SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	Specification of the Event-based monitoring framework v2
Document version:	1.0
Task code:	T3
Deliverable code:	D3.1.2
WP Leader (organisation):	LINAGORA
Deliverable Leader (organisation):	LINAGORA
Authors (organisations):	Christophe GATTI (THA)
Date of first version:	01/02/13

Change control

Changes	Author / Entity	Code of version
Creation of the document	Christophe GATTI	0.1
Add of #section runtime example including figures	Christophe GATTI	0.2
Check spelling	Christophe GATTI	0.3
Taking into account of peer review remarks	Christophe GATTI	1.0

Table of Contents

Table of Figures

1. Introduction

This document is part of work package 3 – Monitoring and Governance. The goal of this document is to refine the business monitoring architecture specification. In a first part, we will briefly remind the context of monitoring business events, and then we will present the detailed architecture of the framework, to conclude finally on the communication with the platform with an example.

2. Business Monitoring Framework

2.1. Architecture

The business monitoring framework is server-side and GWT-based. It means every calculus is done at the server level, then, on information update, the server automatically notifies the client-side (browser). This mechanism paves the way for the Future Internet, in the sense the client will no longer pull the server for information updates.
The framework is designed respecting the model-view-controller pattern.

2.1.1. Overview

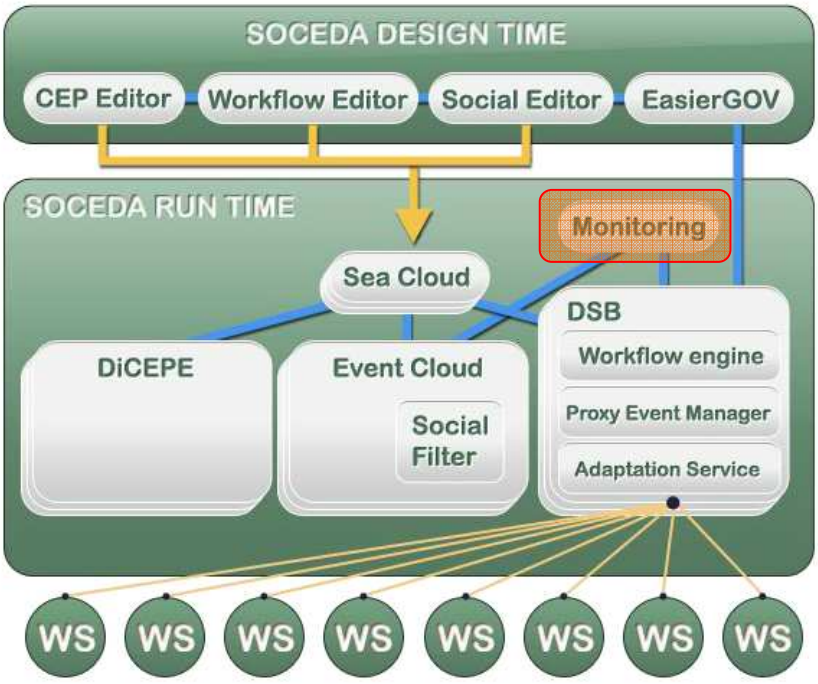

Figure 1 : SocEDA overview

The monitoring framework is part of SocEDA runtime architecture. Once deployed, the framework subscribes to interesting business events, thanks to the SeaCloud, and communicates with the DiCEPE for time-based calculus (e.g. calculus of frequency of one type of events).

2.1.2. Configuration

The framework is configured via an ontology that describes the business domain it applies for. This ontology is the entry-point of the framework. When the framework is launched, the user is asked to select an application domain, and then on submit action, the corresponding ontology is loaded.

Note: For now, all concepts present in the ontology are loaded and monitored. In a future version, there will be a checklist the user can interact with, in order to select only the concept it is

interested into for monitoring. Moreover in this version, the ontologies are embedded into the application.

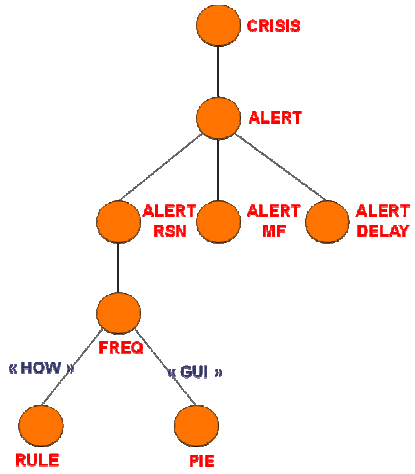

Figure 2 : Ontology "Crisis" example

Using the previous ontology, the frequency of all alerts will be monitored and displayed to the end-user as a pie chart.

Note: Ontology should respect the following notation:

- OWL format,
- Each “measurable” concept, like “frequency” in our case, should be the parent of two leaf concepts, and linked with semantic link with properties “**how**” and “**gui**”. Those properties express, first, how the “measureable” concept will be calculated (e.g. a mathematic formula), then, the display type chosen for the graphical representation of the result of this calculus.

