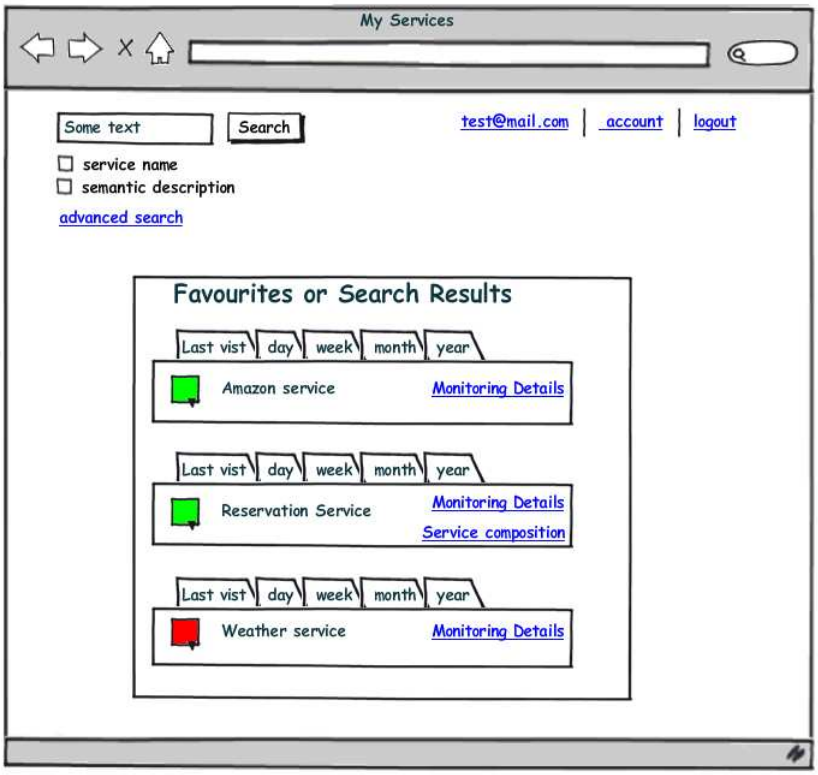

Figure 21. Search / Favourites View

7.2.2 Service Description

By clicking on the service name, the user can probably go to something like a description page for the service (as in all other views here, it is up to the Studio designers to define such screens). The description can contain some basic, aggregated monitoring information in condensed graphical form. By clicking on see the details link, the user can obtain the detailed monitoring information (see Figure 22).

7.2.3 Monitoring Details

On the monitoring details page, the user has a variety of graphical widgets for browsing monitoring data. The user can select what type of data to display and can also select the date range for historical data (slider or date-selection fields). In addition, the user can see the advertised QoS parameters for the service. Where available, alerts can be examined and defined (see Figure 23).

