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

D9.4.1 : C2C e-Commerce Prototype v2

Activity 3:	Use Case Activities
Work Package:	WP9 C2C e-Commerce
Due Date:	M33
Submission Date:	30/11/2010
Start Date of Project:	01/03/2008
Duration of Project:	36 Months
Organisation Responsible of Deliverable:	Hanival
Revision:	1.0
Author(s):	Matteo Villa (TXT), Sven Abels (TIE), Shiva Puram (TIE) Bernhard Schreder (Hanival), 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	10/2010	ToC; first sections	Bernhard Schreder (Hanival)
0.2	11/2010	Contributions to Section 3 and 5	Sven Abels (TIE)
0.3	11/2010	Contributions to Section 5	Emilia Cimpian (seekda)
0.4	11/2010	Updates to section 1, 2 and 8	Bernhard Schreder (Hanival)
0.5	11/2010	Contributions to Section 4 and 5	Matteo Villa (TXT)
0.6	11/2010	Editing all sections, version for internal review	Bernhard Schreder (Hanival)
0.7	11/2010	Corrections after review	All
0.8	11/2010	Final version for submission	All
1.0	30/11/2010	Final format corrections for submission	Julia Wells (ATOS)

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, two e-Commerce C2C prototypes have been developed, which have been built on top of an e-Commerce framework, exploiting technologies and components developed within the technical work packages of SOA4All. The first prototype was available at M24 and demonstrated the innovative functionalities of allowing product resellers to combine the offers of multiple vendors into a single set of products. These products could then be exposed via different export services to multiple channels, including various Web 2.0 platforms, such as Facebook or Twitter.

The second prototype, which is described in this deliverable, further realizes the idea of the “One Stop Cloud Shop” vision. The different services, which support the various e-Commerce operations such as unifying product catalogues and exporting the product data, have been enhanced and updated, based on the feedback gathered by evaluating the first iteration of the e-Commerce prototype. In addition, new services have been included to support the e-Commerce process and enable a full integration with the SOA4All platform. Both the updated services and the completely new functionalities are described in this deliverable in detail.

In addition to the new and updated functionalities of the e-Commerce services, the main focus of the second version of the prototype was the complete integration with SOA4All components and the SOA4All infrastructure, replacing parts that were hardcoded in the previous version. Also, all services have been updated with the necessary annotations and lifting/lowering rules to enable this integration.

The final evaluation and validation of the use case objectives will be reported in a separate deliverable, D9.4.2, which is due the end of the project.

2. Introduction

2.1 Purpose and Scope of the deliverable

Deliverable D9.4.1 represents the second implementation of the C2C e-Commerce Prototype. The purpose of this document is to provide a technical description of the prototype, as released at M33. 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 to demonstrate 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 in 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 work packages 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. The descriptions and installation instructions provided in this document expand, enhance and supersede the previous technical documentation for the C2C e-Commerce prototype, provided in D9.3.1.

This deliverable reflects the improvements and extensions which were designed and implemented for the second version of this prototype. These developments were based on the feedback gathered while testing and evaluating the first prototype.

A final evaluation report, which will include a validation of the Use Case’s objectives, 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 in D9.2.1, 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. It reflects the final state of the e-Commerce prototype, documenting the changes after M24.
- **Section 5** provides a more **technical description** of the prototype’s components, as implemented in the prototype’s final version.
- **Section 6** provides prototype **installation** instructions.
- Finally, **Section 7** provides the conclusions of this deliverable.

3. C2C e-Commerce 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 and D9.3.1.

Overall, the SOA4All e-Commerce use case allows users to access a range of different modules and developments of both, the SOA4All project results in general and the WP9 use case implementations in specific:

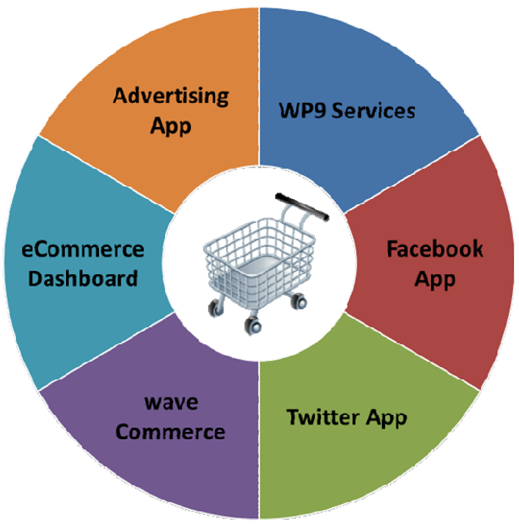
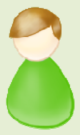


Figure 1: The SOA4All e-Commerce Use Case

3.1 Roles

The scenario involves several people as described in D9.2.1. In D9.3.1, a new role has been added: Silvio, who wants to use the SOA4All services to increase the number of visitors of his website. This updated deliverable D9.4.1 summarizes the roles and provides a recapitulation of the scenario.

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.

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

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

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 would not be as flexible, any more as things tend to be highly connected.

