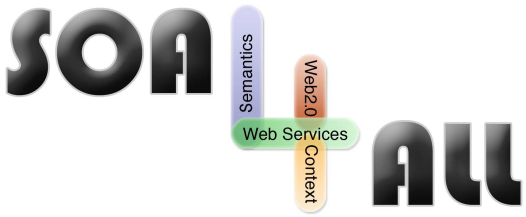

Project Number: **215219**

Project Acronym: **SOA4All**

Project Title: **Service Oriented Architectures for All**

Instrument: **Integrated Project**

Thematic Priority: **Information and Communication Technologies**

D2.2.1 Service Consumption Platform Design

Activity:	Activity 1 - Fundamental & Integration Activities	
Work Package:	WP2 - SOA4All Studio	
Due Date:		M6
Submission Date:		29/08/2008 Resubmission: 12/03/2009
Start Date of Project:		01/03/2008009
Duration of Project:		36 Months
Organisation Responsible of Deliverable:		iSOCO
Revision:		2.14
Authors:	Guillermo Álvaro Rey iSOCO Sven Abels TIE Nikolay Mehandjiev UNIMAN Freddy Lecue UNIMAN Matteo Villa TXT	
Reviewers:	Reto Krummenacher UIBK Marc Richardson 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	19/05/2008	First proposal of ToC	Guillermo Álvaro Rey
0.2	24/06/2008	Initial draft version with contributions for 2nd Plenary Meeting (Nice)	Guillermo Álvaro Rey
0.3	04/07/2008	Merged additional changes in new template	Guillermo Álvaro Rey
0.4	28/07/2008	Merged contributions	Guillermo Álvaro Rey, Sven Abels, Matteo Villa
0.5	30/07/2008	Version with architecture diagrams	Guillermo Álvaro Rey, Sven Abels, Matteo Villa, Nikolay
0.6	05/08/2008	First complete draft	Guillermo Álvaro Rey, Sven, Matteo Villa, Nikolay
0.7	08/08/2008	Corrections and expanded components	Guillermo Álvaro Rey, Sven, Matteo Villa, Nikolay Mehandjiev
1.0	08/08/2008	Version for internal revision	Guillermo Álvaro Rey, Sven, Matteo Villa, Nikolay Mehandjiev
1.1	27/08/2008	Comments from reviewers	Guillermo Álvaro Rey, Sven Abels, Jose Manuel Gomez, Nikolay Mehandjiev, Martin Carpenter
1.2	29/08/2008	Fixed final issues	Guillermo Álvaro Rey
1.3	12/09/2008	Additional corrections	Guillermo Álvaro Rey, Sven, Matteo, Nikolay
2.0	22/12/2008	First draft for resubmission	Guillermo Álvaro Rey
2.8	29/01/2009	New contents from everyone	Guillermo, Matteo Villa, Nikolay Mehandjiev, Freddy
2.9	30/01/2009	Revisions on sect 1.1, sect 2.1, sect 2.2 and chapter 5	Matteo Villa
2.10	03/02/2009	Refined contents	Guillermo Álvaro Rey
2.12	04/02/2009	Refinements	Guillermo, Matteo Villa, Freddy Lecue

2.13	09/02/2009	Completed draft for internal reviewers	Guillermo Álvaro Rey, Matteo, Freddy Lecue
-	23/02/2009	Internal review	Reviewer: Reto Krummenacher (UIBK)
-	23/02/2009	Internal review	Reviewer: Marc Richardson (BT)
2.14	05/03/2009	Addressed comments by reviewers	Guillermo Álvaro Rey, Matteo, Freddy Lecue
Final	10/03/2009	Overall format and quality revision	Malena Donato (ATOS)

Table of Contents

List of Figures and Tables

List of Figures

List of Tables

Glossary of Acronyms

Acronym	Definition
D	Deliverable
DRM	Digital Rights Management
EC	European Commission
GUI	Graphical User Interface
GWT	Google Web Toolkit
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IPR	Intellectual Property Rights
ISP	Internet Service Provider
KPI	Key Performance Indicator
NLP	Natural Language Processing
POS	Part-of-speech
QoS	Quality of Service
RIA	Rich Internet Application
RDF	Resource Definition Framework
REST	Representational State Transfer
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 Services
WSDL	Web Services Description Language
WSML	Web Service Modeling Language
WSMO	Web Service Modeling Ontology
WWW	World Wide Web
XML	eXtended Markup Language

Executive summary

SOA4All is promoting a new service world scenario, in which billions of parties expose services catering to every type of user. As with information on the WWW, these services will be discovered and consumed on-line. In order to achieve this scenario, several important challenges must be overcome. These include not only making so many services available on-line, but also permitting their efficient consumption by users lacking a technical background.

One consequence of a service world containing such a large number of services is that the task of finding the most suitable service for each particular case will be especially complex. While having many options to choose from is a benefit that we want to take advantage of, it involves the necessity of having an efficient way of interacting with the appropriate services, and in the most suitable (i.e., personalised, as we will see) fashion, in order to fulfil the desires of service consumers.

The Service Consumption Platform, the design of which is the scope of this deliverable, follows the WSMO approach towards service consumption, tackling the issue of finding the right services by allowing service consumers to define Goals that will be used to facilitate the discovery of previously semantically enhanced services. This method of interacting with services by expressing objectives in abstract fashion is, in light of the vast number of services involved, much more efficient than directly browsing service repositories.

In addition to that approach, and in line with the actual tendency in the Web nowadays, the Service Consumption Platform stresses the idea of personalisation. This involves considering not only the location of bespoke processes that might meet users' needs but also considering how existing services might be tailored to do so. The Service Consumption Platform emphasizes the characteristic of personalisation by applying customized constraints to the services, thanks to the use of formalised and structured contextual factors.

From the point of view of life cycle of services, consumption plays a central role: Services are previously semantically annotated in the Service Provisioning Platform, and the consumption of those can be analysed later on. It is also worth noting the presence of the Service Consumption Platform within the SOA4All Studio, which will provide some useful infrastructure services and also user interface components that will enable a holistic look and feel for the several platforms of the work package.

In this deliverable, we review and describe the different architectural components that we foresee as necessary in order to cover the functionality that we expect the platform to have, focusing on the issues that we have highlighted in this summary (personalisation of services by means of context, goal configuration, integration within the SOA4All Studio). We believe that by following the design covered by this deliverable, the development of a fully functional Service Consumption Platform as envisaged by SOA4All will be possible.

1. Introduction

1.1 Structure of the Document

This deliverable is structured in the following way:

- **Chapter 1** provides introductory explanations for the deliverable, addressing the purpose and motivations for the Service Consumption Platform within the project (Section 1.2), also placing this platform in context with the rest of architectural components of the project (Section 1.3), and with the Use Case work packages (Section 1.4).
- **Chapter 2** provides the **functional specifications** of the Consumption Platform, defining and classifying the main functionalities that it will support (Section 2.1) and the main use cases for end-users (Section 2.2).
- **Chapter 3** provides the **technical specifications,** starting from an overview of the architecture and its components, showing how these are necessary in order to provider the functionalities identified in the previous chapter.
- **Chapter 4** focuses on the various layers of the architecture, defining for **each single component low-level specifications** on their internal structure and the services they need to expose.
- **Chapter 5** reports the conclusions of the deliverable.

1.2 Purpose and Motivation

SOA4All is proposing a new paradigm where billions of services will be available for the users to interact with them. Thus, in order to enable an interaction with the right services, amongst a huge number of them, we will need an easy-to-use platform that allows service consumers to discover and invoke the services they desire.

The current deliverable addresses the design of this Service Consumption Platform. This platform will be responsible for the invocation and execution of semantically enhanced services provided by the Service Provisioning Platform, which is itself described within other deliverables [8][11].

Regarding service discovery, the Service Consumption Platform will have to enable its users (in this case, SOA4All "service *prosumers*" –as they can act both as service providers and service consumers– when acting with their role of consumers) to efficiently find the services that fulfil their needs. In deliverable D2.1.2 [8], we address how the service modelling tools to be developed in this project will make a large number of services available, by dealing with two different WSMO-variants of plain full WSMO [2]: WSMO Lite [4] and MicroWSMO [5]. This way, traditional Web services (including SOAP-based [1] and RESTful services [6]), enriched with semantic annotations, will be available in our platform.

