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

D9.3.1 : C2C e-Commerce Prototype v1

Activity 3:	Use Case Activities
Work Package:	WP9 C2C e-Commerce
Due Date:	M24
Submission Date:	28/02/2010
Start Date of Project:	01/03/2008
Duration of Project:	36 Months
Organisation Responsible of Deliverable:	TXT
Revision:	3.0
Author(s):	Matteo Villa (TXT), Giovanni Di Matteo (TXT) Sven Abels (TIE), Shiva Puram (TIE) Bernhard Schreder (Hanival), Michael Zaremba (Seekda), Emilia Cimpian (Seekda)
Reviewers:	Maria Maleshkova (OU), Dario Cerizza (Cefriel)

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**
PP	Restricted to other programme participants (including the Commission)	
RE	Restricted to a group specified by the consortium (including the Commission)	
CO	Confidential, only for members of the consortium (including the Commission)	

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	01/2010	ToC; first sections	S. Abels (TIE)
0.2	01/2010	ToC restructuring, TXT initial contributions to 5.1.2	M. Villa (TXT)
0.3	02/2010	First integrated version: chapters 2, 4 done by TXT TIE contributions integrated (sections 3, 7) HANIVAL contributions integrated (sections 6.1, 5.2 still to be refined) SEEKDA contribution to 6.2 and initial contribution to 5.1.1	M. Villa (TXT) S. Abels (TIE) B. Schreder (HANIVAL) E. Cimpian (Seekda)
1.0	02/2010	First release for internal review TIE completed section 4.2 HANIVAL completed sections 5.2 and 4.2 SEEKDA completed section 5.1.1	M. Villa (TXT) S. Abels (TIE) B. Schreder (HANIVAL) E. Cimpian (Seekda)
1.1	02/2010	Final formatting	M. Villa (TXT)
	02/2010	Internal Revision	Maria Maleshkova (OU), Dario Cerizza (Cefriel)
2.0	02/2010	Revised document addressing internal reviewers' comments	M. Villa (TXT) E. Cimpian (Seekda)
3.0	02/2010	Revised document addressing internal reviewers' comments	S. Abels (TIE)

Table of Contents

List of Figures

List of Tables

Glossary of Acronyms

Acronym	Definition
B2B	Business-to-Business
B2C	Business-to-Consumer
BPM	Business Process Management
C2C	Consumer-to-Consumer
CCBS	Customer Care and Billing System
CRM	Customer Relationship Management
D	Deliverable
DC	Digital Channel
EC	European Commission
QoS	Quality of Service
SOA	Service Oriented Architecture
SWS	Semantic Web Services
WP	Work Package

1. Executive summary

The work package 9 scenario shows how the SOA4All results can be applied in the e-Commerce domain. The scenario goes clearly beyond a classical use case in a way that it does not only use and apply the results provided by the project. Instead of this, it also adds own ideas and developments to SOA4All, allowing the use case to show the innovation that SOA4All brings to e-Commerce in a future looking and highly flexible web 2.0 environment. The purpose of this scenario is to demonstrate the SOA4All vision by telling a real-world story around the complete set of SOA4All components in a highly practical way so showing the usefulness of the project results.

Based on this scenario, a first e-Commerce C2C Prototype has been released at month 24. The prototype is build on top of an e-Commerce Framework, and exploits technology and components developed within SOA4All technical workpackages. The innovative functionalities of this prototype allow product resellers to combine services of multiple vendors into one unique set of products, which is exposed into various web 2.0 platforms including Facebook and Twitter, realizing a “One Stop Cloud Shop” vision.

The Prototype is composed of a set of components exposing a common set of services supporting the various e-Commerce operations, such as unifying product catalogues, exporting product data for external platforms.

The Prototype also includes a set of process templates and Goals that can be easily re-used to facilitate the creation of new specific e-Commerce processes. Thanks to such functionalities, the Prototype can fully support three main category of end-users: product sellers, web-shop owners (product re-seller) and shop customers.

A specific set of testing activities for each of them is planned and will drive the evaluation of this prototype.

2. Introduction

2.1 Purpose and Scope of the deliverable

Deliverable D9.3.1 represents the first implementation of the C2C e-Commerce Prototype. The purpose of this document is to provide a technical description of such, as released at M24. The prototype is built from the framework infrastructure defined into Task T9.2, following the scenario described in D9.2.1, extended in order to include an additional role: the *advertiser*. This is a very interesting and promising role for the e-Commerce domain, and it enhances the chances of exploitation by WP9 partners. The scenario is quickly summarised at the beginning of this deliverable for clarity.

The Prototype aims at demonstrating the applicability of SOA4All technology in the e-Commerce domain. It highlights the innovative aspects showing how product resellers can combine services of multiple vendors into one unique set of products, which is exposed into various web 2.0 platforms including Facebook and Twitter.

This deliverable provides the technical details of the architecture of the prototype, and how it is exploiting the components developed within the technical workpackages of SOA4All. The document also provides a detailed description on the main functionalities and APIs/Services offered by the prototype. Installation instructions are also included.

For each functionality, the document also shows the progress towards a full implementation and evaluation.

An improved version of the prototype is expected at month 33 (D9.4.1), which will be based also on the feedback gathered while testing and evaluating the first prototype.

A final evaluation report is expected at month 36 (D9.4.2)

2.2 Structure of the document

The document is organised as follows:

- **Section 3** summarises the **WP9 scenario**, which was previously described intD9.2.1, for the sake of clarity.
- **Section 4** describes the **architecture** of the prototype demonstrator, highlighting the involvement of SOA4All components and the infrastructure specifically created within WP9.
- **Section 5** provides a more **technical description** of such components, as currently implemented in the prototype.
- **Section 6** describes how the **prototype evaluation** is planned, and the progress in the development of the full scenario scheduled by the end of the project.
- **Section 7** finally provides prototype **installation** instructions.
- Finally, **Section 8** provides the conclusions of this deliverable.

3. C2C eCommerce Scenario

This section gives a short overview about the WP9 scenario. The purpose is to allow the reader to get a quick reminder of the main goals and actors of the scenario. For a more detailed description, please refer to D9.2.1.

3.1 Roles

The scenario involves several people as described in D9.2.1. In the new M24 prototype, a new role has been added: Silvio, who wants to use the SOA4All services to increase the number of visitors of his website.

Name	Role	Summary
Arian	Buyer	Visits Nadas Facebook page, sees some personalized products and decides to buy one of them.
Nada	Reseller	Wants to generate income on various popular web 2.0 sites. Uses SOA4All to connect services of various partners in order to expose product information to Facebook, Twitter, her own Webshop, etc.
Theodore, Esteban, Claus	Sellers	Want to increase their sales by offering web services, allowing resellers to retrieve a product list and to order a specific product.
Silvio	Advertiser	Wants to increase accesses to his web portal using the SOA4All WP9 Social Advertising Services.

3.2 Theodore, Esteban and Claus

Theodore, Esteban and Claus are from different companies, which are all producing products from the textile industry, reaching from footwear to T-Shirts for all seasons and covering both, male and female clothes. They are responsible for the e-Commerce part of their companies.

One day they read about SOA4All and they understand that SOA4All makes it simple to provide and consume services. They then decide to make their offers available to everyone via SOA4All.

They create a small service that allows people to:

- Get a list of their products and product descriptions,
- Request the price and availability of a product,
- Place a product order.

They use the SOA4All studio to add their service to SOA4All. Two of them have used web services before and already have suitable product services they want to expose to potential resellers. The third one decides to create a RESTful service.

They use the SOA4All tools to annotate their services and to describe them graphically with some semantic information based on common e-Commerce ontologies, like eCl@ss[1] for product categorization and the GoodRelations[2] for details on their product definitions and order services. They can do all of this using the graphical components of SOA4All – without any knowledge of the technologies behind this – just by using drag & drop to annotate their service elements. Afterwards they click the save button to add the service to SOA4All.

3.3 Nada

Nada wants to sell some goods to generate some side income. She has registered a business allowing her to buy and sell products. Nada is skilled in IT: she uses a lot of web 2.0 platforms including Facebook, Twitter and even an alpha version of Google Wave and she even owns a small webshop where she adds textile products manually from time to time. However, via her webshop she is only making a small number of sales and the product descriptions are usually outdated. She spends most of her time that she would like to instead invest into her webshop, updating prices and availabilities or to remove or add products. Also she has no way of automatically aligning her offered products with the Web 2.0 platforms she is using. For example, she also has created an eBay shop to sell and auction some of her products, but needs to manually synchronize the two shops.

Nada wants to change this and to do this more efficiently, thus saving time and being able to spend it making more sales opportunities. She wants to be able to automate product listings and she is convinced that adding her products to her web 2.0 sites would significantly increase her sales. However, automating things is a highly technical work and even if she develops a piece of software to connect her shop to the product data of her supplier, then she would make herself bound to this supplier and wouldn't be as flexible, any more as things tend to be highly connected.

Nada chats with some friends about her problem and one of them recommends SOA4All identifying the following benefits to her:

- It would help her to easily discover supplier services that she can use as a product data source.