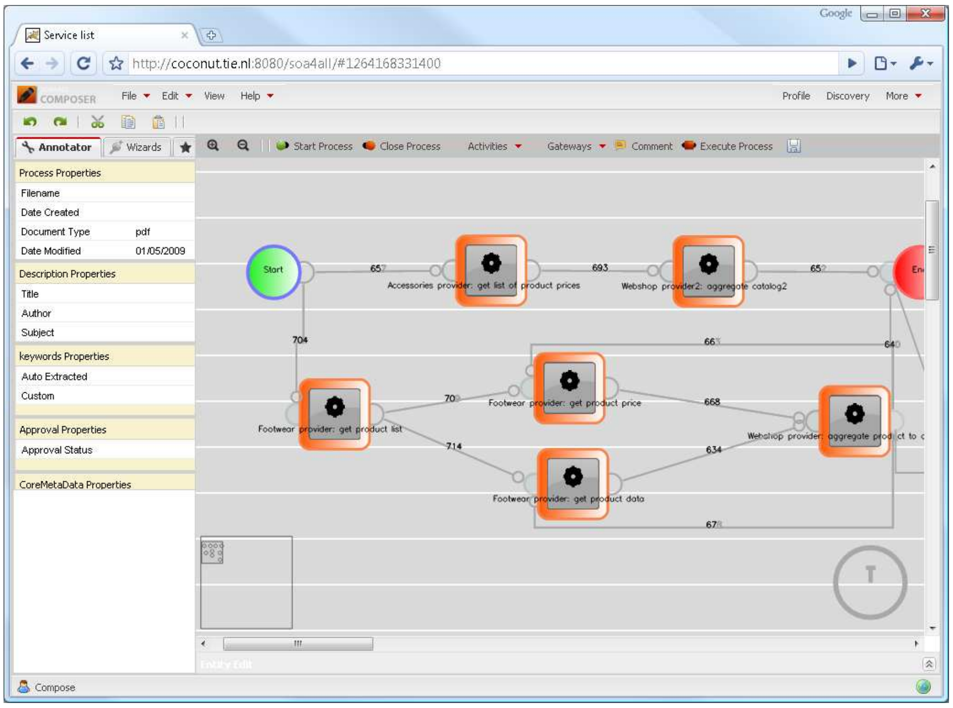
Nada chats with some friends about her problem and one of them recommends SOA4All identifying the following benefits to her:

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

- o 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.

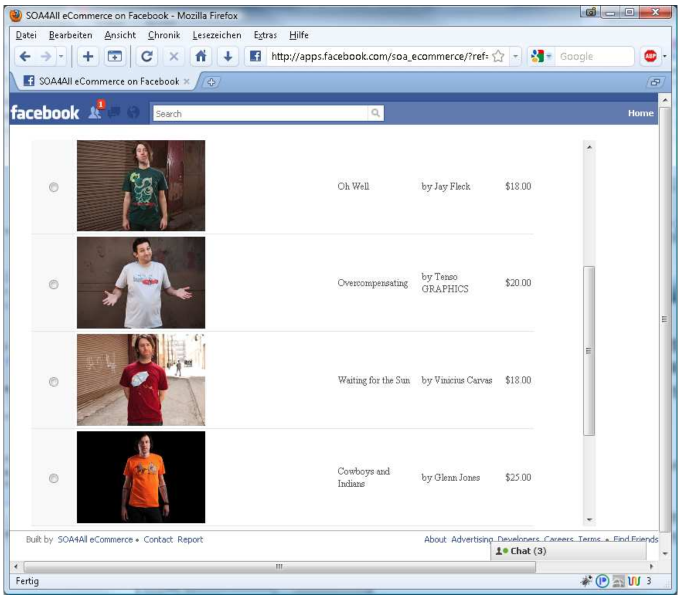
She then starts searching for suitable services using the SOA4All Discovery Platform and finds many services related to products. She uses the SOA4All process editor to create her own personalized process based on those services:


Service list
http://coconut.tie.nl:8080/soa4all/#1264168331400
COMPOSER
File
Edit
View
Help
Profile
Discovery
More
Annotator
Wizards
Start Process
Close Process
Activities
Gateways
Comment
Execute Process
Start
Accessories provider: get list of product prices
Webshop provider2: aggregate catalog2
Footwear provider: get product list
Footwear provider: get product price
Footwear provider: get product data
Webshop provider: aggregate product to c
Compose

Process Properties	
Filename	
Date Created	
Document Type	pdf
Date Modified	01/05/2009
Description Properties	
Title	
Author	
Subject	
keywords Properties	
Auto Extracted	
Custom	
Approval Properties	
Approval Status	
CoreMetaData Properties	

Figure 2: The D2.4.1 Dashboard showing parts of the WP9 scenario

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.


SOA4All eCommerce on Facebook - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://apps.facebook.com/soa_ecommerce/?ref=
Google
SOA4All eCommerce on Facebook
facebook
Search
Home
Built by SOA4All eCommerce • Contact Report
About Advertising Developers Careers Terms • Find Friends
Chat (3)
Fertig

Oh Well	by Jay Fleck	$18.00
Overcompensating	by Tenso GRAPHICS	$20.00
Waiting for the Sun	by Vinicius Carvas	$18.00
Cowboys and Indians	by Glenn Jones	$25.00

Figure 3: 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 suspicion 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 RESTful 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 RESTful 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 Final Prototype: section 4.1 describes all the components that are forming the prototype, classifying them into three main categories. Section 4.2 describes the relationship with the SOA4All architecture, and which component is accessed by the WP9 scenario actors.

