



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

D2.2.3 – Service Consumption Platform Second Prototype

Activity N:	Activity 1 - Fundamental & Integration activities	
Work Package:	WP2 – SOA4All Studio	
Due Date:	31/08/2010	
Submission Date:	30/08/2010	
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 Matteo Villa Freddy Lecue Irene Celino	ISOCO TXT UNIMAN CEFRIEL
Internal Reviewers:	Daniele Dell'Aglio (CEFRIEL), Alistair Duke (BT)	

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	14/07/2010	Document Initialized	Guillermo Álvaro Rey
0.2	28/07/2010	First contributions from all partners	Freddy Lecue, Matteo Villa, Irene Celino
0.3	11/08/2010	Stable draft, changed all screenshots in Section 2	Guillermo Álvaro Rey
0.4	13/08/2010	Complete version with conclusions	All
0.5	16/08/2010	Version ready for internal revision	Guillermo Álvaro Rey
0.6	25/08/2010	Comments from internal review	Daniele Dell'Aglio
0.7	27/08/2010	Comments from internal review	Alistair Duke
1.0	30/08/2010	Applied changes. Final Version	All

Table of Contents

EXECUTIVE SUMMARY	6
1. INTRODUCTION	7
2. SPICES FUNCTIONAL SPECIFICATION: USING THE SERVICE CONSUMPTION PLATFORM	8
2.1 INTERACTING WITH SPICES: GENERAL OVERVIEW	8
2.2 OPENING A SERVICE	10
2.2.1 <i>Using the Search widget</i>	10
2.2.2 <i>Browsing categories</i>	11
2.2.3 <i>Selecting a favourite service</i>	12
2.2.4 <i>Selecting a recommended service</i>	13
2.3 CONSUMING A SERVICE	13
2.3.1 <i>Dealing with authentication</i>	14
2.4 GETTING PERSONALISED RESULTS	15
2.4.1 <i>Example (Pre-Annotation of Service Description using the User Profile based Ontology):</i>	15
2.4.2 <i>Example (Illustration of a User Profile based Service Personalisation):</i>	16
2.5 INSPECTING MORE DETAILS ABOUT A SERVICE	16
2.6 CREATING FEEDBACK AND BOOKMARKING	17
2.7 GETTING RECOMMENDATIONS	18
3. IMPLEMENTATION DETAILS	19
3.1 DEVELOPMENT OF SPICES AS A DECOUPLED MODULE OF THE SOA4ALL STUDIO	19
3.2 SEMANTIC-BASED SERVICE INVOCATION	19
3.2.1 <i>Example</i>	21
3.3 INTEGRATION WITH ISERVE	22
3.4 INTEGRATION WITH THE STORAGE SERVICES	23
3.5 INTEGRATION WITH THE LINKED USER FEEDBACK SERVICE	23
3.6 INTEGRATION WITH THE AUDITING SERVICE	24
3.7 INTEGRATION WITH THE UI WIDGETS	24
3.8 INTEGRATION WITH THE RECOMMENDATION SYSTEM	25
3.9 KEYWORD-BASED AND CATEGORY-BASED DISCOVERY	25
3.9.1 <i>Example: Keyword-based SPARQL query</i>	26
3.9.2 <i>Example: Category-based SPARQL query</i>	26
4. INSTALLATION & USAGE	27
4.1 REQUIREMENTS & PREPARATIONS	27
4.1.1 <i>For End-Users</i>	27
4.1.2 <i>For Administrators</i>	27
4.2 INSTALLATION (DEPLOYMENT)	28
4.3 EXECUTION	28
5. CONCLUSIONS	29
6. REFERENCES	30
ANNEX A. RELATED PUBLICATIONS	31

List of Figures

Figure 1: SPICES: Left panel and dashboard area featuring some services and listings	9
Figure 2: Examples of portlet buttons:	9
Figure 3: Left-hand panel	10
Figure 4: Search widget.....	10
Figure 5: Search results listing	11
Figure 6: Categories Tree.....	12
Figure 7: Category results listing	12
Figure 8: “My Actions” panel.....	13
Figure 9: Recommendations panel.....	13
Figure 10: Service GUI for inputs	14
Figure 11: Results of a service invocation	14
Figure 12: Request of API credentials by the platform.....	15
Figure 13: Service Personalisation	16
Figure 14: Service Details tab.....	17
Figure 15: Feedback Creation and Bookmarking buttons	17
Figure 16: Service-based Recommendations	18
Figure 17: Semantic-based service invocation sequence diagram	20

Glossary of Acronyms

Acronym	Definition
D	Deliverable
EC	European Commission
GUI	Graphical User Interface
GWT	Google Web Toolkit
GXT	Ext-GWT
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
NLP	Natural Language Processing
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
SPICES	Semantic Platform for the Interaction and Consumption of Enriched Services
SWEET	Semantic Web sErvice Editing Tool
SWS	Semantic Web Service
T	Task
UI	User Interface
W3C	World Wide Web Consortium
WP	Work Package
WS	Web Service
WSDL	Web Services Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WWW	World Wide Web
XML	eXtensible Markup Language

Executive summary

This deliverable describes the Second Prototype of the Service Consumption Platform, lately named **SPICES** (Semantic Platform for the Interaction and Consumption of Enriched Services), the tool of the SOA4All Studio where end-users are able to interact with services and consume them in a lightweight manner.

This prototype documentation goes through the main characteristics of the platform and the functionality it provides to end-users. SPICES is described as a “personalised homepage” for the consumption of semantically enriched services, where end-users are able to find the services they are interested in and interact with them in an easy yet personalised manner. Several examples of use are given so the first part of the document can be seen as a user guide of the platform.

The deliverable also covers the most relevant implementation details, with a special stress on the integration with the rest of architectural components of SOA4All, which can be considered as one of the strongest characteristics of the platform. To list some of the aforementioned integration details, SPICES makes use of the services retrieved from the service repository iServe; it interacts with the Linked User Feedback framework for the ratings, comments and taggings on services; it retrieves recommendations from the Recommendation System, etc.

Finally, this document gives installation instructions for the platform, which can be deployed as a standalone Web application.

1. Introduction

The Service Consumption Platform, renamed both internally within the project and externally as **SPICES** (Semantic Platform for the Interaction and Consumption of Enriched Services) to better suit other tools of the platform such as SWEET and SOUR (provisioning tools described in D2.1.4 [1]), is a Web-based tool where end-users can interact with services as consumers, discovering the ones they are most interested in easily.

The Service Consumption Platform was first addressed in its design document D2.2.1 [2], and the First Prototype was described in D2.2.2 [3]. SPICES is the evolution of that prototype; the current deliverable describes the characteristics of this Second and final prototype. Some functionalities have been added to the previous version, and there is a higher level of integration with respect to the rest of components of the SOA4All architecture.

For instance, the services that are available within this platform come from iServe [4], a common service repository central to the project also accessed by the provisioning tools used to semantically annotate services. This way, the lifecycle of services is properly closed and an increasing number of services are automatically made available to SPICES by accessing the external repository.

An important change with respect to the previous version of the Consumption Platform is the decoupled approach of the SOA4All Studio that has implied the deployment of each tool in a separate basis. Hence, SPICES can be used as part of the SOA4All Studio dashboard¹, but also independently, for it is deployed as a standalone tool at its own location². It is worth noting that the decoupled version of SPICES is accompanied by a complete description of the platform at <http://soa4all.isoco.net/spices/about>.

The decoupling of the Studio also has implications with respect to the installation of a new instance of the platform, as well as regarding its source code, which has been opened, as with the rest of the SOA4All Studio components. This document also addresses these concerns.

The rest of the document is organised as follows: Section 2 covers the functional specification of SPICES including examples of use; Section 3 discusses implementation and integration details; Section 4 addresses installation instructions; Section 5 summarises the main conclusions.

An Annex A has been included with publications related to the contents of this deliverable: a poster about SPICES presented at ESWC 2010 and two papers submitted to ISWC 2010 on the Web API authentication issue and on the subject of service personalisation.

¹ Online version of the SOA4All Studio dashboard: <http://coconut.tie.nl:8080/dashboard/>

² Online version of SPICES: <http://soa4all.isoco.net/spices>

2. SPICES Functional Specification: Using the Service Consumption Platform

This section discusses the functional characteristics of SPICES from an end-user perspective, and can be considered as a basic user guide. Examples of use are given throughout the following subsections to better illustrate the way the platform can be used in order to interact with and consume services. Particularly, each of the following subsections covers:

- Section 2.1: A general overview on the platform and its main characteristics,
- Section 2.2: The different ways to open a service,
- Section 2.3: How to actually consume a service,
- Section 2.4: How to obtain personalised results,
- Section 2.5: How to get more information about a service,
- Section 2.6: How to provide feedback on a service or bookmark it,
- Section 2.7: How to get recommendations.