Therefore, the new service consumption paradigm that SOA4All is promoting has also to take into account the new versions of WSMO, and thus our platform will find the right services based in the annotations made in the Service Provisioning Platform. A key requirement for this platform will be enabling a proper mechanism with which the users can find their desired services. As we understand it, the WSMO approach of defining Goals – in several ways that we will address later on – will be the most efficient method to find the proper services. Especially if we take into account that there will be a huge number of them, preventing users from just browsing through a directory to find the ones that satisfy their needs. It is worth noting, however, that the use of Goals by non-technical service consumers within the platform will be transparent for them, as they won't even need to know what a Goal is nor the formalisms that lie underneath. A Goal will be a conceptual artefact that we will use

in the backend to capture user requirements, enabling the discovery and consumption of services for them, but the non-technical user won't be interested in the machinery that permits it, so we will abstract him from these technical issues.

Additionally, Service Consumption, as envisaged by SOA4All, will stress the importance of the user, facilitating a personalised access to the services. This is, we believe, not only a specific characteristic of our project, but a global tendency of the Internet. The average user in the Web today needs to experience a personalised access to the Web sites and applications he interacts with, therefore enhancing the default settings of applications with data and configurations relevant to him. Between the numerous examples that can be found all across Internet, we can highlight the proliferation of personalised home sites, such as iGoogle[1] or netvibes[2], but also the fact that many other Web applications of different kinds treat user configuration as one of their most prominent characteristics. The approach of SOA4All towards personalisation of services will be to use contextual factors to apply customized constraints to the services.

Furthermore, in the light of Web 2.0, the user is important not only as an individual, but also we have to consider him as part of a social network that determines his experience. Social network sites, such as Facebook[3] or Linkedin[4], to name a few, place the user in context with his connections. As a result, the actions of Web 2.0 users are conditioned by their presence in their social network. New services are taking into account the fact of users being in a community, and in a similar fashion SOA4All will address this issue, by considering the information that comes from the community as a particular case of contextual information used to modify the characteristics of a given service.

We believe that the issue of personalisation is of major importance in the context of SOA4All, where there will be a large number of users who have specific individual goals to be addressed by the combination of certain functionalities, amongst those provided by a vast amount of services. Giving the users the chance to experience those services in a customised fashion will be key in order to permit them fulfil their desires more suitably. One could argue that in this scenario it can be extremely difficult to discover and adapt exactly the functionalities that a service consumer may need. In fact, the number and heterogeneity of services and user needs and contexts in SOA4All is an issue for complexity. However, the use of formalised and structured contextual information [10] will permit the desired adaptation to occur in an organised fashion.

To sum up, the Service Consumption Platform is a key architectural component of SOA4All in order to enable a world where a huge number of services will be available to be consumed. The Service Consumption Platform design covered by this deliverable will satisfy the motivations explained herein, which are of major importance to reach the ambitious objectives of the project.

1.3 Alignment with the Overall SOA4All Architecture

We describe here the alignment of the Service Consumption Platform with the SOA4All architecture defined in work package 1, and with the rest of architectural components of the project. Being the Service Consumption Platform part of the SOA4All Studio [20], many characteristics about its integration are similar to the other platforms (i.e., Provisioning [8][11] and Analysis [14]) which conform the SOA4All Studio, but we will also explicitly address and

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

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

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

[4] http://www.linkedin.com/

point out those details regarding the particular platform being described in this deliverable.

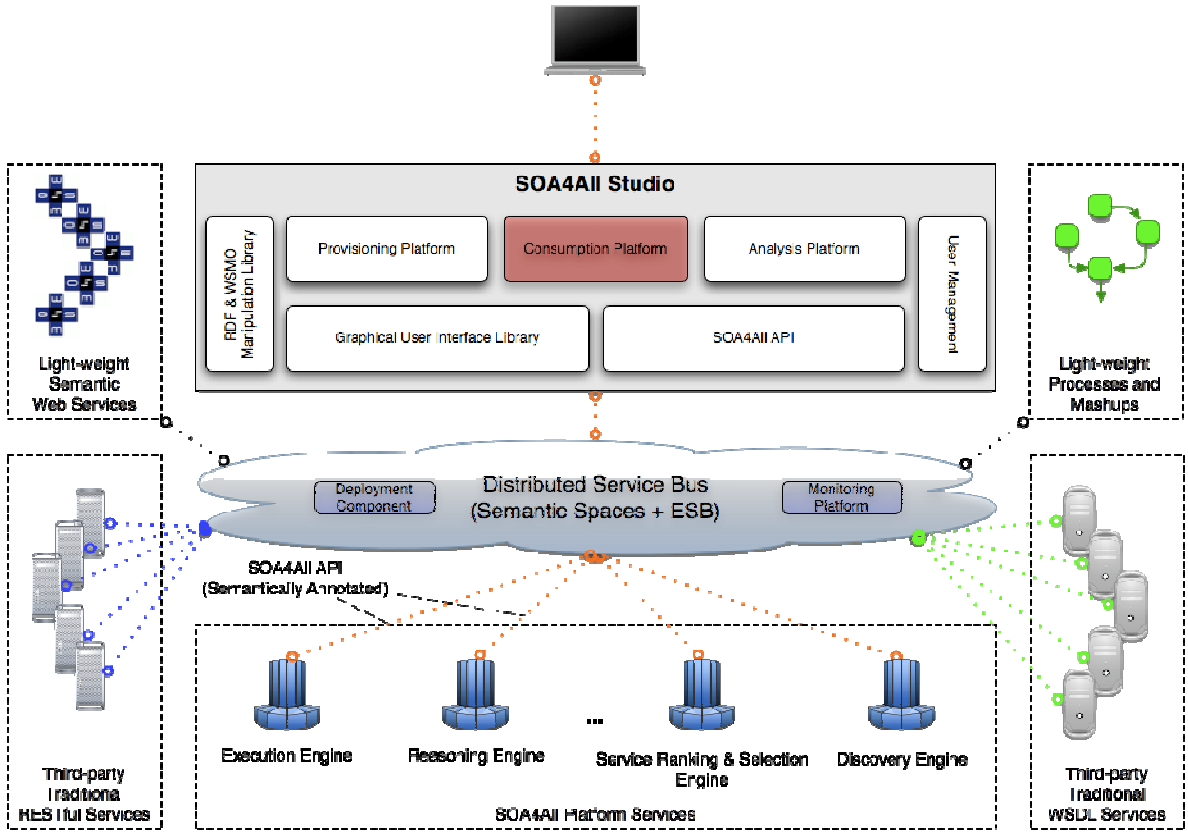

Figure 1: Service Consumption Platform within the SOA4All Architecture

In Figure 1, a high-level view on the overall SOA4All architecture is depicted. The SOA4All Studio stands there as the user gateway to SOA4All, providing a Web-based interface for service provisioning, consumption and analysis. The Service Consumption Platform is therefore a central architectural component of SOA4All, enabling the interactions of users with the service world as service consumers.

In turn, the SOA4All Studio is integrated with the rest of architectural components of the project through the Distributed Service Bus (DSB) described in WP1 [21], depicted as a cloud in the figure, meaning that from the user's perspective, the DSB appears as a cloud of services.

The Service Consumption Platform will provide the means for consuming services previously annotated with the languages defined within WP3. In particular, it will support interactions with WSDL-based services annotated with WSMO-Lite [4], and with RESTful services annotated with MicroWSMO [5]. By using those languages, and through the integration components previously addressed, it will be possible to use the resources that the other work packages make available, such as performing service discovery (WP5) or dealing with composite execution of services (WP6).

Finally, integration of the Service Consumption Platform within the other SOA4All Studio building blocks will be possible thanks to the SOA4All Studio Infrastructure Services and UI Components [20]. We will leverage those integrating services in order to show a holistic graphical user interface, but also to achieve tasks that are common to all of the platforms within the SOA4All Studio, such as managing user profiles or interacting with the repositories in Semantic Spaces.

1.4 Alignment with the Use Case Work Packages

Achieving the objectives of SOA4All requires the delivery of a service consumption interface which is tuned to the skills and tasks of our target users. The long-term aim of SOA4All is to open up service consumption to everyone, yet at first instance our target users are those found in the SOA4All case studies (WP7-9).

Within the current deliverable we shall focus on those aspects of the case studies most relevant to its work – namely the consumption of services.

1.4.1 End-user Integrated Enterprise Service Delivery Platform

WP7 *End-User Integrated Enterprise Service Delivery Platform* use case focuses in the EU Services Directive that targets at facilitating and harmonizing the consumption and provisioning of services within the EU. "Service" in this context means all sorts of economic services and includes consulting, construction, maintenance, advertising, tourism, etc. The Directive's vision is "**to make progress towards a genuine Internal Market in Services so that, in the largest sector of the European economy, both businesses and consumers can take full advantage of the opportunities it presents"**. By supporting the development of a truly integrated Internal Market in Services, the Directive will help realize the considerable potential in terms of economic growth and job creation of the services sector in Europe. For this reason, the Services Directive is a central element of the renewed Lisbon Strategy for growth and jobs. Moreover, by providing for administrative simplification, it also supports the better regulation "agenda". Besides simplifying, accelerating and unifying the administrative processes of all member states, a special focus is on providing services between multiple administrations within a single state or between different states. This requires establishing new communication mechanisms between service providers and administration offices. A major element to accomplish the simplification of administrative procedures from the constituent's point of view is to install so-called single points of contact at the administrations that are responsible for guiding the constituent throughout the entire process and to provide as much help and information as is necessary.