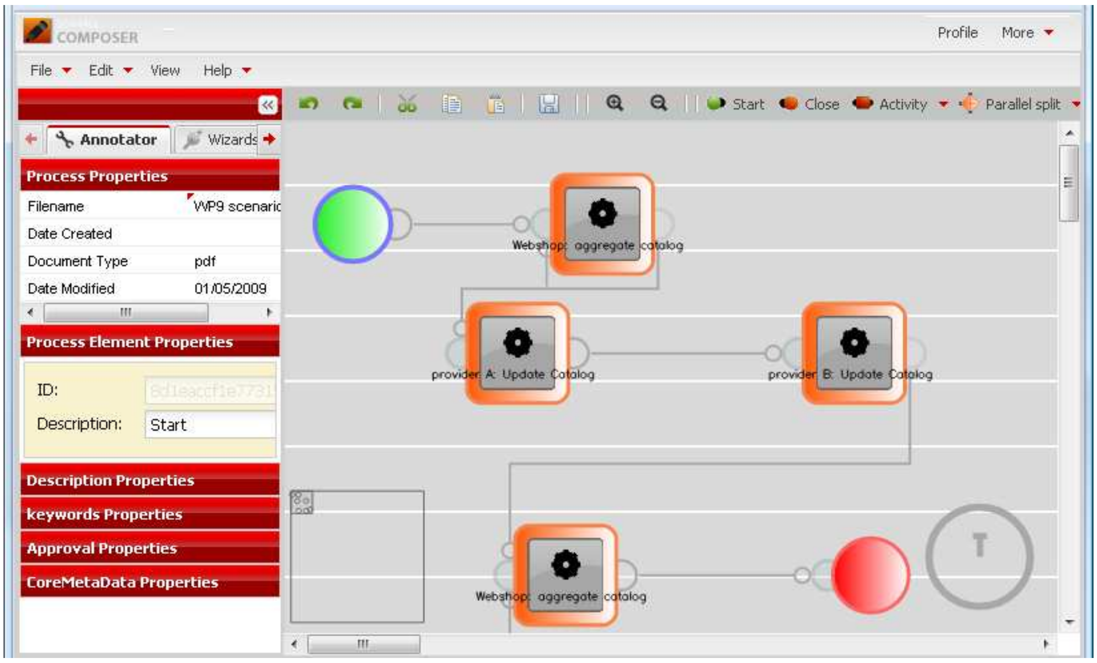
[1] http://www.eclass-online.com

[2] http://purl.org/goodrelations

- It would allow her to stay flexible as she can model complex processes easily using a graphical process composer.
- It would allow her to do a fully automatic integration by directly connecting all services starting at the supplier's product catalog and ending with her product presentation. – No more manual updates of product data.
- It would allow her to optimize the overall quality of connected services. For instance, this can be achieved by minimizing their prices and ensuring seamless connections between services as well.
- It would allow her to 'send' her data everywhere as long as services permit this: to her webshop, to Facebook, to Twitter and to virtually any other web 2.0 platform that she already uses or will become available in the future. Unlike RSS feeds, her product information would be seen as part of the platforms.
- She does not have to deal with storing outdated product data and worrying about the best description terms. She can simply rely on getting product descriptions from the cloud the moment she needs it. No need to store the description at her server if she does not want.
- She can consider context information for her sales items. For example, the following data could be considered (based on the privacy settings of the visitor):
    - the profile of the site visitor (gender, education, age, country, language, …),
    - the profiles of his/her contacts/friends (e.g. birthday dates, current locations, …),
    - the device currently used by the site visitor (e.g. an iPhone, a car navigation system, a PC, a kiosk in an hotel/shop/airport),
    - the current location of the site visitor and eventually the weather and traffic conditions at the location,
    - the current activity of the user (e.g. travelling, working, shopping, …),
    - the scheduled activities of the site visitor.

Nada visits the SOA4All Studio and creates a user profile with the SOA4All Profile Editor. Surprisingly, she notices that she can even reuse her OpenID for logging in, which she uses on many other websites as well.

She then starts searching for suitable services using the SOA4All Discovery Platform and finds many services related to products. She uses the SOA4All process editor to create her own personalized process based on those services:



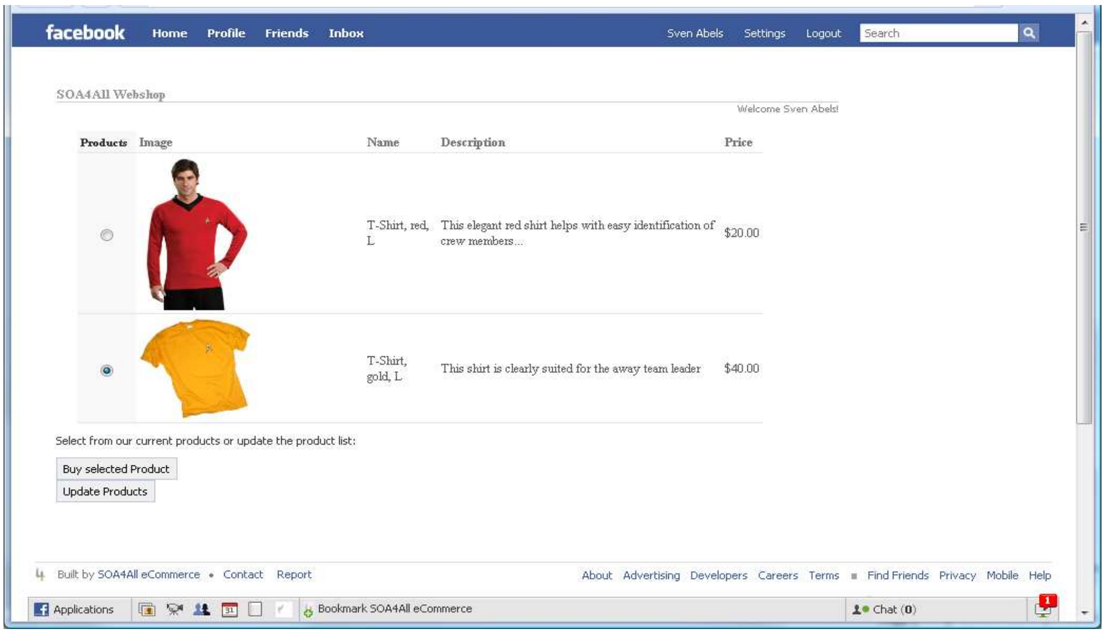
Filename	WP9 scenaric
Date Created	
Document Type	pdf
Date Modified	01/05/2009

ID: 8d1eaccf1e7731

Description: Start

Figure 1: The Process Editor showing a WP9 process

After a while Nada visits the SOA4All studio again and extends the process, which gathers the product definitions from external suppliers by adding additional distribution channels. For this purpose, Nada creates a new process in the process editor, which combines her webshop and the service outputs from Claus, Theodore and Esteban and feeds them into a syndication service, which acts as a mediator for her product data. From this syndication service, Nada feeds the data via SOA4All to different platforms. She starts with Facebook and decides to extend it with Twitter, eBay and the newly announced Google Wave beta.


facebook
Home
Profile
Friends
Inbox
Sven Abels
Settings
Logout
Search
Built by SOA4All eCommerce • Contact Report
About Advertising Developers Careers Terms Find Friends Privacy Mobile Help
Applications
Bookmark SOA4All eCommerce
Chat (0)

SOA4All Webshop

Welcome Sven Abels!

Products	Image	Name	Description	Price
○		T-Shirt, red, L	This elegant red shirt helps with easy identification of crew members...	$20.00
●		T-Shirt, gold, L	This shirt is clearly suited for the away team leader	$40.00

Select from our current products or update the product list:

Buy selected Product

Update Products

Figure 2: The SOA4All Facebook App

3.4 Arian

Arian is a friend of Nada but had not seen her for 6 months, when they met at a social event and talked about her desire to set up a webshop. He asks himself what Nada is doing now and finally finds Nada's Twitter page and her Facebook profile. He sees that Nada has been an active user and is now selling clothes via her Facebook page and surprisingly these clothes even fit to his profile. He decides to buy one of them. He clicks on the product, pays with his Facebook credits and happily receives the products a few days later. Summarizing, Arian never notices that he actually deals with SOA4All. From his viewpoint, everything is happening in the background but all courtesy of the project.

3.5 Silvio

Silvio represents a very large association of magazines. Such magazines have a web-presence, and want to invest into on-line advertising. Silvio wants to increase accesses to his web portal: he is currently using a banner-based strategy, but he cannot reach all the sites he would like, and moreover he has the suspect that web users tend to “avoid” clicking on banners.

Silvio knows about the e-Commerce framework that is based on SOA4All technology. This framework allows to implement an innovative “collaborative advertising” paradigm, where web-users are encouraged to click on banners thanks to a credit gathering mechanism. Moreover, this functionality is community-oriented, as a user who brings a new “friend” will receive extra credits. Such credits can be spent later on by users in the web Sellers shops. Silvio understands the potentials of web-shops built with the SOA4All e-Commerce framework and the attractive idea of collaborative advertising, so he decides that he wants to have his banners enabled for this “collaborative advertising” concept..

Thanks to the WP9 e-Commerce framework, he is provided with the REST services to add to his banner in order to turn it into a “collaborative advertising banner”.

Moreover he can provide his new collaborative-enabled banner as a semantically-annotated REST service, so that SOA4All users can find it from the Consumption Platform, and include it into their processes or applications.

4. Prototype Architecture

This section describes the architecture of the prototype: section 4.1 describes all the components that are forming the prototype, classifying them into three main categories. Section 4.2. shows the relationship with the SOA4All architecture, and which component is accessed by the WP9 scenario actors. Finally, section 4.3 describes how the architecture will be enhanced for the future planned extensions, which will be incorporated into the final version of the prototype at month M33.

