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

D2.2.2 – Service Consumption Platform First Prototype

Activity N:	Activity 1 – Fundamental & Integration activities	
Work Package:	WP2 – SOA4All Studio	
Due Date:		M18
Submission Date:		03/09/2009
Start Date of Project:		01/03/2008
Duration of Project:		36 Months
Organisation Responsible of Deliverable:		ISOCO
Revision:		1.0
Author(s):	Guillermo Álvaro Rey Iván Martínez Matteo Villa Giovanni Di Matteo	ISOCO ISOCO TXT TXT
Internal Reviewers:	Freddy Lecue John Davies	UNIMAN BT

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
PU	Public	**X**

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	06/07/2009	Document Initialized	Guillermo Álvaro Rey
0.2	20/07/2009	Included RS and UI widget integration paragraphs	Giovanni Di Matteo, Matteo Villa
0.3	24/07/2009	Completed first version	Guillermo Álvaro Rey
0.4	28/07/2009	Refined contents.	All
0.5	29/07/2009	Version for internal reviewers	All
0.6	31/07/2009	Internal Review	Freddy Lecue
0.7	04/08/2009	Internal Review	John Davies
0.8	17/08/2009	Addressed internal review comments	Guillermo Álvaro Rey
0.9	31/08/2009	Final modifications	All
1.0	03/09/2009	Minor edits and inclusion of draft paper #2	Guillermo Álvaro Rey

Table of Contents

List of Figures

Glossary of Acronyms

Acronym	Definition
D	Deliverable
DRM	Digital Rights Management
EC	European Commission
GUI	Graphical User Interface
GWT	Google Web Toolkit
GXT	Ext-GWT
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IPR	Intellectual Property Rights
ISP	Internet Service Provider
KPI	Key Performance Indicator
N3	N-Triple
NLP	Natural Language Processing
POS	Part-of-speech
QoS	Quality of Service
RIA	Rich Internet Application
RDF	Resource Definition Framework
RDF(S)	RDF Schema
REST	Representational State Transfer
RS	Recommender System
SA-REST	Semantic Annotations for RESTful Services
SAWSDL	Semantic Annotations for WSDL and XML Schema
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SWS	Semantic Web Service
T	Task
UI	User Interface
UNSPSC	United Nations Standard Products and Services Code
W3C	World Wide Web Consortium
WP	Work Package
WS	Web Service
WSDL	Web Services Description Language
WSML	Web Service Modeling Language

Acronym	Definition
WSMO	Web Service Modeling Ontology
WWW	World Wide Web
XML	eXtensible Markup Language

Executive summary

This deliverable describes the First Prototype of the Service Consumption Platform, the part of the SOA4All Studio where end-users are able to consume services, discovering the most suitable ones for them in a lightweight manner.

The document begins with a description of the platform, mentioning what is already implemented and can be used. Specifically, the platform is described as a “personalized homepage” for the consumption of semantically enriched services, with a portal layout where different services can be opened and consumed in separate “portlets”[1] (i.e., non-overlapping windows containing different user-interface components), and a lefthand panel that acts as a starting point for the consumption process in different ways. Some general implementation details and installation guidelines are also covered.

The most important issues and challenges regarding the automation of service consumption from semantic service descriptions are discussed, also addressing the problem of the heterogeneity of authentication solutions. Another important aspect of the platform is its adaptability to satisfy different user needs, and we explain how we tackle this issue by making use of the Recommendation System.

Finally, special emphasis is placed on the fact that the platform is not a standalone component, but an interconnected piece inside the SOA4All Studio, which uses the functionalities of many other architectural components of the project. Thus, the document also highlights the integration of the platform with other components of the project.

[1] http://en.wikipedia.org/wiki/Portlet

1. Introduction

The Service Consumption Platform, first described in D2.2.1 [1], is the part of the SOA4All Studio [2] where users can interact with the service world as service consumers, discovering the most suitable ones for their purposes in a lightweight manner.

Based in Ext-GWT[2], as the rest of the SOA4All Studio, the platform presents itself via a portal layout[3] where users can open several services, or sets of services (and in the future, goals) and interact with those in a lightweight manner in a configurable way reminiscent of Web2.0 sites such as iGoogle[4] or NetVibes[5].

Services can be consumed in this platform as long as they have been semantically enriched within the Service Provisioning Platform (WSDL services with the WSMO-Lite Editor, and RESTful services with SWEET [3][4]). We have placed more emphasis in the consumption of the latter, but we are at the same time working on a generic way of consuming WSDL services too. In the future, the processes composed within the Process Editor [5] will also be consumable in this platform.

Enabling this kind of consumption of services at the conceptual level (i.e., from the semantic annotations attached to their syntactic definitions) requires moving from that layer to the execution one underneath, as well to a presentation layer above. Suitable input and output adapters and lowering (from semantic to syntactic level) and lifting (from syntactic to semantic level) mechanisms have to take place, in an automatic and transparent manner, from the semantic annotations. This document explains how the Consumption Platform is tackling these issues (while we refer to the first short paper included in Annex A for further information) in order to allow users to easily consume services.

Another challenge we face within this platform has to do with authentication issues, for many of the existing service providers require the implementation of heterogeneous authentication and/or authorization mechanisms. So far, we have implemented the most direct and straightforward ones, while sketching the big picture of the problem, in order to be able to solve it properly. We refer to the second short paper in Annex A for the work in progress in this respect.

We also explain an important characteristic of the platform, which is the stress placed on personalisation, as it adapts itself to satisfy better the needs of each user. In order to achieve this, the platform makes the necessary calls to the Recommendation System (RS, [6]), which is able to recommend items for a user, given his past usage of the platform, and his current context.

The interaction with the RS shows that the Consumption Platform is not a component on its own, but a piece of the whole SOA4All Studio which is interconnected with other ones. We analyse in which way and via which calls these interactions take place.

In the rest of the document, we describe the general characteristics of the platform in Section 2, discussing how we are tackling the important issues in Section 3, and addressing the integration of the platform with other components in Section 4. Section 5 discusses future work and challenges, and concludes the deliverable.

[2] http://extjs.com/products/gxt/

[3] http://extjs.com/examples/pages/portal/portal.html

[4] http://www.google.com/ig

[5] http://www.netvibes.com/

2. Functional Specification of the Service Consumption Platform

2.1 Description of the platform

The Service Consumption Platform[6] has been designed to support the interaction with many services in different ways, with a portal-style visualization layout. This is in line with Web 2.0 “personalized homepages” such as iGoogle, NetVibes or My Yahoo![7], where the end-users can interact with the content they want and organize it in the way that suits them most. The Service Consumption Platform aims at being one of these *startpages*, only that for the consumption of semantically enriched services instead of content in the form mainly of feeds.

With this approach, the platform contains a left panel that allows users to open services in a number of different ways. These services, and sets of services, are opened in the main dashboard view, where several of them can coexist organized in three different columns, as depicted in Figure 1.

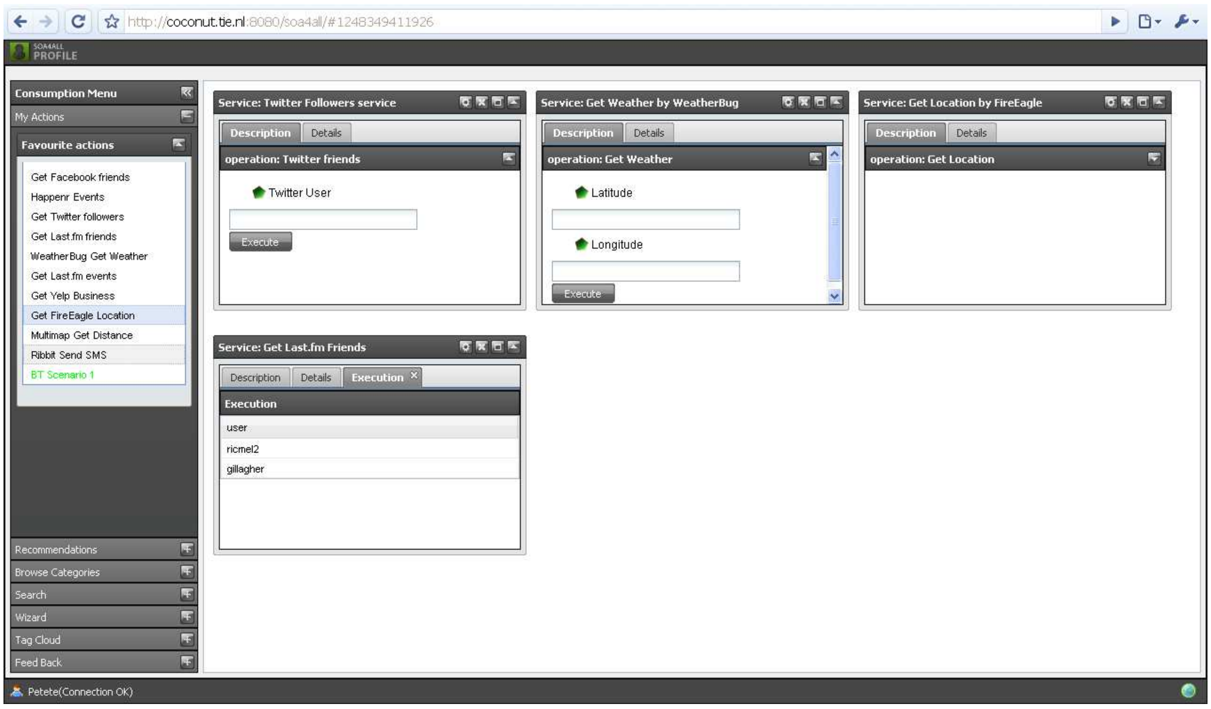


Figure 1: Consumption Platform: Left panel and dashboard area with some services

Following the portal kind of layout, individual “portlets” can be maximised to occupy the whole dashboard area (and conversely minimised to their initial size), as well as contracted to occupy only the header (and conversely expanded to their initial size), and obviously closed when they are not going to be used anymore. Figure 2 shows some examples of buttons to perform these actions. It is worth mentioning that some extensions have been developed to the Ext-GWT `Portal` class into an `AdaptivePortal` class, in order to support the desired extra features. This extended class is conveniently shared in the SOA4All Studio in order to allow other client modules (such as the Analysis one) to make use of it, and also to mutually benefit from the possible improvements (e.g., enabling an easy addition of extra columns).

[6] Available online at http://coconut.tie.nl:8080/soa4all/

[7] http://my.yahoo.com/

Figure 2: Examples of portlet buttons: Close, maximise, contract; Close, expand, minimise

The lefthand panel, depicted in Figure 3, contains several ways to allow end-users to find the services they need. It is divided into the following areas:

- "My Actions": Usual actions of the user and bookmarked ones will be displayed in this section, allowing them to be re-opened when required.
- "Recommendations": The platform will recommend (through the Recommendation System) items to the user in a proactive way, as we will see in Section 3.2.
- "Browse Categories": A hierarchical way of navigating through a taxonomy, which is at this stage fixed.
- "Search": A direct way of finding services with a keyword based query, making use of the underlying Discovery services.
- "Wizard": It will guide the steps of the users to simplify service consumption. (Not available yet.)
- "Tag Cloud": Finding services through selecting popular tags used to describe them displayed in a tag cloud format.