4.1 Prototype Components

The WP9 Final Prototype is based on both SOA4All components and the "WP9 e-Commerce 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:

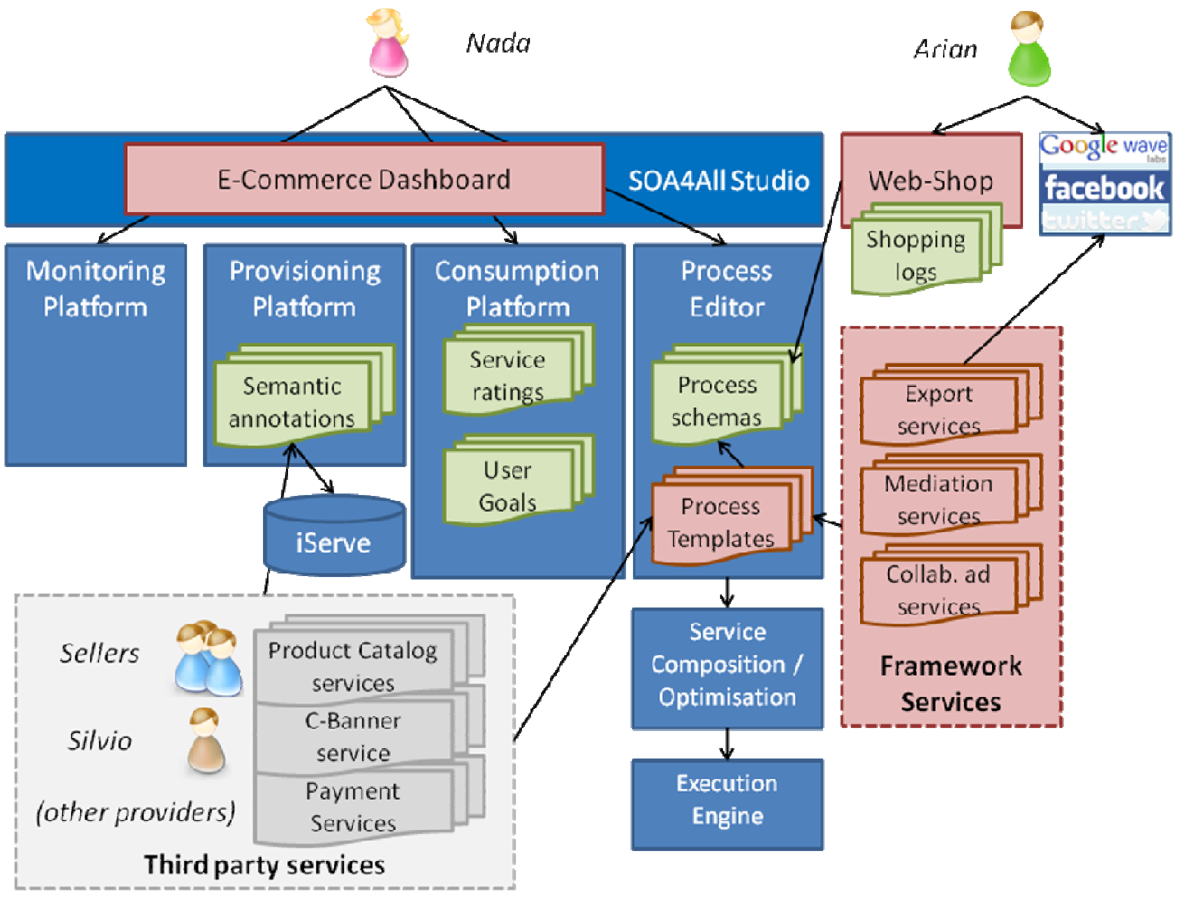


Figure 4: Graphical Representation of the Prototype components

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:
   - **e-Commerce Dashboard:** this dashboard (GUI) allows a quick access to typical e-Commerce services (e.g. payment services) and to related parts of the SOA4All infrastructure. At the same time it also provides an example for end users showing

them how they can integrate own modules into the e-Commerce dashboard by using the e-Commerce code as an example.

- **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, Twitter and Google Wave.
- **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 Final Prototype includes the following services provided by Sellers and Advertisers (light-grey colour in Figure 3):
    a. **Product Services** (both as WSDL and RESTful), provided by the Sellers (Theodore, Esteban, Claus).
    b. **Payment Services**, exploiting existing payment services.
    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 RESTful 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 processes 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.

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 and Advertiser (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 RESTful) or Banner service. Semantic annotations are then stored in the Semantic Space.