4.1 Prototype Components

WP9 demonstrator is based on both SOA4All components and the “WP9 eCommerce Framework”, which was described in D9.2.1. The following picture provides a graphical representation of the components forming the Framework, and their relationship with SOA4All tools and WP9 users:

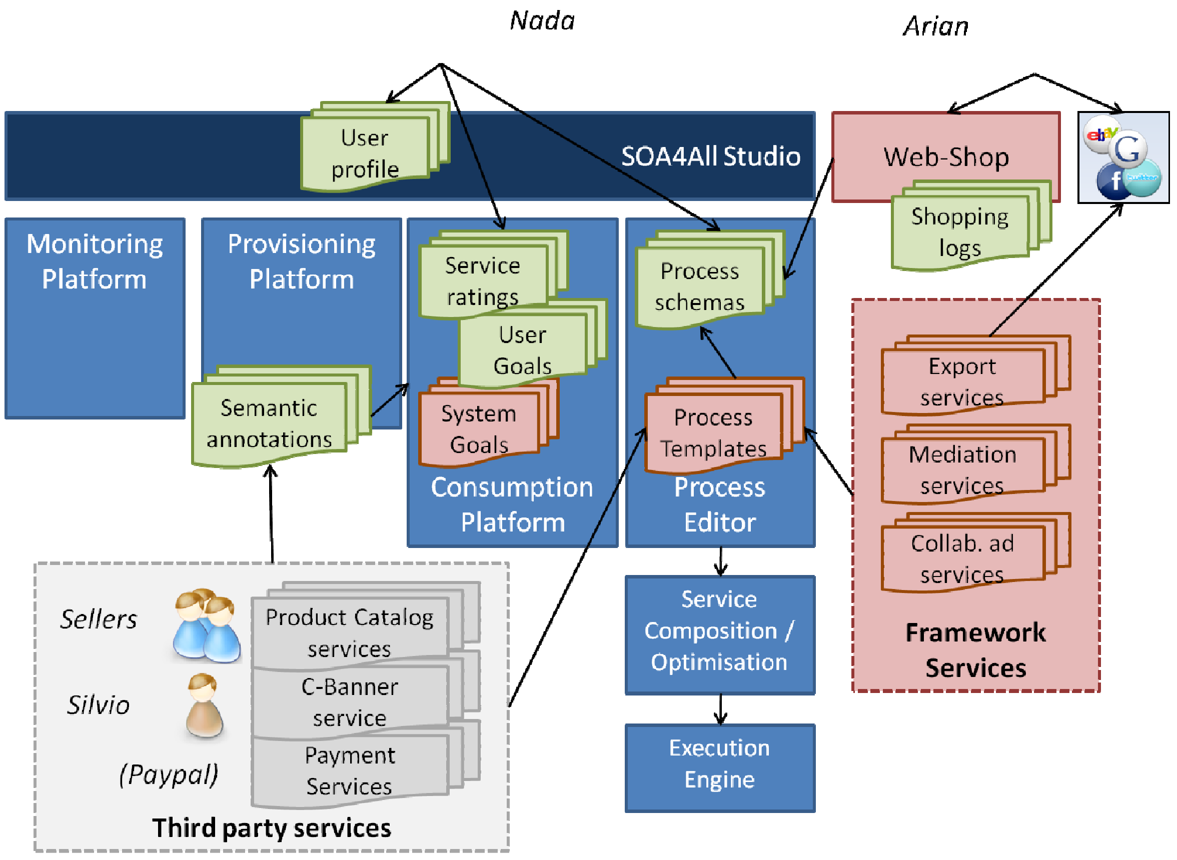


Figure 3: Graphical Representation of the Prototype components

We can identify three main categories of components.

1. **e-Commerce Framework components:** these components (represented in pink colour in Figure 3) are developed within WP9 and form the basis to help end-users to build e-Commerce applications on top of SOA4All technology. They include:

- **a pre-defined set of goals,** to help end-users (i.e. Nada) to identify service providers in the area of product catalogues. These goals are available in the Consumption Platform.
- **a library of Process Templates**, specialised in the e-Commerce area (catalogue management, order forwarding, e-payments) to help Nada to compose her processes in a faster and easier way than starting from scratch.
- **framework services,** required to complete and to optimise the scenario. These services can be included in Nada's processes in the Process Editor, and they include:
    - **support services:**
        - *Mediation Service*, to merge information coming from different catalogue providers (i.e. product category names, unit of measures, etc..).
        - *Collaborative Advertising Services,* to incentivise the use of advertising in the web-shop, by supporting a collaborative paradigm.
    - **a set of multi-channel export services**, enabling Nada to export her catalogue to multiple platforms such as Facebook and Twitter.
- **web-Shop setup:** this webshop will be used out of the box without any change in the existing webshop. Exposed web services and existing integration features will be used to show that the products selected by Arian are connected to the shop.

2. **Third Party Services:** WP9 demonstrator includes the following services provided by Sellers and Advertisers (light-grey colour in Figure 3):
    a. **Product Services** (both as WSDL and REST), provided by the Sellers (Theodore, Esteban, Claus).
    b. **Payment Services**, exploiting existing payment services such as Paypal[3]
    c. **Collaborative-banner Services**, provided by Silvio, returning text and image of a collaborative-enabled banner.

3. **User-generated content:** while working with the e-Commerce prototype, all the users are generating specific content (green colour in Figure 3). This content includes:
    - **semantic annotations**: generated by third-party service providers (both for REST and WSDL services).
    - **goals and service ratings**: mainly generated by Nada while searching for suitable services from the Consumption Platform.
    - **process schemas**: generated by Nada while composing her e-Commerce processed starting from the pre-defined templates.
    - **shopping logs**: generated by Arian while visiting the web-shop and purchasing items. These logs are gathered by the SOA4All Monitoring Platform for further analysis purposes.

[3] http://www.paypal.com

All these components are described in detail in Chapter 5

4.2 Architecture

The architecture of the WP9 demonstrator is based on the SOA4All architecture as described in D1.4.1A. Depending on the specific WP9 scenario user, different SOA4All components are used.

4.2.1 Service Providers (Theodore, Esteban, Claus, Silvio)

Sellers and Silvio are considered as “service providers”. As such, they are working with the **Provisioning Platform** (SWEET & SOUR), in order to semantically annotate their Product services (either WSDL or REST) or Banner service. Semantic annotations are then stored in the Semantic Space.

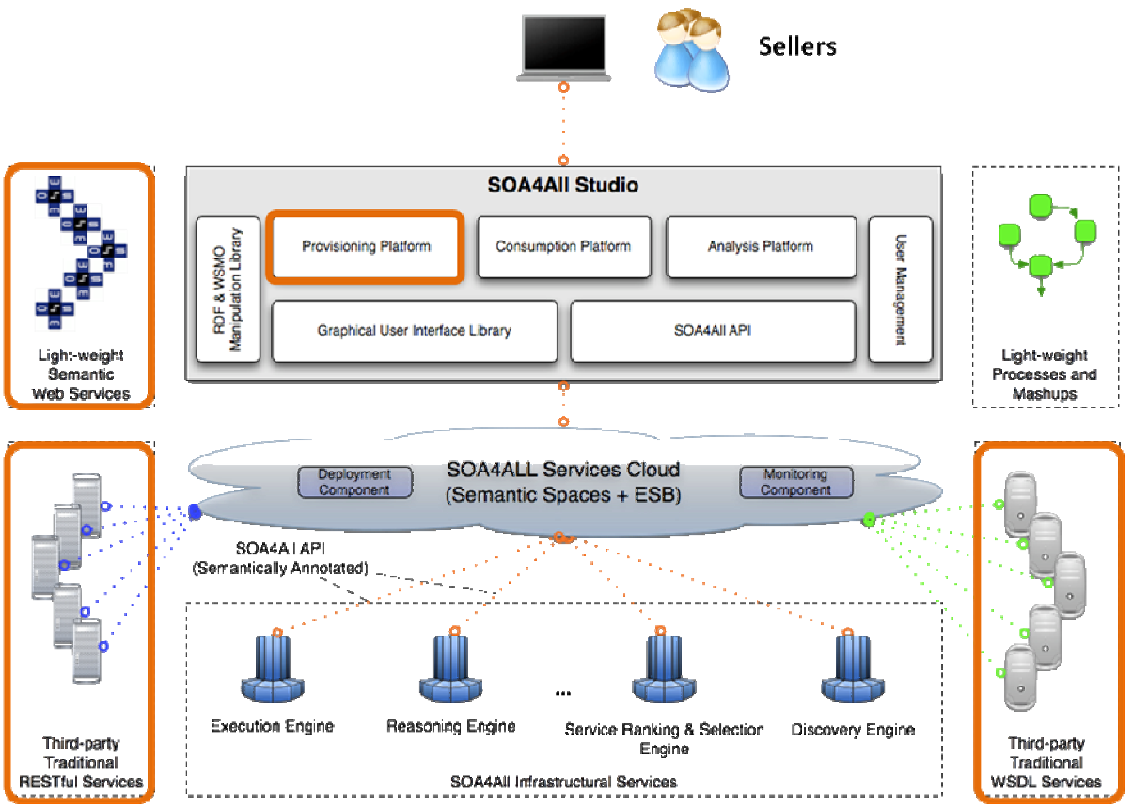

Sellers
SOA4All Studio
RDF & WSMO Manipulation Library
Provisioning Platform
Consumption Platform
Analysis Platform
User Management
Graphical User Interface Library
SOA4All API
Light-weight Semantic Web Services
Light-weight Processes and Mashups
SOA4ALL Services Cloud (Semantic Spaces + ESB)
Deployment Component
Monitoring Component
SOA4All API (Semantically Annotated)
Third-party Traditional RESTful Services
Third-party Traditional WSDL Services
Execution Engine
Reasoning Engine
...
Service Ranking & Selection Engine
Discovery Engine
SOA4All Infrastructural Services

Figure 4: Relationships with SOA4All architecture for Service Providers

4.2.2 Nada

Nada is working with the **Consumption Platform** (SPICIE – see [4]) in order to identify those services she needs to build her application. In additon, she is working with the **Process Editor** in order to compose required e-commerce processes, and finally she is accessing the **Analysis Platform** in order to monitor her web-shop activities. It should be noticed that each of these front-end components will be accessing SOA4All Infrastructural Services (i.e.