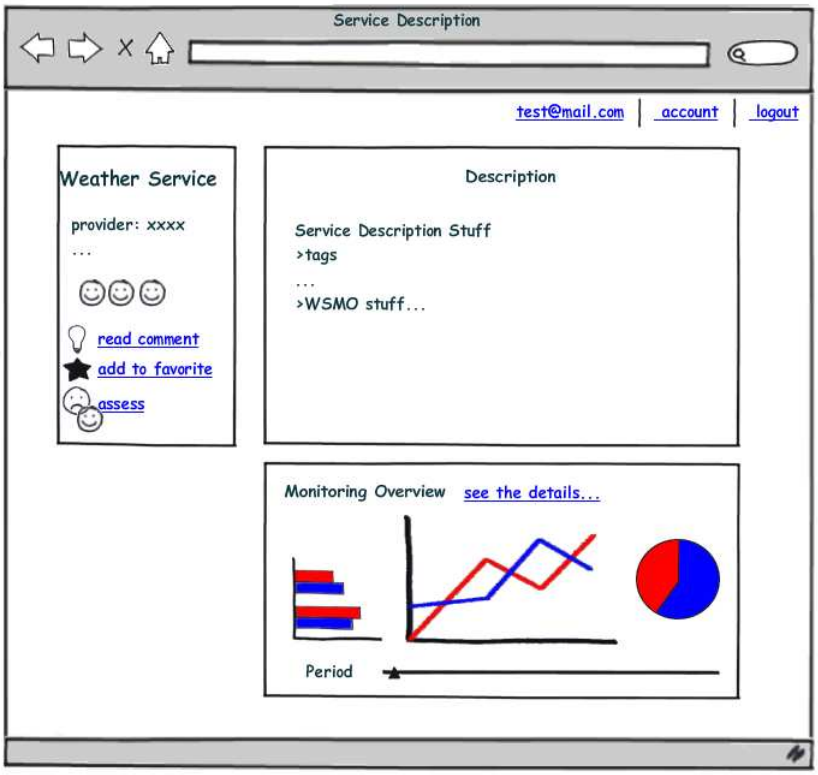

Figure 22. Service Description

Figure 23. Monitoring Details

7.2.4 Defining Alerts

In order to be notified of events such as QoS violation the user must be able to define alerts (essentially filters triggering notification events or other types of actions). The mock-up in Figure 24 illustrates this functionality and shows the type of visual controls that can be involved.

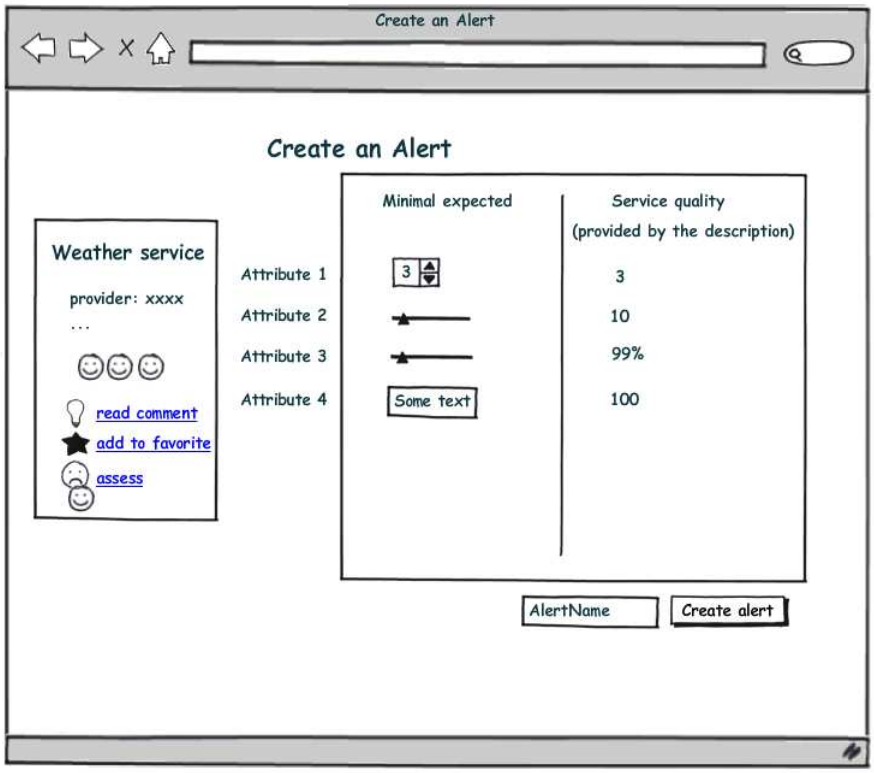

Figure 24. Defining Alerts

7.2.5 Monitoring Information in the Composition Editor

As mentioned above, for composed services or for processes, it is important to present the monitoring information in the context of the composition. Therefore, the editors used to compose services should be leveraged when displaying the monitoring information. In Figure 25, a Reservation service composed of 3 other services is illustrated. The Weather service is in orange indicating an alert associated to it. The user can see the monitoring averages associated to Weather or the list of alerts raised by it. Note that the average values are presented next to the values for the entire composed service (Reservation). This helps put the values of one service in perspective with regard to the entire execution values, therefore showing the relative impact of the service. As in other views, the user can always choose to obtain monitoring details or create new alerts, depending on the context.