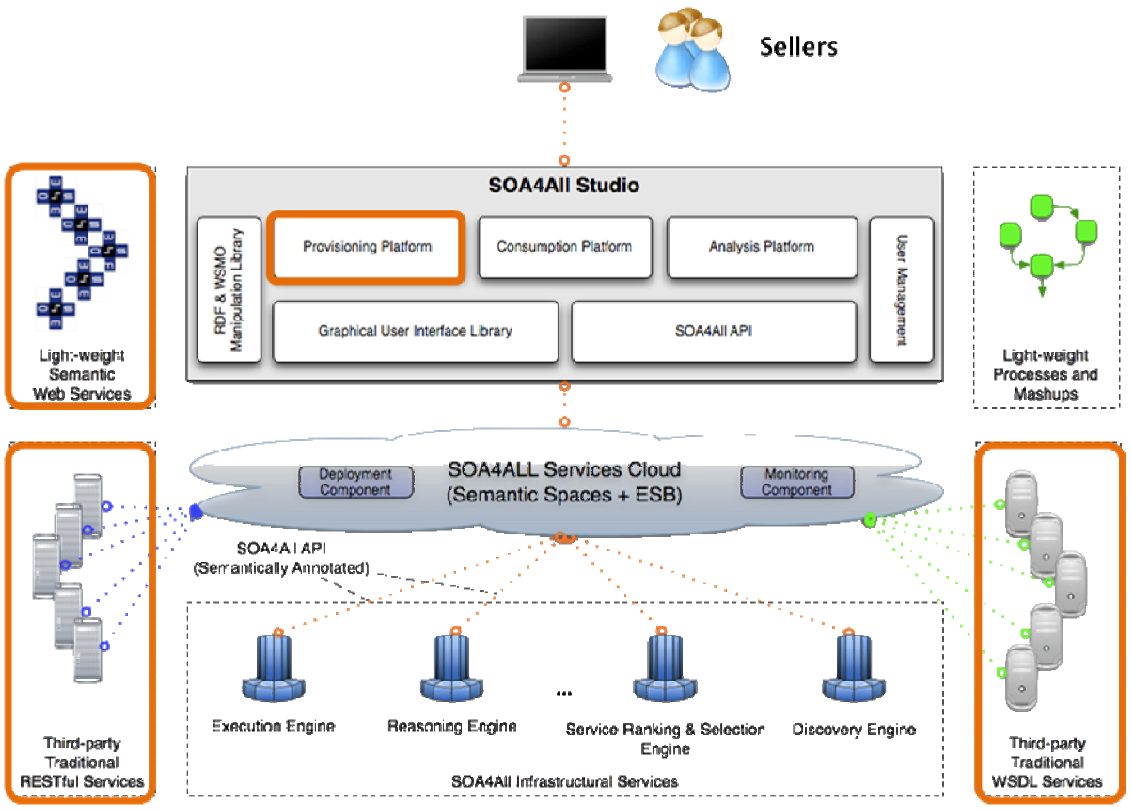

Sellers
SOA4All Studio
RDF & WSMO Manipulation Library
Provisioning Platform
Consumption Platform
Analysis Platform
User Management
Graphical User Interface Library
SOA4All API
Light-weight Semantic Web Services
Light-weight Processes and Mashups
Deployment Component
SOA4ALL Services Cloud (Semantic Spaces + ESB)
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 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 addition, she is working with the **Process Editor** in order to compose the required e-commerce processes, and finally she is accessing the **Analysis Platform** in order to monitor her web-shop activities. It should be noted that each of these front-end components uses SOA4All Infrastructural Services (i.e. Discovery, Execution, etc…), as described in deliverable D1.4.2A.

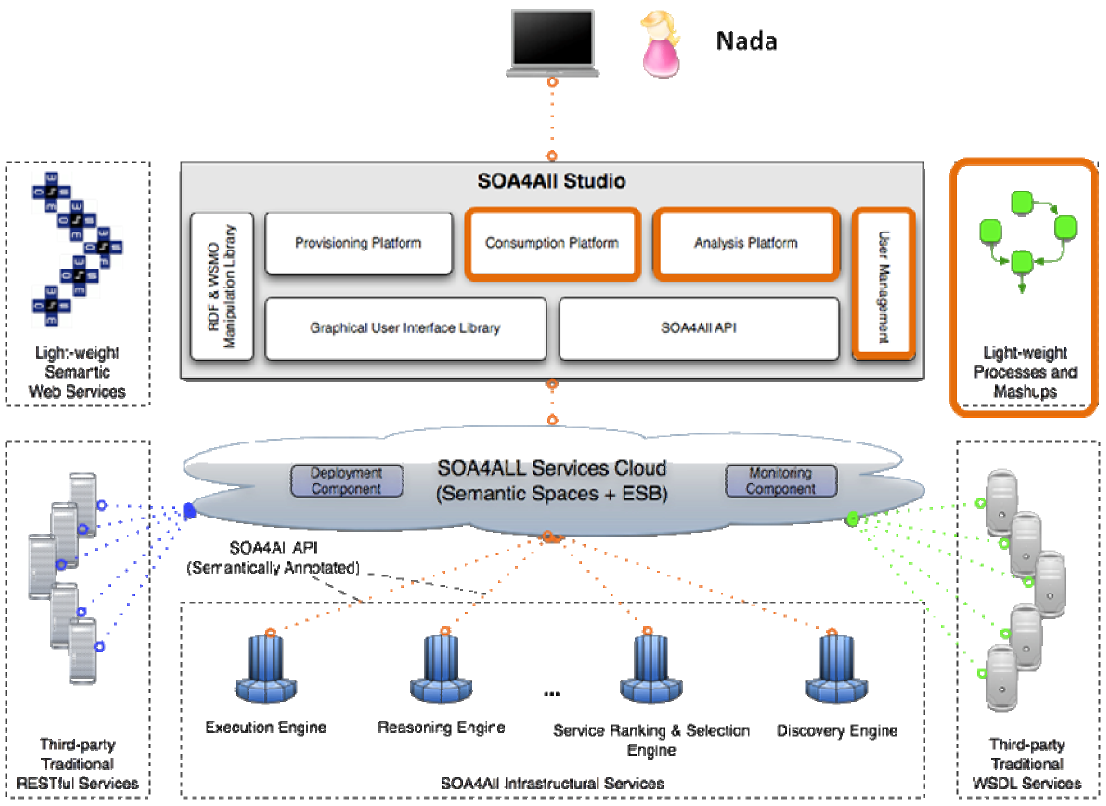

Nada
SOA4All Studio
RDF & WSMO Manipulation Library
Provisioning Platform
Consumption Platform
Analysis Platform
User Management
Graphical User Interface Library
SOA4All API
Light-weight Semantic Web Services
Light-weight Processes and Mashups
SOA4ALL Services Cloud (Semantic Spaces + ESB)
Deployment 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 6: 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.