Let us now enumerate the main requirements that we have identified as relevant to the work package

- This scenario specifies more specific user profiles (i.e., users with some domain specific skills in areas of business, economy such as a civil servant) than WP8 and WP9 scenarios. Since such users have limited skills such as:
- Basic WWW skills, able to navigate hyperlinks, fill WWW pages, communicate via email, including document attachments, etc. (for the customer point of view)
- Not a trained IT professional, background in public administration or regulations (civil servant i.e., public administrator)

the consumption platform requires considering them by providing tools ease-of-using.

More specifically civil servants require to:

- retrieve and consume services by using natural language e.g., structured query interface where the civil servant can specify functionality of the service and some context parameters
- browse descriptions of services in order to discover functionalities they require (e.g., by browsing services that are eligible to register business)
- select services (from a list of service specifications) that are relevant (e.g., the cheaper service)
- customize atomic and composite (process-based) services e.g., by feeding

information from service specifications (say documents required) to the Customer via email

- bookmark its favorite atomic and composite services

In addition administrators should:

- edit/explore details of (atomic or composite) services according to a graphical representation of profiles and processes. It will make them
    - Easy to edit/annotate (by browsing ontologies and dragging/dropping concept descriptions), score, comment suggested solutions
    - Easy to share the templates created, together with specifications about the context in which the solution works (helpful for customers with close profiles)

Finally, the scenario attached to WP7 requires to consider context by

- adapting services. For instance potential solutions of the natural language request "Open a new business in Paris within EC" should be an adaptation of solutions of the request "Open a new business in Madrid within EC". The context applicable could be specified as: "For privately owned businesses opening wholly-owned retail subsidiaries in another European country".

In general, the case study focuses on a software-facilitated service at the top level which hides an administrative process stringing together pure Web services and human-enacted services. The role of the service consumption platform within this work package will be the support of the consumption of these Web services and their personalisation to the specific circumstances of each customer. Ease-of-use is achieved by hiding complexity and appropriate design of representations and tool interfaces, this depends on the profile of target end users and the complexity of the target processes.

1.4.2 W21C BT Infrastructure

WP8, *BT W21C Infrastructure* case study aims to provide user-friendly facilities for advanced service discovery, annotation, composition, consumption and monitoring using the existing Web21c infrastructure [22] and BT's newly acquired Ribbit platform[5], exposing core BT communication capabilities. The first scenario is about designing a simple application by composing existing services, and the second scenario is about enabling bulk resellers to implement innovative business ideas using unbranded services provided by BT, or to integrate said services into their business processes.

The Service Consumption Platform will be used by the different profiles stipulated in the case study: (i) "Telecom service enthusiasts", and (ii) "Telecom and social network entrepreneurs". Both of them will be, supposedly, more technically advanced than the civil administrators and customers from the previous case study, but at the same time, they might not have a deep technical background and still want to interact with services as consumers. Ease-of-use in the Service Consumption Platform is therefore a strong requirement, as lightweight methods for discovering and interacting with services are needed.

In terms of generic functionality, we have the following requirements towards the platform:

1. Different (and light) ways of finding the right services, which satisfy the needs of the users who interact with the platform.
2. Interacting with personalised services, and also ranked following contextual factors.
3. Usable without knowledge of ontologies, WSDL or programming languages.

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

4. Web browser-based, drag and drop easy to use interface.

Therefore, the purpose of the Service Consumption Platform within this work package will be to support the process of consuming services in a manner reflecting the personal requirements of the users. Compared to the previous case study, here the technology is more complex and the services are mostly technology-based, which makes this case study equally challenging in terms of end user interface and tool facilities.

1.4.3 C2C Service eCommerce

The WP9 Use Case is focused on the definition of one holistic and real-world oriented C2C eCommerce framework, providing an easy way for end users to use third party services offered through the framework and enabling them to build eCommerce applications. To convert the vision into reality, a C2C service platform under design must be capable to provide discovery facilities and support users with the integration and composition of these services

The potential user of C2C platform will be rather not searching for individual services (e.g., a credit card approval service), but for complete solutions to address his/her business needs (e.g., services to run Web shop): the platform will allow users to customize these services as per their individual requirements, execute them, but also easily integrate these services into their own already existing environments

More in detail, the WP9 end-user is going to use the consumption platform for the goals of finding services and executing them. WP9 will develop a “service broker” based on the Seekda platform adapting their discovery and adaptation process with the Consumption Platform services. This would provide the means of searching for both traditional and semantically annotated Web services.

From a technical point of view, the end-user will be interacting with a dedicated GUI specifically developed within WP9, in order to have a coherent framework. Such GUI will be invoking the Consumption Platform Services.

2. Service Consumption Platform: Overall View

In this section, we will specify the main functionalities that the Service Consumption Platform is going to support (Section 2.1). We will also specify a set of intended use cases (Section 2.2) within the platform, so we will be able to refer to them when describing each of the components.

2.1 List of Functionalities

We cover here the main characteristics of the Service Consumption Platform from the perspective of the functionalities that it will support.

We can identify five main areas of functionalities:

1. **Goal manipulation.** This group of functionalities is related to how end-users will be able to create and access goals: this can happen by choosing pre-defined goals from a repository, or to express them in natural language style or through the use of keywords. It is also possible to define "parametric" goals that can be completed by end-users according to their needs.
2. **Personalisation.** This area covers all such aspects related to how the Platform will grant a personalised access to end-users. Personalisation can be in terms of a dynamic customisation of the elements that the platform displays to an end-user (i.e., the list of suggested goals for him), and in order to achieve this the Consumption platform will need to store and to retrieve contextual information with other SOA4All Studio components.
3. **Service discovery and invocation**. The core group of functionalities that the platform will offer to end-users is related to discovery and to invocation of services, either by relying on a goal or by selecting a particular service.
4. **User Feedback.** This group of functionalities allows users to express their feedback on goals (and retrieve it) thanks to various techniques such as tagging (assigning relevant keywords to them), evaluations (associating textual comments to them) or ratings (to express their usefulness in a scale).
5. **Support.** Finally, some support functionalities are required in order to import and to browse taxonomies (hierarchical structures of terms that will be used to classify services and Goals).

The following Table 1 summarizes the aforementioned functionalities that the Service Consumption Platform will offer for each of the five areas identified – we provide a unique ID for each functionality for future cross-referencing:

Func. ID	Function Area	Description
F#01	**Goal manipulation**	Retrieve Goals from a repository
F#02		Store Goals into a repository
F#03		Lightweight completion of Goals
F#04		Users will be able to express their desires in plain language.
F#05	**Personalisation**	Retrieve context information from platform-based storage

F#06		Request context information from the Recommendation Manager component
F#07		Set up a context-specific filter for supporting service discovery/composition
F#08		Set-up a context-specific filter for finding suitable templates of service composition
F#09		Set-up a context- and template-specific filter for finding suitable service components.
F#10		Use available recommendations to personalise the experience of the user within the platform
F#11		Retrieve recommendations from the Recommendation System by providing user's contextual information
F#12	**Service discovery and invocation**	Discover services that satisfy a Goal
F#13		Invoke services automatically by relying on a Goal
F#14		Invoke a service choosing from a list of services
F#15		Interact with services with instructional assistance (wizards)
F#16	**User Feedback**	Ability to associate tags to Goals
F#17		Ability to retrieve Goals by selecting tags
F#18		Ability to rate Goals
F#19		Ability to retrieve Goal ratings
F#20		Ability to produce reviews on Goals
F#21		Ability to retrieve Goal reviews
F#22	**Support**	Browse through a taxonomy and select several categories
F#23		Ability to import an external taxonomy

Table 1: List of Service Consumption Platform functionalities

In order to better define how these functionalities will be offered to end-users, the following picture (Figure 2) depicts an initial mockup version of the platform. For instance, the focus on Goal manipulation can be recognised by the presence of Goals within the platform (even though the wording will be rephrased to abstract when possible the use of Goals and the machinery that supports the platform). These are used (and hence the "Run" button in the mockup) to discover and invoke services (third set of functionalities), and the resulting consumption, and also the group of "Popular Goals" will be personalised to better satisfy the needs of the users (second set of functionalities). User feedback is present in the stars

(ratings) of the Goals, and also because it is possible to check the comments on them, and the tags associated to them, or provide new ones. Finally, support for taxonomy browsing through categories is also depicted in the left navigation panel.