2.1 Interacting with SPICES: General overview

As described in the previous prototype documentation [3], the design of the platform now known as SPICES intends to support the interaction of end-users with many services in different ways, through a portal-style visualization layout. SPICES aims at being a *startpage* for the consumption of semantically enriched services in a similar fashion to well-known Web 2.0 “personalised homepages” such as iGoogle³, NetVibes⁴ or My Yahoo!⁵.

The platform contains a left panel that allows users open services in different ways (as we will see, by searching, browsing through a taxonomy, by opening bookmarked services, and by obtaining recommendations). These services, and sets of services, can be opened in the main dashboard view, where several of them can coexist organized in three different columns, as depicted in Figure 1. The first and more obvious difference with respect to the first prototype version is the change in the look and feel of the platform, which has been enabled by the new skin available for the whole SOA4All Studio (D2.4.3, [5]).

³ <http://www.google.com/ig>

⁴ <http://www.netvibes.com/>

⁵ <http://my.yahoo.com/>

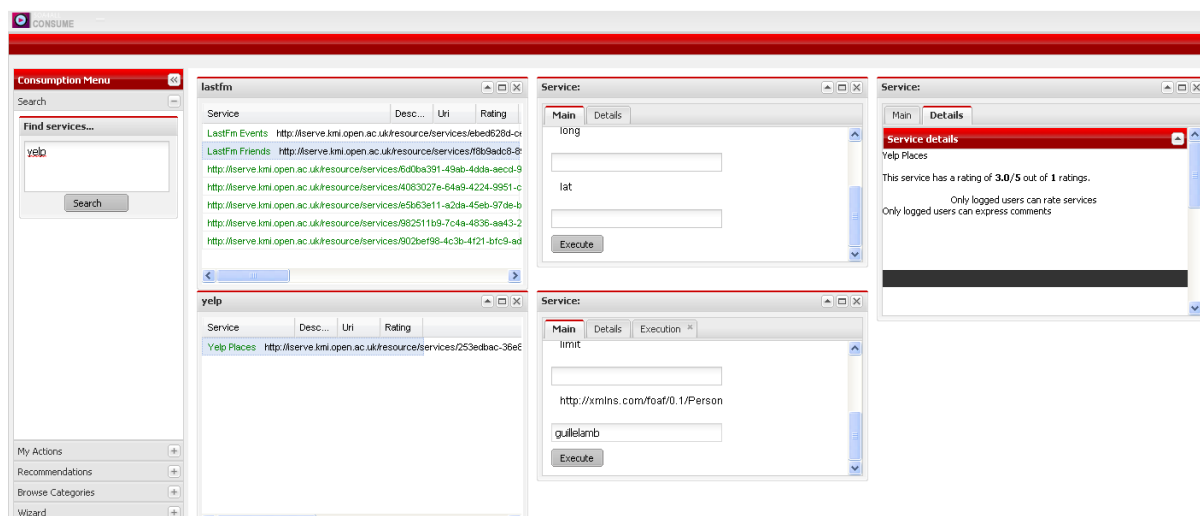


Figure 1: SPICES: Left panel and dashboard area featuring some services and listings

Following the portal kind of layout, individual “portlets”⁶ 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 in the portlets.



Figure 2: Examples of portlet buttons:
Contract, maximise, close; Expand, minimise, close

The left-hand panel, depicted in Figure 3, is the starting point of interaction in the platform and contains several ways to allow end-users to find the services they need. It is divided into the following areas:

- **Search:** A direct way of finding services with a keyword based query.
- **My Actions:** Services bookmarked by the user are displayed in this section, making easy and quick to re-open them when required.
- **Recommendations:** Sets of services are recommended through the Recommendation System to the users.
- **Browse Categories:** A hierarchical way of selecting services by navigating through a taxonomy.
- **Wizard:** It displays a draggable panel with information about the platform, suggested examples and a link to the SPICES “about” page⁷, which has more information.

⁶ <http://en.wikipedia.org/wiki/Portlet>

⁷ <http://soa4all.isoco.net/spices/luf>



Figure 3: Left-hand panel

Besides the Wizard option, the rest of menus trigger options by which services can be opened, as we will see in the next subsection.

2.2 Opening a service

The main objective of an end-user within the platform is to open services in order to interact with them. We list below the different ways through which a service may be opened in the main dashboard area, namely by i) searching, ii) browsing through categories, iii) selecting a favourite service, and iv) selecting a recommended service.

2.2.1 Using the Search widget

Users can write text in the search input box, depicted in Figure 4, to find services related to a particular keyword. Once the "Search" button is selected, a list of services is shown in a new portlet in the main dashboard area, including more information that can be relevant for the end-user such as the average rating for each particular service, helping him to decide which option to choose.

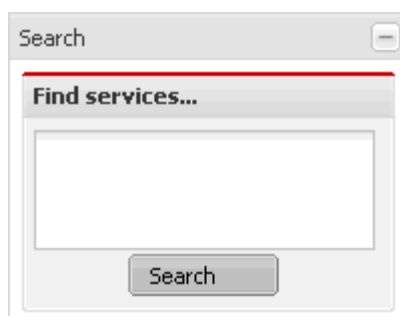
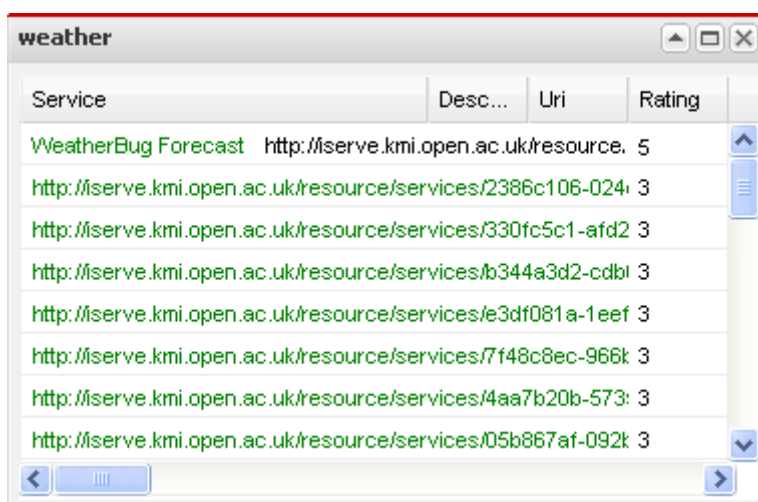


Figure 4: Search widget

For example, if a user wants to interact with a weather service, he might type that particular word and he'll be presented with a portlet containing relevant weather services, as illustrated in Figure 5.



Service	Desc...	Uri	Rating
WeatherBug Forecast		http://serve.kmi.open.ac.uk/resource/	5
		http://serve.kmi.open.ac.uk/resource/services/2386c106-024/	3
		http://serve.kmi.open.ac.uk/resource/services/330fc5c1-afd2/	3
		http://serve.kmi.open.ac.uk/resource/services/b344a3d2-cdb/	3
		http://serve.kmi.open.ac.uk/resource/services/e3df081a-1eef/	3
		http://serve.kmi.open.ac.uk/resource/services/7f48c8ec-966/	3
		http://serve.kmi.open.ac.uk/resource/services/4aa7b20b-573/	3
		http://serve.kmi.open.ac.uk/resource/services/05b867af-092/	3

Figure 5: Search results listing

Amongst the presented services, the end-user is able to select one or more to be opened. The list of services for the particular search is not immediately closed and can in fact coexist with other searches made by the user.

It is worth noting that, when possible, the result listing will provide extra information in addition to simply returning the service URI. For example, the first service annotations in the example have an associated label “WeatherBug Forecast” which makes it easier for the end user to recognize the service. (Please note that a given set of annotations do not necessarily have descriptive labels associated.) Also, it has a relevant and higher rating, and the table also reflects that. (In Sections 2.5 and 2.6 we will address where these ratings come from.)

2.2.2 Browsing categories

The other option by which an end-user is able to obtain a list of services is by browsing through a taxonomy of categories and choosing one in particular amongst them. In this case, the Categories widget depicted in Figure 6 features the Service Finder RDF taxonomy⁸, so users can find services relevant to each of those by browsing through them and selecting the one they are interested in.