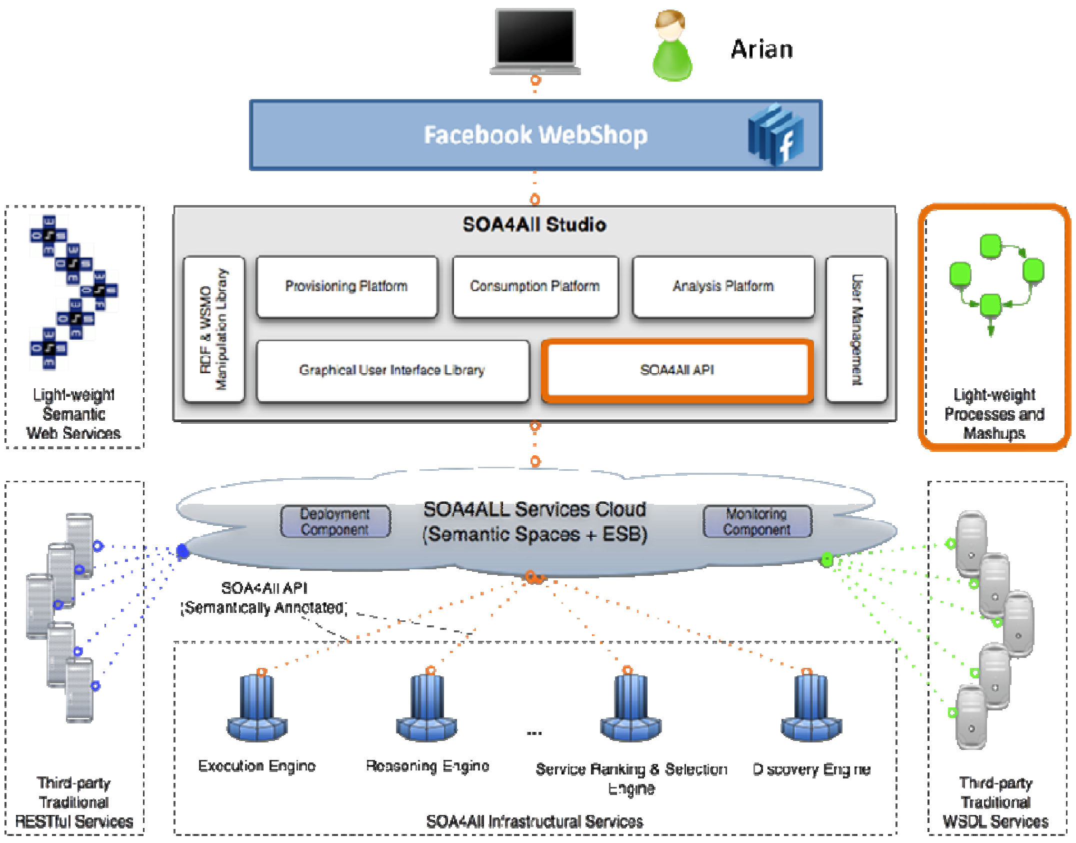

Arian
Facebook WebShop
SOA4All Studio
RDF & WSMO Manipulation Library
Provisioning Platform
Consumption Platform
Analysis Platform
User Management
Graphical User Interface Library
SOA4All API
Light-weight Semantic Web Services
Light-weight Processes and Mashups
SOA4ALL Services Cloud (Semantic Spaces + 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 7: Relationships with SOA4All architecture for Arian

5. Prototype Technical Implementation

This section summarises the final 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 are explained in the subsections concerning the concrete service.

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. 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 services 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 services to transparently merge all these characteristics into a single product definition.

Three mediation services are provided for coping with this task. The first one, the Aggregation Service (described in the C2C s-Commerce Prototype v1) transparently invokes the services define by the three service providers, offering the transparent invocation off all the operations allowed by the services, and returning the composition of the individual invocation results. For a detailed description of this service please see [1].

The other two services, Temporal Repository and Query services, allow a provider to store an entire list of products in a third-party database, so that the products can be accessed directly from this database.

5.1.1.1 Database

The database used for storing the products consists of only one table, *mainProduct*, with the following fields:

Table 1: Main Product

Element name	Element type
id	**char(30) character**
name	**text character set**
price	**float**

orderURL	**text character set**
imageURL	**text character set**
description	**text character set**
details	**text character set**
gender	**text character set**
size	**text character set**
specialOffer	**tinyint(4) default '0'**

The primary key is the **id** of a product, considering that no two products are defined using the same id. The information regarding the service that originally provided a product is not maintained, as there is no need to invoke the services once their products are stored in the database.

5.1.2 The Temporal Repository Service

The Temporal Repository service provides the functionality of storing a list of products. It provides mediation functionality to cope with the different formats used by the providers, as listed in the following table:

Table 2: 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 service is available at http://coconut.tie.nl:8181/services/Temprep and provides the operation ***storeProductList***, with a list of products as parameter. The id of every product from the list is compared with the ids of the product already stored in the repository. If the product with the same id is already in the database, the operation returns a message that informs the user that the product was previously added. If the id of a product is not recognized, the product is added and the user informed about this.
The service's WSDL is available at: http://coconut.tie.nl:8181/services/Temprep?wsdl

5.1.3 The Query Service

This service queries the database populated using the Temporal Repository service. It is available at http://coconut.tie.nl:8181/services/Query, and implements two operations; ***getProductList()*** and ***getProductWithId(String Id)***. The ***getProductList i***s invoked without parameters and returns a list containing all the products from the database. The ***getProductWithId*** operation is invoked with a product id as parameter, and returns the details of the product having that id. If there is no product with the specified id the user receives an error message.