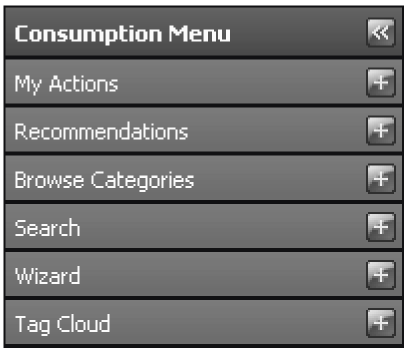
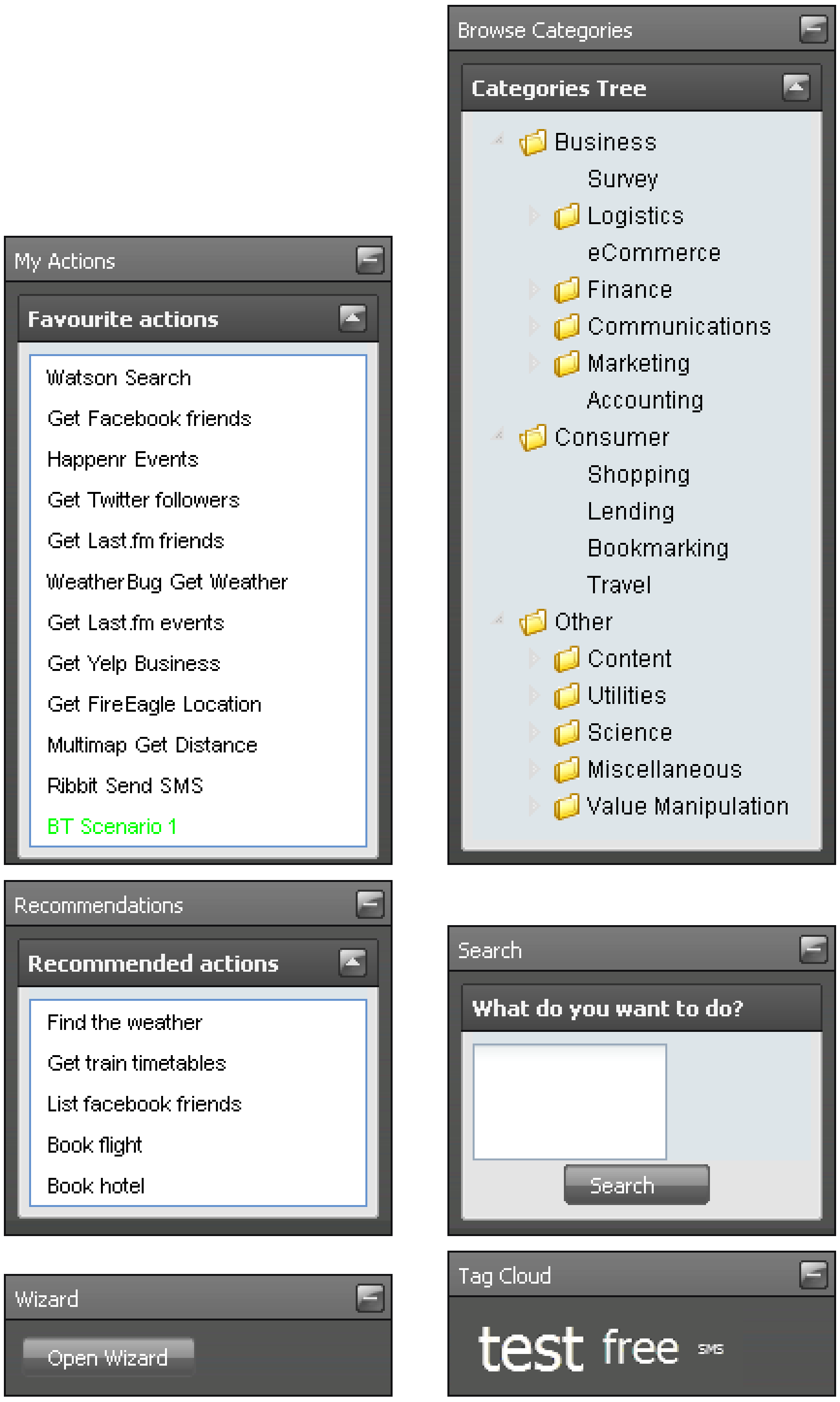


Figure 3: Left Panel

For further illustration, examples of the six aforementioned widgets that belong to the lefthand panel are depicted in Figure 4 in their expanded view.



My Actions

Favourite actions

Watson Search
Get Facebook friends
Happenr Events
Get Twitter followers
Get Last.fm friends
WeatherBug Get Weather
Get Last.fm events
Get Yelp Business
Get FireEagle Location
Multimap Get Distance
Ribbit Send SMS
BT Scenario 1

Browse Categories

Categories Tree

- Business
  - Survey
  - Logistics
  - eCommerce
  - Finance
  - Communications
  - Marketing
  - Accounting
- Consumer
  - Shopping
  - Lending
  - Bookmarking
  - Travel
- Other
  - Content
  - Utilities
  - Science
  - Miscellaneous
  - Value Manipulation

Recommendations

Recommended actions

Find the weather
Get train timetables
List facebook friends
Book flight
Book hotel

Search

What do you want to do?

Search

Wizard

Open Wizard

Tag Cloud

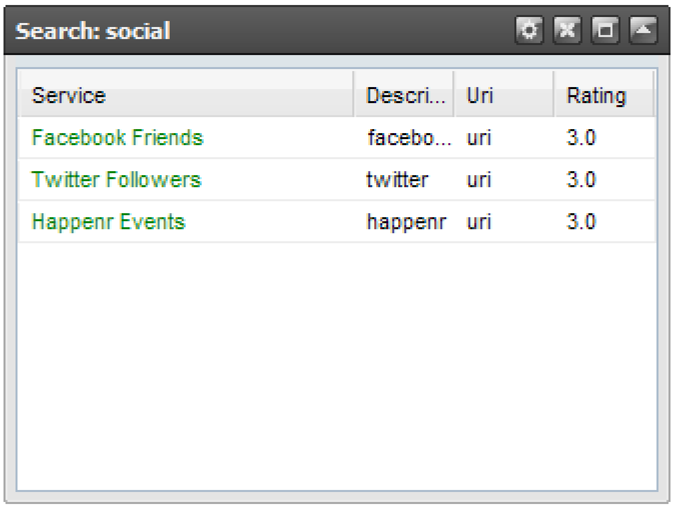
test free SMS

Figure 4: The six widgets in the lefthand panel

Obtaining a list of results

Some of the widgets in the lefthand panel allow the user to directly open a service for invocation. For instance, the “My Actions” and “Recommendations” widgets will contain services that can be opened with just one click.

Additionally, in some cases the widgets in the left panel result in a list of services, which are displayed in a portlet within the dashboard area. This is the case of the search results and selections through categories or tags. In this case, this intermediate step helps users choose the most suitable service from a list such as the one in Figure 5.



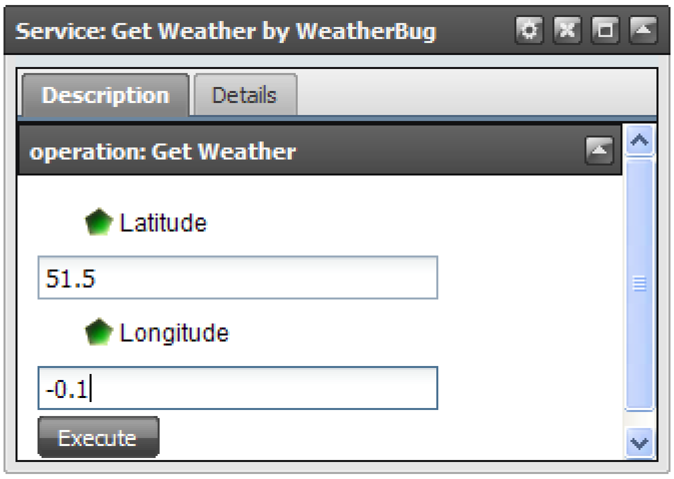
Search: social

Service	Descri...	Uri	Rating
Facebook Friends	facebo...	uri	3.0
Twitter Followers	twitter	uri	3.0
Happenr Events	happenr	uri	3.0

Figure 5: Search results showing a list of suitable services

Consuming a Service

Finally, a service can be consumed. In order to do so, when the service is selected, either from one of the options in the lefthand panel or from a list of results, a new portlet is opened. This portlet contains some information about the service, and gives the user the ability to invoke it by filling the necessary information. Section 3 gives more detail about the process of creating the forms and processing the user inputs from the service descriptions. Figure 6 depicts an example of interface presented to the user for a weather service, with a Latitude and a Longitude values to be filled in order to successfully interact with the service.



Service: Get Weather by WeatherBug

Description | Details

operation: Get Weather

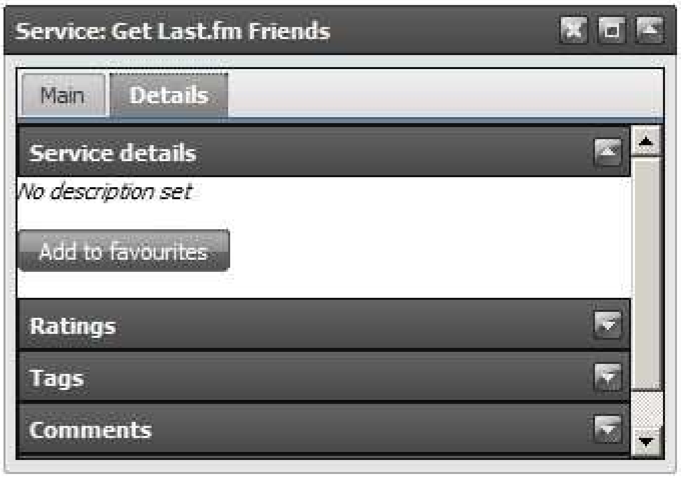
Latitude: 51.5

Longitude: -0.1

Execute

Figure 6: Service GUI for inputs

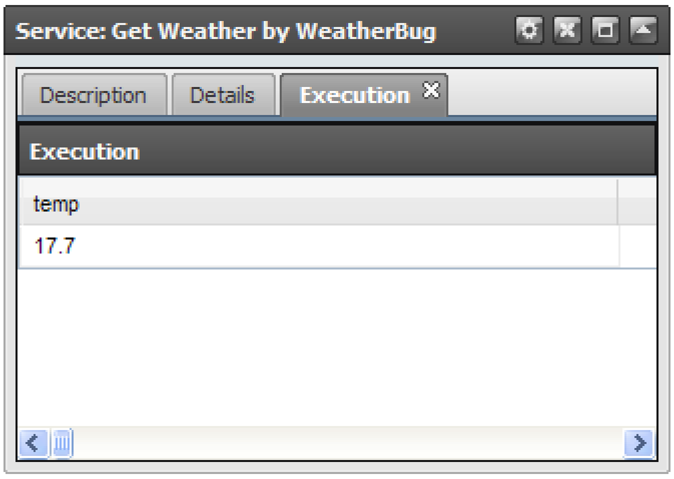
The "Details" tab displays additional information about the service. At this stage, feedback provided by other users is displayed there, in the form of ratings, tags and comments, through the interaction with the Feedback Framework, as explained in Section 4.2. In this area, it is also possible for a user to provide extra feedback information. In the short term, it is also expected to include here complementary information about the service, such information coming from the Analysis Platform. Figure 7 depicts the Details tab of a service.



