SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	Final specification of the event-based monitoring framework and implementation
Document version:	1.0
Task code:	T3
Deliverable code:	D3.1.3
WP Leader (organisation):	LINAGORA
Deliverable Leader (organisation):	LINAGORA
Authors (organisations):	Christophe GATTI (THA)

Date of first version: 24/10/13

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

1. Introduction 6
2. Business Monitoring Framework 7
2.1. Architecture 7
2.1.1. Overview 7
2.1.2. Configuration 7
2.1.3. Use-case 9
2.1.4. Project 9
2.2. Runtime 10
2.2.1. Detailed conception 10
2.2.2. Deployment 11
2.2.1. Execution 11
3. Conclusion 13

Table of Figures

Figure 1 : SocEDA overview 7
Figure 2 : Ontology "Crisis" example 8
Figure 3 : Runtime example 9
Figure 4 : Eclipse project 10
Figure 5 : Runtime welcome page 11
Figure 6 : Runtime main page 12

1. Introduction

The main purpose of this document is to report on the activities carried out in the Business Monitoring Framework development. The outcome of task 3.1.3 is the final implementation of the Business Monitoring Framework. Accordingly the section 2.2 of this document is dedicated to describe the implementation of the prototype.

The other main goal of task 3.1.3 is the verification and validation of the prototype. Actually, this activity is carried out with two objectives in mind: one is testing the prototype implementation of the Business Monitoring Framework; the other (possibly more important) is to test the effectiveness of the SocEDA technology used in the implementation of the prototype.

Finally, Section 2.2.1 illustrates the results of the use-case-based runtime execution.

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

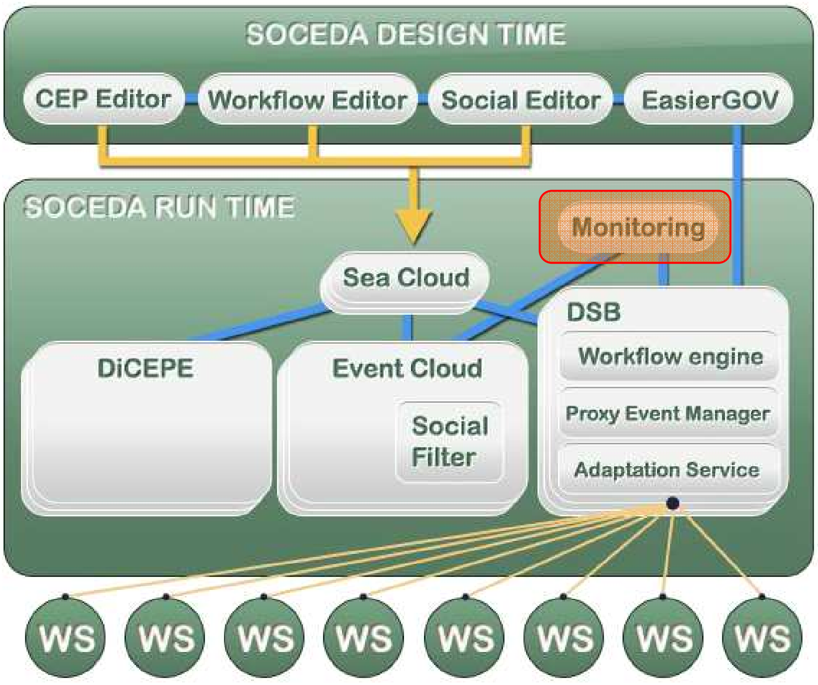


Figure 1 : SocEDA overview

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.