⁸ <http://www.service-finder.eu/ontologies/ServiceCategories>

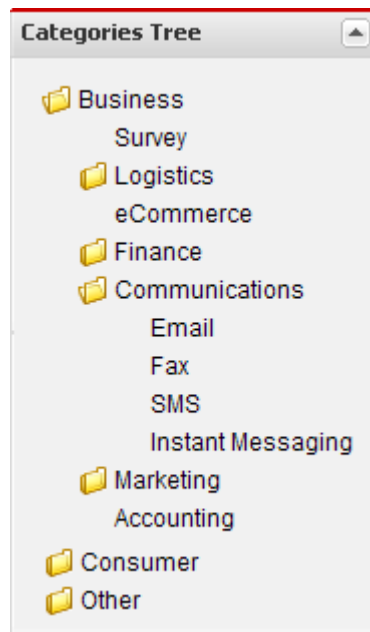


Figure 6: Categories Tree

For example, a user interested in sending text messages over the phone could navigate through the categories in order to select Business → Communications → SMS, hence triggering a list of SMS related services, as shown in Figure 7.

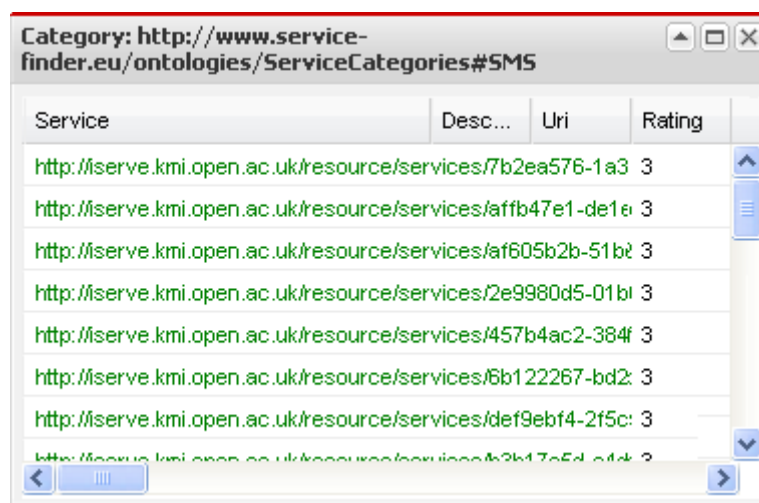


Figure 7: Category results listing

2.2.3 Selecting a favourite service

Besides the previously explained ways of opening a service after obtaining a list of results through search or categories, a service may be opened directly from the left-hand panel. In this case, the Favourites widget ("My Actions" panel) depicted in Figure 8 features services previously bookmarked by the user (see Section 2.6 on how to bookmark a service). The selection of one of these services opens it in the main dashboard area.

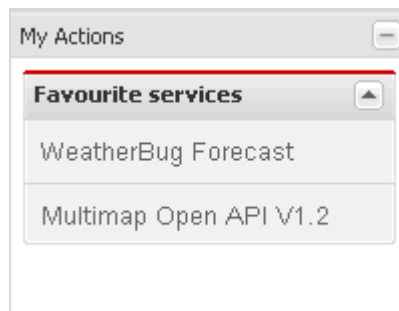


Figure 8: "My Actions" panel

2.2.4 Selecting a recommended service

Another way of directly opening a service from the left-hand panel is by selecting one of the recommendations that come from the Recommendation System, as we will see in Section 2.7. An example of this panel featuring three recommendations is depicted in Figure 9. It is worth noting that the platform is able to provide explanations to the end-user about why each of the services has been suggested to him, by showing the relation in a "tooltip" when the mouse is moved over the name of the service.

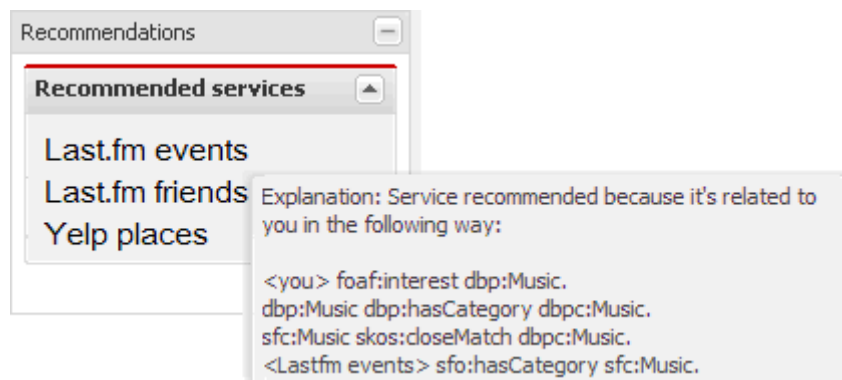


Figure 9: Recommendations panel

2.3 Consuming a service

Once a service is opened in a new portlet in the main dashboard area, the end-user is presented with a form containing input boxes that have to be completed in order to specify the necessary information to invoke the service.

Following the example of a particular weather service, the required input could be the geographic coordinates of the location from which the forecast is to be retrieved. Figure 10 illustrates the inputs from which the end-user could complete the expected data, in this case the latitude and longitude information.

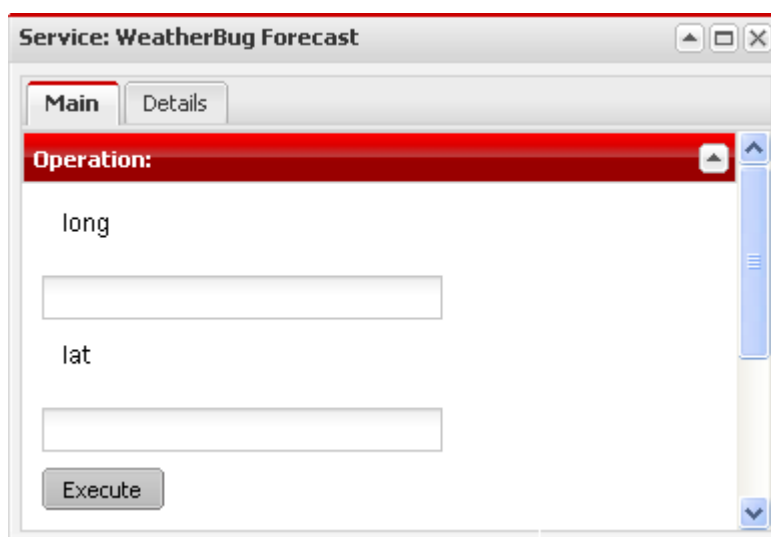


Figure 10: Service GUI for inputs

Each time a user invokes a service, a new “Execution” tab is displayed in the service portlet. This way, the results of different interactions can be compared, and discarded (by clicking the ‘x’ handler on the right-hand side of the tab) when desired. The complex process of transformations and HTTP calls including lowering, invocation, lifting, etc., happens behind the scenes (as covered in Section 3.2), and the user is presented with the result in an understandable manner. Figure 11 follows the weather service example with the display of the execution result (weather forecast information) in a new and separate “Execution” tab.

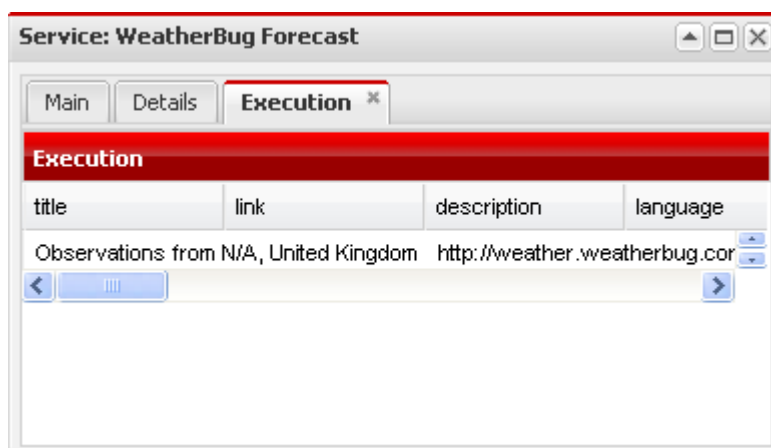


Figure 11: Results of a service invocation

2.3.1 Dealing with authentication

As discussed in the previous deliverable, one of the main issues with services that require authentication is that the semantic annotations are not enough to automatically enable their consumption. SPICES approaches this subject by trying to guide the end-user when possible in an active manner and handling the credentials in future interactions with the same service seamlessly. The following Figure 12 illustrates how the platform requests necessary API key credentials in the first use of a particular service. Once the credentials get stored, this request is not repeated in the subsequent interactions of any user with the same service, as the credentials will be already available.