No description set

Figure 7: Service details tab

When a user executes a service, the choices are sent to the server-side module, which processes them and actually interacts with the service placing the call at the execution layer. The response is parsed and processed back so the results can be presented to the user in an additional “Execution” tab, such as the one from the invocation of the weather service in Figure 8. It is worth noting that several execution tabs can be displayed after different invocations of the same service.



temp
17.7

Figure 8: Results of a service execution

2.2 Implementation Details and Installation Instructions

The Service Consumption Platform is part of and is fully integrated into the SOA4All Studio, and as such it does not make sense on its own as a separate entity. In fact, the use of the underlying Infrastructure Services as well as the UI Components provided by T2.4 [2] makes our platform fully dependent on those, not to mention the other project resources invoked.

Being a part of the SOA4All Studio, the Consumption Platform is implemented in two different modules inside `soa4all-dashboard`:

- `soa4all-dashboard-gwt-module-consumptionplatform`: This is the module with the frontend functionality that is able to create the GUI and communicate with the server-side modules.
- `soa4all-dashboard-consumptionplatform-service`: This is the module with the backend functionality that the client module accesses.

Importantly enough, the Consumption Platform service module is not the only service module that the client module is able to access, for it is also wired to other service modules inside the dashboard which provide extra functionality, such as `soa4all-dashboard-auditing-service` for tracing the interactions of the users within the platform or `soa4all-dashboard-feedback-service` to interact with the Feedback Framework.

In order to install and test the functionality of the Consumption Platform prototype, the whole SOA4All Studio needs to be deployed. The procedure for the installation and setup of the Studio is described in Section 4 of the D2.4.2 deliverable [8]. Once the Studio is running, the Consumption Platform can be accessed through the Dashboard by clicking on the "SOA4All Consume" button, as illustrated in Figure 9.

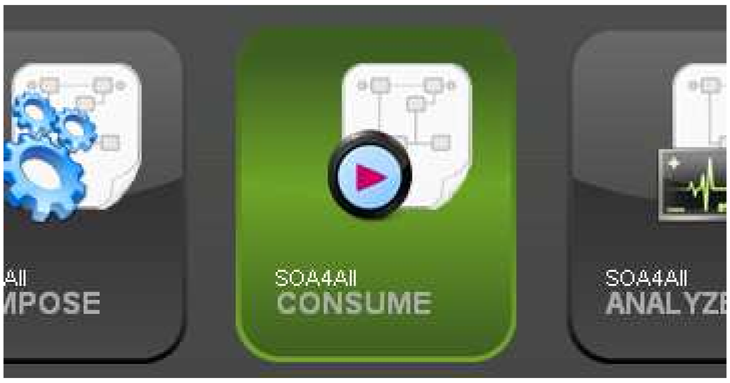


Figure 9: Consume button in the SOA4All Studio dashboard

However, end-users do not need to install anything to use the Studio. The only thing that they need is a web browser, as they may simply invoke the SOA4All application by calling the following web address, without needing to install any plugin:

http://coconut.tie.nl:8080/soa4all

3. Challenges

3.1 Automating Service Consumption

The Service Consumption Platform should not be seen as just a layer on top of discovery in the sense that it merely permits the discovery of services, in several ways, for invocation. In fact, there are a number of challenges to overcome in order to provide an efficient and automated way to consume services.

The ideal scenario of SOA4All is that once services are annotated (via the Provisioning Platform tools: WSMO-Lite Editor for WSDL services, and SWEET for RESTful services), these can be directly consumed from the Consumption Platform.

However, the process of automating service consumption is not so straightforward and indeed there are some issues to be tackled, which we refer to in this section. Firstly, the platform has to deal with the service annotations in order to produce the input for users to enter, and to give the responses in a meaningful way. (See section 3.1.1.) Secondly, there are several authentication mechanisms, especially when we consider the new trend of REST services, that prevent automation. We cannot disregard the fact that API definitions are intended for humans to understand and deal with, and not machines. (See section 3.1.2.)

3.1.1 From service descriptions to easy consumption

The Service Consumption platform operates at a conceptual (semantic) level, thanks to the annotations performed with the Provisioning Platform tools, hence dealing with instances as inputs and outputs for the (semantically enriched) services. However, the actual execution of services still happens at the same communication level, requiring an interchange of messages, typically in XML in traditional WS services, but not restricted to those, for they might also be through invocation of URLs and involving other format of responses such as JSON in RESTful services.

The Consumption Platform needs to handle the conceptual-level data and their mappings to the communication-level messages, thanks to the grounding information attached to the service. This information describes how the semantic data should be written in an XML form that can be sent to the service, and how XML data coming back from the service can be interpreted semantically by the platform. These two processes are respectively known as “lowering” and “lifting”, and they are depicted in the bottom part of Figure 10.

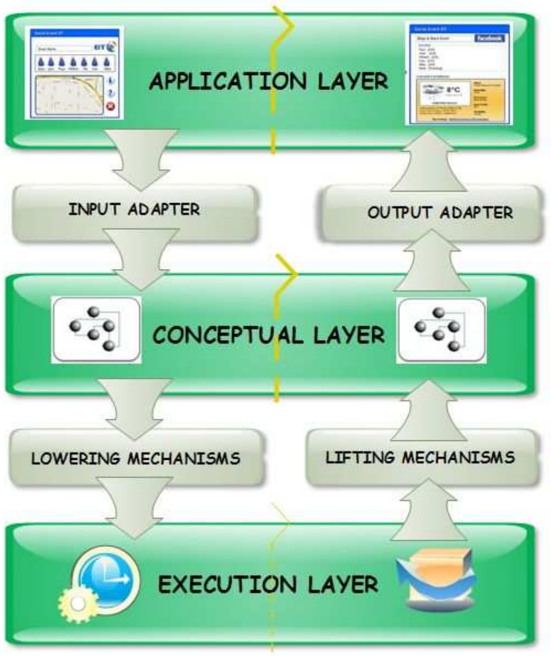


Figure 10: Layers involved in Service Consumption

While the lowering and lifting mechanisms are important, they deal with only half of the problem, for the conceptual level will also need to be mapped into graphical elements in the Consumption Platform. Firstly, because the platform needs to display relevant forms and input boxes for the users to complete with the information they desire. Secondly, because it is desirable that the platform shows the results of the execution of the service in a format more suitable for an end-user. It is important to note that we are aiming at permitting very different kinds of users, including non-technical ones, interact with the service world, so we want to hide the semantics to them when they are not needed.

We refer to the first paper attached in Annex A ("Towards a lighter way of consuming semantically enriched services") for further explanations on our position in this respect.

Additionally, the status of these adapters and mechanism in the Consumption Platform is briefly covered here:

- Input adapter: We have already set up the basis for creating a generic input form from the semantic information of the service. This `AnnotationsExtractor` creates the interface that an end-user can complete in order to interact with the service, indicating the relevant concepts that need to be entered.
- Lowering and Lifting mechanisms: Right now, these mechanisms are still hardcoded for a small set of services, but we are at the same time aiming to integrate with the Lifting and Lowering services provided by WP3, which will take place after M18. This way, when there are lifting and lowering mechanisms available, our platform can treat those steps automatically, as the actual mechanisms are out of scope of this platform.
- Output adapter: At this stage of development, the responses are displayed in generic tables to the end-users. It is also foreseen to offer more options for creating more suitable mappings from the conceptual layer in the future.

3.1.2 Authentication issues

There is another important challenge we have to face in order to enable the possibility of easily consuming services just by analysing the semantic annotations used to describe them. This challenge is related to the heterogeneity of authentication and authorization methods used by different kinds of services. In particular, RESTful services have very different ways to let third parties interact with them, and they usually ask for a key that they can provide on request. While it is feasible to ask for and store these keys manually for a small set of services, these methods prevent automation.

We refer to the second paper included in Annex A, a draft paper currently in preparation. The most important outcome is that we have already identified several types of authentication that service providers use in order to allow third-party websites interact with them. We plan to adapt the WSMO-Lite ontology to support these authentication types. Additionally, we have already taken into account the some of the simple cases, namely:

- services that use no method for authentication at all,
- services that authenticate via an API key sent in the url,
- services that require OAuth.

So far, for the latter two cases, the API keys and OAuth credentials are still stored by hand, while further progress on automating the gathering of keys and credentials –related to a user's profile when applicable– is expected for the future.

3.2 Platform adaptation

Under an end-user point of view, one of the main goals the Consumption Platform wishes to achieve is to facilitate the interactions with the user, assisting him to work with the anticipated high volumes of SOA4All services. One of the ways followed to accomplish this aim has been to exploit the Recommendation System (RS) functionalities, which give suggestions to the user basing on his behaviour and on his past interactions with the platform.

The Recommender System is totally integrated in the SOA4All Consumption Platform in a transparent way: the final user of the platform will receive simple and intuitive indications about potentially useful services, without having to deal with any configuration or data request. The system will study in a silent fashion the user actions and interests, providing him the most suitable suggestions in the most appropriate way, in function of the current context.

Currently, the RS exposes two kinds of functionalities to the Consumption Platform, both based on the concept of collaborative filtering. Content-based recommendation will be developed and integrated in the second half of the project.

The first kind of approach is totally based on user profiles. The SOA4All infrastructure logs all the human interactions with the Consumption Platform, so the RS can use them to automatically create users' outlines.

As soon as a user logs into the platform, this component contacts the Recommender System to check whether any suggestion is available for his profile. The RS returns back a list of potentially interesting services for the specified user, together with their degree of possible interest.

The suggested services are then displayed in the main menu, positioned on the left of the dashboard, in a specific section called "Recommendations". In case of new suggestions the title in the menu changes colour, becoming red, to attract the visitor's attention.