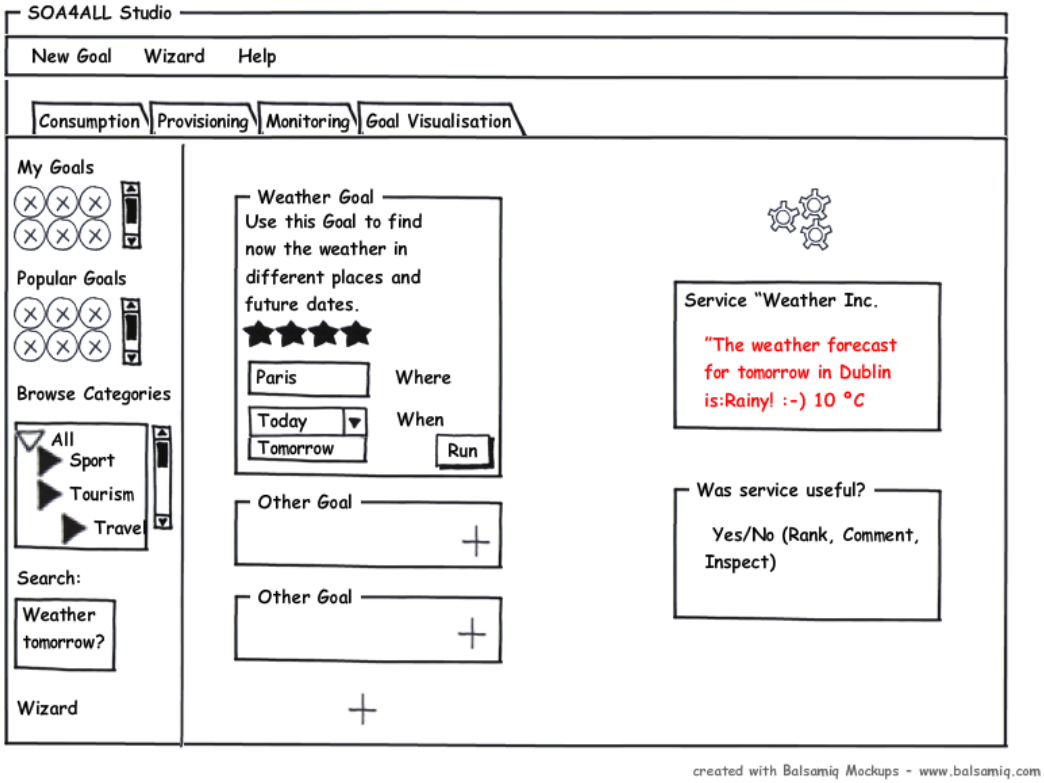

Figure 2: Service Consumption Platform Mockup

2.2 Description of Envisaged Use Cases

In this section, we show the main use cases envisaged for the platform, which are in line with the vision and functionalities previously depicted. These will be useful in the next section, when we describe the components of the platform, as we will refer to these use cases from there.

Some use cases are related to handling predefined Goals found through taxonomy browsing (Uc#01) or with a natural language query (Uc#02) or by selecting tags (Uc#03), and the modification of those is also envisaged (Uc#04). Other use cases treat the adaptation of the platform to suit the user needs (Uc#05) as well as the adaptation of the services themselves (Uc#07). Another use case in the platform is the support for feedback on Goals (Uc#06). The actual discovery of services and their selection is another use case (Uc#08), as well as using a wizard to consume them (Uc#09). Finally, support for importing a new taxonomy is also expected (Uc#10).

The following Table 2 summarises these use cases and shows which functionalities are required in order to support each of them:

Use Case Id	Description	Func. Required
Uc#01	Direct Goal invocation, found via Goal taxonomy browsing	F#01, F#03, F#12, F#13, F#22
Uc#02	Direct Goal invocation, found via NLP query	F#04

Uc#03	Finding Goals through browsing tags	F#17
Uc#04	Modification of Goal, and invocation	F#02
Uc#05	Platform adaptation	F#10, F#11
Uc#06	Feedback on Goals	F#16, F#17, F#18, F#19, F#20, F#21
Uc#07	Service adaptation	F#05, F#06, F#07, F#08, F#09
Uc#08	Discovery of services and choosing which one to invoke	F#12, F#14
Uc#09	Use of wizards to consume services	F#15
Uc#10	Import taxonomy	F#23

Table 2: List of use case descriptions

In order to describe the use cases, we will refer to a set of artefacts that will be part of them. For instance, a taxonomy to categorize de Goals will be needed (GTax), as well as several different Goals (G1, G2,… Gn), which will be WSMO Goals described in WSML. Other artefacts will be Web services (WS1, WS2), WSDL- and REST-based, as well as the semantic enrichments of those (SWS1, SWS2) in WSMO-Lite and MicroWSMO. A tag cloud (GTagC) that reference the Goals will be needed too, and for a particular Goal we will refer to its ratings (G1_r), comments (G1_c) and tags (G1_t). Another elements are a natural language query (Query), and a WSML ontology (Ont1) and a particular concept from it (Conc1) that will be used to refine a Goal description. Regarding the users, we will refer to different profiles (Prof1, Prof2) in order to address personalisation issues in the platform.

In Table 3, we summarize the list of artefacts that we will refer to in the use cases:

Artefact Id	Description	Type
GTax	Goal taxonomy	UNSPSC
G1	Goal	WSML
G2	Goal	WSML
...	Goal	WSML
Gn	Goal	WSML
GTagC	Goal tag cloud	RDF
SWS1	SWS	WSMO-Lite/MicroWSMO
WS1	WS	WSDL/REST
SWS2	SWS	WSMO-Lite/MicroWSMO
WS2	WS	WSDL/REST
Query	Natural language query	String
Ont1	Ontology	WSML
Conc1	Concept	WSML

Prof1	User profile	Profile
Prof2	User profile	Profile
G1_r	Ratings	RDF
G1_c	Comments	RDF
G1_t	Tags	RDF

Table 3: List of use case artefacts

For each of the ten use cases that we intend to cover, we give a brief description, referring to the artefacts, inputs and outputs involved in each of the steps. In Annex A, we also show a very schematic use case diagram and a sequence diagram per use case.

2.2.1 Use Case #1: Direct Goal Invocation, found via Goal taxonomy browsing

A user browses a taxonomy of Goals GTax, until he finds and selects one that is suitable for him (G1), and the Goal is opened. The user completes the missing information and invokes the Goal. He gets the result personalised to his needs.

Artefacts: GTax, G1, G2...Gn, SWS1, WS1
Input: Taxonomy of Goals (GTax)
Output: Results from the execution of a service (WS1)

2.2.2 Use Case #2: Direct Goal Invocation, found via NLP query

A user queries the platform in natural language (Query) and is presented with a list of suitable Goals (G1…Gn). He chooses one of them (G1) and completes it and invokes it as before.

Artefacts: GTax, G1, G2...Gn, SWS1, WS1, Query
Input: Natural language query (Query)
Output: Results from the execution of a service (WS1)

2.2.3 Use Case #3: Finding Goals through browsing tags

A user selects some tags from the tag cloud (GTagC), and a list of Goals matching that criteria is presented, and he chooses one of them (G1).

Artefacts: G1, G2...Gn, SWS1, WS1, GTagC
Input: Tag cloud for Goals (GTagC)
Output: Selected Goal (G1)

2.2.4 Use Case #4: Modification of Goal, and Invocation

A user modifies an open Goal (G1) from the component GUI view and in the code. He adds a new ontology (Ont1) concept to the Goal (Conc1). He saves the Goal and invokes it.

Artefacts: G1, G2, SWS1, WS1, Ont1, Conc1
Input: Goal (G1) + ontology (Ont1)
Output: New Goal (G2) + Results from the execution of a service (WS1)

2.2.5 Use Case #5: Platform adaptation

A user is presented with some recommended Goals. He invokes a Goal and the platform will recommend new services. He logs out and another user with a different profile logs in. He is presented with a different set of recommended Goals.

Artefacts: G1, G2...Gn, SWS1, WS1, Prof1, Prof2
Input: User profiles (Prof1, Prof2), user interactions
Output: Adapted platform

2.2.6 Use Case #6: Feedback on Goals

A user has an open Goal (G1), and checks the comments made on it (G1_c), the global rating it has (G1_r), and the associated tags (G1_t). He also rates, comments on and tags the Goal.