7.2.6 Goals that Match Execution Templates

The composition of services will result in several process templates. The view illustrated in Figure 26 will show the different Goals that trigger the execution of each kind of process template, and to what degree.

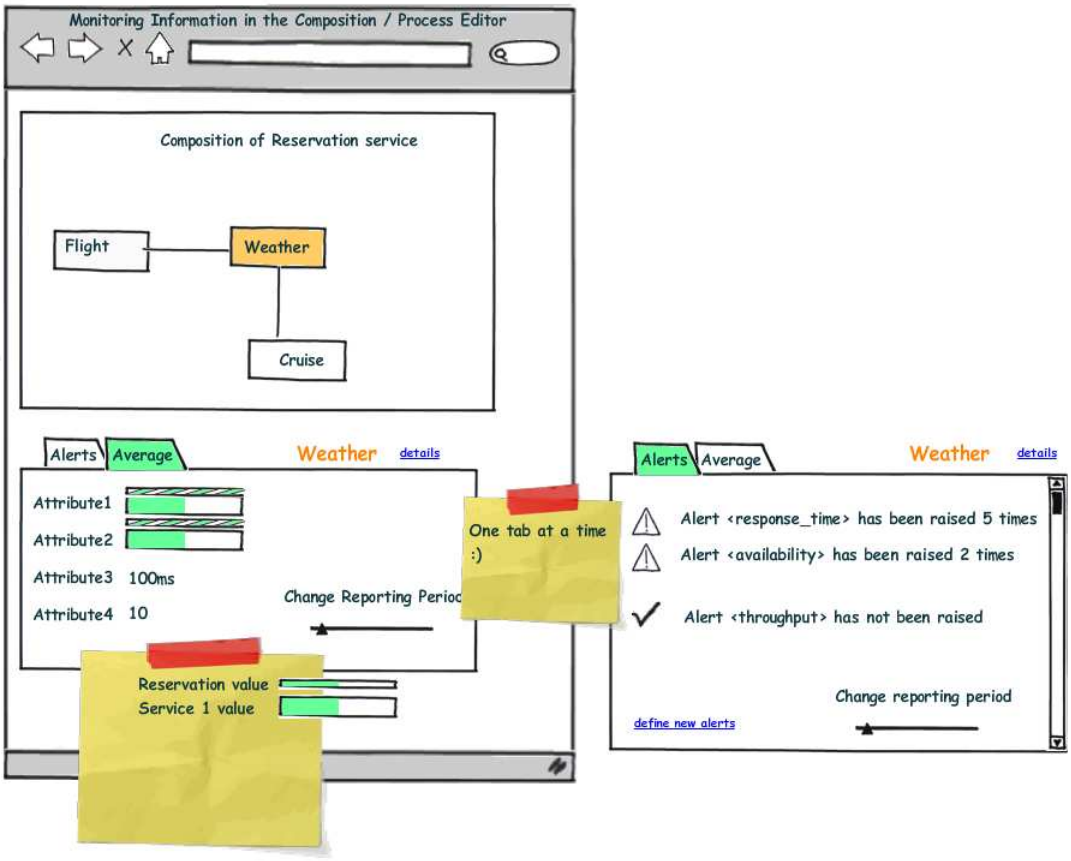

Figure 25. Monitoring Information in the Process Editor

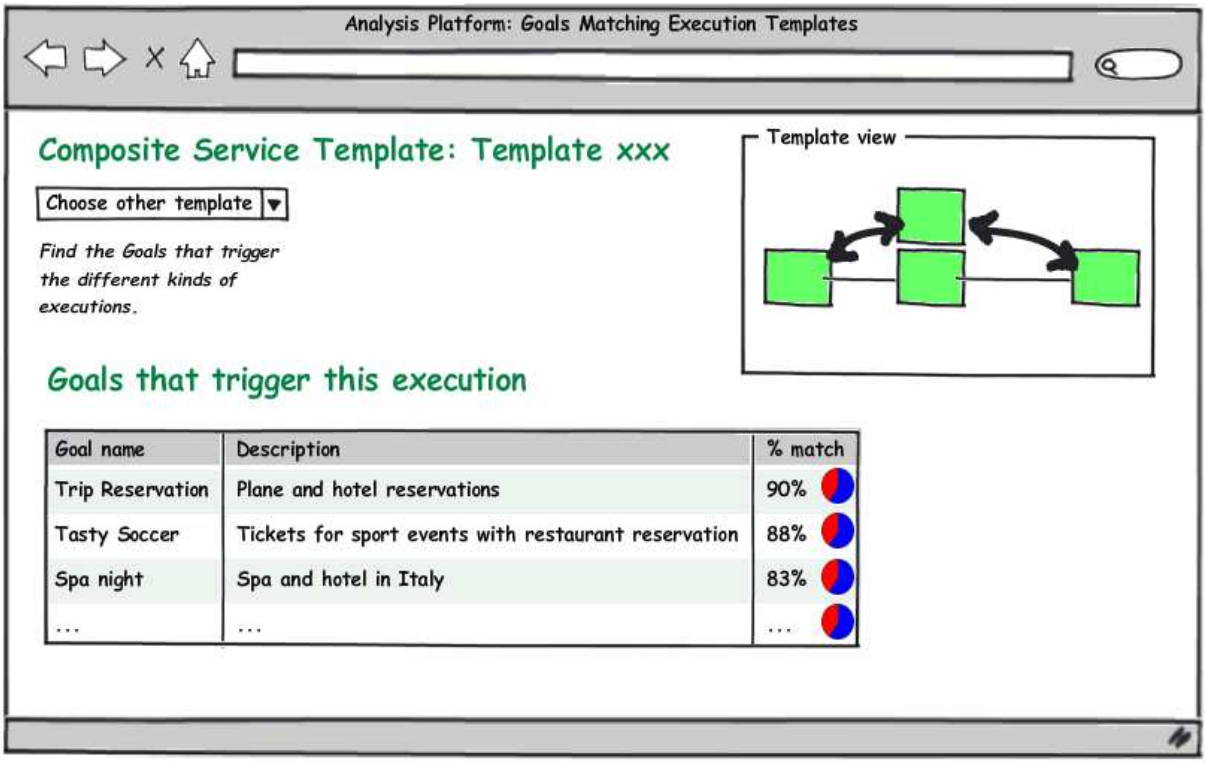

Figure 26. Goals Matching Execution Templates

8. Conclusion and next steps

The Analysis Platform will obtain data from the SOA4All infrastructure and the Studio editors, will analyze and extract knowledge from it and will present meaningful information to the end-users as well as to the automatic adaptation mechanisms developed in SOA4All.

This document sets the architectural and technological basis for the development of the Analysis Platform. It describes in detail the components that will be created as well as the existing foundations which are to be extended in order to provide the complete required functionality, as identified through the requirements analysis. The document also provides visual descriptions of the future Analysis user interface and describes the usage scenarios in which it will be used.

The Analysis Platform has strong connections to other tasks in the SOA4All DoW. In particular it is naturally related to T1.6 for monitoring event extraction, to T2.4 for general Studio integration, to T2.6 for integration of monitoring information in process views and to T6.5 for runtime adaptation of process executions based on event analysis. Beside these tasks, T2.3 has connections with the entire set of services provided by WP2. As part of the next steps, collaboration with all these identified tasks is crucial and will be particularly pursued.

As described in the DoW, the first prototype of the Analysis Platform will be developed by M18. All major functionalities will be available by M18, even if some will be of a rudimentary nature. The BEP and MM will be functional, while KOPE and SENTINEL will benefit from extensions and initial integration into the overall platform. Also by M18, the user interface will be aligned with developments in the Studio. Lastly, M18 will bring initial integration with the DSB and the Semantic Space. A fully functional and complete prototype will be available by M30, completely integrated in the Studio and the overall SOA4All infrastructure.

