	SocEDA ***Cloud based platform for large scale social aware EDA*** ANR-10-SEGI-013	

SocEDA

Document name:	Specification of the Event-based monitoring framework v1
Document version:	1.0
Task code:	T3
Deliverable code:	D3.1.1
WP Leader (organisation):	PetalsLink
Deliverable Leader (organisation):	PetalsLink
Authors (organisations):	Christophe GATTI (THA)
Date of first version:	06/04/12

Change control

Changes	Author / Entity	Code of version
Creation of the document	Christophe GATTI	0.1
Review of the plan	Christophe GATTI	0.2
Add of several figures	Christophe GATTI	0.3
Introduction of Monitoring Framework cut as MVC model §2.4	Christophe GATTI	0.4
Add of explanations about GUI §2.4.3	Christophe GATTI	0.5
Add of explanations about monitoring component following meeting with project coordinator	Christophe GATTI	0.6
Various corrections and conclusion added	Pierre CHATEL	0.7
Taken into account of corrections of P.CHATEL	Christophe GATTI	1.0

Table of Contents

Table of Figures

1. Introduction

This document is part of SocEDA components specification and introduces the Business *Event Monitoring* framework. It is the first version of the specification, identified as deliverable D3.1.1 of the WP3. This specification will evolve within the future deliverables D3.1.2 and D3.1.3.
The goal of SocEDA project is to develop and validate an elastic and reliable federated SOA architecture for dynamic and complex event-driven interaction in large highly distributed and heterogeneous service systems.
Consequently, this first chapter will discuss about notions directly related to the project, such as Business Events, Complex Event Processing, Semantics and Ontology. The following chapters will focus more on the monitoring framework specifications.

1.1. Semantics and ontologies

An **ontology** formally represents knowledge as a set of concepts within a domain, and the relationships between those concepts. It can be used to reason about the entities within that domain and may be used to describe the domain.
For example, one could consider an ontology to represent information about scheduled flights. E.g. which airlines offer direct flights between two particular airports, what reservations passengers have (i.e. first class, business class etc.), which airlines operate flights using certain aircrafts, which flights are operated by which airlines, which aircrafts are manufactured by a given company, and so on.
This aspect is important to elicit because as the monitoring framework we intend to build involves monitoring business-related activity and each (business) operator, such as a flight controller to follow the aforementioned example, could be interested in a specific information to monitor. In the scope of this deliverable, semantics are used to categorize, organize, and sort information related to different business topics, for monitoring purposes

1.2. Complex Event Processing

Complex event processing is a technology that allows correlating *basic events* to *complex events*. Climbing up the ladder from technical to (complex) business events, complex event processing decouples providers of technical information and consumers of semantically rich information.

