SocEDA

Cloud based platform for large scale social aware EDA

ANR-10-SEGI-013

Document name:	Adaptation Service (AdS) implementation v.1
Document version:	0.1
Task code:	
Deliverable code:	
WP Leader (organisation):	
Deliverable Leader (organisation):	
Authors (organisations):	
Date of first version:	14/02/13

Change control

Changes	Author / Entity	Code of version
Creation of the document	Anne-Marie BARTHE-DELANOË Sebastien TRUPTIL / Armines	V1 (draft)

Table of Contents

1. Introduction

1.1. Purpose

The adaptation service shows the value added of the SocEDA platform in the context of the response to a need of change of the workflows.

With its complex-event processing architecture, French project SocEDA can contribute significantly. The management of the adaptation need would indeed be facilitated by the increased situational awareness provided by SocEDA's platform for event processing. In addition to that, SocEDA would ensure a timely and adequate diffusion of information to relevant actors.

This document aims at describing this adaptation service implementation in its first version (i.e. the detection part of the tool), following the concepts described into deliverable D.2.4.2.

1.2. List of Acronyms

Acronym	Definition
A	Army
P	Police
O	Office of infrastructures
MF	Meteo France (weather forecast)
RSN	Radiation Survey Network (measures radioactivity)
EDF	Electricité De France (Operates nuclear plants in France)
RNA	Representative of the National Authority (prefect)
F	Firemen
MEMS	Mobile Emergency Medical Service
WSDL	Web Semantic Description Language
ESB	Enterprise Service Bus
JBI	Java Business Integration
SA	Service Assembly
SU	Service Unit

BC	Binding Component
BPEL	Business Process Execution Language
XML	eXtended Markup Language

2. Adaptation service in SocEDA

2.1. Objectives

First of all, we have to remind that the Adaptation Service in the SocEDA project is a tool designed to situation, i.e. processes, to the evolution of the context. As explain in D2.4.2, to define the evolution of the processes, two steps are needed:

1. Detection: this step consists in detecting an evolution of the situation that could not be solved by the ongoing processes or making the current process not relevant to the current situation,
2. Adaptation: when an evolution is detected, the adaptation step is executed to modify the ongoing processes in order to make them relevant to the current situation.

In a few words, the Adaptation Service has to allow on one hand to detect if the ongoing processes meet the requirements of the current situation, on the other hand to adapt the ongoing processes if necessary.

The Adaptation Service in the SocEDA project is a tool designed to provide agility (as seen as the combination of detection and adaptation). It allows on one hand to detect if the on going processes meet the requirements of the current situation, on the other hand to adapt the on going processes if necessary.

2.2. Principles of the approach to detect evolutions

The approach to detect evolutions is based on several steps:

1. Define the situation model and it to obtain reference situation model and current situation model
2. Make a comparison between reference situation model and current situation model
3. Calculate the difference between these two model to check the adequacy of the running processes with the current model situation (representing the situation in the real world).

Implementation of these steps is developed in the next section.

3. Implementation of the detection part

3.1. Principles of detection

As fully explained in deliverable D2.4.2, the detection part of the Adaptation Service is based on the comparision of two models of the situation:

- The model representing the expected situation (i.e. reference situation model), updated with events coming from the monitoring of the processes,
- The model representing the real situation on the field (i.e. current situation model), updated with events coming from the field (sensors, reports, etc.).

These models are made using modelers (Signavio distribution, adapted to our case), allowing the users to caracterize the situation models. They are updated with incoming events, emitted by the CEP.