References

[1] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)

[2] A. Mos, A. Boulze, S. Quaireau, and C. Meynier, “Multi-layer perspectives and spaces in SOA”, In Proceedings of the 2nd international Workshop on Systems Development in SOA Environments (SDSOA'2008), ICSE'08, Leipzig, Germany, May 2008.

[3] F. Baude, V. Legrand, V. Lestideau, Large Scale Service Deployment - Application to OSGi. 3rd Int. conf. on Autonomic and Autonomous Systems (ICAS 2007), June 2007, Athens.

[4] van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.): Business Process Management, Vol. 2678. Springer (2003) 1-12

[5] Hepp, M., Leymann, F., Domingue, J., Wahler, A., Fensel, D.: Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management. In: Lau, F.C.M., Lei, H., Meng, X., Wang, M. (eds.): ICEBE. IEEE Computer Society (2005) 535-540

[6] Watson, H.J., Wixom, B.H.: The Current State of Business Intelligence. Computer 40 (2007) 96--99

[7] Alves de Medeiros, A.K., Pedrinaci, C., van der Aalst, W., Domingue, J., Song, M., Rozinat, A., Norton, B., Cabral, L.: An Outlook on Semantic Business Process Mining and Monitoring. Proceedings of International IFIP Workshop On Semantic Web \& Web Semantics (SWWS 2007) (2007)

[8] Pedrinaci, C., Domingue, J., Alves de Medeiros, A.K.: A Core Ontology for Business Process Analysis. 5th European Semantic Web Conference (2008)

[9] Motta, E., Lu, W.: A Library of Components for Classification Problem Solving. The Open University (2001)

[10] Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26 (1983) 832--843

[11] Uschold, M., King, M., Moralee, S., Zorgios, Y.: The Enterprise Ontology. Knowledge Engineering Review 13 (1998) 31--89

[12] Fox, M.S.: The TOVE Project Towards a Common-Sense Model of the Enterprise. IEA/AIE '92: Proceedings of the 5th international conference on Industrial and engineering applications of artificial intelligence and expert systems. Springer-Verlag, London, UK (1992) 25--34

[13] Gangemi, A., Borgo, S., Catenacci, C., Lehmann, J.: Task taxonomies for knowledge content. EU 6FP METOKIS Project D07 (2004)

[14] Muhlen, M.z.: Workflow-based Process Controlling. Foundation, Design, and Implementation of Workflow-driven Process Information Systems., Vol. 6. Logos, Berlin (2004)

[15] van Lessen, T., Nitzsche, J., Dimitrov, M., Karastoyanova, D., Konstantinov, M., Cekov, L.: An Execution engine for BPEL4SWS. 2nd Workshop on Business Oriented Aspects concerning Semantics and Methodologies in Service-oriented Computing (SeMSoc) in conjunction with ICSOC (2007)

[16] Norton, B., Pedrinaci, C., Domingue, J., Zaremba, M.: Semantic Execution Environments for Semantics-Enabled SOA. it - Methods and Applications of Informatics and Information Technology Special Issue in Service-Oriented Architectures (2008) 118--121

[17] Fensel, D., Kerrigan, M., Zaremba, M. (eds.): Implementing Semantic Web Services: The SESA Framework. Springer (2008)

[18] Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B., Pedrinaci, C.: IRS-III: A broker-based approach to semantic Web services. Web Semantics: Science, Services and Agents on the World Wide Web 6 (2008) 109--132

[19] van Dongen, B.F., van der Aalst, W.M.P.: A Meta Model for Process Mining Data. In: Missikoff, M., Nicola, A.D. (eds.): EMOI-INTEROP, Vol. 160. CEUR-WS.org (2005)

[20] Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge Acquisition 5 (1993) 199--220