Artefacts: G1, G1_r, G1_c, G1_t
Input: Ratings (G1_r), comments (G1_c) and tags (G1_t)
Output: Updated ratings (G1_r), comments (G1_c) and tags (G1_t)

2.2.7 Use Case #7: Service Adaptation

The service the user plans to invoke is adapted according to its particular context.

Artefacts: GTax, G1, WS1, Prof1
Input: Goal (G1), user profile (Prof1)
Output: Context based consumption of a service (WS1)

2.2.8 Use Case #8: Discovery of services and choosing which one to Invoke

A user applies a Goal (G1) to discover services, and then he chooses which one to interact with (SWS1).

Artefacts: G1, SWS1, SWS2, WS1, WS2
Input: Goal (G1)
Output: Results from the execution of the selected service (WS1)

2.2.9 Use Case #9: Use of Wizards to consume services

A user applies a Wizard to find and define a Goal and consume a service.

Artefacts: G1, SWS1, WS1
Input: Wizard selections
Output: Results from the execution of a service (WS1)

2.2.10 Use Case #10: Import taxonomy

A user imports a new taxonomy (GTax) in order to use it within the platform.

Artefacts: GTax
Input: External taxonomy (GTax)
Output: Loaded taxonomy (GTax)

3. Service Consumption Platform Design Overview

In this section, we will cover the main technical details and characteristics of the Service Consumption Platform. We begin with an overview on its general architecture in Section 3.1, and then address the main technical details in Section 3.2. We will cover each of the architectural components in detail in Section 4.

3.1 Architecture Overview

In Section 1.3, we have addressed the main characteristics of the project, focusing on the most relevant issues for the SOA4All Studio, and which are the roles of the different platforms and components involved in it. It is worth noting the central position of the Service Consumption Platform within the lifecycle of services. It is in this platform where the users interact with the services as “service consumers”, discovering and invoking them, after the necessary annotations to services have been provided in the Service Provisioning Platform, and taking place the analysis of these executions both at runtime and after finished.

Figure 3**Error! Reference source not found.** highlights how the Service Consumption Platform makes use of the Infrastructure Services provided by the SOA4All Studio. These, in turn, make use of the underlying architectural components and resources provided by WP1 as part of the SOA4All Runtime, including the different repositories in Semantic Spaces, where the relevant data will be stored.

The figure represents five major logical layers, which are defined at a logical level in the next subsections, containing different components of the platform.:

- **Goal Manipulation Manager**, which stresses the importance of Goals and enables their use within the platform.
- **Personalisation Manager**, which uses contextual factors in order to provide a personalised experience to users.
- **Service Discovery and Invocation Manager**, which deals with the interactions of users with the platform to consume services.
- **User Feedback Manager**, which supports ways for the users to provide feedback on the Goals.
- **Taxonomy Support Manager**, which provides additional resources to deal with taxonomies..

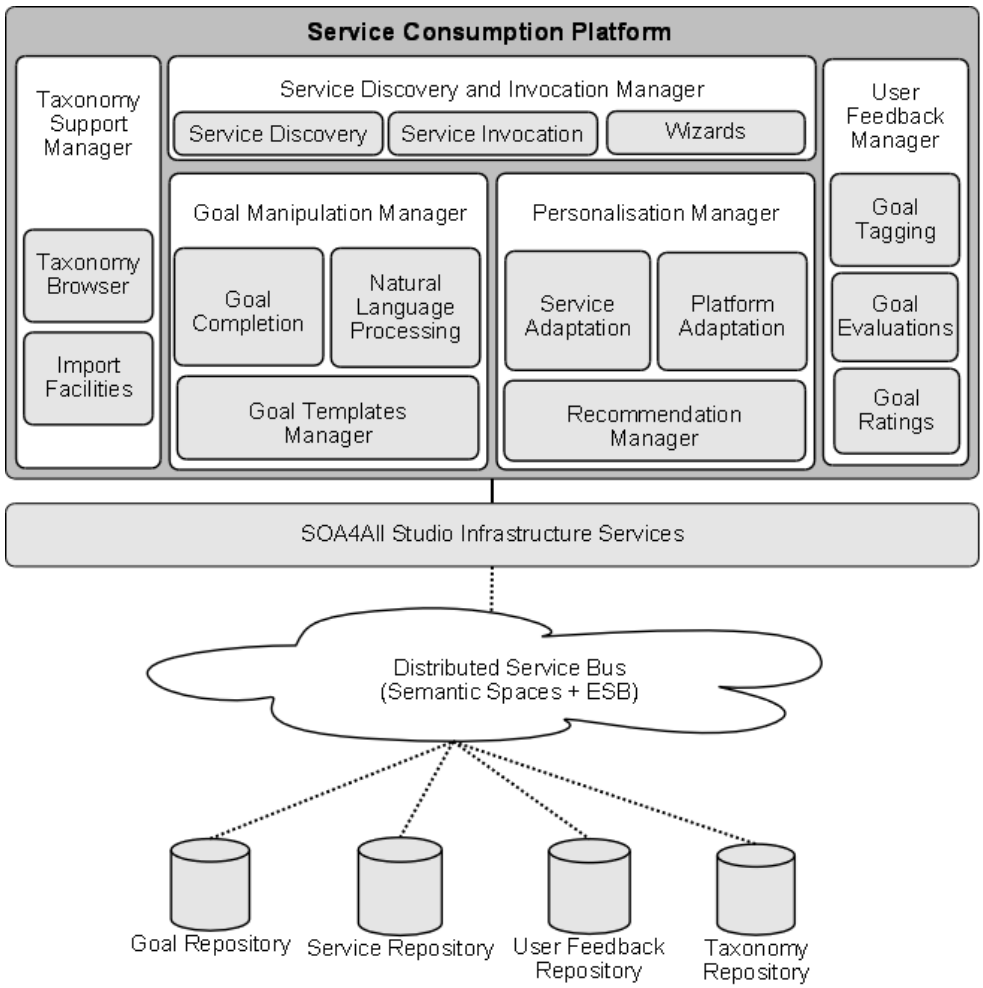

Figure 3: Service Consumption Platform Architecture Overview

Components shown within each layer are actual software components: their detailed functionalities, input/outputs and the way they communicate with each other is detailed into Section 4. Additionally, as shown in Figure 3**Error! Reference source not found.**, the Consumption Platform is accessing information from the following repositories in Semantic Spaces, as we will see:

- Goal Repository, where the Goal definitions (previously transformed from WSML into RDF) will be stored.
- Service Repository (a WP5 component on its own, which in turns interacts with the Semantic Spaces), from where we will retrieve the semantically enriched services (WSMO-Lite and MicroWSMO service descriptions).
- User Feedback Repository, where the tags, ratings and evaluations of Goals will be.
- Taxonomies Repository, where we will include an initial taxonomy, but which will allow users to include new ones.

It is worth noting that the interaction with these repositories won't be achieved directly but taking advantage of the Storage Services provided as part of the SOA4All Studio Infrastructure Services [20].

Finally, the outputs of the Service Consumption Platform will be the executions of the services and thus the corresponding logs that the Service Analysis Platform will use.

3.1.1 Goals Manipulation Manager

As we have mentioned before, we make a huge emphasis on the use of Goals within this platform, in order to allow users to find the most suitable services for their needs. Hence, a central layer of the platform is the Goal Manipulation Manager, which includes different components that support the use of Goals as an efficient and decoupled mechanism to find services. We will begin by making use of plain-WSMO Goals in order to allow users to express their desires formally, and will extend them in order to support lighter Goals and even not completely defined (open) ones, aligning our research with the notation developed within work package 3.

Central to the Goal manipulation layer is the Goal Templates Manager component, which supports the interaction of the platform with stored Goals. It will make use of the Storage Services provided by the SOA4All Studio, which in turns interacts with the WP1 facilities, in order to store and retrieve the desired Goals and information about them.

Another component, the Goal Completion component, is in charge of enabling different ways for the users to modify the available Goals in order to define with more precision their desires. Coming from previously existing Goals, service consumers will be able to refine the expressions of what they need (the Goals) by modifying them graphically, or through their code.

In a similar way, the Natural Language Processing (NLP) component will be in charge of transforming the users desires expressed in natural language into formal Goal representations. We will make use of and extend Knowledge Tagger (KT), which in turns uses GATE (General Architecture for Text Engineering, [15]), the Language Technology tool, in order to match the informally expressed sentences (i.e., in natural language) of the users into formally defined and structured Goals.

3.1.2 Personalisation Manager

Another very important aspect that we have also stressed within the Service Consumption Platform is the characteristic of personalisation. We believe that presenting a personalised experience to the users is the better way to satisfy their needs in the Service Web such as the one envisioned by the project, where the potentially huge number of services will make it difficult to find the most suitable in each case.