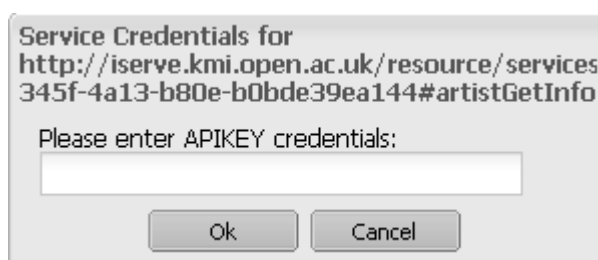


Figure 12: Request of API credentials by the platform

2.4 Getting personalised results

The personalised consumption of services is facilitated by semantic descriptions of services (i.e., using WSMO-Lite due to its simple but not simplistic model), the end-user profile and the context. Therefore, the formal model required to represent the semantics of Web services, their functional parameters (e.g., inputs and outputs), the user profile and the context based information is provided by a domain ontology.

According to these descriptions, semantic connections can be established between an input parameter of a service and a field of a user profile (in the following we focus on user profile description but the model could be easily extended to context based descriptions without loss of generalities), using their semantic similarity. The similarities are judged using a matchmaking function between the two semantic descriptions using the domain ontology.

In case matchmaking function is valid (i.e., subsumption relationships between service and profile description), the input parameter is automatically assigned with the value referred to in the user profile, but with a warning flag, informing about the incompleteness (i.e., only subsumption, no perfect equivalence) of the personalisation. In any case, it is up to the end-user to validate the personalised parameterisation of services. Finally, in case of incompatibilities, no personalisation is performed.

Since the semantics on services descriptions and user profiles are rarely described using the same ontology, the personalisation approach requires a pre-step of semantic annotation of services by means of the user profile ontologies. Such annotations are performed using the SWEET tool [6] and are stored as new service descriptions in iServe. Therefore any service descriptions relevant to the User profile are semantically annotated using the OWL vocabulary [7] i.e., mainly Concept (`rdfs:subClassOf`, `owl:equivalentClass`) and Properties (`rdfs:subPropertyOf`, `owl:equivalentProperty`) modifiers that respectively define concepts and properties through class and property axioms.

2.4.1 Example (Pre-Annotation of Service Description using the User Profile based Ontology):

The following RDF snippet simply links two properties of a user profile and a service description i.e., the property `http://profile.soa4all.org#lastName` of the user profile is equivalent to the property `http://xmlns.com/foaf/0.1/surname` of a service description available in the database.

```
<rdf:Description rdf:about="http://profile.soa4all.org#lastName">
  <owl:equivalentProperty rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
</rdf:Description>
```

Given these further semantic annotations a WSMO-Lite reasoner has been used to infer the subsumption relationships between semantic descriptions of services and user profiles.

2.4.2 Example (Illustration of a User Profile based Service Personalisation):

Figure 13 depicts a personalisation of a service from Last.fm using a user profile. For instance, the Person parameter field (left hand side) has been instantiated by “hidayat” (according to the information stored in the profile: see right-hand side, and the RDF linked operated on properties). Such a personalization requires the end-user to login first, and then the system is able to connect the relevant information by means of semantic technologies.

Service: Last.fm friends

Main Details

operation: user.getFriends

http://xmlns.com/foaf/0.1/Person
hidayat

limit

integer

APIKey

Execute

```

<rdf:Description rdf:about="http://profile.soa4all.org#av1andr1">
  <hasOpenID>"http://av1andr1.myopenid.com/"</hasOpenID>
  <lastName>hidayat</lastName>
  <firstName>aviandri</firstName>
  <hasEMail>aviandri@gmail.com</hasEMail>
  <hasWebsite>aviandri.net</hasWebsite>
  <hasPhoneNumber>123124312</hasPhoneNumber>
  <hasOccupation>RA</hasOccupation>
  <hasConnSkype>aviandri</hasConnSkype>
  <hasSNLinkedIn>aviandri</hasSNLinkedIn>
  <hasConnMSN>aviandri</hasConnMSN>
  <hasSNFacebook>aviandri</hasSNFacebook>
  <hasInterests>computer</hasInterests>
  <hasSNTwitter>aviandri</hasSNTwitter>
  <hasConnYahoo>aviandri</hasConnYahoo>
  <hasBirthDay>11/08/84</hasBirthDay>
  <hasConnICQ>aviandri</hasConnICQ>
  <hasConnAOL>aviandri</hasConnAOL>
  <hasSNMySpace>aviandri</hasSNMySpace>
  <hasLocation>Manchester</hasLocation>
</rdf:Description>

```

Figure 13: Service Personalisation

2.5 Inspecting more details about a service

A different tab (labelled “Details”) within a service portlet contains additional information about the service. It has a pointer to the set of annotations, and also features feedback data provided by other end-users.

The link to the service annotations directly points to the Linked Data URI of the set of annotations in iServe. Thus, any user can inspect the details of the annotations by HTML-browsing inside iServe. Also, more advanced users will be able to get the RDF of the annotations there.

The feedback information is retrieved from the Linked User Feedback (LUF) service⁹, also known as the Feedback Management Framework (described in D2.1.4 [1]), which is now a decoupled REST service from which SPICES is able to retrieve user generated feedback about the services in the form of (i) ratings, (ii) comments and (iii) taggings. (Note that we use the term “tagging” to define a set of tags created over a resource by a tagger on a given date, as in the Tag Ontology¹⁰.)

This information is displayed in the “Details” tab through some widgets (a Rating widget with

⁹ <http://soa4all.isoco.net/luf/about>

¹⁰ “Taggings reify the n-ary relationship between a tagger, a tag, a resource, and a date”, <http://www.holygoat.co.uk/projects/tags/>

stars for the average rating, a Tag Cloud widget for the aggregated taggings, and a table with the previous comments) as shown in Figure 14.

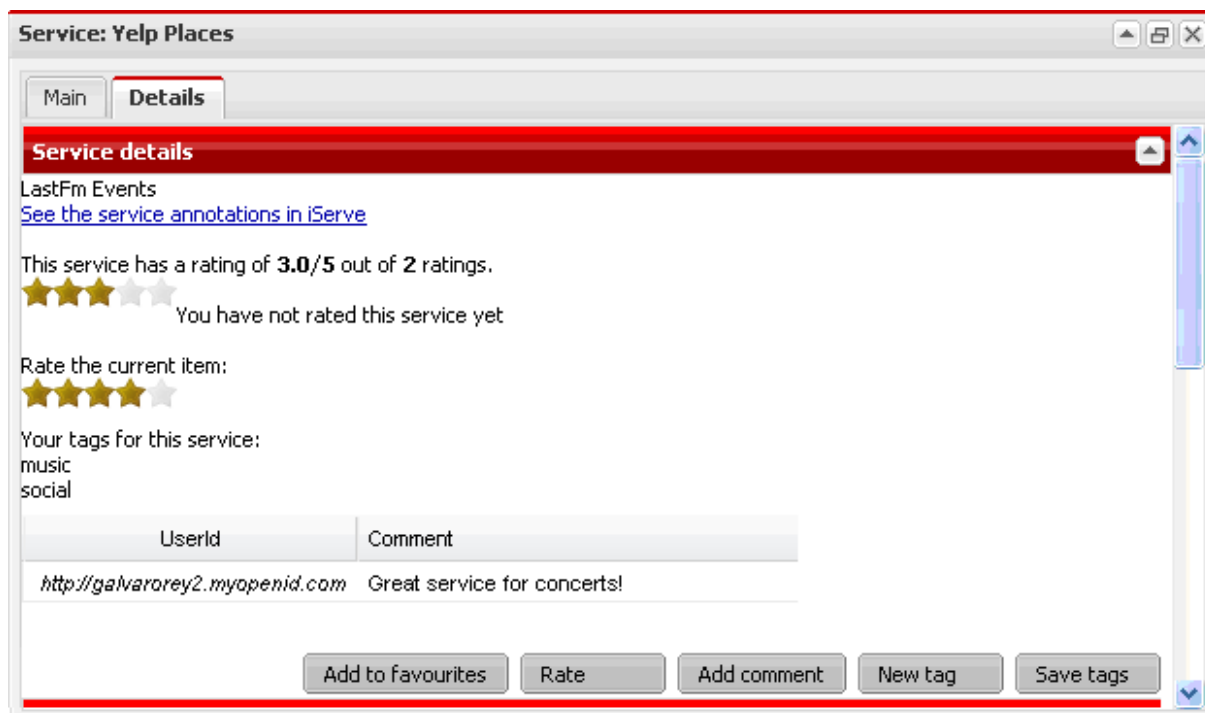


Figure 14: Service Details tab

Interestingly enough, as covered in D2.1.4, the user-generated feedback information that SPICES retrieves from LUF has not necessarily been produced within this platform, but – thanks to the decoupled approach followed by the Studio – also from iServe too.