[21] Schreiber, G., Akkermans, H., Anjewierden, A., de Hoog, R., Shadbolt, N., de Velde, W.V., Wielinga, B.: Knowledge Engineering and Management: The CommonKADS Methodology. MIT Press (1999)

[22] Studer, R., Benjamins, R., Fensel, D.: Knowledge Engineering: Principles and Methods. Data Knowledge Engineering 25 (1998) 161-197

[23] Hohpe, G., Woolf, B.: Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions. Addison-Wesley Longman Publishing Co., Inc., Boston, USA (2003)

[24] Fensel, D., Krummenacher, R., Shafiq, O., Kuehn, E., Riemer, J., Ding, Y., Draxler, B.: TSC - Triple Space Computing. Journal of electronics & Information Technology (Elektrotechnik & Informationstechnik), special issue on ICT research in Austria (2007)

[25] Luckham, D.C.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley Longman Publishing, Boston, USA (2001)

[26] Web Services Distributed Management: Management Using Web Services (MUWS 1.1) Part 1 OASIS Standard, 01 August 2006 Available at: http://docs.oasis-open.org/wsdm/wsdm-muws1-1.1-spec-os-01.pdf

[27] Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.1 OASIS Standard, 01 August 2006 Available at: http://docs.oasis-open.org/wsdm/wsdm-mows-1.1-spec-os-01.pdf

[28] Basic Profile Version 1.0, 16 April 2004, Available at: http://www.ws-i.org/Profiles/BasicProfile-1.0.html

[29] Web Services Resource 1.2 (WS-Resource) OASIS Standard, 1 April 2006, Document identifier: wsrf-ws_resource-1.2-spec-os, Available at: http://docs.oasis-open.org/wsrf/wsrf-ws_resource-1.2-spec-os.pdf

[30] Web Services Base Notification 1.3 (WS-BaseNotification), OASIS Standard, 1 October 2006, Document identifier: wsn-ws_base_notification-1.3-spec-os, Available at: http://docs.oasis-open.org/wsn/wsn-ws_base_notification-1.3-spec-os.pdf

[31] (WSA, 2004) Web Services Addressing (WS-Addressing), W3C Member Submission, 10 August 2004, Available at: http://www.w3.org/Submission/ws-addressing/

[32] The COMET Programming Model, http://en.wikipedia.org/wiki/Comet_(programming)

[33] Google, The rocket-gwt Framework, http://code.google.com/p/rocket-gwt/wiki/Comet

[34] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, Wil M. P. van der Aalst: The ProM Framework: A New Era in Process Mining Tool Support. ICATPN 2005: 444-454

[35] Bondi A. B., Characteristics of scalability and their impact on performance, Proceedings of the 2nd international workshop on Software and performance, Ottawa, Ontario, Canada, 2000, ISBN 1-58113-195-X, pages 195 – 203

[36] Gómez-Pérez, J. M. and Corcho, O. 2008. Problem-Solving Methods for Understanding Process Executions. *Computing in Science and Engg.* 10, 3 (May. 2008), 47-52. DOI= http://dx.doi.org/10.1109/MCSE.2008.78

[37] Pedrinaci, C., Lambert, D., Wetzstein, B., van~Lessen, T., Cekov, L., Dimitrov, M.: SENTINEL: A Semantic Business Process Monitoring Tool. Ontology-supported Business Intelligence (OBI2008) at 7th International Semantic Web Conference (ISWC), Karlsruhe, Germany (2008)

[38] W. Hur, H. Bae, and S.-H. Kang. Customizable Workflow Monitoring. Concurrent

Engineering, 11(4): 313–325, 2003.

[39] C. Pedrinaci, I. Markovic, F. Hasibether, and J. Domingue. Strategy-driven business process analysis. In 12th Conference on Business Information Systems (BIS), 2009.

[40] Galizia, S., Gugliotta, A., Pedrinaci, C.: A Formal Model for Classifying Trusted Semantic Web Services. 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand (2008)