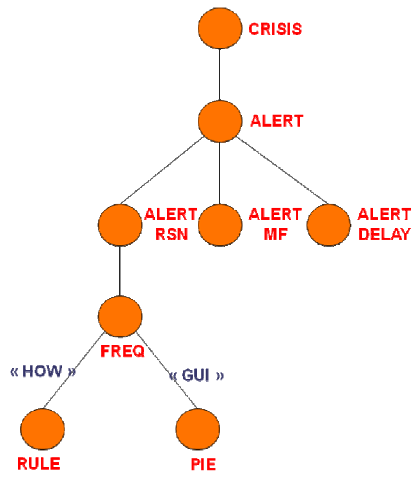


Figure 2 : Ontology "Crisis" example

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

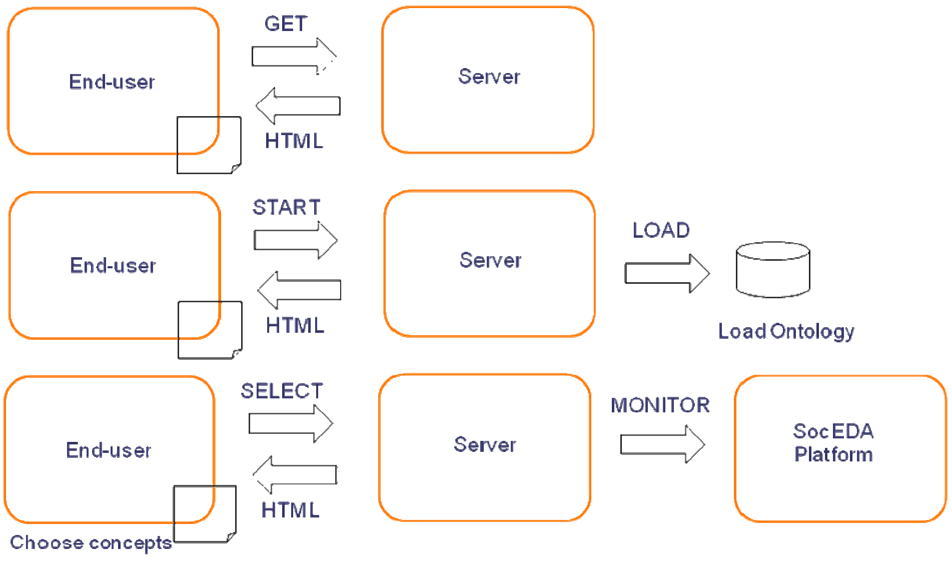


Figure 3 : Runtime example

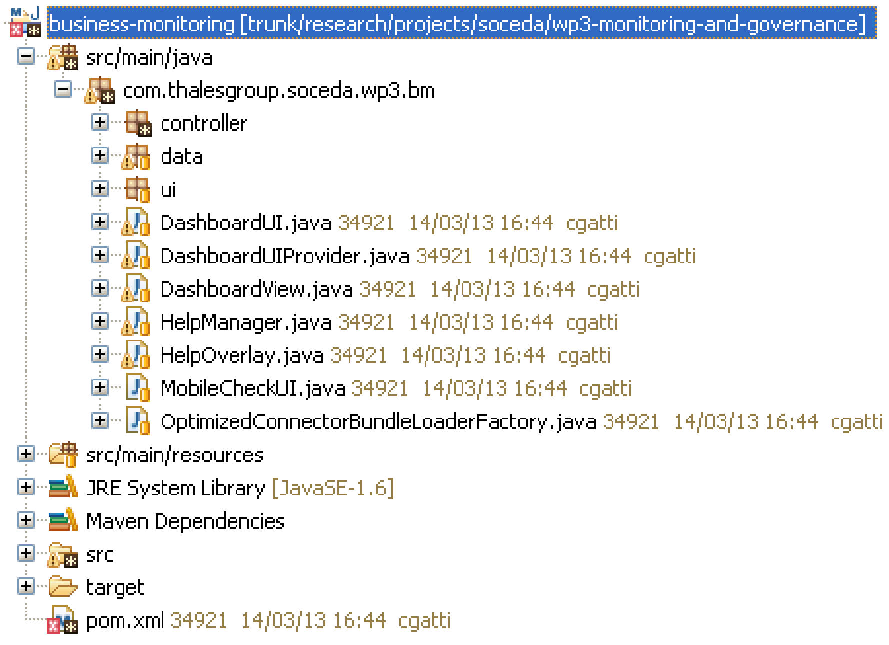
This example summarizes what has been explained previously. The following steps are executed:

- 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**".



```
business-monitoring [trunk/research/projects/soceda/wp3-monitoring-and-governance]
  src/main/java
    com.thalesgroup.soceda.wp3.bm
      controller
      data
      ui
      DashboardUI.java 34921  14/03/13 16:44  cgatti
      DashboardUIProvider.java 34921  14/03/13 16:44  cgatti
      DashboardView.java 34921  14/03/13 16:44  cgatti
      HelpManager.java 34921  14/03/13 16:44  cgatti
      HelpOverlay.java 34921  14/03/13 16:44  cgatti
      MobileCheckUI.java 34921  14/03/13 16:44  cgatti
      OptimizedConnectorBundleLoaderFactory.java 34921  14/03/13 16:44  cgatti
  src/main/resources
  JRE System Library [JavaSE-1.6]
  Maven Dependencies
  src
  target
  pom.xml 34921  14/03/13 16:44  cgatti
```

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

2.2.1. Detailed conception

In order to communicate with the platform, the business monitoring framework uses the following dependency:

```
<artifactId>management-client</artifactId>
<groupId>com.ebmwebsourcing.esstar.esmanagement</groupId>
<version>1.0-SNAPSHOT</version>
```

This dependency is responsible for events subscriptions and notifications.

When the framework is launched, an instance of this client is created. Then, the user selects an ontology that is parsed, and selects the concepts he wants to monitor.

These concepts are linked to a graphical representation and an effective “complex event”.

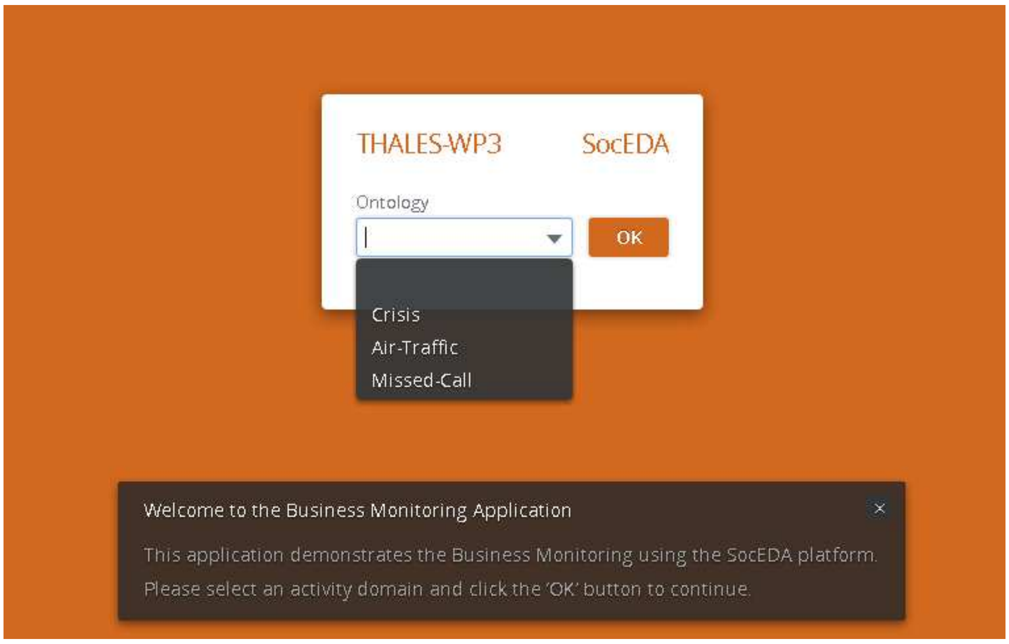
For each of selected concept, the client subscribe to it.