5.1.4 Collaborative Advertising Service

5.1.4.1 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 RESTful 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

5.1.4.2 Resources

Collaborative Advertising Services have been implemented as RESTful 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

5.1.4.3 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)

5.1.4.4 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 expose her product data, offers and other information on 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 500 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. Facebook was selected, because its broad user scope will guarantee a high impact of the SOA4All WP9 prototype and also because it fits well to the scenario and with the prototype implementations.

The Facebook application designed and implemented in the scope of WP9 allows Nada 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[3].
- **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 application itself. For the next version of the WP9 e-Commerce prototype, this information will be transformed by the Facebook application into an ontological representation of user context and then sent to the WP9 e-Commerce 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[4]. 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_e-Commerce/.

In the final version of the prototype described in this deliverable, the Facebook application has been updated to work with the new version of the WP9 e-Commerce process, which was deployed on the SOA4All infrastructure and is executed by its own execution engine. The new process utilizes the support services described in Section 5.1. As the new process is now fully integrated with the SOA4All infrastructure, as an additional benefit, Nada can now monitor the involved services, and can notice problems, such as unavailable vendor services. In such a case, the process still would not fail, but the currently exposed products in the Facebook application will not feature the faulty service’s offers.

5.2.2 Twitter

The Twitter micro-blogging service has been integrated with the current version of Nada’s e-

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

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

Commerce process, as first defined for the M24 prototype. After the product data has been aggregated from several product suppliers, and the different data formats have been aligned, 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.

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 trigger 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 products 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 appearing, which need to be tweeted within 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 Execution 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 shortened, 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 RESTful 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 about which it should be twittered
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[5] 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

For the M33 prototype, the main change was that a new process was deployed, which is executed by the SOA4All infrastructure. Also, the different support services described in

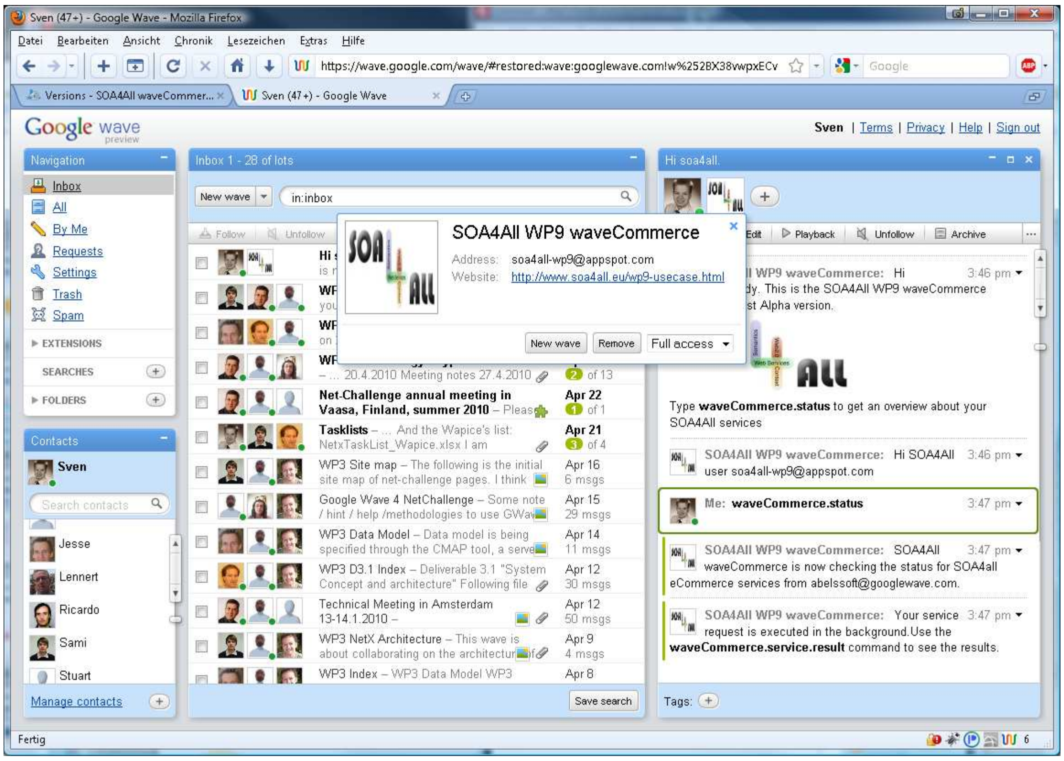
[5] Available at http://yusuke.homeip.net/twitter4j/en/index.html

Section 5.1 have been utilized. In addition, the authentication process to post tweets to Nada's Twitter account has been changed. This reflects the changes that happened to the Twitter API since the M24 version of the prototype was developed. In the new version, OAuth is used as the single allowed authentication method.

5.2.3 Google Wave

A Google Wave[6] integration has been developed by the SOA4All team. This integration was achieved by implementing a new product on top of Google's Cloud service AppEngine. This product directly accesses the SOA4All infrastructure and links parts of it to Google Wave, therefore making it usable by everyone.

In month 27 of the project, Google announced to stop actively developing its Google Wave platform and to release it under open source. Although the open source release will ensure that Google Wave is still available, the consortium assumes that the popularity will drop, as Google does not actively push the platform. Because of this, the development of the SOA4All Google Wave was stopped in month 27 allowing WP9 to focus on the other developments. Nevertheless, the Google Ware adapter is already fully available as a project result (with a limited feature set) and could be used by users at any time.