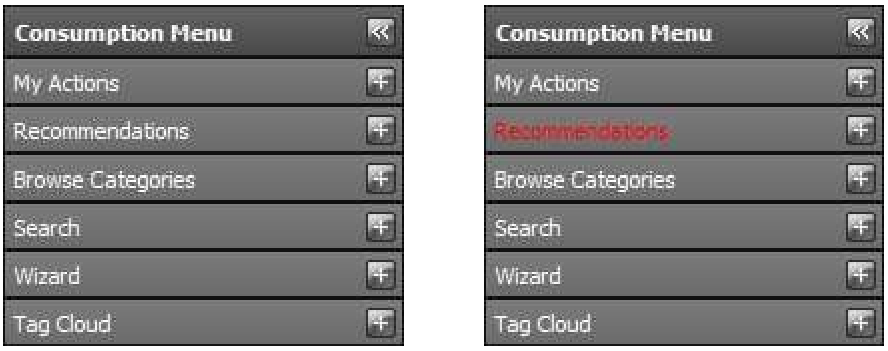
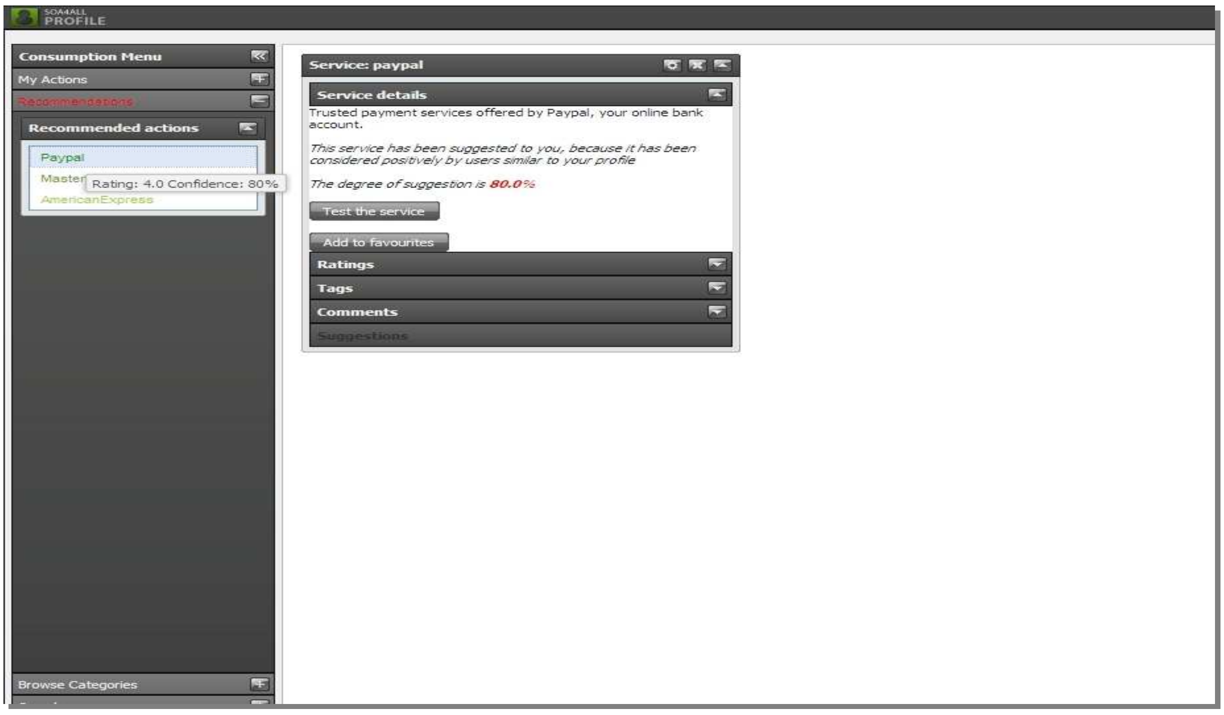


Figure 11: Recommendation area in CP main menu: “No suggestion” vs. “suggestions available” appearance

The user can expand the “Recommendations” section and, if suggestions are present, they are presented in an interactive table. Services names are displayed with different shades of colour: the darker the colour, the more confidence the recommendation has in the relevance of the service to the user. Moving the mouse over a suggested service brings up a tool tip, showing the average rating of the specific service and its suggestion confidence, as it is depicted in the following figure.



Rating: 4.0 Confidence: 80%

Trusted payment services offered by Paypal, your online bank account.

This service has been suggested to you, because it has been considered positively by users similar to your profile

The degree of suggestion is ***80.0%***

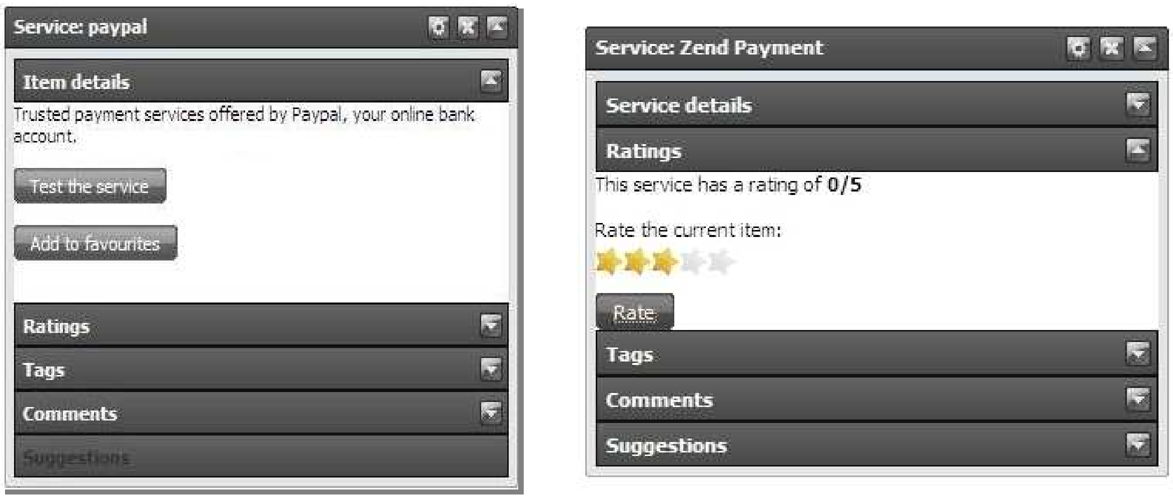
Figure 12: Recommended services by user profile

A click on an item of the table will open the service portlet where all the service information is displayed. In the main details section a brief explanation of the reason of the suggestion is shown, together with the related degree of confidence.

The second kind of suggestion is based both on the user profile and on a selected service. Such sort of recommendations are strongly linked to the currently selected service, so they will be displayed in the bottom area of the service details portlet, in an *ad hoc* section called “Suggestions”.

As soon as the user selects a service, the Consumption Platform retrieves its details and contacts the RS to know if it’s possible to give any suggestion to the user about services correlated to the selected one. If no suggestions are present for the current couple “logged user – selected service”, because there are no services to suggest, or because the user has not yet interacted with the service, this area of the portlet will be collapsed and not selectable, as shown in Figure 13, on the left side.

As soon as the user interacts with the selected service, trying it or giving some positive mark, the RS understands the user's potential interest in that service and supplies back some other services that other users considered in an "Amazon-like" style. In this case, the "suggestion area" is activated and the user has the possibility to expand it as he does for the other sections of the portlet.

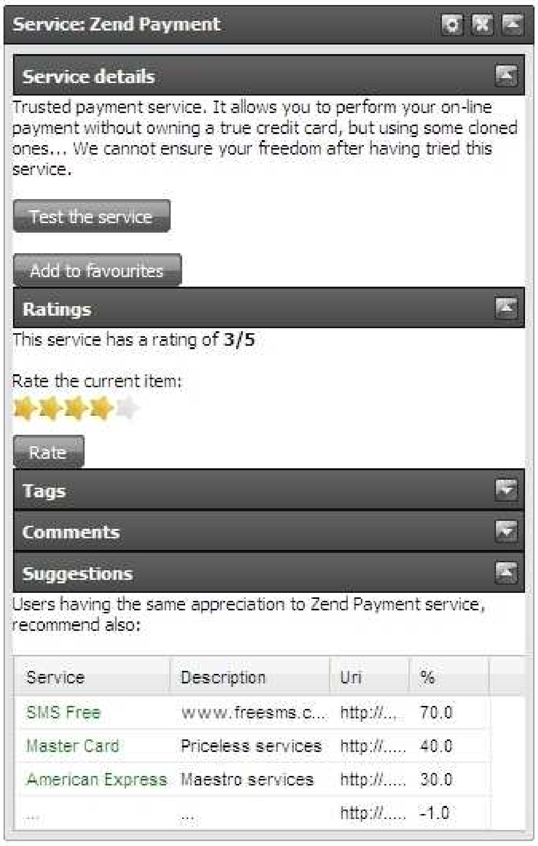


Trusted payment services offered by Paypal, your online bank account.

This service has a rating of **0/5**

Figure 13: Service portlet - "Suggestion" section disabled vs. enabled

In this dedicated area there is a table displaying the names of the suggested services and the percentage of potential interest the user can have about each of them, as shown in Figure 14.



Service: Zend Payment

Service details

Trusted payment service. It allows you to perform your on-line payment without owning a true credit card, but using some cloned ones... We cannot ensure your freedom after having tried this service.

Ratings

This service has a rating of **3/5**

Rate the current item:

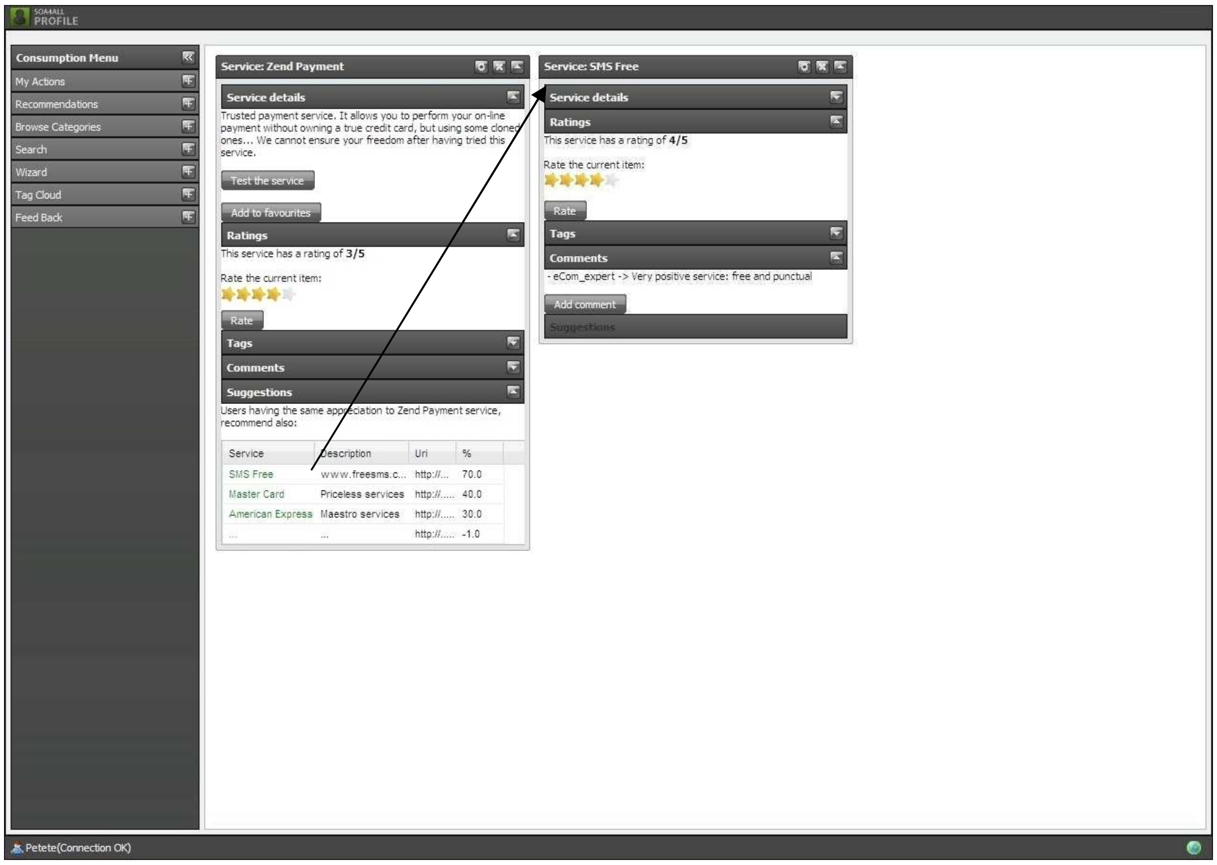
Suggestions

Users having the same appreciation to Zend Payment service, recommend also:

Service	Description	Uri	%
SMS Free	www.freesms.c...	http://...	70.0
Master Card	Priceless services	http://.....	40.0
American Express	Maestro services	http://.....	30.0
...	...	http://.....	-1.0

Figure 14: Service portlet with suggested services table displayed

When a user clicks on a service name in this table, a new service will be opened in a separate portlet, as depicted in Figure 15, Such in a way the user will be able to have more information about the suggested service and interact with it.