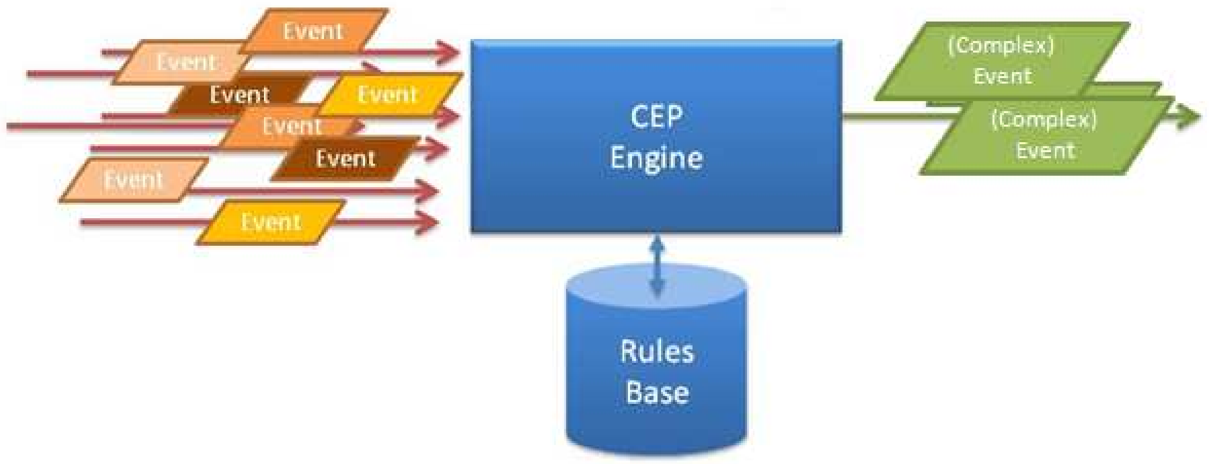


Figure 1 : Complex Event Processing

As mentioned previously, (complex) business events are deduced from one or several technical event(s) through rules-based complex event processing, thus providing a higher level of abstraction.

To make the connection with the use-case *Crisis Management* – that is about a situation in which a large quantity of radioactive substance is accidentally released in the atmosphere, due to a critical accident in a French nuclear plant – where multiple and heterogeneous actors are involved, communication between all of them is ensured through events, and particularly, through business events, as shown by the following figure:

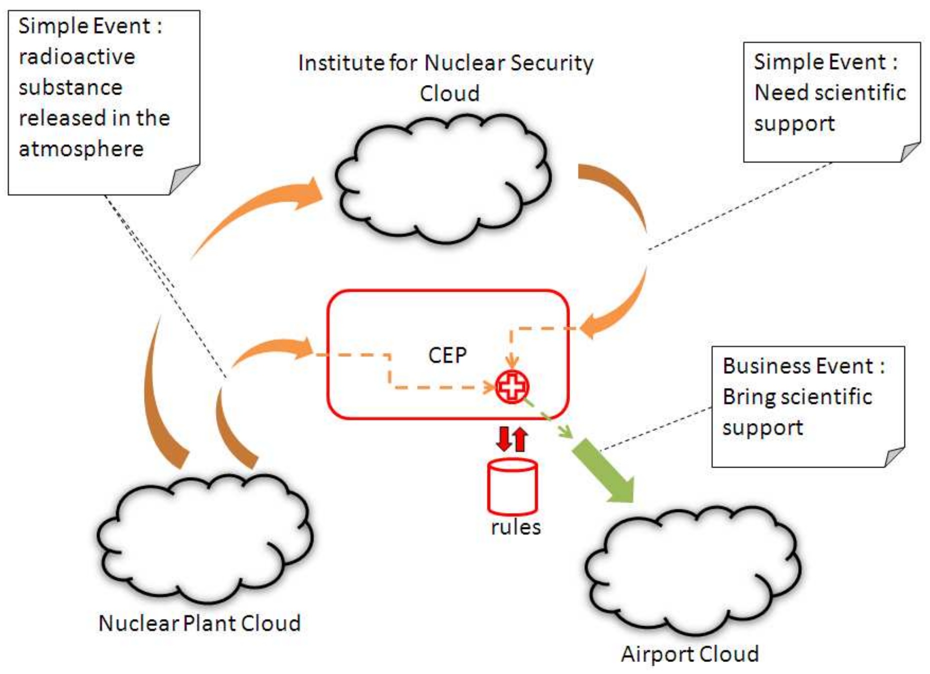

Simple Event : radioactive substance released in the atmosphere
Institute for Nuclear Security Cloud
Simple Event : Need scientific support
CEP
Business Event : Bring scientific support
rules
Nuclear Plant Cloud
Airport Cloud

Figure 2 : Example of Business Event

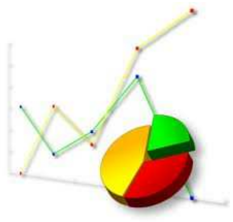
In this example, the nuclear plant sends an event to the IRSN institute, warning them about a release of hazardous gas into the air. The IRSN institute, once notified, sends an event at its turn, evoking the need of scientific experts to be on place so as to measure the extent of damage.
The complex event processor, being notified of all the events, consults its rules (using input events), finds a match (a result expressed by the correlation of both events), and computes a business event, at destination to the airport (the nearest airport from the scientific experts location), requesting the urging transport of those people.