2.1.3. Use-case

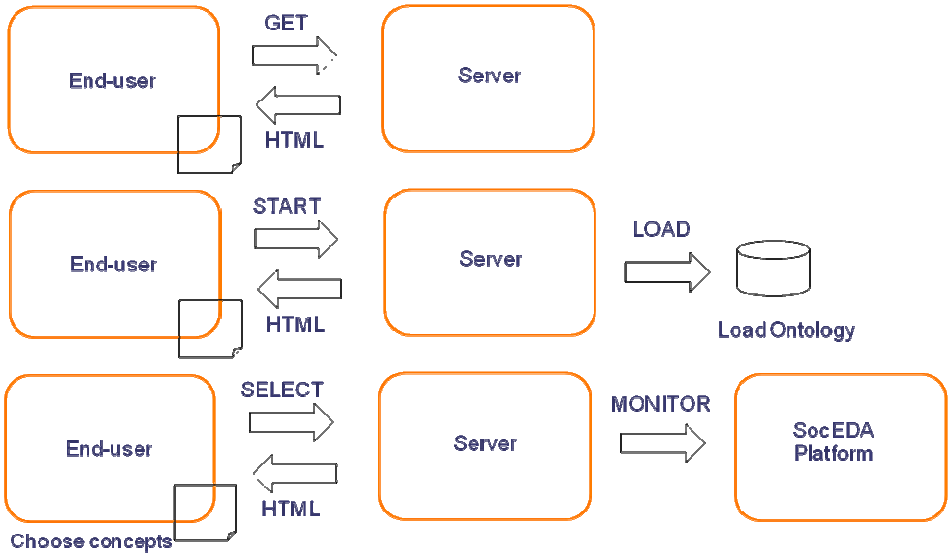

Figure 3 : Runtime example

This example summarizes what has been explained previously. The following steps are executed:

- The user launches the application,
- The user selects an application domain,
- The user chooses the concepts it wants to monitor,
- The user gets provided with a dynamic graphical interface of monitoring activity.

2.1.4. Project

To be used, the project has to be compiled, either by compiling the whole platform or by compiling it only, or to be used directly with the binary provided as web archive, deployed in a java-support application server (tomcat, jetty). Both compilations require "**Maven**" tool in version 2.x.x. Just place yourself inside the project directory and type "**mvn install**". This will update / download all the dependencies needed to compile the project.

Once compiled, the project is ready to be loaded by eclipse, using right-click on the project explorer view then selecting "**import / maven / existing maven projects**".

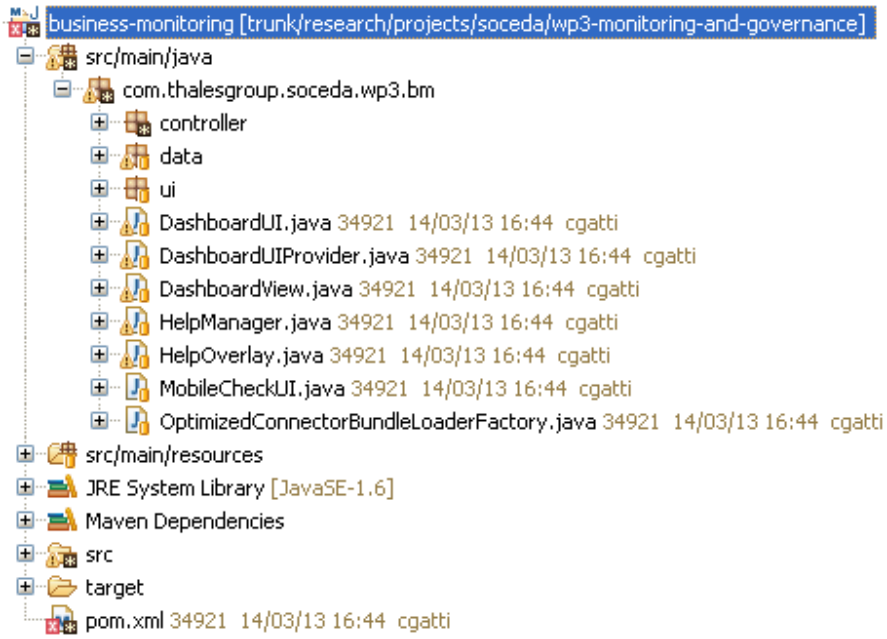

Figure 4 : Eclipse project

The project is composed of the following packages:

- **controller**, which is the main component for ontology parsing (using Jena) and communication with:
  - The DiCEPE for pushing time-based calculus, and executing calculus itself otherwise,
  - The SeaCloud for managing subscriptions to events (using UserManagementClientSOAP generic client).
- **data**, which contains the model representation of treated events,
- **ui**, which contains the different graphical representations managed yet,
- The classes composing the main view.

2.2. Runtime example

2.2.1. Deployment

Once the web archive is compiled, using previous indications use the administration console of tomcat to deploy the web-app.