Service: Zend Payment

Service details

Trusted payment service. It allows you to perform your on-line payment without owning a true credit card, but using some cloned ones… We cannot ensure your freedom after having tried this service.

Ratings

This service has a rating of **3/5**

Rate the current item:

Tags

Comments

Suggestions

Users having the same appreciation to Zend Payment service, recommend also:

Service	Description	Uri	%
SMS Free	www.freesms.c...	http://...	70.0
Master Card	Priceless services	http://.....	40.0
American Express	Maestro services	http://.....	30.0
...	...	http://.....	-1.0

Service: SMS Free

Service details

Ratings

This service has a rating of **4/5**

Rate the current item:

Tags

Comments

- eCom_expert -> Very positive service: free and punctual

Suggestions

Figure 15: Selection of a suggested service

4. Integration with other components

As mentioned before, the fact that the Consumption Platform is a component of the SOA4All Studio makes it possible for the platform to use the Infrastructure Services and UI components of the Studio, as well as the interaction with other server-side modules available in the SOA4All Studio. Additionally, the Consumption Platform is also able to interact with other architectural components of the project, via the DSB.

This section covers how the Consumption Platform is tied to each of these other components.

4.1 Storage Services (T2.4)

The Storage Services are of utmost importance for many components of the Studio, and the Consumption Platform is not an exception. These services allow us to easily store and retrieve triplets into the Semantic Spaces, which eventually will be connected below to those services.

Anyway, the fact that the Semantic Spaces are connected below is transparent for our platform as a client of those services, for there is already a working version of the Storage Services. The Consumption Platform does not care if the repositories are on one machine or distributed into the Spaces, as we will interact with them through the same API.

The interaction with the Storage Services happens from the consumption server-side module via RESTful calls to the available GET method, constructing the desired SPARQL query and sending it in the header, which returns the desired RDF information, to be parsed conveniently on reception:

```
STORAGE_SERVICE_URL/repositories/<repository-id>
```

Direct interaction with the Storage Services in the Consumption Platform so far happens only in one direction, retrieving information. It is worth noting though that the platform also inserts information into the Semantic Spaces through the Storage Services (e.g., feedback information, logs), but in an indirect manner via other services, as we will see in the following two subsections.

We retrieve the following information in our direct calls to the Storage Services:

- Service annotations stored in RDF. (We make different calls to obtain the necessary details of a given service from the `AnnotationsExtractor`.)
- Categories stored in RDF. (Making the necessary calls to browse through the RDF taxonomy.)

4.2 Feedback Framework (T2.1)

As we stated before, the Feedback Framework is implemented in another server-side module (`soa4all-dashboard-feedback-service`) within the Dashboard of the SOA4All Studio, making it possible to handle ratings, comments and tags associated to different items.

Our platform needs to interact with this server-side functionality, and it will be possible to do so by wiring it conveniently too with our client-side module. Then, it is only a matter of asynchronously invoking the necessary methods and handling the responses with the callback functions. For example:

```
((FeedbackClientServiceAsync)
RemoteServiceHelper.getInstance().setupRemoteService((ServiceDefTarget)
GWT.create(FeedbackClientService.class))).rateItem(itemId,rating,userId,
rateItemCallback);
```

The methods we are invoking are:

- `rateItem`
- `getRating` (Both by a specific user, and the average by all users)
- `commentItem`
- `getComments` (Both by a specific user, and the ones made by all users)
- `tagItem`
- `getTags` (Both by a specific user, and the ones assigned by all users)

As stated before, the two-way interaction with the repository of feedback information (in the Semantic Spaces as well) is done through a separated server-side module, and not directly though the invocation of the Storage Services.

4.3 Auditing Service (T2.4)

Other server-side functionality that we need is the ability to record the users' interactions within the platform (e.g., a user opens a service, a user invokes a service, etc.). The tracking of these actions will be necessary for the Recommendation System to perform its computations, as well as for the Analysis Platform.

Again, for the Consumption Platform to interact with the auditing server-side functionality (in a server-side module: `soa4all-dashboard-auditing-service`) we have wired the platform client-side module to it.

In this case, the method that we will be accessing asynchronously to record each action that we want to (as long as it is expected in the ontology defined in D2.7.1 [6]) will be "`logAction`". In this example, we are tracking an invocation:

```
((AuditingClientServiceAsync)
RemoteServiceHelper.getInstance().setupRemoteService((ServiceDefTarget)
GWT.create(AuditingClientService.class))).logAction("ItemInvocation",
getCookie(), logParams, logActionCallback );
```

4.4 UI Widgets (T2.4)

The Consumption Platform exploits some common User Interface items, provided by T2.4, to perform actions or display information, keeping a common graphical interface with the whole SOA4All Studio environment. In particular, two widgets are used: the "rating widget" and the "tag cloud widget", both in the context of the "Details" tab of a service portlet.

The former is used in the "Rating" section, depicted in Figure 16, to display the average rating for the selected service and to allow the user to express his mark for the displayed service.

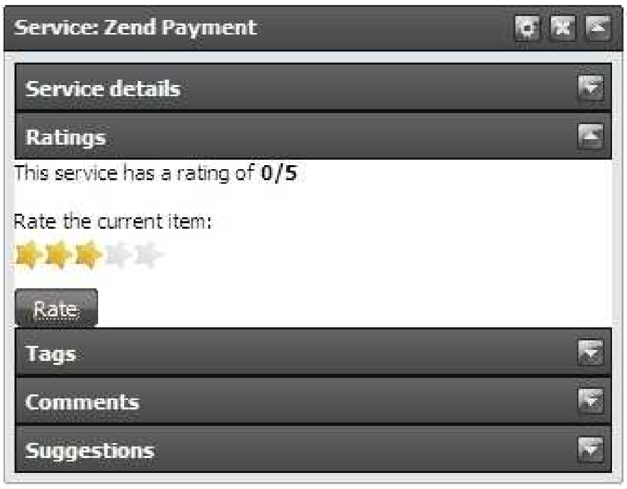


Figure 16: Rating Widget integrated in a service portlet

The latter is used in the “Tags” section, shown in Figure 17, to display the tags users have labelled the service with.

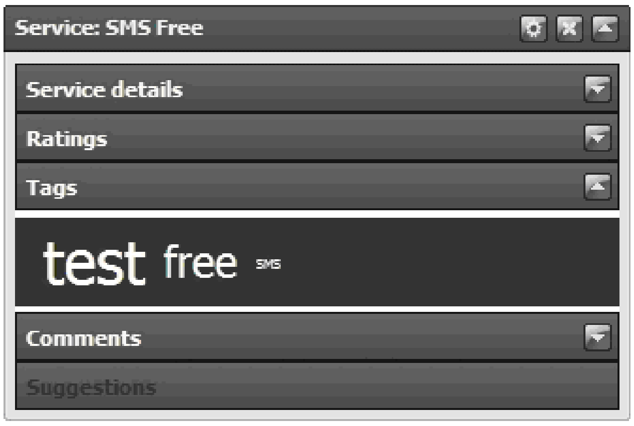


Figure 17: Tag Widget integrated in a service portlet

4.5 Recommendation System (T2.7)

The Consumption Platform performs its integration with the Recommender System through the invocation of the RS APIs.

There are different kinds of API invocations, according to the type of recommendation algorithm to be used: up to now a cyclic calls to the *getServiceRecommendationByUser* function have been implemented, together with ad hoc calls to *getServiceRecommendationByUserAndService* one.

All the interactions between the users and the Consumption Platform are automatically logged into the semantic space, so the RS can periodically retrieve them to elaborate them together with the previous ones, generating always more accurate user profiles.

Each time a user logs into the Consumption Platform, the Recommender System is contacted to check whether any suggestion is available for the current user. The RS loads its off-line computed data and returns back a list of potentially interesting services for the specified user, together with their degree of possible interest. It should be underlined that these suggestions are not produced at the moment the RS is queried, but they are the result of a batch computation.

The same interaction is performed periodically to update the suggestions to the user in real time. The period elapsing between two invocations is the same time as between the updates of the users' profiles by the RS.

To retrieve this kind of recommendations, the CP invokes the *getServiceRecommendationByUser* method of the RS APIs, providing the userURI of the currently logged user and the maximum number of services to be returned.

The other implemented case, where RS suggestions are made available to the end user, is the "Amazon-like" suggestion, based on the couple "current user – current service". As soon as the user selects a service, the CP retrieves the service details from the Semantic Spaces and contacts the RS to know if it is possible to give any suggestion to the user about services correlated to the one just selected. To ask for this information, the *getServiceRecommendationByUserAndService* function of the RS APIs is invoked, passing as parameters the logged user userURI, the current service serviceURI and the maximum number of returned services.

4.6 Keyword-based discovery (WP5)

The Service Consumption Platform gives the ability to end-users of finding services in several ways, as we have covered throughout the document. In this line, the easiest way for users to find services will be by specifying a query with the objective of retrieving a list of suitable services. Of course, our platform is not able to reply to those queries by itself, but uses the underlying SOA4All Discovery component in order to retrieve the relevant sets of services.

While it is intended to enrich the search options with NLP techniques in the future, as well as by performing Goal-based, Input/Output-based discovery, what is available by now is the keyword-based kind of discovery. Hence, we perform calls to the relevant method of the Discovery API (D5.3.1, [7]) and use SPARQL queries over the service descriptions in order to obtain a list of suitable service identifiers.

5. Conclusions

The first prototype of the Service Consumption Platform, described in this deliverable, already provides functionality for the consumption of semantically enriched services. While there is a long way ahead to improve it and implement new functionality, the foundation has been laid, on which we will build in the coming months of the project.

The platform is already integrated with many components of the SOA4All Studio (Storage Services, UI Components, Feedback Framework, Auditing Service, Recommendation System, etc.), and other architectural components of the project, such as Discovery, making this platform part of a global toolbox, and leveraging many functionalities provided by other components. We believe that this is an important characteristic of the platform, which is not on its own as a separated tool, but belongs to and it is accompanied by other modules. Again, the most important work in this respect of performing the first integrations will be leveraged in the future for subsequent alignments with different modules, such as the Analysis Platform.