1.3. Business event monitoring

In the scope of SocEDA, services communicate using events. It is one particular aspect of monitoring. Focusing on business event monitoring, requires focusing first on the concept of monitoring.
Monitoring is the action of supervising activities in progress to ensure they are on-course and on-schedule in meeting the objectives and performance targets.
The monitoring in SocEDA is based on supervising business events. It does not include enforcement (i.e. taking decisions while following-up the monitoring of business events). It graphically reports business events notifications and evolutions to end-users and has no influence over the overall operation of the platform. At this point, we consider the business event monitoring.

“Let's take a step back to where we originally thought we were going with technologies such as enterprise resource planning systems and data warehouses. Clearly, the intent was a good one - know your customer, react to problems faster through better business intelligence and research situations before making significant business decisions. Emerging from this quagmire was the concept of business activity monitoring (BAM), i.e., managing a process such as a customer interaction and monitoring the flow of that process to ensure it stays on track. BAM monitors business processes in real time in an effort to support operational improvements. BAM was the first step in a journey that has since evolved into specialist technologies that help manage the organization operationally and strategically. One of those technologies is business event monitoring (BEM). Where BAM typically concerns itself with managing a single business process, BEM is generally concerned with monitoring all current processes to provide meaningful alerts and analytics to users. Think of BEM as real-time data mining.”[1]

BEM is a means of filtering the tsunami of information about business events that pours through the company every day by recognizing those data points that are most important to particular persons or workgroups.



[1] http://www.information-management.com *by Sam Barclay*

2. Monitoring Framework

2.1. Architecture

The business event monitoring framework is developed respecting the MVC pattern.

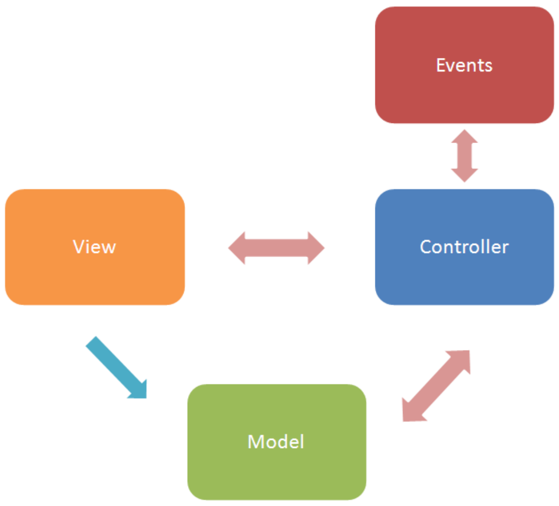


Figure 3 : MVC Pattern

The MVC pattern stands for Model, View, and Controller. It is a software design for interactive computer user interfaces that separates the representation of information from the user's interaction with it.
The following parts discuss about each part of it.

Note:
The next version of this document will include distributes aspects – i.e. replication of the components, placement of part of the different parts of the framework into clouds, and so on.
For the moment, the business event monitoring framework is divided in three parts, a model, a controller, and a view.

2.1.1. Model

The model is clearly, in our case, all the events in the system at a given time. All events being raised into the platform are categorized by Topic. A *Topic* is a logical grouping of events And refer to the notion of Topic defined in the document *D3.2.1 Specification of the governance infrastructure v1*. For example, considering the UC Air Traffic Flow Management, some events will correspond to the flight characteristics, others to hotel reservation while others will be related to the weather, and so on. Thus, in the platform, subscriptions are done by Topics. There are events Topics producer.

Moreover, every business event is enriched by metadata. This metadata expresses a triple of a variable, its current value, and a description – one example of metadata could be “incoming flight current arrival speed is 100m/s”. Moreover, an ontology could then associate a data to a datatype, for example the possible dimensions for a given data.

Note:
In the current implementation of the SocEDA components that the business monitoring depends on, metadata can be placed on business events, but conversion from WS-Notification to RDF typed events loses this information. This will be corrected in future release of the platform.

2.1.2. Controller

For the controller part, a monitoring component has been defined. It is in charge of collecting business events that end-user has subscribe to – using Topics definition – and makes computations over collected events data.