2.6 Creating Feedback and Bookmarking

As explained in the previous subsection, any user interacting with the platform, either if he is logged in or not, can inspect the additional information about a service, including the feedback information created by other users. However, in order to produce new ratings, comments and taggings about a given service, and also for bookmarking them, an end-user needs to be logged in within the platform.

In order to log in to SPICES, users need to enter the platform through the main SOA4All Studio installation, logging in with their OpenId as described in D2.4.3 [5]. Once logged in, users can select the SOA4All Consume option and enter SPICES already logged in.

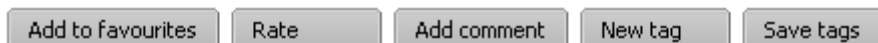


Figure 15: Feedback Creation and Bookmarking buttons

When a user is logged in into SPICES, he sees in the Details tab the buttons depicted in Figure 15, and he is able to generate new feedback information about a particular service by using them:

- A **rating** can be created by selecting the number of stars (1 to 5) in the rating widget and pressing the “Rate” button.
- A **comment** can be inserted by pressing the “Add comment” button and writing the text in the modal panel that shows up.

- A **tagging** (set of tags) can be created by pressing the “New tag” button for each new tag, and eventually the “Save tags” button.

In addition to creating new feedback about the services, a logged-in user can bookmark the ones he is interested in by pressing the “Add to favourites” button. This is not only useful for making these services available in the left-hand panel, as explained in Section 2.2.3, but also in different SOA4All Studio platforms such as the Process Editor [9] (in this case, so the bookmarked services can be inserted into a process).

2.7 Getting recommendations

Recommendations in SPICES come in two different flavours from the Recommendation System. While the deep explanations on how these recommendations are computed is contained in a separate deliverable (D2.7.2, [8]), we mention here how they take place within SPICES from the end-user point-of-view:

- For logged-in users, their experience within the platform will result in some generic recommendations shown in the left-hand panel, as explained in Section 2.2.4. (See `getRecommendationByUser` method described in Section 3.8)
- For all users, there are recommendations per-service in the bottom part of the Details tab. (See Section 3.8: If they are logged in, via the `getRecommendationByUserAndService` method; otherwise through the `getRecommendationByService` one.) These recommendations include a degree of confidence (in percentage) on the quality of the suggestion, as depicted in Figure 16.

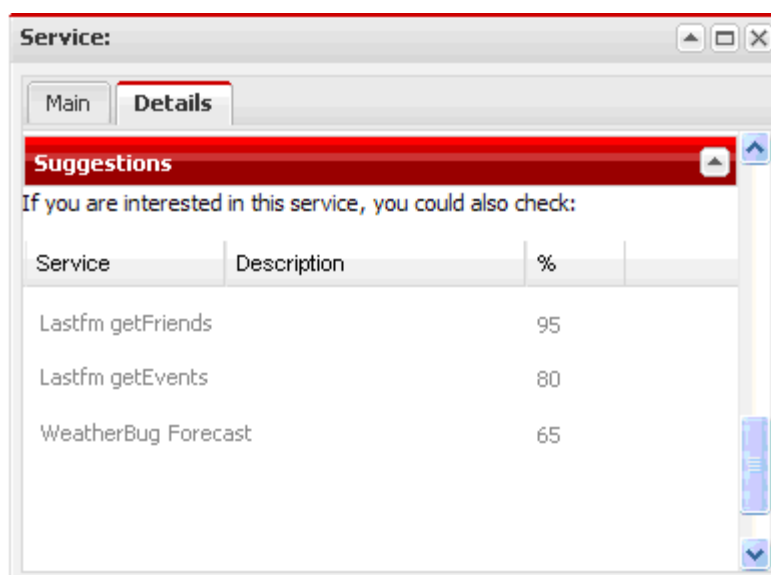


Figure 16: Service-based Recommendations

3. Implementation Details

This section discusses the most relevant implementation details of SPICES, which we have considered worth explaining in order to further illustrate its characteristics.

First of all, it is important to point out that SPICES, as a component of the SOA4All Studio, has taken a decoupled approach lately, as addressed in Section 3.1. This decoupled approach does not prevent the platform from sharing some common functionalities with other SOA4All modules, such as the semantic invocation facilities that are explained in Section 3.2. Furthermore, with SPICES being a decoupled module but also a part of a bigger picture (the SOA4All Studio), integration aspects are particularly relevant. The following subsections cover these integration issues with other important SOA4All components: iServe (Section 0), the Storage Services (Section 3.4), the Linked User Feedback service (Section 3.5), the Auditing Service (Section 3.6), the UI Widgets (Section 3.7), and the Recommendation System (Section 3.8).

3.1 Development of SPICES as a Decoupled Module of the SOA4All Studio

In the previous version of the SOA4All Studio, every single module, such as the Consumption Platform, was bundled into a monolithic software application (i.e., the SOA4All Studio). However, during the last period of the project, this approach was considered inefficient and thus the whole work package took a decoupled one instead.

This fact implies that the development of SPICES still benefits from the SOA4All Studio Infrastructure Services and UI Components (covered in D2.4.3 [5]) by importing the relevant `.jar` files into the project, while at the same time the deployment of the platform can be carried out independently and as part of the SOA4All Studio dashboard installation.

For instance, the GXT (Ext-GWT¹¹) development of SPICES has benefited from the new look and feel applied to the whole Studio, and therefore the new interface of the platform features the palette with the silver-cherry theme, as showcased in the many screenshots of Section 2.

From the development point of view, a new consumption project with a client-side module (with the frontend functionality that is able to create the GUI) and a server-side module (with the backend functionality) have been enabled. These two modules are bundled together into a Web application in a `.war` file.

A continuous build tool takes care of generating the `.war` file for the platform, which can be deployed in a Tomcat independently from the rest of the SOA4All Studio, as covered in Section 4.2.

Importantly enough, even if decoupled from the main installation of the SOA4All Studio, it can be used in conjunction with it (e.g., for logging-in purposes) and many functionalities are accessed from other SOA4All architectural components via REST service calls, as we will see in some of the following subsections.

3.2 Semantic-based Service Invocation

This section explains how SPICES can automatically invoke services based on semantic descriptions of user input and to provide service responses at semantic level.

The following Figure 17 provides an overview of the whole process:

¹¹ <http://www.sencha.com/products/gwt/>

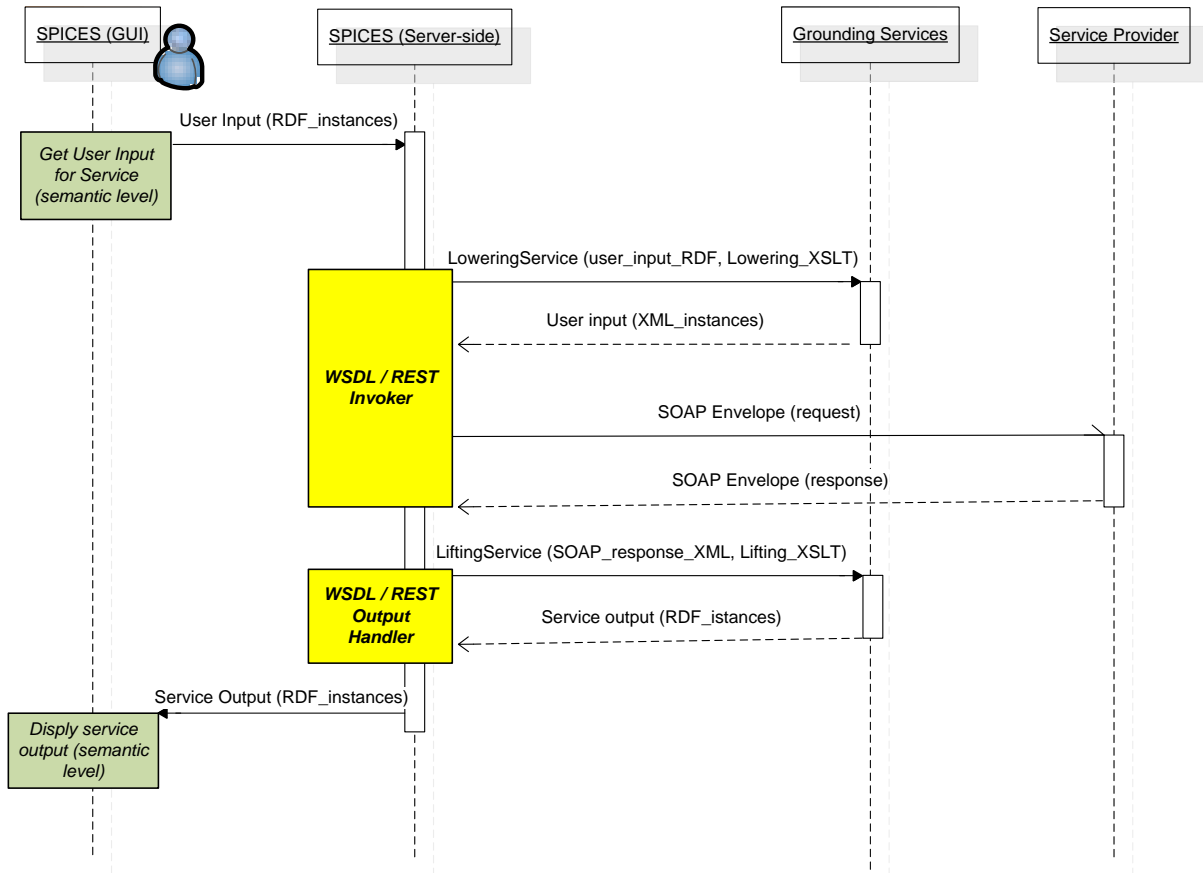


Figure 17: Semantic-based service invocation sequence diagram

Based on a service semantic description (i.e., retrieved from iServe), the SPICES GUI can collect user input at semantic (RDF) level.

Such input is streamed at server-side, where it is handled by a “**WSDL/REST Invoker**” Component: this is the core component for the whole invocation chain.

This component interfaces with the Grounding Services developed into WP3 (D3.4.1, [10]), which can perform lifting and lowering operations. Necessary information in order to perform such lifting / lowering operations is the availability of a lifting and a lowering schema: These schemata can be generated thanks to the Grounding Editor, as explained into D3.4.1 [10].

As an example, we report here the **main API** to be invoked in the “WSDL/REST Invoker” component, to invoke a WSDL service:

```
public String executeWsdService (String wsdlURL, String serviceName,
String operationName, String lifted_input, String lifting_schema_url,
String lowering_schema_url)
```

Input parameters are:

- **wsdlURL**: The URL of the WSDL file.
- **serviceName**: The name of the service to be invoked, in the WSDL file.
- **operationName**: The name of the service operation to be invoked.

- **lifted_input**: This is the user input in RDF, as provided at GUI level.
- **lifting_schema_url**: URL of the lifting schema to be used.
- **lowering_schema_url**: URL of the lowering schema to be used.

The API returns the service output transformed at RDF level, or an empty string in case of failure.

After collecting all necessary inputs, the “WSDL/REST Invoker” binds to the required service and invokes the Lowering Service in order to transform user input provided as RDF instances into XML instances, which are then encapsulated into the service request (i.e. into the SOAP message in case of WSDL services).

At this point the service can be directly invoked. The last operation is handled by the “WSDL/REST Output Handler” component, which transforms the service output from XML to RDF instances, thanks to the Lifting Service. Such RDF output is finally passed to the GUI to be displayed to the user.

3.2.1 Example

The following example shows how to use the “executeWsdIService” API.

The service to be used is a very simple one: it takes user input and appends a “hello” string in front of it:

```
http://demos.txt.it:8056/getCatalogVersion/VersioningService?WSDL
```

The service name is “VersioningService” and operation name is “GetVersion”.

A lifting and a lowering schema have been created thanks to the grounding editor, using the purchase order ontology PO.rdf:

Lifting schema:

1. http://stronghold.ontotext.com:8080/storage/repositories/playground/files/Sample_Matteo_lifting_schema_both.xsl

Lowering schema:

2. http://stronghold.ontotext.com:8080/storage/repositories/playground/files/Sample_Matteo_lowering_schema_both.xsl

These mappings are quite simple: An RDF entity `oms:has_productName` should be used as the service input, while the service output should be mapped to the property `oms:has_name` of the `oms:USAddress` entity.

As a sample user-input, we use the following RDF:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:oms="http://anonymous.generated/iOnto#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description rdf:about="http://anonymous.generated/iOnto">
    <owl:imports rdf:resource="http://anonymous.generated/iOnto"/>
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Ontology"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://anonymous.generated/iOnto#_omsitemmyVersion">
    <rdf:type rdf:resource="http://anonymous.generated/iOnto#item"/>
    <oms:has_productName
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">myVersion</oms:has_productName>
  </rdf:Description>
</rdf:RDF>
```

Actually, the mapping should take the entity `oms:has_productName` and map it as an input for the “VersioningService”. In this example, the user input is “myVersion”.

The WSDL Invoker component transforms this input into XML and generates the following SOAP request message:

```
<?xml version='1.0' encoding='utf-8'?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <getVersion xmlns="http://test/">
      <String xmlns="">myVersion</String>
    </getVersion>
  </soapenv:Body>
</soapenv:Envelope>
```

As we can see, the user input “myVersion” has been correctly placed into the operation “getVersion” as an input parameter.

After invoking the service, we get the following SOAP response:

```
<?xml version='1.0' encoding='utf-8'?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <ns2:getVersionResponse xmlns:ns2="http://test/">
      <return>hellomyVersion</return>
    </ns2:getVersionResponse>
  </S:Body>
</S:Envelope>
```

We notice that service output is “hellomyVersion” (the string “hello” is appended before user input “myVersion”).

This response is finally transformed back to RDF:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:oms="http://anonymous.generated/iOnto#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:daml="http://www.daml.org/2001/03/daml+oil#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="http://anonymous.generated/iOnto">
    <owl:imports rdf:resource="http://anonymous.generated/iOnto"/>
  </owl:Ontology>
  <oms:USAddress rdf:about="http://anonymous.generated/iOnto#_omsUSAddresshellomyVersion">
    <oms:has_name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      >hellomyVersion</oms:has_name>
    </oms:USAddress>
  </rdf:RDF>
```

We see how in the RDF above there is a resource `oms:_omsUSAddresshellomyVersion`, instance of `oms:USAddress`, that has a property `oms:has_name` with value “hellomyVersion”, which means that for the Consumption Platform end-users, the service invocation output will be “hellomyVersion”.

3.3 Integration with iServe

In the previous version of the Consumption Platform, the managed services were retrieved from the Storage Services in an ad-hoc manner, but there was not a formal repository of services central to the project. Now, this repository exists as iServe [4], the platform for the seamless publication and discovery of semantic Web services described in D2.1.4 [1], which is used by the provisioning platform tools SWEET and SOUR to store annotations on RESTful and WSDL services, respectively.

iServe exposes those annotations through an API and as Linked Data, thus enabling a SPARQL endpoint, which is quite convenient for the purposes of SPICES, as we can retrieve exactly the information desired about the services (e.g., the operations related to a service, its inputs and outputs, the concepts related via `sawsdl:modelreference`, etc.) when necessary.

The SPARQL endpoint through which SPICES interacts with iServe is located at: <http://iserve.kmi.open.ac.uk/data/execute-query>. We also highlight the GUI through which test queries can be sent: <http://iserve.kmi.open.ac.uk/browser.html>.

The integration of SPICES with iServe is of great importance, for it implies that any new service added to the service repository is automatically available within the consumption platform. Thanks to the SPARQL access to iServe, SPICES will be always up-to-date with respect to the services that can be opened.