SOA4All WP9 waveCommerce

Address: soa4all-wp9@appspot.com

Website: http://www.soa4all.eu/wp9-usecase.html

New wave | Remove | Full access

Hi … is r…		
WP… you…		
WP… on …		
WP… – … 20.4.2010 Meeting notes 27.4.2010		2 of 13
Net-Challenge annual meeting in Vaasa, Finland, summer 2010 – Pleas	**Apr 22**	1 of 1
Tasklists – … And the Wapice's list: NetxTaskList_Wapice.xlsx I am	**Apr 21**	3 of 4
WP3 Site map – The following is the initial site map of net-challenge pages. I think	Apr 16	6 msgs
Google Wave 4 NetChallenge – Some note / hint / help /methodologies to use GWav	Apr 15	29 msgs
WP3 Data Model – Data model is being specified through the CMAP tool, a serve	Apr 14	11 msgs
WP3 D3.1 Index – Deliverable 3.1 "System Concept and architecture" Following file	Apr 12	30 msgs
Technical Meeting in Amsterdam 13-14.1.2010 –	Apr 12	50 msgs
WP3 NetX Architecture – This wave is about collaborating on the architectur…f	Apr 9	4 msgs
WP3 Index – WP3 Data Model WP3	Apr 8	

…ll WP9 waveCommerce: Hi …dy. This is the SOA4All WP9 waveCommerce …st Alpha version. 3:46 pm

Type **waveCommerce.status** to get an overview about your SOA4All services

SOA4All WP9 waveCommerce: Hi SOA4All user soa4all-wp9@appspot.com 3:46 pm

Me: **waveCommerce.status** 3:47 pm

SOA4All WP9 waveCommerce: SOA4All waveCommerce is now checking the status for SOA4all eCommerce services from abelssoft@googlewave.com. 3:47 pm

SOA4All WP9 waveCommerce: Your service request is executed in the background.Use the **waveCommerce.service.result** command to see the results. 3:47 pm

Figure 8: SOA4All Google Wave Integration

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

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.

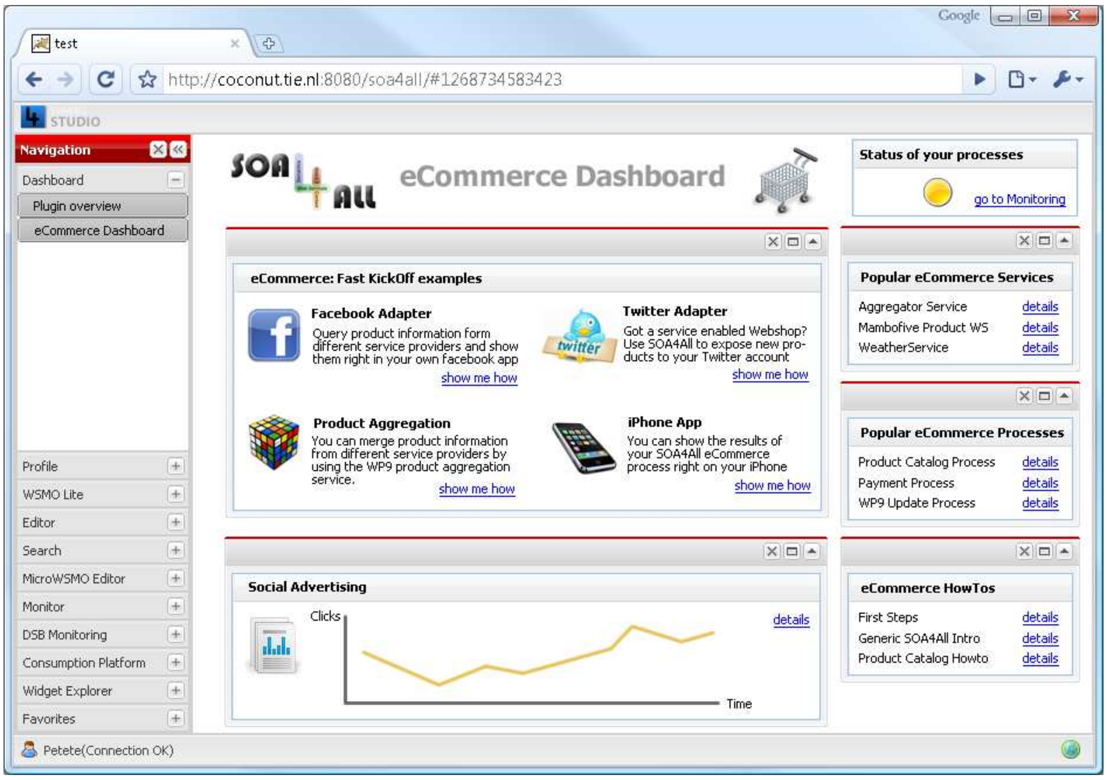
5.3.1 Mambofive

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.

For demonstrating the usability of such a real world webshop, demonstration products and databases have been developed and shown in various events such as the ICT 2010 conference. This has allowed potential e-Commerce users to see the applicability of the research results in production environments, which makes the results much more tangible and attractive for non-research and for non-technical people.