At design time, a business expert writes CEP rules using CEP editor, to express business events. Those business events will thus have no producer. At runtime, they are thus unknown by the governance. The monitoring component – the controller is the sole component – will subscribe to the SeaCloud (please refer to the SeaCloud specification document for more information) for a Topic (classification of business events by Topics will be decided at design time) and that’s only when the DiCEPE (please refer to the DiCEPE specification document for more information) will notify business events of those Topics that the business event monitoring framework will be able to transform them if needed and translate them to the end-user view.

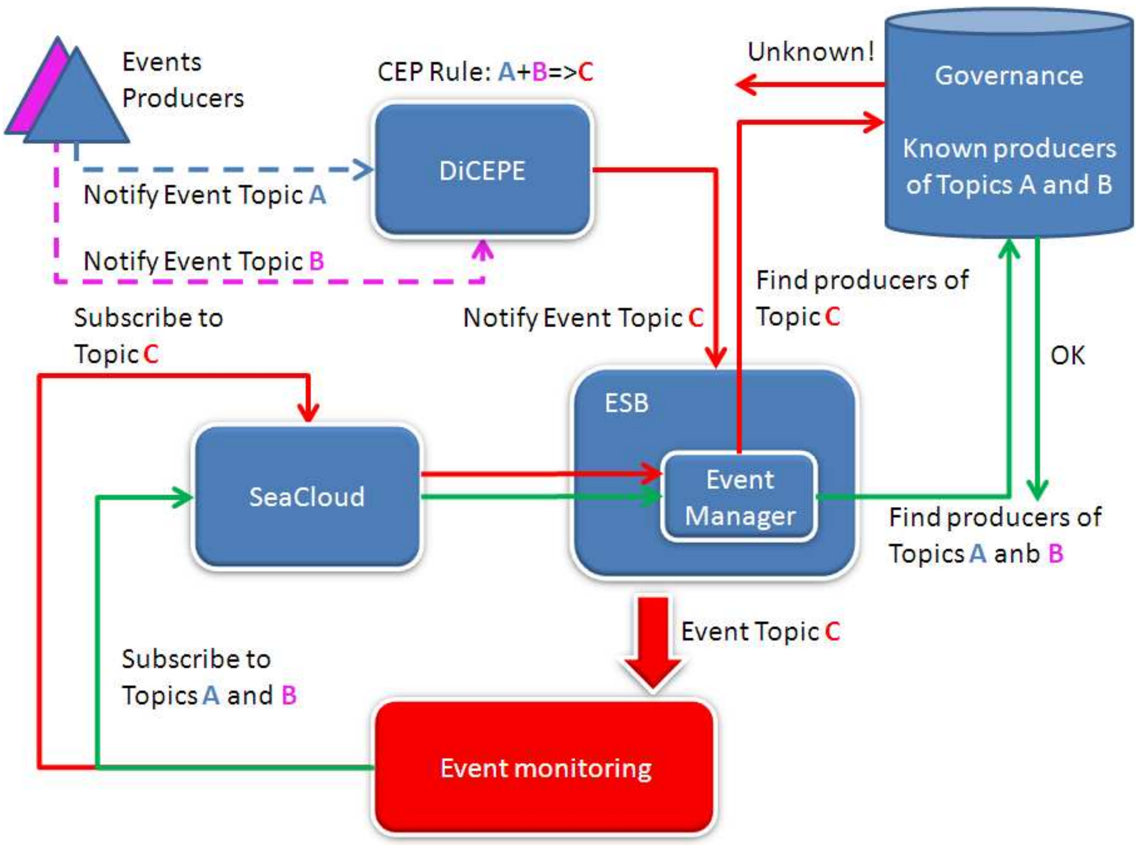


Figure 4 : Business event monitoring at runtime

The previous figure illustrates two cases of figure:

1. The subscription is done at the SeaCloud level, asking about interest for Topics A and B of events:
    a. The SeaCloud translate the request to the corresponding EventCloud (one deployed BY Topic) that ask the governance for Topics producers. Those are known (EasyESB is connected to the Topics producers).
    b. When events issuing from those Topics are raised, the controller of the business event monitoring framework will be notified immediately.
2. The subscription is fone at the Seacloud level, asking about interest for Topic C – Topic C that has been defined for containing business events:
    a. When asked, the governance hasn't the sufficient knowledge about who is the producer of this Topic.
    b. When events of Topic A and events of Topic B are raised, if the match (CEP rule) is done between some of them, the DiCEPE generates a business event of Topic C (it has the CEP rule to do so) that immediately reaches the monitoring component.