3.4 Integration with the Storage Services

The Storage Services, part of the SOA4All Studio Infrastructure Services (see D2.4.3, [5]), 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 RDF triplets to and from the Semantic Spaces connected to those services. Anyway, the fact that the Semantic Spaces are connected is transparent for our platform as a client of those services; SPICES does not care if the repositories are on one machine or distributed into the Spaces, as the interaction is done 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>
```

The online version of the Storage Services, with which SPICES interacts, is located at the following location: <http://coconut.tie.nl:8080/storage>.

Direct interaction with the Storage Services in SPICES happens only in one direction, retrieving information (for example, for the categories stored in RDF). 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.

3.5 Integration with the Linked User Feedback service

In the M18 prototype of the Consumption Platform, interaction with the Feedback Management framework was done by accessing a different server-side module located at the same installation of the SOA4All Studio. However, in the same spirit of decoupling the components of the Studio, SPICES now interacts with the so-called Linked User Feedback service (LUF¹², the evolution of the Feedback Management framework as a decoupled REST service, described in D2.1.4 [1]) through its REST API.

This way, end-users in SPICES are able to rate, comment and associate tags to services (see Section 2.6), as well as inspect the previously created user-generated feedback

¹² <http://soa4all.isoco.net/luf/about>

(Section 2.5). Interestingly enough, thanks to the LUF service being decoupled from this particular platform, the ratings, comments and taggings are shared between SPICES and iServe (i.e., feedback produced within the two platforms are available to both of them).

The calls to the LUF API to produce new ratings, comments and taggings are, respectively, POST calls to:

- <http://soa4all.isoco.net/luf/api/ratings>
- <http://soa4all.isoco.net/luf/api/comments>
- <http://soa4all.isoco.net/luf/api/taggings>

In addition, information about a particular rating, comment and tagging is retrieved by SPICES by calling, respectively:

- <http://soa4all.isoco.net/luf/api/ratings/{id}>
- <http://soa4all.isoco.net/luf/api/comments/{id}>
- <http://soa4all.isoco.net/luf/api/taggings/{id}>

Finally, SPICES uses the search method to obtain feedback information associated to a particular service, also in relation to a particular user (to show a logged-in user his previously produced feedback information about a service), by calling:

- <http://soa4all.isoco.net/luf/api/search?itemId={item}&userId={user}>

3.6 Integration with the Auditing Service

The integration of SPICES with the Auditing Service is yet another case where the integration used to happen between server-side modules within the same installation of the SOA4All Studio (i.e., inside a deployed `.war` installation), and now the communication is done through RESTful calls to the Auditing Service API (i.e., the two components do not need to be installed in the same location).

The Auditing Service, described in D2.4.3 [5], has the ability of recording the users' interactions within the platform (e.g., a user opens a service, a user invokes a service, etc.). The tracking of these actions is necessary for the Recommendation System to perform its computation, as well as for the Analysis Platform.

SPICES calls a single method of the Auditing Service, POSTing information (the action itself and additional parameters) into a new `action` resource:

- <http://soa4all.isoco.net/auditing/action>

3.7 Integration with the UI Widgets

SPICES exploits some common User Interface items provided by T2.4 (see D2.4.3 [5]), by importing the relevant `.jar` files into the project, to perform actions or display information, keeping a common graphical interface with the whole SOA4All Studio environment. In particular, three widgets are used:

- The **Rating Widget**, used in the “Details” tab of a service portlet to display the average rating for the selected service and also to allow the user to express his mark for the displayed service (see Figure 14).
- The **Tag Cloud Widget**, also used within the “Details” tab of a service portlet to display the tags users have labelled the service with (see Figure 14).
- The **Taxonomy Selector Widget**, used to create the categories tree so users can browse through the taxonomy in the left-hand panel (see Figure 6).

It is worth noting that the widgets have been adapted to the new skin to fit the new UI of the Studio modules.

3.8 Integration with the Recommendation System

SPICES is integrated with the Recommendation System (RS); at batch time, this integration serves to the RS to gather the needed information to compute recommendations, while at run time, the integration is leveraged to invoke the RS to get recommendations to be displayed within the platform. The RS employs different algorithms to compute recommendations: collaborative filtering techniques are used to get recommendation from the users logs of the Consumption Platform, while various content-based algorithms are used to compute recommendations on the basis of the semantic descriptions of users and/or services. Those algorithms are fully described in Chapter 3 of deliverable D2.7.2 [8].

The RS run-time API is briefly described in the following. Further details are given in Chapter 2 of deliverable D2.7.2.

1. **getRecommendationByUser(*URI user*, *int num*)**: this method provides suggestions based only on the description of the current user.
2. **getRecommendationByUserAndService(*URI user*, *URI service*, *int num*)**: this method provides recommendations based on both the description of the current user and of the service currently seen by the user.
3. **getRecommendationByService (*URI service*, *int num*)**: this method provides recommendations based only on the description of the current service.

SPICES invokes the RS run-time API based on the following criteria:

- **getRecommendationByUser** is invoked when a user is logged in but he is not looking at a specific service.
- **getRecommendationByUserAndService** is invoked when a user is logged in and he is analyzing a specific service.
- **getRecommendationByService** is invoked when a user is not logged in but he is analyzing a specific service.

3.9 Keyword-based and category-based discovery

SPICES gives end-users the ability to look for services in several ways, as we have covered throughout the document. In this line, the easiest way for users to discover services will be by specifying a set of keywords with the objective of retrieving a list of suitable services. Of course, SPICES is not able to reply to those sets of keywords by itself, but uses the underlying SOA4All Discovery components, creating a suitable query with the selected keywords, in order to retrieve the relevant sets of services.

In the previous prototype, the queries were run against the service descriptions contained in the Storage Services; in the present one, the queries act on the service descriptions contained in iServe, which makes it possible for SPICES to find the services annotated with the provisioning tools. Behind the scenes, SPARQL queries are run against the service descriptions in iServe through the aforementioned endpoint in order to obtain a list of suitable service identifiers, but the end-user does not need to know about these technical details, for he is just writing some keywords and not the query himself.

The M18 deliverable described the keyword-based discovery that acts when the Search Widget is used (example of query given below in Section 3.9.1). Now, in addition to the

keyword-based discovery, we also illustrate a query triggered by the selection of a category in the Taxonomy Selector Widget (in Section 3.9.2).

3.9.1 Example: Keyword-based SPARQL query

A query over the id of the service with the term `searchTerm`, obtaining a list of services and their labels, if available:

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX msm: <http://cms-wg.sti2.org/ns/minimal-service-model#>
SELECT ?service ?labelSer
WHERE
{
    ?service rdf:type msm:Service .
    FILTER regex(str(?service), "searchTerm", "i" ) .
    OPTIONAL { ?service rdfs:label ?labelSer . }
}
```