Discovery, Execution, etc…), as described in deliverable D1.4.2A

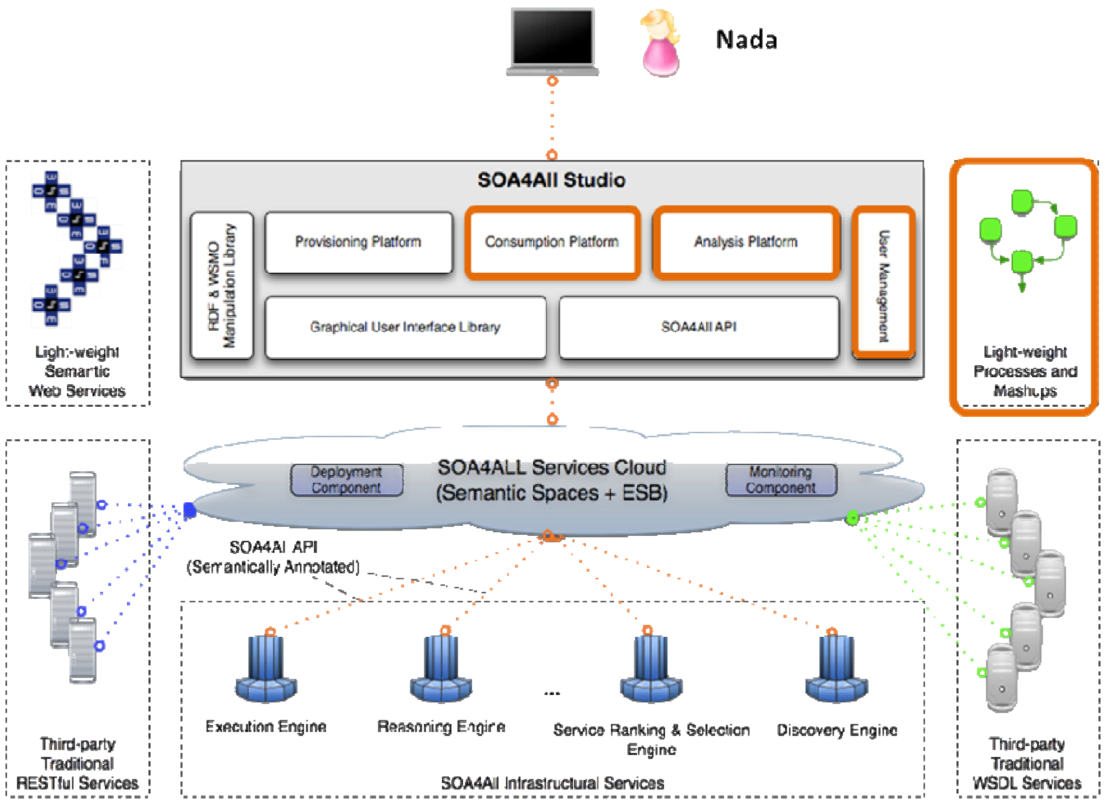

Nada
SOA4All Studio
RDF & WSMO Manipulation Library
Provisioning Platform
Consumption Platform
Analysis Platform
User Management
Graphical User Interface Library
SOA4All API
Light-weight Semantic Web Services
Light-weight Processes and Mashups
SOA4ALL Services Cloud (Semantic Spaces + ESB)
Deployment Component
Monitoring Component
SOA4All API (Semantically Annotated)
Execution Engine
Reasoning Engine
...
Service Ranking & Selection Engine
Discovery Engine
SOA4All Infrastructural Services
Third-party Traditional RESTful Services
Third-party Traditional WSDL Services

Figure 5: Relationships with SOA4All architecture for Nada

4.2.3 Arian

Arian is accessing only the Web Shop, while the SOA4All technology is transparent to him. In the background, the Web-Shop itself (i.e. Facebook Application) is invoking SOA4All Executor via the **SOA4All API** in order to run the e-commerce processes.

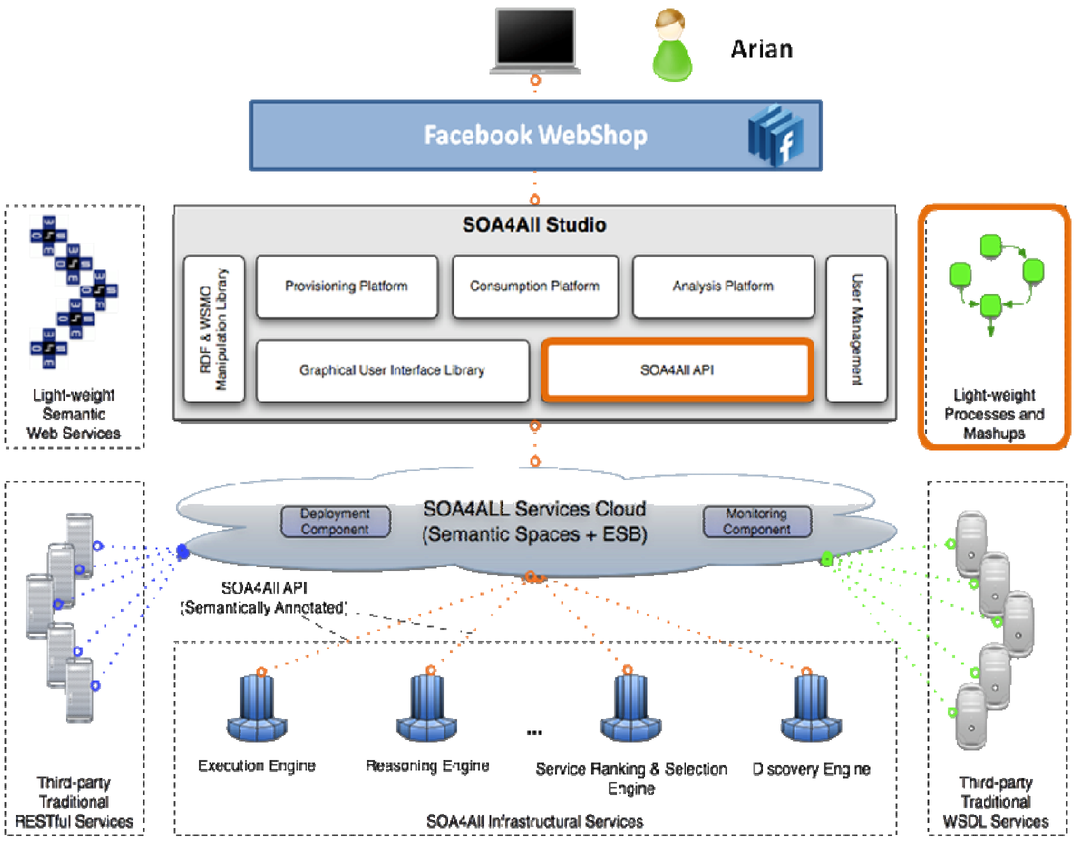

Arian
Facebook WebShop
SOA4All Studio
RDF & WSMO Manipulation Library
Provisioning Platform
Consumption Platform
Analysis Platform
User Management
Graphical User Interface Library
SOA4All API
Light-weight Semantic Web Services
Light-weight Processes and Mashups
SOA4ALL Services Cloud (Semantic Spaces + ESB)
Deployment Component
Monitoring Component
SOA4All API (Semantically Annotated)
Execution Engine
Reasoning Engine
Service Ranking & Selection Engine
Discovery Engine
SOA4All Infrastructural Services
Third-party Traditional RESTful Services
Third-party Traditional WSDL Services

Figure 6: Relationships with SOA4All architecture for Arian

4.3 Planned Extensions

A second release of the Prototype is planned at month 33. This new release will extend and improve this first prototype.

Although it will be based mainly on the feedback gathered during the evaluation and testing on the first prototype, WP9 partners have already identified several areas requiring to be extended or improved.

4.3.1 Framework Services

Both framework support services and export services will be extended:

- *Support services:*
    - **E-Commerce specific services**: an e-Commerce “API” consisting of a set of pre-implemented services would allow users to create e-Commerce specific applications more easily and it would also provide out-of-the-box services for typical e-Commerce situations. For example, the WP9 use case has already created a product aggregation service that allows users to intermix different product data without having to worry about duplicated product information in their web shop
    - **Collaborative Advertising Services***:* a banner service will be implemented, along with a banner service template in order to help advertisers to build their own banner services. The possibility of adding such services dynamically into Nada’s

processes will be explored.

- **Smart Caching mechanism**: During the course of the development, it has turned out that the speed of composed services may be slow if the process consists of many different services. As such, we propose a smart caching mechanism that would allow to fasten the processing of composed services.

- *Export Services :*
  - **eBay and Google Wave integration:** The current prototype implementation shows a tight integration of an e-Commerce process into Facebook via the SOA4All services. In future scenarios, this should be extended for additional backend systems such as eBay and Google Wave.

4.3.2 Framework Templates

e-Commerce process templates: while the current WP9 scenario assumes that Nada is sufficiently experienced in e-commerce, web application development and Web 2.0 applications to design her own process, the 4All aspect of the project should also allow for users with less experience to construct e-commerce processes. Process templates and sample processes will be investigated as possible means to ease the learning curve for users of the process editor.

4.3.3 Web-Shop Set-up

The process of building the web-shop needs to be strengthened, in order to provide better support to users with low technical skills. This will be achieved thanks to the following improvements:

- **e-Commerce specific UI**: It is planned to create an e-Commerce specific set of UI elements allowing users to use SOA4All even faster by reusing typical elements needed for e-Commerce scenarios. The UI will be based on top of the SOA4All studio. It will provide an alternative view of the Dashboard and will therefore provide a new entry point and a new menu structure optimized for e-Commerce scenarios.
- **e-Commerce How-to/examples:** SOA4All is very generic. We therefore foster the creation of e-Commerce specific how-to documents that show in real world examples, how SOA4All would actually be used in modern e-Commerce in a step-by-step guideline.

4.3.4 Third-party services

Integration of third-party services will also be improved. In particular:

- **Integration of Payment Systems:** Payment is the most important step in e-Commerce. It requires a high security and a powerful set of payment options. WP9 therefore suggests to do a pre-defined integration of common payment services and to list them in a specific area. Those services could then be highlighted as “certified SOA4All payment services”.

4.3.5 Use of SOA4All tools

For the second prototype, it is also expected an incremental use of the SOA4All tools:

- **e-Commerce process optimisation:** based on monitoring results, Nada sees how the different provider services are consumed. Services like payment transactions and ordering of products are consumed by the customers of her various web shop front-ends, whether via Facebook, eBay or her own web shop. The SOA4All studio, in particular the Template Generator component within the Process Editor, could help with identifying possible pitfalls and provide useful hints to optimise the processes (i.e. a possible recommendation could be: "nobody seems to buy from supplier X, perhaps you want to drop the service?").
- **Use of context:** as it will be explained in Section 5.2.1, different possibilities to gather context about the actual end-customers exist with the various means offered by social networks. Context information will be utilised in the second prototype to offer better product choices, special offers and other benefits to the customer. Geography, tastes/dislikes and various other data a customer has provided offer insights into his or her buying habits and enable better targeting of products or ads.