We will make use of contextual factors, as introduced by D4.1.1 [10], in order to enrich the interactions of the user with the services. Additionally, the contextual information will be able to enrich the interaction with the platform itself. The Personalisation Manager that we will develop will take care of the different kinds of personalisation envisioned, and will consist of a set of logical interrelated components.

The Recommendation Manager component will bring into the platform the results from the Recommendation System that will be developed in Task 2.7: the goal of this component is to provide recommendations (or suggestions) to users by exploiting similarities *among users* (i.e. suggest services that have been used by similar users) and by exploiting similarities *among users and services* (i.e. suggest services that have been used by similar users and that are similar to the current service).

The Service Adaptation component will focus on presenting the users a personalised (and more suitable) version of the services that they interact with. Contrary to the Service Adaptation component of WP6, which focuses on adaptation at “run time” (e.g., in case that some service is no longer available), here we focus on service adaptation regarding the context, the user profile as well as the recommendation given by the recommendation system “at design time”. To this end user profiles, preferences and recommendations will be analysed to detect information that can be re-used in order to support service consumption. In other words end users will not be requested to provide information they previously provide

(for instance by means of preferences in their profiles) or the information that can be retrieved by recommendation. Therefore, services requested by end users will be personalised according to their profiles or tier recommendation. For instance, a Hotel booking service could be personalised by enabling automatic selection of language according to preferences registered in the end user profiles.

Additionally, the Platform Adaptation component will take the output of the Recommendation Manager in order to change certain elements (such as the set of recommended Goals) in the platform, thus providing the users with a personalised version of it. For instance, platform functionalities such as Goal categories or user's suggested Goals will change according to the contextual factors of the user.

3.1.3 Service Discovery and Invocation Manager

The end-users (SOA4All users acting as "service consumers") of this platform will have the objective of interacting with services (discovering and invoking them); and that is the reason for the Service Discovery and Service Invocation component on top of other building blocks. As we have pointed out, service consumers will generally perform these actions by defining their intentions in formal Goals, and their consumption will be enhanced by contextual factors that will make possible a personalised experience for them.

Different components will expose their functionality allowing users to achieve these tasks: The Service Discovery component, and the Service Invocation component. These will use and extend GUI components from the SOA4All Studio, but will focus on the underlying functionality.

Additionally, we will build a set of intelligent wizards, which will further lower the entry barrier to non-technical users, guiding them through the necessary steps to complete the configuration of Goals and their use to discover and interact with services. The Wizards component will cover these functionalities.

3.1.4 User Feedback Manager

Particularly important to our vision on the importance of the Goals are the feedback mechanisms supported by the platform. Service consumers will be able to produce feedback on Goals in similar ways to those which are popular in Web 2.0 sites: (i) by providing light annotations through tagging (attaching key words considered relevant for describing the Goals), (ii) by providing comments or evaluations on the Goals, and (iii) by rating them. We will make use of the Feedback Management Framework of the Service Provisioning Platform [11] in order to enable these kinds of feedback.

3.1.5 Taxonomy Support Manager

The Service Consumption Platform will support ways to deal with taxonomies which other components within the platform will be able to use. Particularly, this logical group includes a Taxonomy Browser component that will allow users to browse through a particular taxonomy, selecting particular items, and an Import Facilities component, which allows users to import any other desired taxonomy into the system, enabling a later interaction with them trough the platform.

3.2 Technical Description of the architecture

From the technical point of view, we will develop the platform using the Java language, and Ext-GWT for the Web side, when necessary, following the general design guidelines explained in DX-UI [9]. Thus, the components depicted in the previous section, summarized here in Table 4, highlighting their relationship with the functionalities of the platform, will communicate with each other by means of their exposed functions with direct Java calls. In Figure 4, we represent this internal communication as well as the different types that will take

place within SOA4All.

Layer	Comp. ID	Component Name	Func. covered
Goal Manipulation Manager	Comp#01	Goal Templates Manager	F#01, F#02
	Comp#02	Goal Completion	F#03
	Comp#03	Natural Language Processing	F#04
Personalisation Manager	Comp#04	Service Adaptation	F#05, F#06, F#07, F#08, F#09
	Comp#05	Platform Adaptation	F#10
	Comp#06	Recommendation Manager	F#11
Service Discovery and Invocation Manager	Comp#07	Service Discovery	F#12
	Comp#08	Service Invocation	F#13, F#14
	Comp#09	Wizards	F#15
User Feedback Manager	Comp#10	Goal Tagging	F#16, F#17
	Comp#11	Goal Evaluations	F#18, F#19
	Comp#12	Goal Rating	F#20, F#21
Taxonomy Support Manager	Comp#13	Taxonomy Browser	F#22
	Comp#14	Import Facilities	F#23

Table 4: Service Consumption Platform list of components

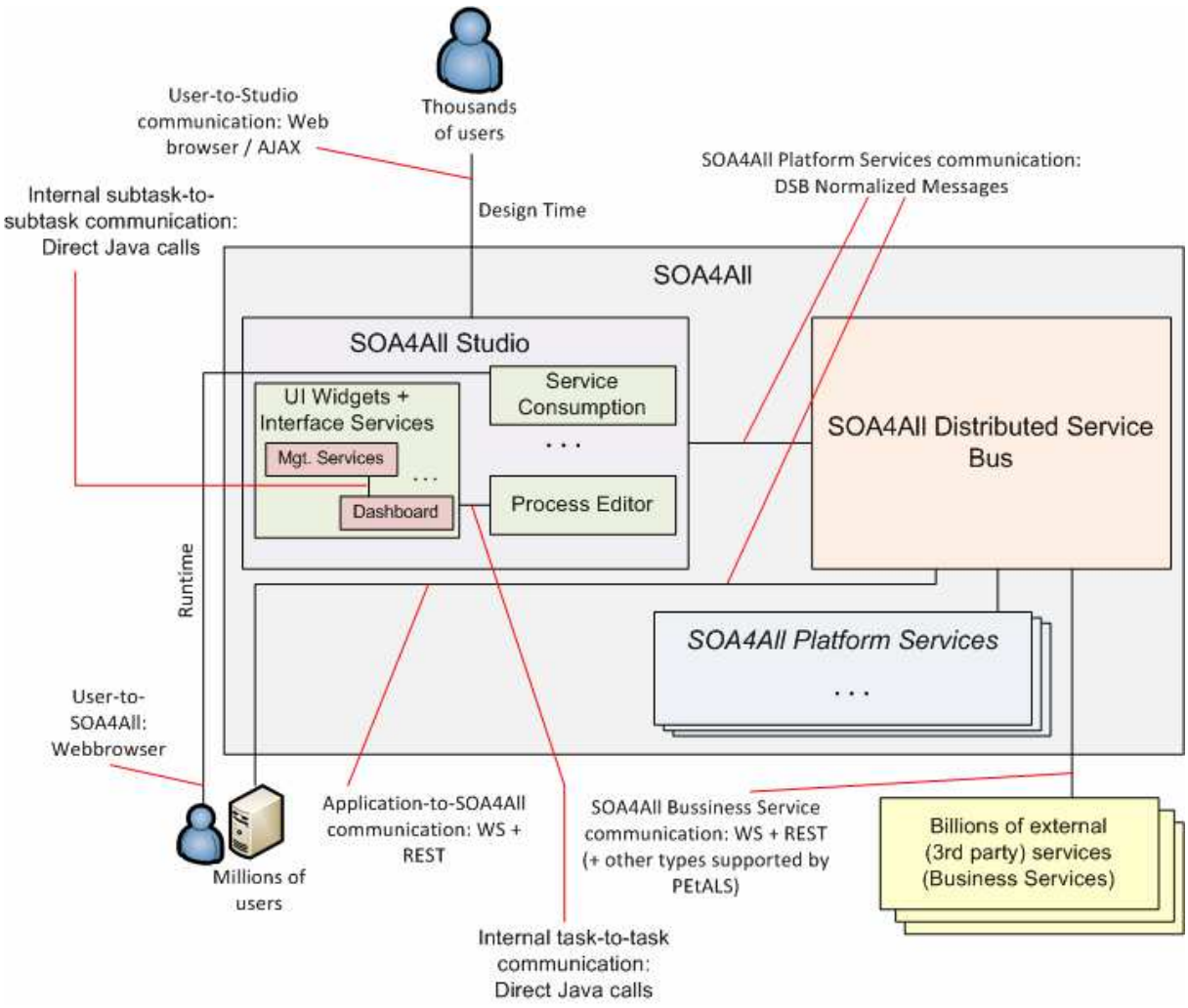

Figure 4: Different types of communication within SOA4All

It is worth noting that these components will make use of underlying facilities provided by the SOA4All Studio [20], both from a graphical point of view (UI Components) and as a support for functionalities (Infrastructure Services). Those functionalities from the SOA4All Studio will be exposed via an API, which our components will call conveniently via direct Java calls.