3.9.2 Example: Category-based SPARQL query

A query to retrieve services associated via `sawsdl:modelReference` to the category `http://www.service-finder.eu/ontologies/ServiceCategories#Category`:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX sawsdl: <http://www.w3.org/ns/sawsdl#>
PREFIX msm: <http://cms-wg.sti2.org/ns/minimal-service-model#>
SELECT ?service
WHERE
{
    ?service rdf:type msm:Service .
    ?service sawsdl:modelReference
        <http://www.service-finder.eu/ontologies/ServiceCategories#Category> .
}
```

4. Installation & Usage

In this section, details are given on the requirements to use SPICES, both from an end-user point-of view, and in the case it is desired to install a new instance of the platform independently.

4.1 Requirements & Preparations

The requirements implied to use SPICES differ in the case of just using the platform via Web and in the case of being an administrator willing to install a new instance of the platform.

4.1.1 For End-Users

We address here the normal end-users that simply access and interact with the platform as explained in Section 2, (those users who are referred to as “the ‘4All’ of SOA4All” in D2.4.3 [5], where similar installation instructions are given for the whole SOA4All Studio dashboard).

These users do not need to install anything to use SPICES. The only thing that they need is a modern Web browser. The current prototype implementation described in this deliverable supports the latest stable versions of Firefox (3.5), Chrome (5) and the Internet Explorer (8). Based on this, they may simply invoke SPICES by calling the relevant Web address. It is not necessary to install any plugin.

At the present moment, a running version of SPICES for testing purposes can be accessed by end-users at the following location: <http://soa4all.isoco.net/spices>.

However, please note that this location might change in the course of the open source publishing activities of the SOA4All project.

4.1.2 For Administrators

In order to deploy a new instance of SPICES, the usual requirements for the rest of the SOA4All Studio are valid (i.e., Java and Tomcat), but also an installation of the Recommendation System is necessary in order to enable recommendations in the new platform.

4.1.2.1 Java

All SOA4All Studio developments are based on the Java programming language. As such, a Java Runtime Environment is required. Java can be downloaded for any operating systems including Windows, Linux and MacOS in their current version.

The current prototype of SPICES requires Java 1.6 or newer. The latest version may be downloaded at <http://java.sun.com>.

4.1.2.2 Tomcat

As the SOA4All Studio and its modules are Web-based solutions, the SPICES prototype is available as a Web application. As such, the Tomcat server (6.0 or newer) installation is required in order to setup the prototype. Tomcat is available at the following website: <http://tomcat.apache.org>.

4.1.2.3 Recommendation System

A new version of SPICES does not necessarily need an installation of the Recommendation System to work, but of course if it is desired to have recommendations enabled within the

platform, it is necessary. The process of installing the Recommendation System is explained in D2.7.2 [8].

4.2 Installation (Deployment)

An administrator willing to install a new instance of SPICES can retrieve the relevant .war file and deploy it in a Tomcat installation.

SPICES is using a continuous integration tool also used by other components of the SOA4All Studio called Hudson, located at <http://coconut.tie.nl:8080/hudson>. The latest version of SPICES can be found there by navigating to ConsumptionPlatformDecoupled-SPICES → lastStableBuild → "SOA4ALL Dashboard - Consumption platform" in `soa4all-consumptionplatform-webapp-0.0.1-SNAPSHOT.war`¹³.

Copying (and renaming, if desired) that .war file into the folder `webapps` of a Tomcat installation, and restarting it afterwards will automatically install all the necessary SPICES files.

4.3 Execution

After deploying the SPICES application, it can be accessed by opening a Web browser and navigating to the following URL: <http://localhost:8080/soa4all-consumptionplatform-webapp-0.0.1-SNAPSHOT> (depending on how has it been renamed). This should display the main dashboard area of SPICES as described in Section 2.1.

For updated information and any further questions, please refer to <http://soa4all.isoco.net/spices/about>.

¹³ Also, the link is: [http://coconut.tie.nl:8080/hudson/job/ConsumptionPlatformDecoupled-SPICES/lastStableBuild/eu.soa4all.studiodecoupled.spices\\$soa4all-consumptionplatform-webapp/artifact/eu.soa4all.studiodecoupled.spices/soa4all-consumptionplatform-webapp/0.0.1-SNAPSHOT/soa4all-consumptionplatform-webapp-0.0.1-SNAPSHOT.war](http://coconut.tie.nl:8080/hudson/job/ConsumptionPlatformDecoupled-SPICES/lastStableBuild/eu.soa4all.studiodecoupled.spices$soa4all-consumptionplatform-webapp/artifact/eu.soa4all.studiodecoupled.spices/soa4all-consumptionplatform-webapp/0.0.1-SNAPSHOT/soa4all-consumptionplatform-webapp-0.0.1-SNAPSHOT.war)

5. Conclusions

This deliverable is the documentation of the second prototype of the SOA4All Studio Service Consumption Platform, now known as **SPICES** (*Semantic Platform for the Interaction and Consumption of Enriched Services*), the evolution on the previous version described in M18.

SPICES is the tool of the SOA4All Studio where end-users are able to interact with services and consume them in a lightweight manner. It is intended to be a “personalised homepage” for the consumption of semantically enriched services, where end-users are able to find the services they are interested in and interact with them in an easy yet personalised manner.

The second and final prototype addressed by this deliverable has taken the same decoupled approach as the rest of the SOA4All Studio components, while at the same time new functionalities have been added, such as a better support for both WSDL and REST services, dealing with authentication, etc.

Additionally, it is worth noting that the efforts within the platform have been influenced by external factors such as the increasing importance of the Linked Data paradigm, and thus SPICES caters for what is now referred as Linked Data Services, as well as interacting with further external data in that format (e.g., with the Linked User Feedback service).

This prototype documentation has discussed its main characteristics from the end-user point of view, also reflecting the most important implementation issues, with a particular stress on the integration of the platform with other architectural components of the project, such as the common service annotations repository iServe. Finally, installation instructions for the platform have also been given, explaining how it can be deployed independently.

6. References

- [1] M. Maleshkova, G. Álvaro, A. Simov: Service Provisioning Platform Second Prototype, D2.1.4, EU FP7 SOA4All project, August 2010
- [2] G. Álvaro, S. Abels, N. Mehandjiev, F. Lecue, M. Villa: Service Consumption Platform Design, D2.2.1, EU FP7 SOA4All project, February 2009
- [3] G. Álvaro, I. Martínez, M. Villa, G. di Matteo: Service Consumption Platform First Prototype, D2.2.2, EU FP7 SOA4All project, August 2009
- [4] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecký, and J. Domingue. (2010) iServe: a Linked Services Publishing Platform, Workshop: Ontology Repositories and Editors for the Semantic Web at 7th Extended Semantic Web Conference
- [5] S. Abels, J. Vogel, G. Álvaro, I. Martínez, T. Pariente, A. Simov: SOA4All Studio UI and Infrastructure Services Second Prototype, D2.4.3, EU FP7 SOA4All project, August 2010
- [6] M. Maleshkova, J. Kopecký, C. Pedrinaci: Adapting SAWSDL for Semantic Annotations of RESTful Services. OTM Workshops 2009: 917-926
- [7] M. K. Smith, C. Welty, and D. L. McGuinness. OWL web ontology language guide. W3C recommendation, W3C, 2004.
- [8] I. Celino et al.: Recommendation System Second Prototype, D2.7.2, EU FP7 SOA4All project, August 2010
- [9] J. Vogel et al. SOA4All Process Editor Second Prototype, D2.6.3, EU FP7 SOA4All project, August 2010
- [10] A. Simov. WSMO Data Grounding Component, D3.4.4, EU FP7 SOA4All project, August 2009

Annex A. Related publications

This Annex contains information about a publication and two submissions related to the contents of this deliverable, describing research done within the Service Consumption Platform task:

1. A poster presented at ESWC2010, which is available online (both the poster itself¹⁴ and its description¹⁵).

→ Álvaro, G., Martínez, I., Gómez, J., Lecue, F., Pedrinaci, C., Villa, M., and di Matteo, G.: Using SPICES for a Better Service Consumption. Poster at Extended Semantic Web Conference (ESWC 2010)

Abstract. In this poster we present SPICES (Semantic Platform for the Interaction and Consumption of Enriched Services), a Web-based tool that automates the process of consuming a Web service by making use of the semantic annotations that describe them. SPICES supports both traditional WSDL services and RESTful ones and offers end-users the possibility of interacting with them in an easy yet personalised manner, without the need of advanced technical skills - which were traditionally required-, being the complexity that lies underneath hidden to them. SPICES is being developed within the European project SOA4All.

2. A paper submitted to ISWC2010 on the subject of Web API authentication¹⁶.

→ Maleshkova, M., Pedrinaci, C., Domingue, J., Alvaro, G., Martinez, I.: Using Semantics for Automating the Authentication of Web APIs. Submitted to main track of ISWC 2010.

Abstract. Recent technology developments in the area of services on the Web are marked by the proliferation of Web applications and APIs. The implementation and evolution of applications based on Web APIs is, however, hampered by the lack of automation that can be achieved with current technologies. Research on semantic Web services is therefore trying to adapt the principles and technologies that were devised for traditional Web services, to deal with this new kind of services. In this paper we show that currently more than 80% of the Web APIs require some form of authentication. Therefore authentication plays a major role for Web API invocation and should not be neglected in the context of mashups and composite data applications. We present a thorough analysis carried out over a body of publicly available APIs that determines the most commonly used authentication approaches. In the light of these results, we propose an ontology for the semantic annotation of Web API authentication information and demonstrate how it can be used to create semantic Web API descriptions. We evaluate the applicability of our approach by providing a prototypical implementation, which uses authentication annotations as the basis for automated service invocation.

¹⁴ <http://lab.isoco.net/files/publications/AlvaroEtAlESWC10.png>

¹⁵ <http://lab.isoco.net/files/publications/AlvaroEtAlESWC10.pdf>

¹⁶ Submission also included in the Annex A of D2.1.4, present here for it features an important section on how SPICES deals with the Web API authentication issues.

3. A paper submitted to ISWC2010 on the subject of service personalisation.

→ Lecue, F.: Personalization of Semantic Web Services. Submitted to main track of ISWC 2010.

Abstract. Nowadays web users have clearly expressed their wishes to receive and interact with personalized services directly. However, existing approaches largely syntactic content-based, fail to provide robust, accurate and useful personalized services to its users. Towards such an issue, the semantic web provides enabling technologies to annotate and match services' descriptions with a user' features, interests and preferences, thus allowing for more efficient access to services and then information. The aim of our work, part of service personalization, is on automated instantiation of services which is crucial for advanced usability i.e., how to prepare and present services ready to be executed while limiting useless interactions with users? To this end, we exploit Description Logics reasoning through semantic matching to i) identify useful parts of a user profile that satisfy services requirements (i.e., input parameters) and ii) compute the description required by a service to be executed but not provided by the profile. Finally, the scalability of our approach has been evaluated through its integration in the service consumption of the EC-funded project SOA4All.