5. Prototype Technical Implementation

This section summarises the current state of the technical implementation of all services created for the WP9 e-commerce prototype, which are needed to realise the overall scenario as detailed in Section 3 of this deliverable. In addition, the integration of these services with the SOA4All platform and SOA4All Studio is explained. Finally, details, such as API documentation and usage instructions either are explained in the subsections concerning the concrete service.

It should be noticed that some of the components designed in the Prototype Architecture (section 4) are not currently implemented, as they are planned for the Second Prototype at month M33. Due to such reason, they are not reported in this section.

5.1 Support Services

Support services aim to facilitate the creation of e-Commerce processes, mainly for the reseller role, as defined previously in Section 4.1. Two concrete services are mentioned below, though the pool of available support services is likely to grow in future iterations of the WP9 scenario as more required functionalities are discovered. These services differ from third-party services in the sense that they fulfil specific functions in the WP9 scenario and are unlikely to be offered by external service providers.

5.1.1 Mediation Service

The role of the mediation service in this scenario is the aggregation of product data from different providers.

The three product providers, Theodore, Esteban and Claus define their products using different XML schemas, and allow different operations. The products provided by the sellers have different characteristics, and it is the role of the mediation service to transparently merge all these characteristics into a single product definition. As such, the product provided by the aggregator has all the combined characteristics of the aggregated services products. In addition, the aggregator allows for the transparent invocation off all the operations allowed by the services, returning the composition of the individual invocation results.

5.1.1.1 Overview

The aggregated services describe the products based on different characteristics, described in the following tables:

Table 1: three product services

Element name	Element type
1. Product Web Service:	
id	xs:int

name	xs:string
price	xs:double
buyUrl	xs:string
imageUrl	xs:string
2. IWeb Shop Service:	
id	xs:string
name	xs:string
price	xs:string
buyUrl	xs:string
imageUrl	xs:string
3. Hanival Product WS Service:	
id	xs:int
name	xs:string
price	xs:string
description	xs:string
details	xs:string
imageurl	xs:anyURI
sex	xs:string
size	tns:sizeEnum {S, M, L, XL, XXL}
specialOffer	xs:boolean

The product offered by the **aggregated service** maintains all this information, being characterized by the following elements:

Table 2: aggregated product service

Element name	Element type
id	**xs:string**

name	**xs:string**
price	**xs:float**
orderURL	**xs:string**
imageURL	**xs:string**
description	**xs:string**
details	**xs:string**
gender	**tns:genderEnum {Ladies, Gents, Uni}**
size	**tns:sizeEnum {S, M, L, XL, XXL}**
specialOffer	**xs:boolean**

If one element does not exist in the original product description, the corresponding filed in the aggregated service product is simply marked as “not specified”.

5.1.1.2 Operations

The aggregated service offers all the operations originally offered by all the aggregated services, namely:

getProductList: returns a list containing all the products with all their elements

getProductNameById: returns the name of the product with a given Id

orderProduct: returns true if the product can be ordered and false otherwise

5.1.1.3 Aggregation Service Deployment

The service is available at:

https://sandbox.seekda.com/axis2/services/AggregationService

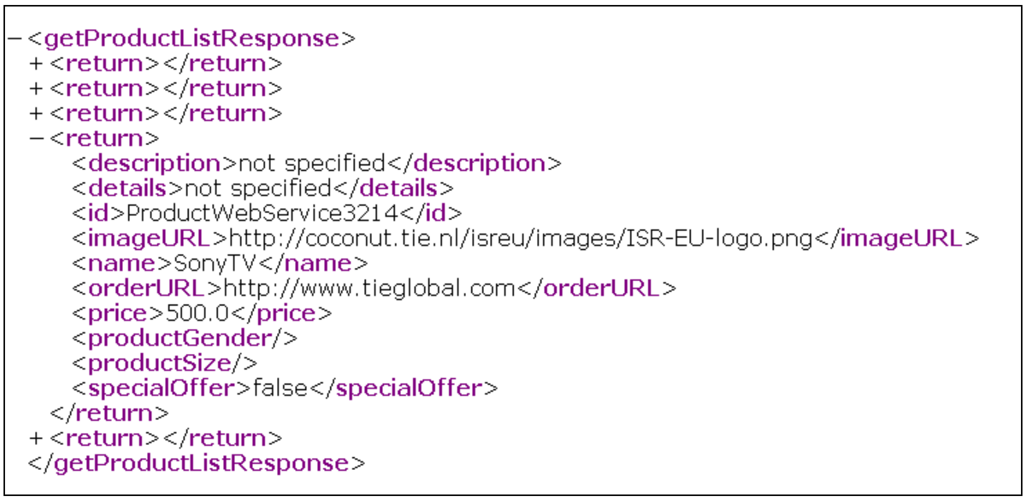
The WSDL file is at:

https://sandbox.seekda.com/axis2/services/AggregationService?wsdl

An example of calling the service by using HTTP GET (the operation for retrieving all products) is shown below:

https://sandbox.seekda.com/axis2/services/AggregationService/GetProductList

The result is the following:



```
- <getProductListResponse>
  + <return></return>
  + <return></return>
  + <return></return>
  - <return>
      <description>not specified</description>
      <details>not specified</details>
      <id>ProductWebService3214</id>
      <imageURL>http://coconut.tie.nl/isreu/images/ISR-EU-logo.png</imageURL>
      <name>SonyTV</name>
      <orderURL>http://www.tieglobal.com</orderURL>
      <price>500.0</price>
      <productGender/>
      <productSize/>
      <specialOffer>false</specialOffer>
    </return>
  + <return></return>
  </getProductListResponse>
```

The result includes all the products offered by the three services. As it can be seen in this example, some of the products are not originally described using all the parameters from the product description as defined in the mediation service. In this case, those parameters are also unspecified in the aggregation service result.

5.1.1.4 Transformations

Implemented in java, the aggregator executes calls to the other three Web Services, every time an operation is performed. It checks the data retrieved from each service in order to remove the null values, and solve the existing heterogeneity problems.

These types of problems are usually referring to simple type conversion, like in the case of the imageURL from xs:anyURI to xs:string. Another type of problems occurs for example when one element contains unrequested information, like it is the case with the price of the products offered by Hanival Product WS, which contain information about the currency. In this case, there are two options for the aggregated service: either remove this information or specify that for the products offered by the other two services this information is not available.

For all the aggregated services, the products are uniquely identified by the element id. The aggregator service maintains this characteristic, but internally creates an id containing information about the service that offers the product (its name) and the original id of the product.

5.1.2 Collaborative Advertising Service

5.1.2.1 Overview

The collaborative advertising service incentivizes users to click on banners, as they will get more credits. This means users will be keener to visit sites offering this type of banners, but also that the number of users will increase thanks to the community-oriented mechanisms of this service: based on a “member get member”, users are rewarded if they bring in a new user). Nada realises that all this will bring additional revenues to her site, and will be more “friendly-seen” to shop visitors, so she decides to include it in her web-shop.

The following table shows the steps that have to be taken by the different stakeholders in order to enable collaborative advertising.

Table 3: Collaborative Advertising Service

Role	Actions
Silvio	Silvio logs into the e-Commerce framework
	Silvio chooses to register as a “*Collaborative*-enabled” advertiser
	The WP9 e-Commerce framework provides him with the *Collaborative* Service invocation code to be included into his banner
	Silvio can build a “c-banner service”, returning the full graphical layout and HTML code of his collaborative-banner. A sample implementation of such service will be provided for the M33 version of the Prototype
	Silvio can now semantically annotate his service (i.e. as a REST service), using the Provisioning Platform
Nada	Nada logs into the e-Commerce framework
	Nada chooses to look for “collaborative advertising services” using SOA4All functionalities (Consumption Platform)
	Nada is presented with a list of possible banner services, that can be included into the web shop, and selects one to include in the web shop
Arian	Arian finds a *Collaborative*-enabled banner in the web shop and clicks on it
	Arian is redirected to the *Collaborative advertising* page, where he can login and be given extra “credits”. After that he is redirected to the advertiser’s site

5.1.2.2 Resources

Collaborative Advertising Services have been implemented as REST services.

There are two main categories of services: *General resource* and *User Resource*

- **General Resource** allows to manage the overall *CollaborativeAD* system: create users, create affiliates, create publishers, get system users
- **User Resource** allows to manage users: modify, delete users, register user actions (click on banner, user registered)

The following sections provide technical details and examples on these resources

5.1.2.3 General resource

http://demos.txt.it/collaborativeAD/rest/

HTTP GET method

This method shows the username of all *CollaborativeAD* registered users, under the form of an XML document.

Examples
Input:

GET of the plain resource url -> http://demos.txt.it/beangarden/rest/

Output:

```
<list>
  <user>user0</user>
  <user>user1</user>
  <user>user2</user>
  <user>user3</user>
</list>
```

HTTP POST Method:

This method creates a *CollaborativeAD* user, or a publisher, or an affiliate, or a credit-earning user action.

Table 4: HTTP post for general resource

INPUT PARAMETER	OPTIONAL	TYPE	DESCRIPTION
username	Yes	String	Username of the user to be created
password	Yes, needed if username parameter is present	String	Password for the user to be created
publishername	Yes	String	Name of the publisher to be created
affiliatename	Yes	String	Name of the affiliate to be created
actionname	Yes	String	Name of the action to be created

value	Yes	Integer	The amount of beans gained for the current action (default value is 0)

5.1.2.4 User resource

http://demos.txt.it/collaborativeAD/rest/{userName}

HTTP GET method:

This method shows the username and number of collected beans of the user represented by this resource, as an XML document.

Example

Input:

GET of the plain resource url -> http://demos.txt.it/beangarden/rest/user0

Output:

```
<user>
  <username>user0</username>
  <bean>7</bean>
</user>
```

HTTP POST Method:

This method registers a credit-earning action made by the selected user resource.

Table 5: HTTP post for user resource

INPUT PARAMETER	OPTIONAL	TYPE	DESCRIPTION
publisherId	No	Integer	ID of the publisher whose banner has been clicked
affiliateId	No	Integer	ID of the affiliate site whose Web page houses the banner which has been clicked
actionId	No	Integer	ID of the action performed (up to now the only possible values are 1 – “banner clicked” and 2 – “registration to BeanGarden”)

HTTP DELETE Method:

This method deletes the current user resource.