5.4 e-Commerce Dashboard

For making the access of users to SOA4All in e-Commerce scenarios easier, the WP9 team has implemented an e-Commerce dashboard. This dashboard allows a quick access to typical e-Commerce services (e.g. payment services) and to related parts of the SOA4All infrastructure. At the same time it also provides an example for end users showing them how they can integrate their own modules into the e-Commerce dashboard by using the e-Commerce code as an example.



eCommerce: Fast KickOff examples

Facebook Adapter
Query product information form different service providers and show them right in your own facebook app
show me how

Twitter Adapter
Got a service enabled Webshop? Use SOA4All to expose new products to your Twitter account
show me how

Product Aggregation
You can merge product information from different service providers by using the WP9 product aggregation service.
show me how

iPhone App
You can show the results of your SOA4All eCommerce process right on your iPhone
show me how

Popular eCommerce Services	
Aggregator Service	details
Mambofive Product WS	details
WeatherService	details

Popular eCommerce Processes	
Product Catalog Process	details
Payment Process	details
WP9 Update Process	details

eCommerce HowTos	
First Steps	details
Generic SOA4All Intro	details
Product Catalog Howto	details

Figure 9: WP9 e-Commerce Dashboard

6. 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:

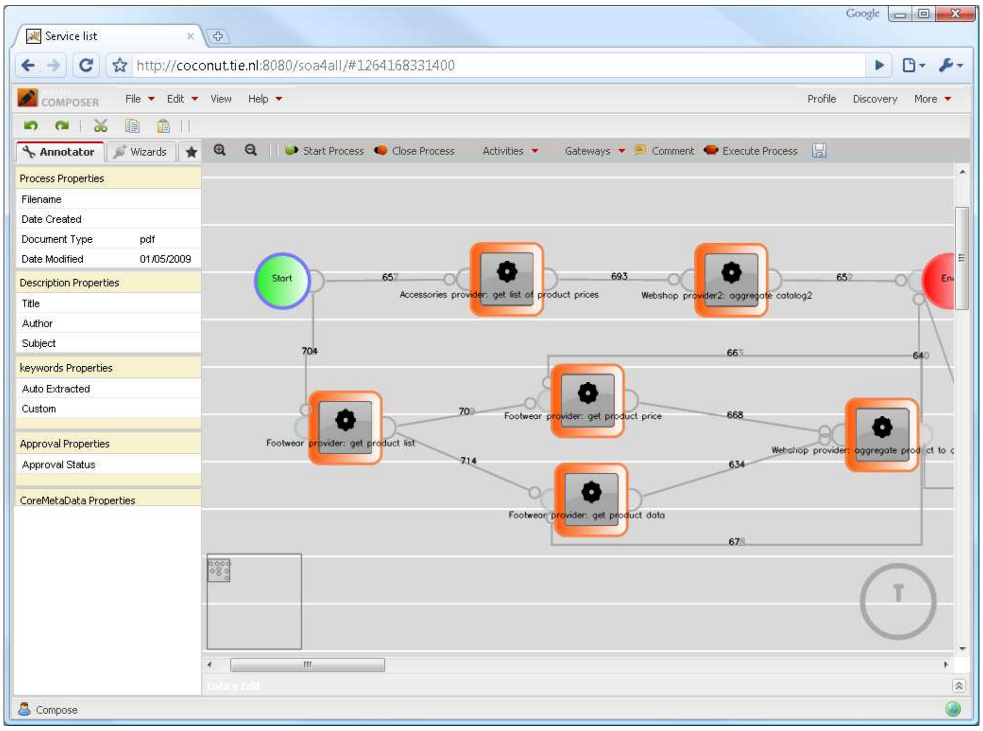
http://localhost:8080/soa4all-dashboard

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

As SOA4All is based on a decentralized approach, many modules including the WP9 e-Commerce dashboard are loaded from remote addresses. If you want to install the e-Commerce dashboard locally, then you need to download a second WAR file and place it in the tomcat folder. You can download the e-CommerceDashboard.war at

http://coconut.tie.nl:8080/hudson

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


Service list
http://coconut.tie.nl:8080/soa4all/#1264168331400
COMPOSER
File
Edit
View
Help
Profile
Discovery
More
Annotator
Wizards
Start Process
Close Process
Activities
Gateways
Comment
Execute Process
Start
Accessories provider: get list of product prices
Webshop provider2: aggregate catalog2
Footwear provider: get product list
Footwear provider: get product price
Footwear provider: get product data
Webshop provider: aggregate product to c
657
693
652
704
663
640
70
668
714
634
678
Compose

Process Properties	
Filename	
Date Created	
Document Type	pdf
Date Modified	01/05/2009
Description Properties	
Title	
Author	
Subject	
keywords Properties	
Auto Extracted	
Custom	
Approval Properties	
Approval Status	
CoreMetaData Properties	

Figure 10: The D2.4.1 Dashboard showing parts of the WP9 scenario

7. Conclusions

This deliverable describes the implementation of the final WP9 e-Commerce Prototype, based on the scenario and on the infrastructure designed described initially in D9.2.1. It documents the changes and updates developed after the release of the first prototype in M24, and includes changes to the services, APIs, as well as completely new functionalities to better support the different e-Commerce processes and to integrate with the SOA4All infrastructure.

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 descriptions on the prototype architecture and its implementation. The evaluation of the prototype will be conducted in the last phase of the project. The respective evaluation and validation results will then be reported in the final deliverable of Work Package 9, D9.4.2, due M36.

8. 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 e-Commerce 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. S. Abels, B. Schreder, M. Villa, M. Zaremba, H. Sheikhhasan, S. Puram: SOA4All deliverable D9.2.1. e-Commerce Framework Infrastructure Design, August 2009.
6. M. Villa, G. Di Matteo, S. Abels, S. Puram, B. Schreder, M. Zaremba, E. Cimpian: SOA4All Deliverable D9.3.1. *C2C e-Commerce Prototype v1*, February 2010.