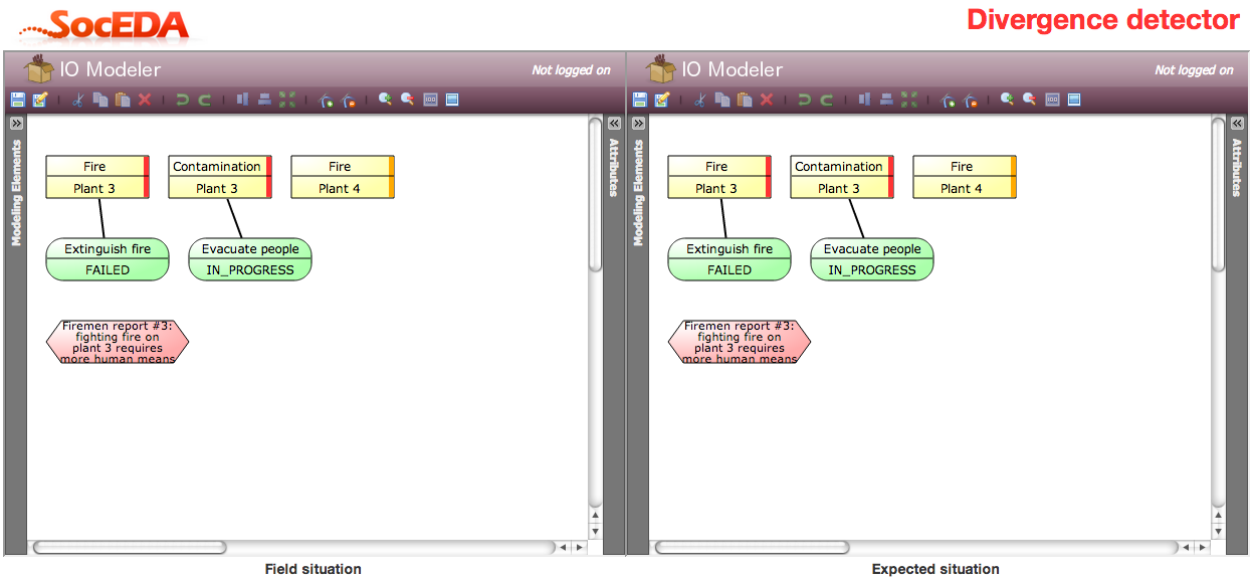


Figure 1. Graphic description of field situation model and expected situation model.

These models are then compared and the distance ∂ between them is calculated. If the calculated ∂ is over the given ∂ threshold, the Adaptation Service informs the users to the need to adapt the processes and helps them to take a decision concerning the level of adaptation (partial or total redefinition, or re-execution of an activity).

3.2. Model format

Both expected and field situation models are XML files. They follow the XSD file (cf. Appendix 1).

This XSD file is in its first version for the moment (it will cover a higher range of concepts in the future versions). For the moment, it concerns the concepts of risk, service, consequence and the links between services and risks, as seen in Figure 2 and Figure 3.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<situationModel>
    <services>
        <service id="S_000456" description="Extinguish fire" state="IN_PROGRESS"/>
        <service id="S_000463" description="Set evacuation perimeter" state="IN_PROGRESS"/>
        <service id="S_000467" description="Evacuate people" state="WAITING"/>
    </services>
    <consequences>
        <consequence id="C_000124" description="Firemen report #3: fighting fire on plant 3 requires more human means"/>
    </consequences>
    <risks>
        <risk id="R_000123" description="Fire" gravity="HIGH" localisation="Plant 3"/>
        <risk id="R_000126" description="Contamination" gravity="HIGH" localisation="Plant 3"/>
    </risks>
    <linkService_Risks>
        <linkService_Risk id="LSR_123" serviceId="S_000456" riskId="R_000123"/>
        <linkService_Risk id="LSR_124" serviceId="S_000456" riskId="R_000126"/>
        <linkService_Risk id="LSR_128" serviceId="S_000467" riskId="R_000126"/>
    </linkService_Risks>
</situationModel>
```

Figure 2. Example of a reference situation model file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<situationModel>
    <services>
        <service id="S_000456" description="Extinguish fire" state="FAILED"/>
        <service id="S_000467" description="Evacuate people" state="IN_PROGRESS"/>
    </services>
    <consequences>
        <consequence id="C_000124" description="Firemen report #3: fighting fire on plant 3 requires more human means"/>
    </consequences>
    <risks>
        <risk id="R_000123" description="Fire" gravity="HIGH" localisation="Plant 3"/>
        <risk id="R_000126" description="Contamination" gravity="HIGH" localisation="Plant 3"/>
        <risk id="R_000186" description="Fire" gravity="MEDIUM" localisation="Plant 4"/>
    </risks>
    <linkService_Risks>
        <linkService_Risk id="LSR_123" serviceId="S_000456" riskId="R_000123"/>
        <linkService_Risk id="LSR_128" serviceId="S_000467" riskId="R_000126"/>
    </linkService_Risks>
</situationModel>
```

Figure 3. Example of a current situation model file

3.3. Models comparision

These two XML files are compared by the XMLCompare library we have developed. The XMLCompare library uses the facilities of the XMLUnit tool (1.3 version) [XMLUnit, 2009].