Besides, the Service Consumption Platform uses some architectural components from the SOA4All Runtime (covered by WP1), such as the Semantic Spaces [23] in which we will store and retrieve many artefacts like Goal descriptions or user ratings. The architectural facilities from WP1 also allow us to interact with the rest of components developed in other work packages, such as Discovery component (WP5), via DSB normalised messages.

Finally, it is important to note that the Service Consumption Platform itself exposes its functionalities through an API, making it possible for additional frameworks and resources to access and adapt the characteristics of the platform conveniently. For instance, the Case Study work packages may present an adapted platform for service consumption, invoking the necessary functionalities covered by additional features that they might need.

4. Service Consumption Platform: Detailed Architecture

In this section, we cover the main technical details of the architectural components previously mentioned in Section 3.1. We will refer to the functionalities listed in Section 2.1 and use the particular use cases addressed in Section 2.2 in order to show more explicitly the functionality they address. We will also cover their technical specification that will allow us to develop them to satisfy the needs of the platform.

We will describe the technical details of each of the platform components, grouped in the five logical sets in which we have structured the platform.

4.1 Goal Manipulation Manager

In the Service Consumption Platform, a hypothetical user could transform his non-structured desires (e.g., book a hotel in Rome) into formal a Goal (WSML file containing that information, even though the non-technical user will not need to know about those technicalities) that would match a hypothetical hotel booking service. We will review the different components involved, explaining the different ways the users will have to achieve their purposes.

The architectural components involved that we will cover in the following subsections are:

1. **Goal Templates Manager** component. This component will interact with a repository of Goal templates, so it will make them available for use in the platform.
2. **Goal Completion** component. This component will permit to represent the user's desires in formally specified Goals by filling some gaps or choosing options from drop-down menus, etc., thus completing abstract Goal templates.
3. **Natural Language Processing** component. This component will transform the informally expressed desires of the users into formal Goals by making use of Natural Language Processing techniques.

4.1.1 Goal Templates Manager Component

Description

The Goal Templates Manager component will interact with the Goal Repository in order to retrieve the Goal templates from there. Strictly speaking, the Goals will be stored in the Semantic Spaces made available by the SOA4All Runtime in WP1 [21], and this component will interact with it making use of the Storage Services that the SOA4All Studio makes available as Infrastructure Services. Technically, the sequence diagrams in Annex A (Figure 11, Figure 13, Figure 15 and Figure 17) show an abstraction of what happens between the Goal Templates Manager component and the Goal Repository, because the interaction will be through the invocation of the Storage Services.

Additionally, this component will enable an interaction with the Goal Repository in the other direction, so templates created by users of the Service Consumption Platform can be stored and made available for future use by other service consumers.

Component Id	Comp#01
Functionalities covered	F#01, F#02
Related use cases	Uc#01, Uc#02, Uc#03, Uc#04
Components used	SOA4All Studio Infrastructure Services: Storage Services

Table 5: Goal Templates Manager component summary

The following diagram (Figure 5) summarizes the process that will be followed in order to interact with the Goal Repository. Similarly, in the following components that interact with different repositories in Semantic Spaces, we will make the same abstraction.

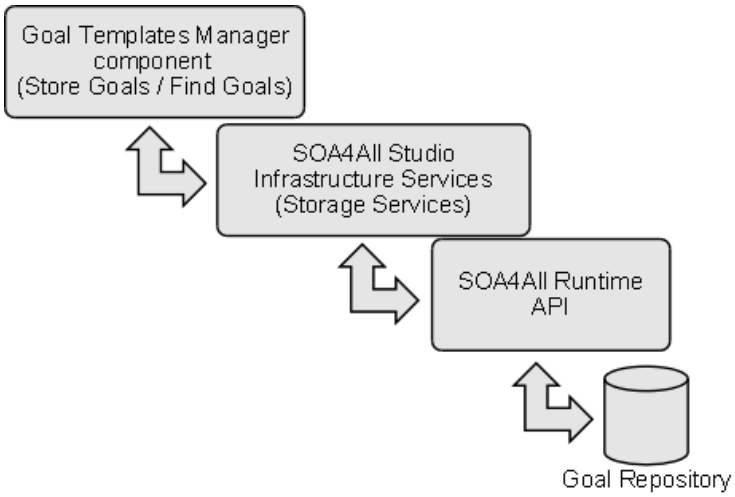

Figure 5: Interaction with the repository through the Storage Services

This component will also be in charge of deriving generic Goals from semantic descriptions of services (in WSMO-Lite and MicroWSMO), making them available into the repository, in order to have semi-automatically generated Goals there that can be used by service consumers.

Technical Specification

The follwing functions will be the main outcome of this component:

1) `StoreGoal(Goal): Int`

   This function receives a Goal expressed in WSML as a parameter, and stores it conveniently into the repository. It returns '0' on success or an error.

2) `FindGoals(taxonomy selection) and FindGoals(tags): Goal[]`

   This function returns a set of Goals, and it can receive either a selection in a taxonomy or a set of tags as input, for there will be different ways to indicate what kind of Goals should be retrieved.

3) `DeriveGoal(WSMO-Lite desc) and DeriveGoal(MicroWSMO desc): Goal`

   This function will derive generic WSMO Goals from service descriptions in WSMO-Lite and MicroWSMO.

4.1.2 Goal Completion Component

Description

The Goal Completion component will enable several ways by which the users will be able to refine the definitions of the Goals, in order to perform more accurate and suitable discoveries of services. This component will support both the modification of Goals by changing their concepts and attributes, and a lightweight and graphical way as well.

The Template Completion component will rely on Goal Templates Manager component previously described in order to retrieve the Goal templates that can be completed in a lightweight fashion. Following this approach, service consumers will be able to express their

needs without having any special knowledge on the platform, by completing specially designed Goals in one of the following ways:

- Choosing from a set of options, by the use of drop-down menus. This will be possible if during the creation of the Goal template several possible values are specified for a particular parameter. The service consumer could use one of these templates by selecting the most suitable option amongst the predefined set.
- Filling some missing information, in established gaps. In this case, the design of the Goal template would need to leave a predefined gap, without the need of specifying possible values that the respective parameter could take, as in the previous way.

In order to make this “completion” possible, we will explore how to expand the traditional plain WSML notation, enriching them with “open parameters”. We begin by considering the two aforementioned possibilities (set of options, predefined gap), as we will see in the technical specification.

Additionally, more advanced users will be able to modify Goals by dragging and dropping concepts into them, placing further constraints on the Goals. We will make use of Watson [24] (an infrastructure component for the Semantic Web that provides functions for interacting with semantic elements) for keyword-based search of WSMO ontologies in the Internet. We will also use WSMO4J [25], the reference implementation of the WSMO API, which enables the creation and manipulation of WSMO objects.

Component Id	Comp#02
Functionalities covered	F#03
Related use cases	Uc#01, Uc#02, Uc#04
Components used	Goal Templates Manager

Table 6: Goal Completion component summary

Technical Specification

The following functions will be exposed by the component:

1) `ShowGoal(Goal)`

This function will translate the code of the Goals, which may be not completely defined, into a graphical view that shows the ways to complete them. While we consider here a first attempt to show a notation for the parts to be completed, we will explore and refine it in the future.

As we mentioned before, we consider two ways by which a Goal definition might be left “open”: (1) By showing a gap that has to be completed with any arbitrary content, or (2) by giving some options from which the users have to choose. This component will translate the “open” Goal definitions into the graphical elements to be presented to the users. For example, from the code shown in Listing 1, the component will generate a graphical representation that the users will be able to complete, as depicted in Figure 6.

```
GoalgoalGetTimetableTrainsFromParis

...

instance testGetTimetable memberOf reqGetTimetable
  origin hasValue _"http://example.com/frenchLocations#paris"
```

```
 dateOfTravel hasValue %%[When][_date]%%
 destination hasValue
%%[Traveling to][" http://example.com/frenchLocations#frenchCity"]%%
```

Listing 1: Abstract Goal example

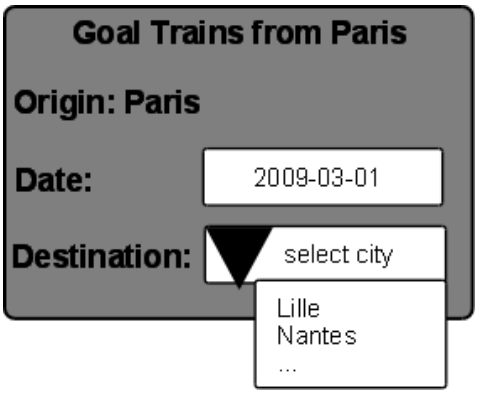

Figure 6: Goal mockup GUI

2) `FillGap(Goal, gap id, content): Goal`

As we have seen, the user will be presented with ways to graphically complete a Goal. This function is complementary and takes care of modifying the Goal when a user fills one of the open gaps, adding the chosen refinement to the Goal.