<ins>Note:</ins>
There is also another monitoring framework provided by the platform, named EasierBSM that complements the business monitoring being specified here, by monitoring lower level metrics such as execution time and latency of services more or less related to the underlying bus itself.

2.1.3. View

The View in the business event monitoring framework is the end-user GUI – the GUI, on which it will find details of events flows of its interest.
As Topic is a grouping of events, we could imagine a higher-level notion, that is to say a Role for example, that will correspond to a business activity. Indeed, the end-user of the monitoring framework will be provided a web-page with a list of Role, so as it could choose his. The Topics remain a too much technical notion. The end-user has to be abstracted from this technical level.

So, to a Role corresponds a list of Topics. For example, an air traffic controller won't be interested in Topics related to the Pilot of an aircraft, but with Topics issued from Air Traffic Control.

At runtime, when the end-user selects its Role on the web-page, the corresponding list of Topics is displayed. And so as to conform to its need, it will be able to add or remove Topics from a Role, create new Roles, and so on.

Then, it could choose the graphical representation of the information it wishes to monitor, in the sense it will be more adapted to its need. A Generate button is provided, and once clicked, the monitoring UI is generated and monitoring activity can start as shown by the following Figure 4 : Business event monitoring at runtime :

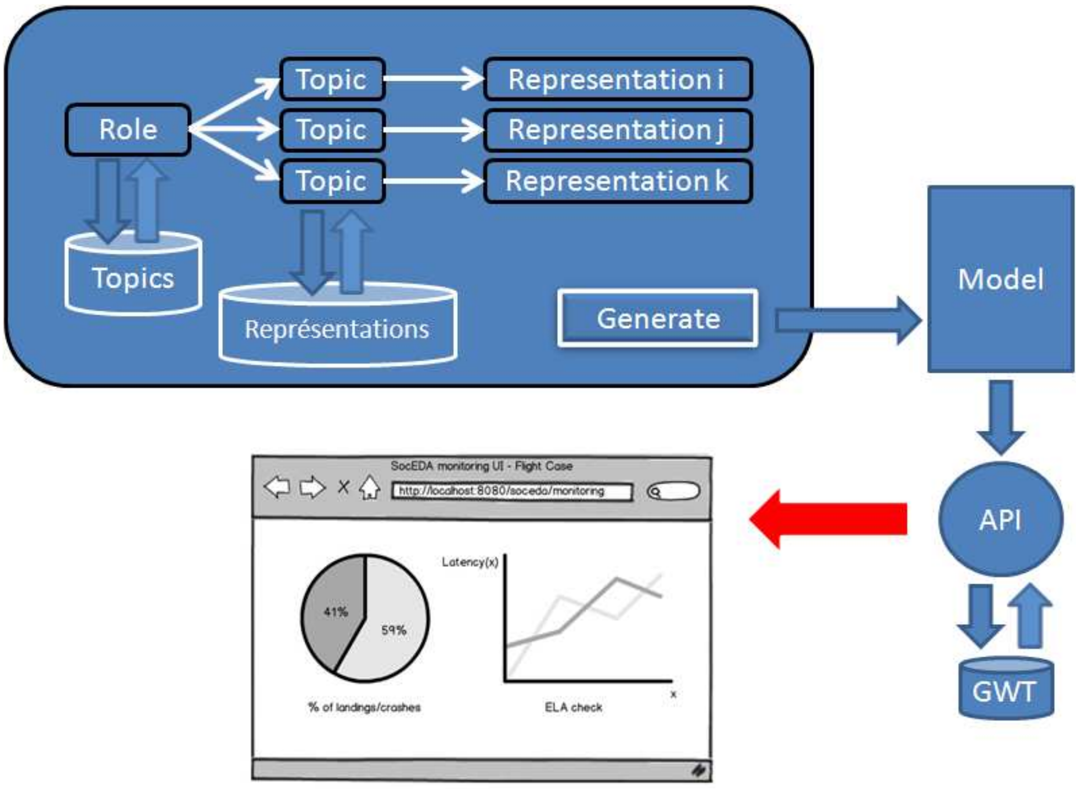


Figure 8: Generation UI process

Note:

1. "The" EventCloud implements events persistence. The business monitoring framework will rely on it for certain type of computation (time-frame related, averages, …). It will thus be invoked, either via the SeaCloud or through a dedicated API.
2. To avoid redundancy, only one controller monitoring component could be deployed, in certain circumstances, for a given Role. That is to say, only one controller will be deployed if for example two end-users depends on the same Role in the monitoring activity. These end users will have separate view components though.This will avoid doing several times the same subscriptions, and doing the same computations. But on the other hand, in case of a very large number of end-users (i.e. Views) using the same Role, replication of the monitoring controller could possibly be a solution to cope with scaling problems

2.2. Integration

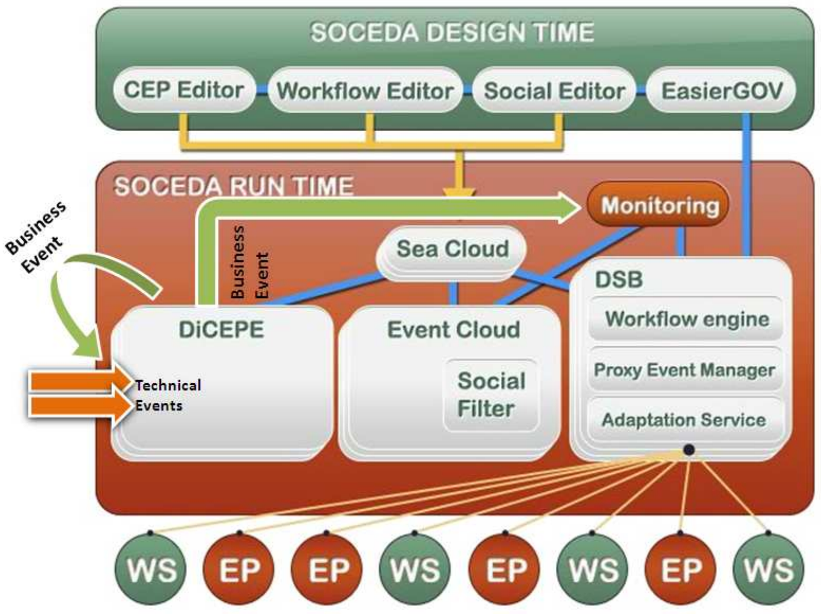


Figure 5 : SocEDA platform architecture

The business event monitoring framework is like an external framework, that once deployed, is able to monitor events according to the end-user needs specifically.
As discussed before, at design time are written CEP rules, to express business events – towards the business event monitoring framework specifically – issued from a correlation of simple/complex events. Those rules are consumed by the DiCEPE at runtime to generate those business events at destination of the business event monitoring framework

Note:

- All subscriptions and events routing are done through the SeaCloud.

3. Conclusion

The differentiating feature of the monitoring framework we intend to build in the scope of this work package is clearly is business-oriented nature. This particularity emerges both from a need clearly expressed by the project and use-case requirements, as well as a natural extension of more classical lower-level form of monitoring (such as the one of EasierBSM). Being a natural fit for this project, the business monitoring has nonetheless an impact on related components of the SocEDA runtime such as the SeaCloud and DiCEPE. The basic implications of this impact have been specified in this document and will be refined and detailed in upcoming D3.1.2 and D3.1.3. It is indeed only through optimal use of assets provided by the platform (e.g. computation power of the DiCEPE) that the business monitoring will achieve its goals and adhere to the distributed philosophy of the project.