HTTP PUT method:

This method updates the password and/or the collected beans number of the current user resource.

Table 6: HTTP put for user resource

INPUT PARAMETER	OPTIONAL	TYPE	DESCRIPTION
password	Yes	String	New password to be set for the current user resource
beans	Yes	Integer	Number of credits already earned by the current user resource

5.2 Multi-Channel Export Services

Export services are created for the WP9 scenario, in order for the shop owner – Nada – to distribute her product data, offers and other information to different distribution channels. These services make use of existing APIs of the various targeted platforms to either directly post data to the platform (e.g., see the Twitter export service), or they transform the data and insert it to a specific repository, which is in turn accessible by the respective platform (again as an example, the Facebook Service has been designed in this manner).

5.2.1 Facebook

With the large existing user base (around 400 million user accounts at the time of this writing) and with a projected steady growth for the near future, Facebook offers an ideal environment for Nada to publish her product data and attract new potential customers. Please note that the WP9 team has chosen to use Facebook as a first example for web 2.0 environments mainly because of the broad user scope which will guarantee a high impact of the SOA4All WP9 prototype and also because it goes well with the scenario and with the prototype implementations.

The Facebook application designed and implemented in the scope of WP9 allows her to display her products directly from her own Facebook profile, respectively to allow users to add the Facebook application to their own profiles. Whenever a user visits the Facebook application, the currently available products and offers are updated by invoking the WP9 e-Commerce process deployed on SOA4All infrastructure.

As explained previously in D9.2.1, Facebook offers two different ways for application developers to work with Facebook data and API calls: applications built in an IFrame and applications based on the Facebook Markup Language (FBML). For the Facebook shop application built in WP9, the IFrame solution has been used, as it provides a more fine-grained approach to social data, and allows the Facebook application to query additional data from visiting users.

A Facebook application offers the unique opportunity to gather the social context of a visiting user. This information can be applied as contextual hints regarding the interests of the given user and enables the adaptation of the application, respectively the contents (in this case the selection of products) shown to the user. We use a version of the Facebook API, based on a REST-like interface, to collect the most interesting information about any given visiting users.

As an example for the basic Facebook API method in use, see Table 7. The amount of data that can be gathered from a user's Facebook profile depends on a few parameters: First, the user must of course have provided information for some areas (such as the fields for a user's favourite media). Furthermore, the privacy settings constrain which kind of information can be queried from any given user (i.e., a user might have restricted some information to be visible to his friends or affiliation network only). Finally, the available data depends on an existing session, which usually means that the Facebook user has to authorize an application to access his or her information. The user data can then be used by the Facebook application to customize the offered products.

Table 7: Querying a Facebook User

Facebook API Method	***Users. getInfo***
Description:	Given a list of Facebook UIDs, this method returns a data structure containing the user info elements. The following basic information is available, even if a user has not (yet) authorized the FB app for their profile: # uid # first_name # last_name # locale # current_location # sex # affiliations (regional type only)
URL:	http://api.facebook.com/restserver.php
Formats:	XML, JSON
HTTP Method(s):	GET
Requires Authentication:	true

The returned elements from Users.getInfo contain a wide variety of information, in addition to the core information shown in the method description above. For WP9, we have selected a subset of this information, which is most applicable to the task of gaining context information, which allows us to adapt the kind of products shown to a user:

- **activities** - User-entered "Activities" profile field. No guaranteed formatting[4].
- **birthday** - User-entered "Birthday" profile field. No guaranteed formatting as it is based on the user's locale. The additional field **birthday_date** provides a guaranteed format.
- **education_history** – Contains a list of school information, as **education_info** elements, each of which contain **name**, **year**, and **concentration** child elements.
- **interests** - User-entered "Interests" profile field. No guaranteed formatting.
- **movies** - User-entered "Favorite Movies" profile field. No guaranteed formatting.
- **music** - User-entered "Favorite Music" profile field. No guaranteed formatting.
- **relationship_status** - User-entered "Relationship Status" profile field. Is either blank or one of the following strings: Single, In a Relationship, In an Open Relationship, Engaged, Married, It's Complicated, Widowed.
- **significant_other_id** - the id of the person the user is in a relationship with. Only shown if both people in the relationship are users of the application making the request.
- **tv** - User-entered "Favorite TV Shows" profile field. No guaranteed formatting.
- **work_history** - list of work history information, as **work_info** elements, each of which contain **location**, **company_name**, **position**, **description**, **start_date** and **end_date** child elements. If no work history information is returned, this element is blank.

Currently, this information is used to select which products are shown to the visiting user of the Facebook application, based on a simple contextual filtering mechanism inside the appilication itself. For the next version of the WP9 eCommerce prototype, this information will be transformed by the Facebook application into an ontological representation of user context and then sent to the WP9 eCommerce process (which is currently invoked whenever a user clicks on "Update Products" in the Facebook application).

We have used a dedicated Java library to interact with Facebook inside the e-Commerce application, called facebook-java-api[5]. The library has been made available under an MIT licence. The SOA4All WP9 Facebook application has been deployed to Facebook and is available at http://apps.facebook.com/soa_ecommerce/.

5.2.2 Twitter

The Twitter micro-blogging service has been integrated with the current version of Nada’s e-Commerce process, as defined for the M24 prototype. After the product data has been aggregated from several product suppliers, and the different data formats have been mediated between, Nada can chose from several export services to distribute the data. She has selected Twitter as one of the distribution channels. The product data should be transformed by this service to a short tweet, which is sent as a status update to Nada’s Twitter account. The tweet itself includes an announcement, containing a short notification about the new product, respectively the special offer, and a link to the webshop.

[4] „no guaranteed formatting“ means that there is no formal specification on how those fields are expressed within facebook, i.e. no formal syntax definitions

[5] available at http://code.google.com/p/facebook-java-api/

The following section first lists several requirements, which were defined for the Twitter service, leading to specific design decision for the service itself, and the WP9 e-Commerce process.

- **Products and offers, which need to be announced via Twitter, should be selectable**

For Nada, it makes no sense to just tweet about every product in her aggregated product catalogue. Interested customers would be flooded by announcements, which offer no direct benefit in comparison to more conveniently browsing Nada's Facebook shop application, or her dedicated webshop. Only special products and offers valid for a certain time period, or related announcements should warrant a tweet. For the WP9 e-Commerce process, we assume that the different product suppliers will from time to time launch such offers, which Nada wants to forward to her customers. Thus only products flagged as "special offers" by the vendors should be forwarded to the Twitter service.

This requirement means that the Twitter service needs to filter out any product from the aggregated list, in order to only tweet about the flagged subset of products. For the next version of the prototype, this functionality will likely be included with the process itself. I.e. there will be a separate service to filter out the products, as this functionality will be decoupled from the Twitter service, which should only offer functionalities for the generation of tweets.

- **Product information should be tweeted regularly**

This requirement leads to the more specific question about the consecutive invocation of the process, and the intervals between each invocation. The WP9 e-Commerce process combines all products from the different suppliers and then prepares this information for distribution via different channels, and social networks. As such the different networks have different requirements on when this data should be available.

The different product suppliers add new products at all times, which means that at any given point, new product offers could be existing which need to be tweeted in a short timeframe. While for Facebook, the product data should be up to date, whenever a user actually browses the products shown via the application, for Twitter, the product data should be sent as "real-time" as possible. Twitter differs from Facebook in this sense, as it is a "push" network, instead of a classic "pull" infrastructure. As a side note, each time the process is executed, the data is normally distributed via all available and attached distribution services. If needed, this behaviour can be easily changed in the SOA4All process editor, by introducing conditional branches to the process. I.e. the execution of one specific branch – sending product information to the Twitter service – could be dependant on the actual invocating party (i.e., the service client).

As a solution to the above, ideally the WP9 e-Commerce process is executed quite frequently, for example once an hour. Nada should customize the process in the SOA4All studio to actually set up a time interval for the process execution. The SOA4All infrastructure has to respect this setting, and – after deployment via the Executor platform service, has to guarantee automated execution once during each time interval.

- **The information contained within each Tweet should be as short as possible**

As Tweets are constrained to 140 characters, only basic information about an offer can be included in the tweet. This will include at least a, possibly shortend, product title as well as a link to the specific product in Nada's webshop. In order to keep the message as short as possible a URL shortening service has to be used to transform the webshop URL. For this, the bit.ly URL shortening service has been selected, since it offers a comprehensive RESTful API to automate the shortening of URLs.

Table 8 below shows the most important method made available by bit.ly, the actual process of shortening an URL given as a parameter to the service.

Table 8: bit.ly URL shortening

Twitter REST API Method	***/shorten***
Description:	Given a long URL, /shorten encodes it as a shorter one and returns it.
URL:	http://api.bit.ly/shorten
Formats:	XML, JSON
HTTP Method(s):	POST
Requires Authentication:	true

Bit.ly offers several additional features and services, which can be leveraged in the e-Commerce process. As an example, bit.ly collects statistical information on the traffic and referrer data linked to a shortened URL. Table 9 shows the available REST-ful Web Service, which allows Nada to find out how many of her Twitter followers clicked on a specific link.

Table 9: bit.ly URL statistics

Twitter REST API Method	***/stats***
Description:	Given a bit.ly URL or hash, return traffic and referrer data.
URL:	http://api.bit.ly/stats
Formats:	XML, JSON
HTTP Method(s):	GET
Requires Authentication:	true

- **Nada should be able to provide her Twitter account data via the SOA4All Studio**

As part of creating her overall SOA4All profile in the Studio, Nada will be able to provide any additional account information for other external networks and services. For the first prototype version described in this deliverable, we have provided the means to customize this as a parameter file, through in the future we expect the SOA4All Studio to take care of this (i.e. the profile in the Studio would include the Twitter account).