The platform is already taking into account semantic service descriptions in order to produce generic visual elements that end-users will interact with. More work on this aspect is to be explored in the future, in order to further satisfy more specific UI requirements. Additionally, integration with external (WP3) lowering and lifting mechanisms services are envisaged after M18 as well.

Another challenge with respect to automation of service consumption is the way of dealing with authentication issues. While we have explored the basic characteristics, there is still work to be done in order to achieve a generic and reasonable solution. Furthermore, additional challenges are expected, not only in terms of storage of keys and security, but regarding parts of the process where the participation of a human cannot be easily circumvented; for example, when asking a service provider for a key. Not to mention service-provider requirements such as branding ones, which will be unfeasible for machines to *understand*, no matter the degree of annotations attached to their descriptions.

With respect to the important factor of Context, we have already performed several steps into what we call Platform Adaptation, by interacting with the Recommendation System, which is able to provide recommendations based on the user and the actions being performed at a given moment. The challenge after M18 is to extend the adaptation based on context to what we call Service Adaptation, actually modifying the results of the execution of services.

Finally, it is worth noting that for the first prototype we have not placed emphasis on the consumption of processes, neither to the use of goals as the anticipated key artifact to be used to discover and interact with services. This is due to the fact that the architecture of the project has not yet matured to fully support those artifacts, but we will of course accommodate the platform for their use after M18, with the objective of complementing and enriching its functionalities to fulfill its design.

6. References

[1] G. Álvaro, S. Abels, N. Mehandjiev, F. Lecue, M. Villa: Service Consumption Platform Design, D2.2.1, EU FP7 SOA4All project, February 2009

[2] S. Abels et al. SOA4All Studio First demonstrator + Interface Specification, D2.4.1, EU FP7 SOA4All project, February 2009

[3] G. Álvaro, L. Cekov, N. Mehandjiev, L. Xu, S. Abels, J. Vogel, A. Simov, M. Maleshkova: Service Provisioning Platform Design, Project Deliverable D2.1.1, EU FP7 SOA4All project, February 2009.

[4] C. Pedrinaci et al. SOA4All Provisioning Platform First Prototype, D2.1.3, EU FP7 SOA4All project, August 2009

[5] J. Vogel et al. SOA4All Process Editor First Prototype, D2.6.2, EU FP7 SOA4All project, August 2009

[6] D. Cerizza, G. Álvaro, G. Di Matteo, G. Ripa, A. Turati, M. Villa. Recommendation System First Prototype, D2.7.1, EU FP7 SOA4All project, August 2009

[7] S. Agarwal, M. Junghans, O. Fabre, I. Toma. First Service Discovery Prototype, D5.3.1, EU FP7 SOA4All project, August 2009

[8] S. Abels, J.-P. Lombardi, J. Vogel, T. Pariente, G. Álvaro, I. Martínez, M. Dimitrov, A. Simov : SOA4All Studio UI and Infrastructure Services First Demonstrator & Interface Specification, D2.4.2, EU FP7 SOA4All project, August 2009

Annex A. Short Paper(s)

This Annex contains two publications related to the content of this deliverable, describing research done within the Service Consumption Platform task. The first one has already been accepted in JSWEB'09, and the second one is still work-in-progress:

1. G. Álvaro, I. Martínez, C. Ruiz: "Towards a lighter way of consuming semantically enriched services". Accepted in JSWEB 2009.
2. G. Álvaro, C. Pedrinaci, P. Grenon: "The 'key issue' in automating service consumption". Draft of paper to be submitted on completion to a relevant conference.

Towards a lighter way of consuming semantically enriched services

Guillermo Álvaro1, Iván Martínez1, and Carlos Ruiz1
Intelligent Software Components, Madrid, Spain
galvaro, imartinez, cruiz@isoco.com

Abstract. The use of semantics to enhance Web Service descriptions paves the way to automating their discovery, composition and consumption. Previous efforts in the area of Semantic Web Services such as the top-down approach of the traditional WSMO conceptual framework implied the realization of Semantic Web Services and Goals into WSML files, where eventually the traditional services were grounded. We believe that this approach has lacked flexibility in order to involve a large number of users interacting with the service world. In the FP7 SOA4All project[8], we explore a new paradigm that uses lightweight versions of that framework, in order to deal with the problem with a bottom-up approach, covering the service descriptions with semantic annotations. In this paper, we describe our particular approach towards service consumption, where we will need to enable mechanisms to move from the conceptual level to the execution level underneath and the application level on top of it.

1 Introduction

The Web has evolved to a paradigm where new content is produced constantly at a rate that we could not imagine not so long ago. In the so-called Web 2.0 users act both as consumers and as producers of content, therefore being prosumers (term coined by Alvin Toffler in his book The Third Wave [1]), contributing to the exponential growth of information available in Internet nowadays.

In contrast, the number of WS-based services that the Web currently exposes is rather low[9], and SOA is largely still an enterprise specific solution exploited by and located within large corporations used mainly for integration.

In order to reach a service world with a much more relevant number of services, we believe that the same concepts of Web 2.0 should be applied, therefore promoting the role of "service prosumer". It is also worth noting other approaches to services such as the REST architectural style [2], quite suitable for the purpose of opening up the service world.

[8] Service Oriented Architectures for All, http://soa4all.eu/
[9] According to seekda.com the number of WSDL services available online on June 19, 2009 was 28.409.

We also believe that the use of semantics to enhance Web Service descriptions will help us automate their discovery, composition and consumption. While there have been efforts in the area of Semantic Web services, they were perceived as rather heavyweight solutions not suitable for bringing the service world to another level. In the FP7 SOA4All project [3] we explore more lightweight solutions for adding semantic information to existing service descriptions.

In this paper, we focus on the aspects to be taken into account in order to provide an efficient consumption of services that have been semantically annotated, while at the same time without requiring users to know about the semantics and technicalities that lie underneath.

The rest of the paper is organized as follows: In Section 2, we describe the particular approach taken into the SOA4All project towards the service world. Section 3 focuses on the particular characteristics of the lightweight annotations that will be used. In Section 4, we elaborate on the different aspects that need to be addressed in order to automate service consumption from their semantic descriptions. We describe the different layers that are present, namely (a) the application layer, (b) the conceptual layer, and (c) the execution layer, and the adapters and mechanisms envisaged to move from ones to the others. Finally, Section 5 covers the challenges and future work, and summarizes the paper.

2 The SOA4All approach

SOA4All aims to contribute to a service world where billions of parties are exposing and consuming services via advanced Web technology. In order to open up the service world to the desired paradigm, SOA4All integrates four pillars, namely (i) Web principles and technology, as the underlying infrastructure for the integration of services at a world wide scale, (ii) Web 2.0, as a means to leverage human-machine cooperation, (iii) Semantic Web technology, in order to abstract from syntax to semantics when required, and (iv) context management, as a way to provide users a customized experience that suits their needs.

By mixing all these technologies, we plan to cater for the needs of different types of users, involving both technical and non-technical people into the creation and consumption of services. Thus, while it is true that semantics play a central role into the envisaged paradigm, we want to abstract users from the technicalities that lie underneath whenever possible.

From the end-user perspective, the main outcome of the project will be the “SOA4All Studio", an easy-to-use integrated framework that will permit different kinds of users, technical and non-technical, the provisioning of new services (both by adding annotations to existing service descriptions and by performing light compositions), and the consumption and analysis of those.

3 Towards lightweight annotations of services

So far, the efforts in implementing Semantic Web Services have taken place in a top-down approach, such as the ones by OWL-S [4] or the WSMO [5] conceptual framework. This implied that Semantic Web Services were realized within ontological frameworks, eventually linking them to WSDL services via grounding mechanisms.

We take a more lightweight approach to implementing Semantic Web Services in a bottom-up direction, enriching existing service descriptions with semantic annotations. This is possible for WSDL descriptions thanks to SAWSDL [6] annotations that link to concepts of the WSMO-Lite ontology [7].

Furthermore, this lightweight approach is also able to take into account REST services thanks to annotations over their descriptions in HTML pages through the use of hRESTS and MicroWSMO [8], therefore opening up the set of services to the large number of providers that expose their functionalities as RESTful APIs. It is worth noting that in the context of SOA4All, tools for adding light semantic annotations have been developed both for WSDL services (WSMO-Lite Editor) and REST-based ones (SWEET, Semantic Web sErvices Editing Tool[9]).

4 From light annotations to easy consumption

The use of light annotations that link service definitions to concepts in the WSMO-Lite ontology permits us work at a conceptual level, dealing with instances as inputs and outputs for the (semantically enriched) services. However, the actual execution of services still happens at the same communication level, requiring an interchange of messages, typically in XML in traditional web services, but not restricted to those, for they might also be through invocation of URLs and involving other format of responses such as JSON in RESTful services.

An important difference of the bottom-up approach we have in SOA4All with respect to the top-down approach of the plain-WSMO model (where WSDL services were grounded into complete WSML [10] services) is that there is no need of a running semantic service execution framework such as WSMX [11] or IRS-III [12], as there actually is in the traditional WSMO approach.

Instead, the Consumption Platform of the SOA4All Studio will need to handle the conceptual-level data and their mappings to the communication-level messages. Web services generally communicate with their clients using XML messages described with XML Schema. On the semantic level, however, Web service inputs and outputs are described using ontologies. A semantic client then needs grounding information that describes how the semantic data should be written in an XML form that can be sent to the service, and how XML data coming back from the service can be interpreted semantically by the client. In other words, the outgoing data must be transformed from an ontological form to XML and, conversely, the incoming data must be transformed from XML to an ontological form. These two processes are known as lifting and lowering, and they are depicted in the bottom part of Figure 1.

While the lowering and lifting mechanisms are important, they deal with only half of the problem, for the conceptual level will also need to be mapped into graphical elements in the Consumption Platform of the SOA4All Studio. Firstly, because the platform will display relevant forms and input boxes for the users to complete with the information they desire. Secondly, as the platform will show the results of the execution of the service in a format more suitable for them. It is important to note that we are aiming at permitting very different kinds of users, including non-technical ones, interact with the service world, so we want to hide the semantics to them when they're not needed.

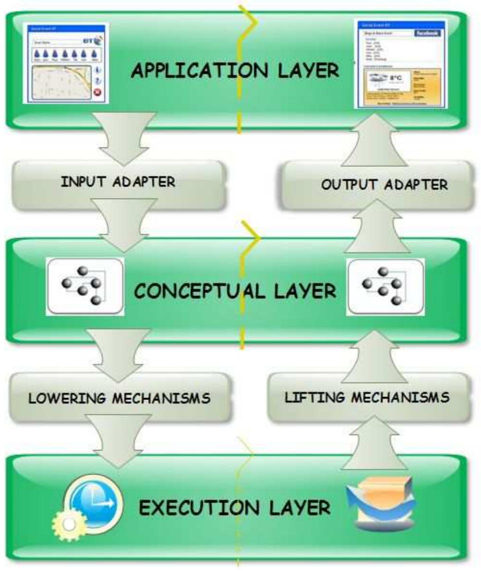


Fig. 1. Layers in the consumption of semantically enriched services

The four aforementioned steps are detailed in the following subsections. We will illustrate our explanations using an example of a service RDF description in Listing 1.1.