2.2.1. Execution

By running the application, the following screen should appear:

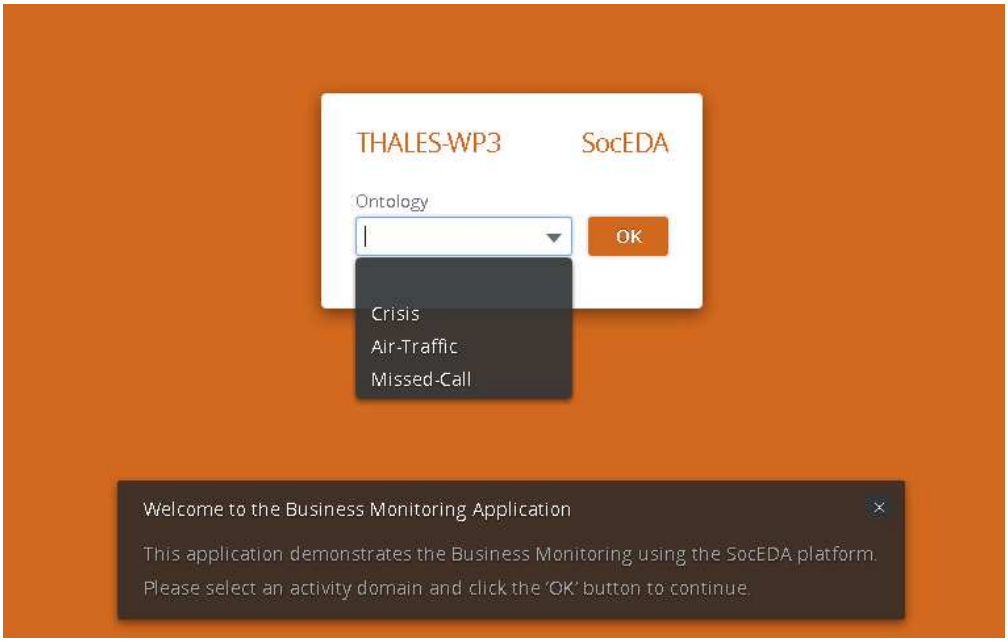

Figure 5 : Runtime welcome page

Then, select the “Crisis” ontology, and press “OK”. The main view should appear as follow:

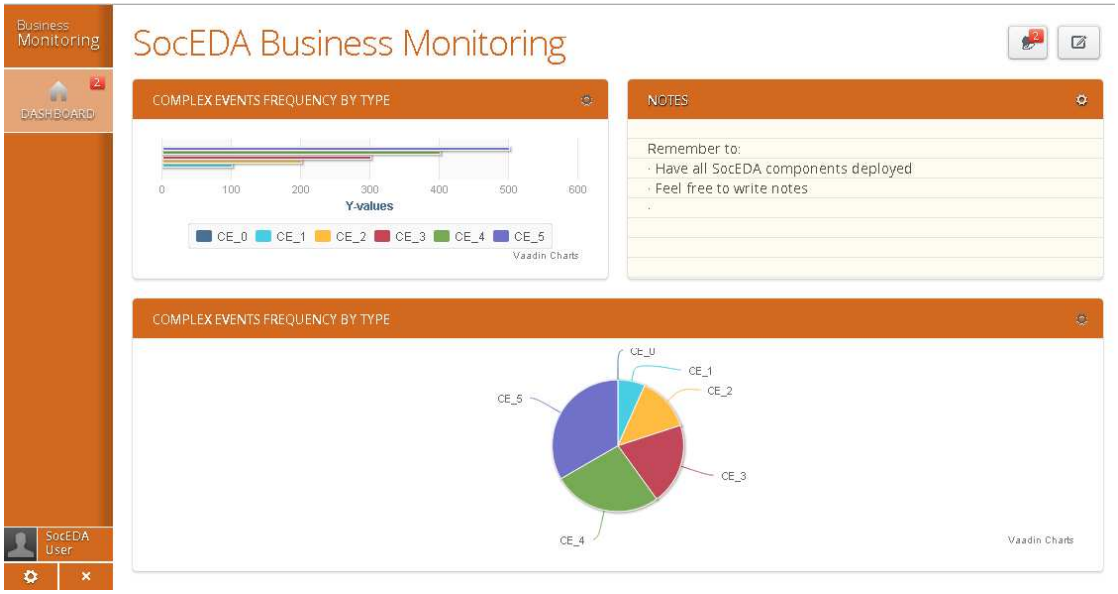

Figure 6 : Runtime main page

In this example, the main page summarizes the frequency of alerts raised by the “Crisis” use-case and sorted by type. The charts automatically refreshes when new alerts come through the platform.

3. Conclusion

This document detailed more precisely the effort spent to reach the goal of obtaining a generic business monitoring framework, with an ergonomic and cute graphical user interface. The framework will be part of the whole SocEDA packaging, available on the SocEDA website, thus abstracting the end-user with deployment and components-communications problematic.

Although the framework is still in beta phase, it will be enriched for M36 with an efficient logger for tracing to the end-user all the connections made with the platform (including the subscriptions, errors management) and, of course, with a new page enabling the end-user to select the concepts it will be interesting to monitor.