All the above design decisions led to the formulation of the following steps, which together define the Twitter WS:

1) The process puts together the aggregated products (similar to the current process)
2) The "special products" are filtered out, resulting in a reduced set of products that should be twittered about
3) For each remaining product in the set, the following steps are followed:
    a. The bit.ly URL shortener Web Service is invoked.
    b. A tweet is created, including a short message with the product title and the product URL.
    c. Tweet the product announcement, using the Twitter WS API, and using the credentials provided by Nada in the SOA4All Studio. The WS method used for this is show in Table 10.

The filtering and repeated invocation of the Twitter WS can be a wrapped WS or actually also modelled as part of the overall e-Commerce process. For most users, it might be too much of a hassle to recreate every step of this process in the SOA4All process editor. Therefore, we have created a service, which abstracts from the fine details, taking in a product list, and sending one tweet for each special product included in this list.

As explained in D9.2.1, we have investigated several libraries to address the Twitter APIs programmatically. For the purposes of the WP9 e-Commerce export services, Twitter4J[6] has been selected. This open source library (released under a BSD-style license) is a pure Java library, which includes built-in OAuth support and requires no additional libraries.

Table 10: Twitter Status Update

Twitter REST API Method	*statuses/update*
Description:	Updates the authenticating user's status. Requires the status parameter specified below. Request must be a POST. A status update with text identical to the authenticating user's current status will be ignored to prevent duplicates.
URL:	http://twitter.com/statuses/update.*format*
Formats:	XML, JSON, RSS, Atom
HTTP Method(s):	POST
Requires Authentication:	true

5.2.3 eBay

Adding eBay as a distribution channel for Nada's products will be realised in the second version of the WP9 prototype. Nada will be able to launch her own shop on eBay, which will provide her with additional options for her products. She might sell products at standard rate,

[6] available at http://yusuke.homeip.net/twitter4j/en/index.html

negotiable prices or auction them off using the eBay API's auction services. The functionalities for the eBay service will be explained in detail in the upcoming deliverable D9.4.1, due M33.

5.2.4 Google Wave

As the Google Wave[7] is still in an early stage of development a possible integration with an e-Commerce platform will be investigated for the second version of the WP9 prototype, due in M33.

5.3 Web-shop Building Services

Finally, as part of the e-Commerce framework and services that should be provided to the users of the reseller role in WP9's scenario, the means to work with existing webshop and e-Commerce solutions have to be provided. The following example highlights the way in which SOA4All results will be linked to existing solutions.

5.3.1 Mambofive

Mambofive is an e-Commerce solution providing all modern Webshop functionalities. It provides a service interface allowing technical experts to access the webshop data using RESTful services. Within the past, this interface has only been usable by experts. With the help of SOA4All, the interface can now be used by anyone that uses the SOA4All results. As such, the interface will be used to receive product data from Theodore.

[7] The API documentation is available at http://code.google.com/intl/de-AT/apis/wave/

6. Evaluation Planning

This section reports the progress of the implementation of the various functionalities of the prototype, and introduces the main tests that will be performed during the valuation phase.

6.1 Status Indicators

This section provides an overview of the prototype status as of month 24. The version of the prototype serves as the basis for the development during the final year of the project. The final version of the prototype will be available in M33. The following status indicators show the progress since the last version, available in D9.2.1.

The prototype is divided into three parts based on the people involved in the scenario described earlier in this deliverable – the "advertiser" is assimilated to the Service Provider, since at the end he will be offering a "banner service". Each part explains the tasks and their status (see table below).

▨	Implemented / finished
	New / unimplemented
B: Buyer, **R:** Reseller**, S:** Seller and Advertiser	

The progress in the table shows the completion of the work in context of the WP9 scenario. In most cases, the scenario relies on the results of the other SOA4All work packages. As such, the following percentage values can be interpreted as follows:

- If an element is marked as 100% then it means that it has been implemented and also fully integrated into the scenario in a perfect and final way.
- If an element is marked as 0% then this functionality is either not yet provided by the corresponding work package or it is available but not yet integrated into the WP9 scenario.
- Any other values mean that an element is already part of the WP9 scenario implementation but it might be only partly realized. For example, some services might still be hard-wired in the first prototype version.

6.1.1 Overall Status

Many of the indicators are marked with an intermediate number such as 60%. This does usually mean that the component is implemented and running in a first version but it also means that this component has room for improvements in various aspects. D2.5.2 [6] will give various suggestions on how those components can be improved. For example, the OpenID login is fully working at the first version (80% finished) although it could be improved in various usability aspects as described in section 4.3 of D2.5.2. On the other hand some of the functionalities planned for M33 are already partially implemented.

From a high-level view, the first prototype of WP9 is in a very good shape and is within the scheduled time plan. It has implemented the most important components first and all critical components are available in a first alpha version (10%-40% completeness) or in a first beta version (40%-70% completeness).

6.1.2 Service Providers and Advertiser

The following table shows the status of each task related to service providers who want to increase the sales of his company by providing services.

Table 11: progress indicators for service providers' tasks

ID	Name	Due to	Progress
S1	Register in SOA4All using Profile Editor	M24	■■■■■■■□□□ 70%
S2	Service annotation using Good Relations[8]	M24	■■■■■■□□□□ 60%
S3	Store service in SOA4All	M24	■■■■■■■■■□ 90%
S4	Monitor service usage	M33	■■■■■□□□□□ 50%
S5	Update sales plan and product definitions	M24	■■■■■■■■□□ 80%
S6	Service annotations using additional ontologies (in comparison to S2)	M33	■■□□□□□□□□ 20%
S7	Update service in SOA4All	M24	■■■■■□□□□□ 50%
S8	Include a collaborative advertising service into advertiser's banner service	M24	■■■■■■■■□□ 80%

6.1.3 Nada

The following table shows the status of each task related to Nada, who wants to generate more income by connecting services through SOA4All in order to expose information to various popular web 2.0 sites such as Facebook, Twitter and eBay.

Table 12: progress indicators for Nada's tasks

ID	Name	Due to	Progress

[8] http://purl.org/goodrelations

R1	Login using OpenID	**M24**	▨▨▨▨▨▨▨▨□□	**80%**
R2	Search for services	**M24**	▨▨▨▨▨▨▨□□□	**70%**
R3	Check ratings and comments for each service	**M24**	▨▨▨▨▨▨▨▨▨▨	**100%**
R4	Check recommendations	**M24**	▨▨▨▨▨▨▨▨□□	**80%**
R5	Test Theodore's service	**M24**	▨▨▨▨▨▨□□□□	**60%**
R6	Add services including Facebook service to bookmark	**M33**	▨▨▨▨▨▨▨▨▨▨	**100%**
R7	Check pre-defined templates to Model	**M33**	□□□□□□□□□□	**0%**
R8	Model process using Composition Editor	**M24**	▨▨▨▨▨▨▨□□□	**70%**
R9	Add or Merge services, parameters based on an automatic approach	**M33**	▨▨▨□□□□□□□	**30%**
R10	Monitor service usage	**M33**	▨▨▨□□□□□□□	**30%**
R11	Rate and recommend services	**M33**	▨▨▨▨▨▨▨▨□□	**80%**
R12	Improve marketing channels based on context	**M33**	▨▨▨□□□□□□□	**30%**
R13	Adjust and extend the Model process	**M33**	▨▨▨▨□□□□□□	**40%**
R14	Add a syndication service to mediate between web 2.0 sites and Nada's product data	**M33**	▨▨▨▨▨□□□□□	**50%**
R15	Add a service to handle payment issues	**M33**	□□□□□□□□□□	**0%**
R16	Replace services and add rating	**M33**	▨▨▨□□□□□□□	**30%**
R17	Execute complex scenario	**M33**	▨▨▨▨▨□□□□□	**50%**

R18	Optimize sales and extend partners	M33	▨▨▨□□□□□□□ 30%
R19	Make use of the collaborative advertising service	M33	▨▨▨▨▨▨▨□□□ 70%
R20	Analyse webshop visitors to update process	M33	□□□□□□□□□□ 0%
R21	Automatic process optimization	M33	□□□□□□□□□□ 0%
R22	Improve process self-Adaptation	M33	□□□□□□□□□□ 0%
R23	Customise User Interface	M33	▨▨□□□□□□□□ 20%
R24	Use Advanced web 2.0 platforms: Twitter, eBay, etc	M33	▨▨▨▨▨▨□□□□ 60%
R25	Monitor the process and compare results	M33	□□□□□□□□□□ 0%
R26	Install a webshop	M33	▨▨▨▨▨□□□□□ 50%
R27	Integrate the webshop with the process model	M24	▨▨▨□□□□□□□ 30%
R28	Reseller can include a “c-banner service”	M33	▨▨▨▨▨□□□□□ 50%

6.1.4 Arian

The following table shows the status of each task related to Arian, who wants to buy some products already seen in Nada’s Facebook page.

ID	Name	Due to	Progress
B1	Visits Nada’s Facebook public account	M24	▨▨▨▨▨▨▨▨▨▨ 100%
B2	Chooses a product and pay	M33	▨▨▨▨▨□□□□□ 50%
B3	Clicks on a collaborative-enabled banner	M33	▨▨▨□□□□□□□ 30%

We assume that subsequent versions of developed software must be tested and validated. In this section we provide a set of initial ideas for comparison and functional evaluation of e-Commerce infrastructure presented in this document.

For comparative evaluation we will provide a comprehensive description of how the solution worked before SOA4All and after the technology was developed (started already in this document in the section 5). A comprehensive validation/evaluation report (D9.4.2) will be based on a thorough evaluation of the SOA4All tools and technologies used in the context of this scenario. This evaluation will include potential user interviews/surveys, and usability testing.

Within this section, we distinguish between three types of tests that will be performed by WP9: (i) Integration tests, (ii) functional tests and (iii) performance tests. In addition to those, we assume that each work package and each tasks will also perform a solid unit tests independently of WP9, as described in the test framework deliverable D1.5.1 of SOA4All.

Integration tests will be performed whenever a deep technical integration of different components is necessary, which have not been developed by the same work package. For example for creating and executing a process, WP9 has to create a process with the results of task 2.6 (WP2) and need to connect the services registered in the distributed service bus (WP1) in order to use them in the process.

Performance tests will allow WP9 to identify bottle necks of the SOA4All developments. In case that such performance problems are detected, WP9 will give immediate feedback to the corresponding task work package leaders.