Listing 1.1. Example service

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:wsl="http://www.wsmo.org/ns/wsmo-lite#"
    xmlns:hr="http://www.wsmo.org/ns/hrests#"
    xmlns:sawsdl="http://www.w3.org/ns/sawsdl#"
```

```
      xmlns:xhtml="http://www.w3.org/1999/xhtml"
      xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
   <wsl:Service rdf:ID="service">
    <rdfs:isDefinedBy rdf:resource="http://...SMSService.php"/>
    <rdfs:label>Send SMS Service</rdfs:label>
    <sawsdl:modelReference rdf:resource="http://...#Consumer"/>
    <sawsdl:modelReference rdf:resource="http://...#SMS"/>
    <sawsdl:modelReference rdf:resource="http://...#Free"/>
    <wsl:hasOperation>
     <wsl:Operation rdf:ID="send.SMS">
      <rdfs:label>SendSMS</rdfs:label>
      <hr:hasAddress>http://...</hr:hasAddress>
      <wsl:hasInputMessage>
       <wsl:Message>
        <sawsdl:loweringSchemaMapping
            rdf:resource="http://...sms-lowering.xsparql" />
        <sawsdl:modelReference rdf:resource="http://...#Recipient"/>
        <sawsdl:modelReference rdf:resource="http://...#Message_Text"/>
        <sawsdl:modelReference rdf:resource="http://...#Sender"/>
        <sawsdl:modelReference rdf:resource="http://...#Message_Title"/>
       </wsl:Message>
      </wsl:hasInputMessage>
      <wsl:hasOutputMessage>
       <wsl:Message>
        <sawsdl:liftingSchemaMapping
            rdf:resource="http://...sms-lifting.xsparql" />
        <sawsdl:modelReference rdf:resource="http://...#SMS"/>
       </wsl:Message>
      </wsl:hasOutputMessage>
     </wsl:Operation>
    </wsl:hasOperation>
   </wsl:Service>
  </rdf:RDF>
```

4.1 Input Adapter

Semantically enriched services will have relevant parts of their syntactic descriptions linked to concepts of the WSMO-Lite ontology. The Input Adapter takes that semantic information in order to produce suitable graphical elements (forms with input boxes, drop-down menus, etc.) so the users can fill them and define their queries to the service.

From the RDF description of the service, the adapter looks for the “InputMessage" concept, and the “modelReference" concepts attached to it. In our example (see lines 22-25) the Input Adapter finds four different concepts to be taken into account, and generates a form with four gaps, depicted in Figure 2.

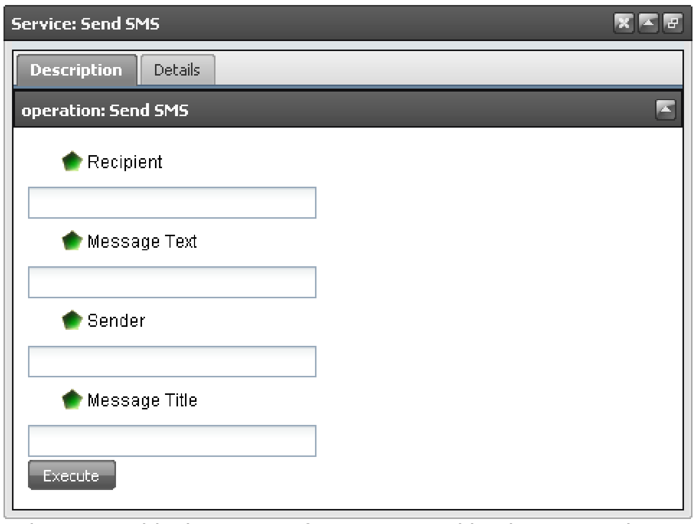


Fig. 2. Graphical User Interface generated by the Input Adapter

We also plan to make use of WATSON [13] as a method to find relevant instances that suit the concepts used for annotating the service descriptions. This way, we will be able to present users a set of sensible options whenever possible.

4.2 Lowering

The lowering mechanisms take care of translating the instances of the conceptual layer to the appropriate messages and calls to the service, depending on the kind of communication envisaged for each case. It is worth noting that our model covers both XML messages over SOAP for WS services, and also the invocation of REST methods via the appropriate URLs.

The lowering mechanisms consider the annotations over the service descriptions in order to map the instances of the conceptual level into XML messages of the syntactic level. In our example, one can spot (lines 20-21) the reference to an external XSPARQL [14] file to be used for this mapping.

4.3 Lifting

Lifting is the process of semantically annotating a source schema S with an ontology O. In our context, the response by the service (in whichever format the service responses: XML, JSON, etc.) is translated to instances. Note that this definition does not restrict the nature of the input schemas used for the lifting process. Furthermore, based on the previous definition we can consider three different approaches to the lifting problem:

- The most basic approach is to create a new ontology based on the source schema and use that ontology for the annotation
- If an existing ontology should be used for the annotation, a mapping between the source schema and the target ontology needs to be created
- Finally, a combination of the first two approaches could be used by creating a new ontology from the source schema and by then mapping this ontology to an already existing one. For this approach existing ontology mapping techniques could be reused.

Again, we rely on an external XSPARQL file, referred within the "OutputMessage" (see lines 30-31) to perform the mapping.

4.4 Output Adapter

Once a service has been executed and our platform has the response information at the conceptual level (thanks to the lifting mechanisms explained before), we are already able to present users the returned instances (in our example, based on the concept found in line 32). However, we believe that it is better to raise the abstraction level in order to provide users a more understandable output. It is important to note again that the typical end-user of our platform won't necessarily be a technical user aware of semantic implementations.

More technically advanced users will be able to create mappings between the response instances at the conceptual level and some more suitable graphical elements that represent the same information in a format more relevant for the user. For example, given a response with an instance of a coordinate concept, it might be better to display a map with the returned point than showing an instance with its attributes representing the latitude and longitude information.

5 Future Work and Conclusions

What we have described in this paper is a service-centric view, but it is important to mention that the WSMO conceptual framework stresses a decoupled approach between semantic Web services and Goals, artifacts that can be used to interact with them by defining some characteristics. As we plan to inherit the advantages of the decoupled approach in the lightweight versions of the model, we will need to take it into account for future adapters and lowering/lifting mechanisms, even though the matching of goals and services itself will happen at the conceptual level.

Another challenge we have to face is related to the heterogeneity of authentication and authorization methods used by different kinds of services. In particular, RESTful services have different ways to let third parties interact with them, and they usually ask for a key that they can provide on demand. While it is feasible to store those keys manually for a small set of services, these methods prevent automation. We have disregarded it in this paper, but will explore ways to overcome these methods in the future.

To sum up, lightweight semantic annotations over descriptions of services can be useful in order to open up the service world to a larger audience. In particular, we have focused here on how these annotations are able to enhance the consumption of services, permitting a higher degree of automation.

We have covered the need to move between the semantic world where the annotations belong to the syntactic level underneath, where the actual communication with the service happens, and also to the application level on top. Achieving this in a graceful manner through the appropriate mechanisms will improve the act of consuming services, therefore contributing to the desired service world.

References

1. Toffler, A.: The Third Wave. Bantam Books, New York (May 1984 (1980))
2. Fielding, R.T.: Architectural Styles and the Design of Network-based Software Architectures. PhD thesis (2000)
3. Domingue, J., Fensel, D., González-Cabero, R.: Soa4all, enabling the soa revolution on a world wide scale. In: Proceeding of the 2nd IEEE International Conference on Semantic Computing (ICSC), IEEE Computer Society (August 2008) 530{537
4. Martin, D., Burstein, M., Hobbs, E., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: Owl-s: Semantic markup for web services. Technical report (November 2004)
5. Lausen, H., Polleres, A., Dumitru: Web service modeling ontology (wsmo). available at: http://www.w3.org/Submission/WSMO/ (Jun 2005)
6. Kopeck_y, J., Vitvar, T., Bournez, C., Farrell, J.: Sawsdl: Semantic annotations for wsdl and xml schema. IEEE Internet Computing 11(6) (2007) 60{67
7. Vitvar, T., Kopeck_y, J., Viskova, J., Fensel, D.: Wsmo-lite annotations for web services. In: ESWC. (2008) 674{689
8. Kopeck_y, J., Gomadam, K., Vitvar, T.: hrests: An html microformat for describing restful web services. In: Web Intelligence. (2008) 619{625
9. Maleshkova, M., Gridinoc, L., Pedrinaci, C., Domingue, J.: Supporting the semi-automatic acquisition of semantic restful service descriptions. In: ESWC 2009 poster session. (2009)
10. De Bruijn, J., Fensel, D., Keller, U., Kifer, M., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L.: Web service modeling language (wsml). W3C Member Submission (June 2005)
11. Haller, A., Cimpian, E., Mocan, A., Oren, E., Bussler, C.: Wsmx - a semantic service-oriented architecture. In: IEEE International Conference on Web Services (ICWS'05), Los Alamitos, CA, USA, IEEE Computer Society (2005) 321{328
12. Domingue, J., Cabral, L., Galizia, S., Tanasescu, V., Gugliotta, A., Norton, B., Pedrinaci, C.: Irs-iii: A broker-based approach to semantic web services. Web Semant. 6(2) (2008) 109{132
13. Sabou, M., Baldassarre, C., Gridinoc, L., Angeletou, S., Motta, E., d'Aquin, M., Dzbor, M.: Watson: A gateway for the semantic web. In: ESWC 2007 poster session. (June 2007-06)
14. Akhtar, W., Kopecky, J., Krennwallner, T., Polleres, A.: XSPARQL: Traveling between the XML and RDF worlds and avoiding the XSLT pilgrimage. In Hauswirth, M., Koubarakis, M., Bechhofer, S., eds.: Proceedings of the 5th European Semantic Web Conference. LNCS, Berlin, Heidelberg, Springer Verlag (June 2008)

The “key issue” in automating service consumption

Guillermo Álvaro1, Carlos Pedrinaci2, and Pierre Grenon2
1 Intelligent Software Components, Madrid, Spain
galvaro@isoco.com

2 Open University, Milton Keynes, UK
fc.pedrinaci,p.grenong@open.ac.uk

Abstract. *... please note: this is a working draft ...*

1 Introduction

(some ideas to be refined for the introduction)

Web services were created with the objective of enabling distributed applications, and from their birth they were seen as the suitable technology to support integration amongst companies in Internet. However, currently there are not many services available in the Web, and therefore the business opportunities and advantages for end-users are less than the initially expected.

Nowadays, Web APIs are being used with more frequency. They seem to be simpler than traditional WS.

In SOA4All[3], we aim at providing an infrastructure (the SOA4All Studio[4]) that enables the semantic annotation of services to support composition, discovery, etc. of services.

Currently, there are many standards, languages and technologies to deal with security and authentication issues. However, regarding Web APIs, there are not so many standards, and hence most of their functionalities, even if public, require authentication to be used, and there exist too many different methods with different protocols.
...
The “interacting with billions of services” in SOA4All is depicted as follows: Semantically enriched versions of traditional services (WSDL-based and RESTful) are produced using WSMO-Lite and MicroWSMO. Then thanks to these annotations, services can be consumed in SOA4All (either in a direct manner or even as part of compositions).

The problem here is that many of the services to be consumed, especially the RESTful ones, are not automatically consumable. This is due to the fact that service providers typically want to have control on which mashups are using their APIs and how, so they ask for some kind of authentication (usually login/password combination or an authentication key, etc., but that's not so simple), and by just annotating a service, it doesn't mean that SOA4All will have the necessary credentials at all.
...
As part of the Consumption Platform of the SOA4All Studio, we have developed an authentication service capable of, based on annotations over the authentication methods of the services, allow third-party applications such as our SOA4All Studio, to interact with the services that request credentials. We present such a service in this paper, and we also provide an ontology that can be used to annotate the authentication methods of services and which our service is capable of interpreting.

2 Related Work

- Mashup systems (Google, Yahoo, etc)
- Auth systems here? (OAuth, etc)
- Yahoo! Query Language (YQL):http://developer.yahoo.com/yql/

3 Analysis of Authentication Methods

As a first step in order to overcome this issue and be able to interact with a greater number of services (not only with the few ones which don't ask for credentials), we want to identify the different kinds of authentication existing out there.

The APIs have been roughly grouped by the different methods of authentication (and authorization) they use, but even within each of those groups, there is a great deal of heterogeneity that complicates things up. What is more, in many cases we have spotted APIs that use a combination of the addressed methods.

3.1 API credentials to be sent in the URL

This is the simplest case, where the service provider issues some credentials which need to be used by the third party to authenticate each query against the service. In this case, the third party includes the so-called “API key”, in plain, as an extra parameter in the URLs used to interact with the service.

The name of the “key" parameter is different amongst different services, and it is worth noting that it might even consist of several parameters, such as “login" and “password", issued to a specific partner. The following two URLs are examples of this kind of authentication, the former one with a key parameter, and the latter with a user and a password:

```
http://ws.audioscrobbler.com/2.0/?method=geo.getevents&lat=51&long=0&api key=xxx
http://happenr.com/webservices/getEvents.php?username=xxx&password=xxx&town=...
```

While including the credentials in the URL is a simple process, it is important that these have to be granted by the service provider beforehand. Here again, there is a great deal of heterogeneity, as different providers tend to issue keys in different manners. To cite some examples, Happenr[5] has a contact form where third party developers can ask for the keys, while in other services such as Bit.ly[6] or Jambase[7] one has to sign in before and then ask for the key. Additionally, in many cases this process is not completely automated and involves a person from the service provider dealing with the creation of the new credentials, and someone from the third party needs to be in contact with them.

Another specific characteristic that present some of the service providers is the scope of the credentials, as the issued key might be only valid for a specific domain. This is the case of the GoogleMaps API[8], which is not a RESTful API but a Javascript container, and where the key is issued to a specific Google account and bound to a specific domain where the map will be located.

3.2 API credentials to be sent somewhere else

In other cases, the service provider issues an API key in a similar way, but it doesn't expect it in the URL of the invoked methods. The API description informs about where the credentials need to be sent in order to authenticate the third party against the service provider.

For example, the Hoovers WSDL API[9] requires third party applications to include the issued key in the SOAP header like this:

```
<soapenv:Header>
<web:API-KEY>YOUR-API-KEY</web:API-KEY>
</soapenv:Header>
```

3.3 HTTPS with HTTP-Auth

In opposition to the previous cases, where generic credentials were used to authenticate each of the calls to the service, some APIs require the credentials of a specific user. This usually happens when the action to be achieved has to do with a personal user account within the service provider (e.g., posting a link to del.icio.us[10] is associated to the del.icio.us user and not to the application calling the API). The HTTPS protocol can be used for this purpose.

Following the example of del.icio.us just mentioned, a third party could allow a user to request a set of tags associated to him (https://api.del.icio.us/v1/tags/get), and his credentials should be sent using HTTP-Auth to do so. Otherwise, a pop-up window will appear informing that the service requires a user and password.

Needless to say, the mashups making use of these kind of APIs (like the del.icio.us one[11]) rely on the fact that users have an account created within the service provider.

3.4 Basic HTTP Authentication

This form of authentication is similar to the previously explained, but using plain HTTP authentication. With this kind of authentication, the credentials of the user can be sent, Base64-encoding the username and the password[12].

An example of an authorization within a third party request (Base64 encoding of the "Aladdin:open sesame" user-password pair):

```
Authorization: Basic QWxhZGRpbjpvcGVuIHNlc2FtZQ==
```

Several services such as blip.fm[13] and simpy[14] use this method, with the login and password of the user accounts in the domains are used.

3.5 Signed requests

In some cases, the credentials provided by the service are not sent in plain, but they have to be used to sign the requests in several fashions. Here again we can see different methods and places for signing the requests, which can be also used in combination with the previous ones. To list a few of these ways to sign methods:

- The Amazon Simple Storage Services (Amazon S3[15] specify a way of using a shared secret to produce a HMAC-SHA1 of selected elements within the request, which needs to be sent along with the access key in the HTTP Authentication header[16].
- Daylife's API[17] specifies a method by which the signature is produced performing an MD5 over the access key, the shared secret and the core input for the call, and then sent in the URL[18].
- The Blip.fm API[19] involves a secret key, a nonce and a timestamp in order to generate a HMAC-SHA1 signature that needs to be sent in the URL along with the other parameters.
- The Seesmic service[20] requires a signature done by performing MD5 over the login+password of a particular user, not over an API key[21].

In order to depict one of these examples, the Amazon S3 example would involve performing the signature as explained in their documentation and then including it in the header like this:

```
Authorization: AWS AWSAccessKeyId:Signature
```

3.6 OAuth

Motivated by the lack of a global solution in the area of authentication amongst websites, OAuth[22] was born. It is an open protocol to allow (simple and standard) secure API authorization, which involves a token-passing mechanism with authorizations on them. OAuth allows a given user to grant access to their private resources on the service provider to the third-party site.

By following this protocol, both service providers and third-party developers can implement the necessary and standard steps of authorizing users of the service provider within the third-party application, and then the external application can use the authorized token to retrieve information of the user inside the service provider. One of the advantages of dealing with a standard method is that developers can find libraries in several programming languages that help them abstract from the whole complex process and re-use them for interactions within different scenarios and amongst different applications.

Even though this protocol has been subject to some critics, specially after a security issue was identified (and addressed in the version 1.0a of the protocol), an increasing number of service providers are enabling this standard way of dealing with the authentication issue. Prominent examples of these are the Twitter API[23], which supports OAuth as one of its authentication mechanisms, or Fire Eagle[24].

3.7 Other custom processes involving tokens

Apart from the aforementioned effort of OAuth, there are several other authentication methods that involve tokens applied in complex ways that don't follow a standard. Most of these protocols are proprietary methods implemented by big players in the service area. To name a few of these custom protocols, Google has AuthSub[25], AOL has OpenAuth[26], Yahoo has BBAuth[27], and other APIs such as the ones by Upcoming[28], Flickr[29] also implement their own methods.

3.8 APIs that only work in certain places

The fact that a service provider exposes its functionalities via an API does not necessarily mean that they can be used for programming applications that run anywhere, even if its methods can be invoked via a RESTful interface. In some cases, the scope of use of the API is limited to a particular site (as it happens with Facebook applications) or set of sites that implement some characteristics (e.g., Google Gadget containers, Facebook Connect sites).

In the case of Google Gadgets[30], “web-based software components based on HTML, CSS, and JavaScript”, we are not facing a proper RESTful API, but a set of Javascript-based resources, so it makes sense that those cannot run outside a special container (i.e., a website able to display gadgets).

However, the case of Facebook is very illustrative, because it has a REST interface which permits developers make calls to them, implementing a custom method for authenticating the requests[31]. However, in principle, an application developed with that API has to run within the Facebook platform, and not in an external site.

The other circumstance in which a third-party website can make use of the Facebook API is to use Facebook Connect[32]. Any website can integrate with Facebook Connect by performing some previous steps, which span from asking the credentials to uploading and configuring some files into their site[33]. This way, they can use the methods of the API, for example to get the (Facebook) friends of a user when he's logged in with his Facebook account.

Thus, even if the integration with these systems is possible, these custom steps involving additional files and configuration are obviously an obstacle on automation.

3.9 Summary of authentication methods

...In order to get an idea about the authentication methods used by different service providers, we refer to the extensive collection of APIs listed in programmableweb.com... (note: even though it is not well organised regarding auth mechanisms; explore how to get numbers of some of the cases, e.g., signed requests, API key sent somewhere else)

...We also include in Table 1 methods that do not require an authentication method at all...

Table 1. Authentication Methods

Authentication Method	Number of APIs
None	102
API credentials to be sent in the URL	536-?
API credentials to be sent somewhere else	?
HTTPS with HTTP-Auth	13
Basic HTTP Authentication	144
Signed requests	?

OAuth	45
Other custom processes involving tokens	?
APIs that only work in certain places	?

4 An ontology of Authentication Methods

ToDo

5 Evaluation

...Refer to the Consumption Platform with link to online version of the SOA4All Studio)

...Annotate several services of the different kinds(, ask for the credentials...) and test that the SOA4All Authentication system deals with them properly.

6 Conclusions

...Some challenges:

- How do we ask for credentials to the sites when it has to be done through a form or via email, etc? Do we assume that the service can be annotated but pending credentials until an "advanced" user does it for everyone?
- We should note that API keys generally have a limited number of uses.
- Also: Service-provider requirements such as branding (see Yelp: they ask to place one of their logos in the mashups created with their API)

Footnotes:

3 Service Oriented Architectures for All, http://soa4all.eu/
4 Available at...
5 http://www.happenr.com/info/api
6 http://code.google.com/p/bitly-api/wiki/ApiDocumentation
7 http://developer.jambase.com/
8 http://code.google.com/apis/maps/
9 http://developer.hoovers.com
10 http://delicious.com/
11 http://delicious.com/help/api
12 http://en.wikipedia.org/wiki/Basic access authentication
13 http://api.blip.fm/
14 http://www.simpy.com/doc/api/rest
15 http://aws.amazon.com/s3/
16 http://docs.amazonwebservices.com/AmazonS3/latest/index.html?RESTAuthentication.html
17 http://developer.daylife.com/docs
18 http://cookbook.daylife.com/docs/DayPI101
19 http://api.blip.fm/
20 http://wiki.seesmic.com/index.php/Main Page
21 http://wiki.seesmic.com/index.php/SeesmicAPIAuthentication
22 http://oauth.net/
23 http://apiwiki.twitter.com/Authentication
24 http://Fireeagle.yahoo.net/developer
25 http://code.google.com/intl/es-ES/apis/accounts/docs/AuthSub.html
26 http://dev.aol.com/api/openauth
27 http://developer.yahoo.com/auth/
28 http://upcoming.yahoo.com/services/api/token auth.php
29 http://www.ickr.com/services/api/auth.spec.html
30 http://code.google.com/intl/en-US/apis/gadgets/docs/spec.html

31 http://wiki.developers.facebook.com/index.php/How Facebook Authenticates Your Application
32 http://wiki.developers.facebook.com/index.php/Facebook Connect
33 http://wiki.developers.facebook.com/index.php/Connect/Setting Up Your Site#Referencing Facebook Connect