2.2.2. 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:

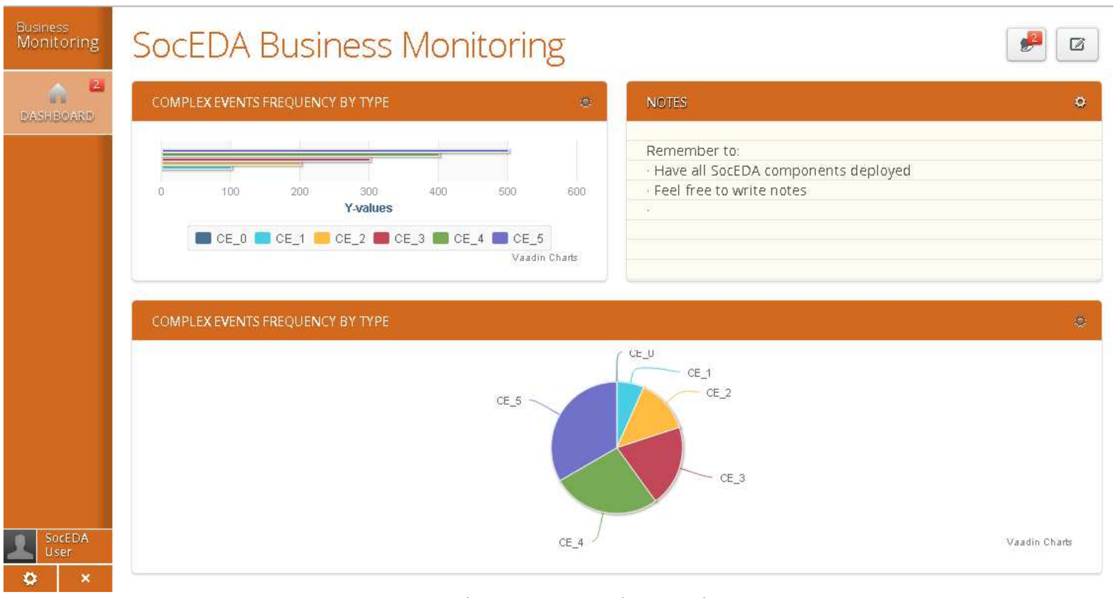


Welcome to the Business Monitoring Application

This application demonstrates the Business Monitoring using the SocEDA platform.
Please select an activity domain and click the 'OK' button to continue.

Figure 5 : Runtime welcome page

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



Remember to:
· Have all SocEDA components deployed
· Feel free to write notes
·

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.
In case error occurs, the user is notified with the ring bell in the top right corner of the UI. Errors are detailed when the user clicks on it.
There is also a free-text editing part, in which the user can enter comments about his live-monitoring, and all modifications will be saved for a future opening.

3. Conclusion

The developers in their work have taken into account different factors of technology creation and distribution, and also contextual issues, for example, the market and society. Important issues regarding the management, quality of technology, acceptance and domain development are adequately performed or deliberated.

The better the technology development process and the more socio-technical factors have been deliberated, the better are the prospects for any technology in terms of acceptance and sustainability.

In order to convert the current solution into an internal/external useable product for the future exploitation, few classes of actions must be addressed like Requirements update, Implementation issues, Service availability and Technology.

Main concerning privacy and security are missing, and are the most important. Specify privacy requirements is necessary, because there are laws that regulate this type of activity, and it is quite complex because different countries have different laws, so it is necessary to take into account the laws of "e-privacy" of all the countries who may be involved.

Implementation issues to make the product suitable for the actual everyday usage is another limitation to date for the exploitation of the Business Monitoring Framework. Therefore a second step in the process of product conversion concerns the detailed specification of user experience-related about the domain, and their implementation, consisting mainly in creating/updating an ontology.

Another limitation that should be removed is service availability. The framework is actually locally deployed, but should be available through the Internet.

It is well known that information technology is moving fast. It is therefore the case of reviewing the technology used in view of the development of the Business Monitoring Framework. If in the future there will be any new/emerging technology that could be used in addition or in replacement of the those currently used or if part of technology currently used in the project activity does not match or does not satisfy the new requirements, must be evaluated, in order to aim to an high exploitation and ensure the product success.

From the other side the current application is already a good proof of concept that helps in understanding how the solution could work even though is currently still a prototype.

The current solution already shows anyway interesting features that could be leveraged.

Finally, thanks to SocEDA, we were able to enrich our knowledge about Complex Event Processing, Scalability, Distributed Systems, and other innovative technologies.