XMLUnit is initially used to checked the conformity of an XML document, after its unmarshalling: the unmarshalled document is compared to the original document. XMLUnit provides us a mean to compare two XML documents together and to check if the documents are similar (similarity is when the nodes are the same in both documents but not necessary in the same order).

But XMLUnit is limited due to its unability to understand the semantic meaning of the id attribute (typed with xsd:ID): this is a true limitation as XMLUnit interprets two nodes (with same place in the hierarchy, same tag name, same attributes, same values) as updated nodes even if the @id value is not the same. Our library answers to this limitation with an analysis of the XPath expressions of the nodes and is able to identify updated nodes from added/deleted nodes.

In a global view, our XMLCompare library is able to have a deep look into any XML file, and detect the similarity or not between two XML files (assuming that these files follow the same XSD to avoid noise into the detection report).

If the documents are not similar, XMLCompare is able to describe each detected difference as an addition/deletion/update of a node/an attribute/a value and gives the matching XPath

expression (which is necesary to have the detail of the detected difference later).

3.4. ∂ calculus

It was arbitrary chosen that any detected difference has a weigth of 1 (whatever the nature of the difference is : addition of a risk, update of a service, etc.). Calculated ∂ is the sum of the number of detected differences times their respective weight.

So, if there are 16 differences between the two XML files, the calculated ∂ is equal to 16.

		Operations			
		Add	Delete	Update	Total per concept
Concepts	Partner	1	0	1	2
	Ressource	0	3	1	4
	Activity	0	3	1	4
	Risk	3	1	0	4
	Consequence	2	0	0	2
	….				
		Calculated ∂ =			**16**

Figure 4. Example of a ∂ calculus.

3.5. ∂ threshold

Among the difference detection, a threshold is defined by the partners. If the calculated ∂ is over this threshold, the Adaptation Service will present the weight of each concept inside the calculated ∂.

3.6. Concept weight

In our example, the 10 detected differences concern: 5 instances of risk, 3 instances of service and 2 instances of consequence. Risk has a weigth of 50%, Service has a weigth of 30% and Consequence a weight of 20%.

		Total per concept	Weigth of concept
Concepts	Partner	2	12,5%
	Ressource	4	25%
	Activity	4	25%

	Risk	4	25%
	Consequence	4	12,5%
	….		
	∂	**16**	**100%**

Figure 5. Example of weight calculus for each concept involved into the ∂.

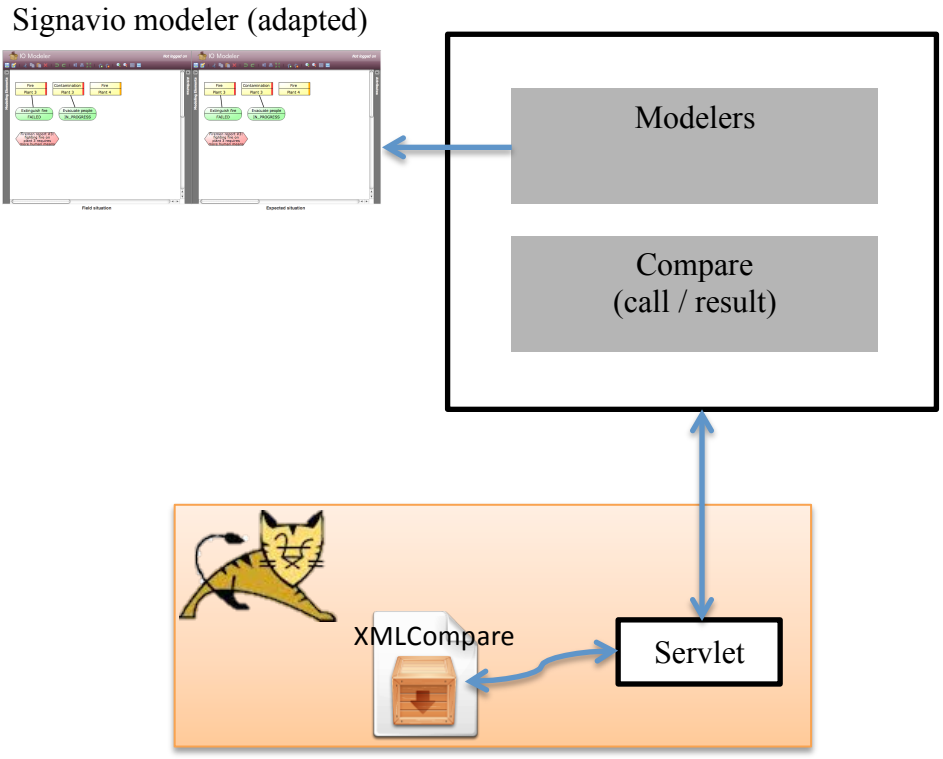
And then, it will advice for adaptation of the processes on the basis of the difference details gathered by the XMLCompare tool.