For functional evaluation we require a set of tests to be defined for each stakeholder of the scenario, namely buyer (Arian), reseller (Nada), advertiser (Silvio) and seller (Theodore, Esteban and Claus). Upcoming sections define procedural guide for testing activities that should be carried out for the developed application. It identifies the tests to be performed, and provides schedules for test activities.

6.2 Tests

This section provides an overview of the tests performed on the sellers, buyer, advertiser and reseller. Deliverable 9.2.1 [5] distinguished between three types of tests that need to be performed:

- ***Integration tests:*** performed whenever a deep technical integration of different components is necessary.
- ***Functional tests:*** a set of functional tests have to be performed for every participant.
- ***Performance tests:*** this tests allow us to identify possible bottle necks in SOA4All development.

Furthermore, [5] defines a set of requirements that need to be fulfilled by the stakeholders, and the time-line for fulfilling these requirements in the duration of the project.

6.2.1 Tests for Service Providers and Advertiser

Four requirements were previously identified for the service providers (sellers and advertiser), out of which three should be fulfilled by M24 of the project. These requirements are listed in the following table:

Table 13: Tests for service providers and advertiser

Req ID	Name	Integration testing	Functional testing	Performance testing	Due to
S1	Seller can register in SOA4All using Profile Editor		Yes		M24
S2, S5 and S6	User can annotate service using Good Relations		Yes		M24
S3 and S7	User can store and update his/her service in SOA4All	Yes	Yes	Yes	M24
S4	User can monitor the usage of his/her service	Yes	Yes		M33
S8	User can include a collaborative advertising service into a banner service	Yes	Yes		M24

6.2.2 Tests for Reseller

A number of requirements for the reseller were identified. As listed in the following table:

Table 14: Tests for reseller

Req ID	Name	Integration testing	Functional testing	Performance testing	Due to
R1	Reseller can login using OpenID		Yes		M24
R2	Reseller can search for Services	Yes	Yes	Yes	M24
R3	Reseller can check ratings and comments for each service	Yes	Yes		M24
R4	User can Check recommendations	Yes	Yes		M24

R5	User can test service registered in S3	Yes	Yes		M24
R6	User can add services including Facebook service to bookmark	Yes	Yes		M33
R7	User can check pre-defined templates to Model	Yes	Yes		M33
R8, R13	User can model, adjust and extend process using Composition Editor	Yes	Yes		M24
R9 and R16	User can add or merge services to the process and execute them. Reseller can replace services and add rating	Yes	Yes		M33
R10 and R25	Reseller can monitor service usage and process; compare results	Yes	Yes	Yes	M33
R11	Reseller can rate and recommend services	Yes	Yes		M33
R12	Reseller can improve marketing channels based on context	Yes	Yes	Yes	M33
R14	User can add a syndication service to mediate between web 2.0 sites and the reseller's product data		Yes		M33
R15	Reseller can add a service to handle payment issues		Yes		M33
R17	Reseller can execute complex scenarios		Yes	Yes	M33
R18	Reseller can optimize sales and extend partners		Yes		M33
R19	Reseller can make use of the collaborative advertising service		Yes		M33
R20	User can analyse webshop		Yes		M33

	visitors to update process				
R21	System can automate process optimization		Yes		M33
R22	System can improve process self-Adaptation		Yes		M33
R23	Reseller can customise User Interface		Yes		M33
R24	Reseller can use Advanced web 2.0 plat-forms: Twitter, eBay etc.		Yes		M33
R27	Reseller can integrate the webshop with the process model	Yes	Yes		M24
R28	Reseller can include a “c-banner service”	Yes	Yes		M33

6.2.3 Tests for Buyer

Three tests were defined for the buyer:

Table 15: Tests for buyer

Req ID	Name	Integration testing	Functional testing	Performance testing	Due to
B1	Buyer can visit reseller Facebook public account		Yes		M24
B2	Buyer can choose a product and pay		Yes		M33
B3	Buyer click on a collaborative-enabled banner		Yes		M33

7. Installation

The installation of the current prototype is very easy. As a system requirement, Java 1.6 and Tomcat 6.0 are expected and need to be installed on the system. Afterwards, the latest WAR file of the soa4all Dashboard needs to be deployed: `soa4all-dashboard.war`. This file is always available via

http://coconut.tie.nl/get-soa4all

Once this file has been downloaded and renamed to `soa4all-dashboard.war`, Tomcat needs to be started. This will automatically install all SOA4All files for you. After starting Tomcat, open a web browser and navigate to the following URL:

http://localhost:8080/soa4all-dashboard

This will show you the welcome screen of the SOA4All Dashboard application, which allows you to access the SOA4All studio including all WP9 components.

For any questions during the deployment, please refer to sven.abels@tieGlobal.com

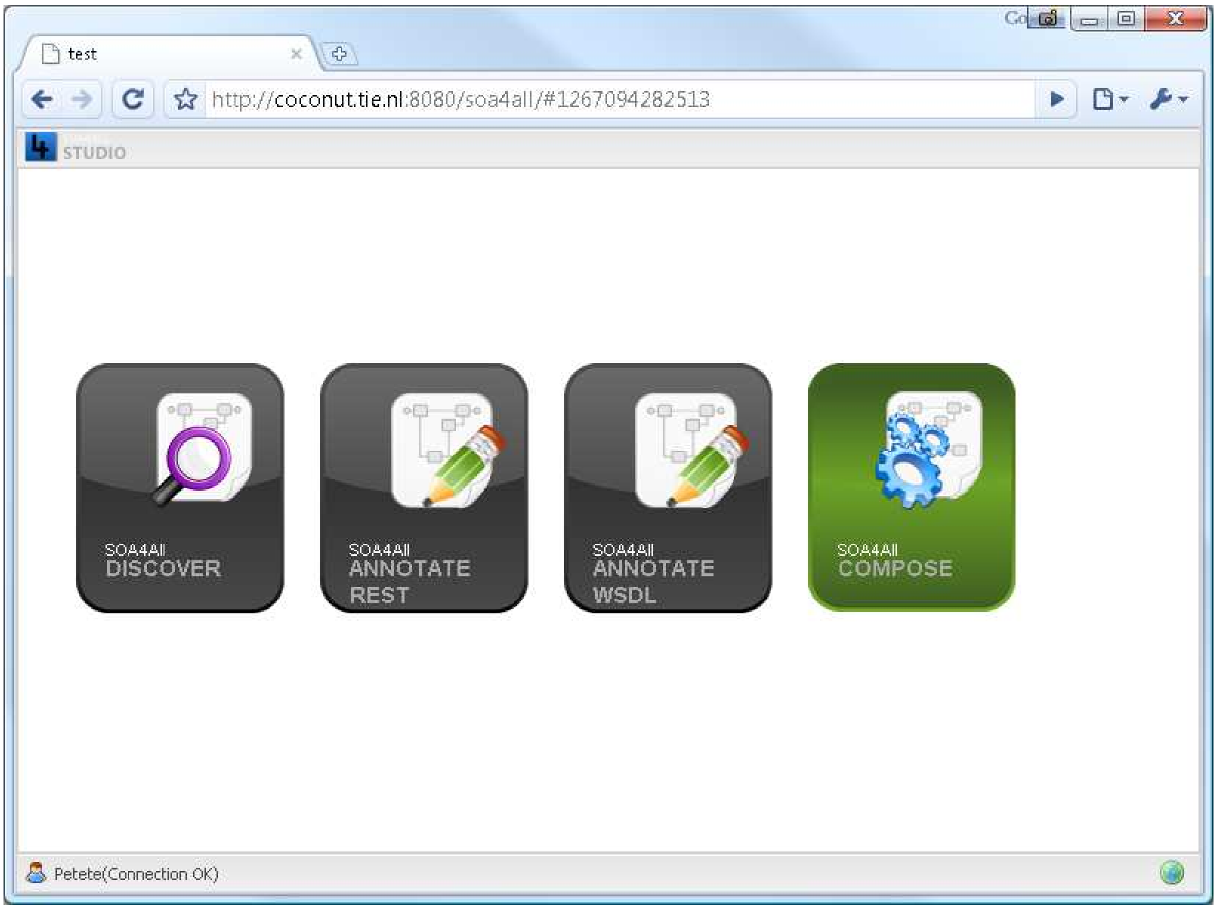


Figure 7: The main SOA4All Studio view

8. Conclusions

This deliverable describes the implementation of the first WP9 e-Commerce Prototype, based on the scenario and on the infrastructure designed into D9.2.1.

The prototype aims to demonstrate the applicability of SOA4All in the e-Commerce domain: this document described how the various functionalities offered by the Prototype are addressing different user needs, allowing them to perform highly innovative operations. The result is an innovative scenario, which can be achieved thanks to SOA4All tools and technology.

The document also provides technical description on the Prototype architecture and its implementation, and provides the planning of its evaluation.

A second prototype will be developed at month M33, addressing feedback gathered during the evaluation of the first prototype and even more innovative functionalities and optimisation. The result expected is a final Prototype, which will be even closer to the end-user real needs, thus offering WP9 partners the possibility to develop and build realistic exploitation plans. The WP9 team has listed ideas for extending the current version with even more e-Commerce specific functionality within section 4 of this document.

9. References

1. Charlie Cheever: Choosing between an FBML or IFrame Application, Facebook Developers Documentation, available at http://wiki.developers.facebook.com/index.php/Choosing_between_an_FBML_or_IFrame_Application, last accessed in August 2009.
2. Schreder, B., Villa, M., Abels, S., Zaremba, M.; Deliverable D9.1.1: Future C2C eCommerce Requirements and Scenario Descriptions, SOA4All: Service Oriented Architectures for All - 215219.
3. eCl@ss. 2004. eCl@ss White Paper, V0.6, 2001, http://www.eclass.de, last access 17.09.2004
4. SOA4All deliverable D2.2.2 “Service Consumption Platform First Prototype”
5. SOA4All deliverable D9.2.1. “eCommerce Framework Infrastructure Design”
6. SOA4All deliverable D2.5.2 “Summative Evaluation Report”