3.7. Our proposal for adaptation part implementation

First of all, we need to define a set of business rules describing how the Adaptation Service can advice users on the loop to choose (partial or total redefinition, etc.), based on the details of each detected difference (thanks to the XPath expression which points exactly the concernent element or attribute or value) and on the weigth of each concept participating into the calculated ∂.

Then, when theses rules are run, the Adaptation Service is able to indicate the best solution(s) for adaptation to the users. But the final choice is let to the users.

4. Technical overview of the Adaptation Service



5. Conclusion

This document gives an overview of the implementation of the first version of the Adaptation Service, i.e. the detection part of this tool.

Further work consists in, first, enhance the XSD file to cover more concepts, secondly, to implement the adaptation part of the tool.

6. References

XMLUnit (2009). XMLUnit 1.3, website : http://xmlunit.sourceforge.net/ (online, January 2013)

7. Appendix 1: XSD file

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">

    <!-- Attributes -->
    <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="serviceId" type="xsd:IDREF"/>
    <xsd:attribute name="riskId" type="xsd:IDREF"/>
    <xsd:attribute name="localisation" type="xsd:string"/>
    <xsd:attribute name="value" type="xsd:string"/>
    <xsd:attribute name="description" type="xsd:string"/>
    <xsd:attribute name="gravity">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="HIGH"/>
          <xsd:enumeration value="MEDIUM"/>
          <xsd:enumeration value="LOW"></xsd:enumeration>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="state">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:enumeration value="IN_PROGRESS"/>
          <xsd:enumeration value="FAILED"/>
          <xsd:enumeration value="WAITING"></xsd:enumeration>
        </xsd:restriction>
```

```
    </xsd:simpleType>
</xsd:attribute>

<!-- Elements - low level -->
<xsd:element name="risk">
    <xsd:complexType>
        <xsd:attribute ref="id" use="required"/>
        <xsd:attribute ref="description" use="required"/>
        <xsd:attribute ref="gravity" use="required"/>
        <xsd:attribute ref="localisation" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="service">
    <xsd:complexType>
        <xsd:attribute ref="id" use="required"/>
        <xsd:attribute ref="description" use="required"/>
        <xsd:attribute ref="state" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="consequence">
    <xsd:complexType>
        <xsd:attribute ref="id" use="required"/>
        <xsd:attribute ref="description" use="required"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="linkService_Risk">
    <xsd:complexType>
```

```
        <xsd:attribute ref="id" use="required"/>
        <xsd:attribute ref="serviceId" use="required"/>
        <xsd:attribute ref="riskId" use="required"/>
    </xsd:complexType>
</xsd:element>

<!-- Elements - high level -->
<xsd:element name="risks">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="risk" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="services">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="service" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<xsd:element name="consequences">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="consequence" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

```
    <xsd:element name="linkService_Risks">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="linkService_Risk" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <!-- Root element -->
    <xsd:element name="situationModel">
      <xsd:complexType>
        <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="consequences" minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="risks" minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="services" minOccurs="0" maxOccurs="1"/>
          <xsd:element ref="linkService_Risks" minOccurs="0" maxOccurs="1"/>
        </xsd:choice>
      </xsd:complexType>
    </xsd:element>
</xsd:schema>
```