3) `ChooseOption(Goal, option id, selection)`

Similarly to the previous function, this one modifies the Goal when a user selects one of the options from a drop-down menu.

4) `AddConstraint(Goal, concept, value)`

It permits introducing new concepts by modifying the code of the Goal, or by introducing them from the taxonomy browser by copy-and-paste or drag-and-drop.

4.1.3 Natural Language Processing Component

Description

We will use Natural Language Processing (NLP) in order to abstract users' desires into formalised Goals, enabling the interaction of non-expert users with services, as they will not even need to actively look for a Goal from which formalise their desires. This component will allow them to express their needs in non-formal language (in natural human language), structuring it with the help of underlying ontologies for this specific purpose.

NLP techniques cover a large area that range from automated generation and manipulation to analysis of natural (or human) languages. For the purposes of this component, we will focus in the Information Extraction subset (a large research area [13] where many tools, such as GATE [15], have already been developed), where the input is a body of natural text and the output is a structured and defined data format.

In particular, we will integrate iSOCO's "Knowledge Tagger" (KT) component and a general purpose ontology about the Goals that will be developed purposefully for this platform. By using KT with a specific ontology of Goals, the NLP component will be able to return instances of particular Goal categories, which, in turn, will be used to find Goals making use of the underlying Goal Templates Manager component, as depicted in Figure 7.

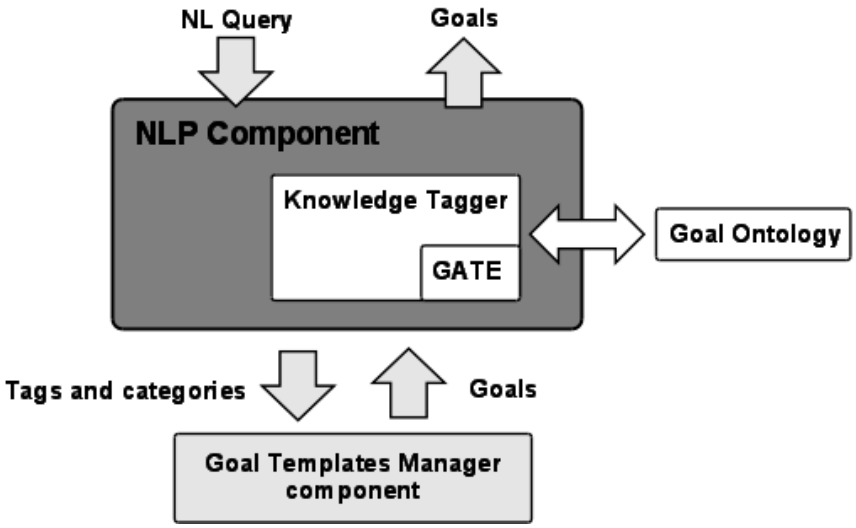

Figure 7: Natural Language Processing component overview

Component Id	Comp#03
Functionalities covered	F#04
Related use cases	Uc#02
Components used	Goal Templates Manager

Table 7: Natural Language Processing component summary

Technical Specification

This component will show the following function:

1) `FindGoalsNLP(string): Goal[]`

The process of finding Goals with a natural language query will begin by tokenizing the string and matching the tokens against a general purpose ontology in order to find some concepts to be used in the next step. There, we will use the relevant concepts as tags and categories when invoking the `FindGoals` function from the Goal Templates Manager, from which we will find the list of Goals that are suitable for the particular expression.

The general purpose Goal ontology that we will use will be aligned with the default taxonomy used for categorization of Goals, UNSPSC (see Section 4.5.1), and we will expand and refine it to offer improved results.

4.2 Personalisation Manager

The Personalisation Manager layer is responsible for both Service and Platform adaptation. In more detail, it presents a personalised view of services and the platform by considering context information. In that way we aim at satisfying requirements from the Service Web, as envisioned by the SOA4All project, where the potentially huge number of services will make it difficult some processes such as relevant and efficient discovery of suitable services.

The following picture (Figure 8) shows that this layer is composed by three main components (detailed in the following paragraphs): the “Service Adaptation”, the “Platform Adaptation” and the “Recommendation Manager” components:

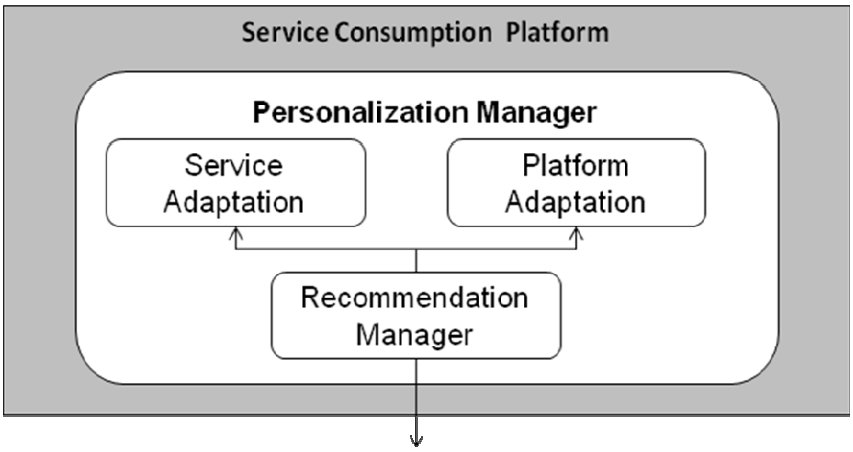

Figure 8: The “Personalisation Manager” layer

More in particular, we will see that the “Recommendation Manager” is mainly responsible for gathering services and goal recommendation information from the other modules of the SOA4All studio, while the “Platform Adaptation” and the “Service Adaptation” components are responsible for providing high-level services that can be invoked in order to personalise the platform. These two components are exploiting the functionalities offered by the “Recommendation Manager”.

4.2.1 Service Adaptation Component

Description

The Service Adaptation component handles the adaptation of services to the context of their use, using information from the Recommendation Manager component, and other available context-specific information. The vision of just-in-time adaptation to context is in unison with the core concepts of semantic services, and it relies on late binding of service components into a composite service identified by its Goal. When a user selects a Goal, or customises a Goal, the context information available at the time will constrain the search for suitable services, and it can also provide “default” values for service input parameters thus parametrising services which are context-aware. If no suitable services are found, the same context-constrained search will be applied to the available service composition templates, and the service components specified by them. The set of suitable services; or service templates and components, will be then passed to the service composition modules developed by WP6.

Component Id	Comp#04
Functionalities covered	F#05, F#06, F#07, F#08, F#09
Related use cases	Uc#01, Uc#02, Uc#03, Uc#04, Uc#07
Components used	Recommendation Manager component

Table 8: Service Adaptation component summary

Technical Specification

The following functions specify the technical details of the functionality and inter-component dependencies of the service adaptation component.

1) `retrieveStoredContext: ContextInformationSet`;

This function will retrieve the set of values for relevant context parameters stored in the current platform.

2) `retrieveCommunityContext: ContextInformationSet`;

This function will retrieve context information from the Recommendation Manager component.

3) `setupServiceFilter(storedContext, communityContext): ServiceFilter`;

This function will fuse storedContext and communityContext information in a ServiceFilter to be used when relevant services to satisfy a Goal are retrieved.

4) `setupTemplateFilter(storedContext, communityContext): ServiceFilter`;

This function will fuse storedContext and communityContext information in a ServiceFilter to be used when relevant service composition templates to satisfy a Goal are retrieved.

5) `setupServiceComponentFilter(serviceTemplate, storedContext, communityContext): ServiceFilter`;

This function will fuse storedContext and communityContext information in a ServiceFilter to be used when relevant service components matching a serviceTemplate are retrieved.

4.2.2 Platform Adaptation Component

Description

The Platform Adaptation component is responsible for providing an automated platform personalisation based on user's context. More in particular, this component will provide a personalised view on Goal categories and will suggest best goals for a given user. This component will exploit the functionalities provided by the "Recommendation Manager" component, making these available as high-level services.

Component Id	Comp#05
Functionalities covered	F#10
Related use cases	Uc#05
Components used	Recommendation Manager component

Table 9: Platform Adaptation component summary

Technical Specification

Functions:

1) `GetUserGoalCategories (user_id) :`

It returns a list of personalised goal categories for a given user

2) `GetUserRecommendedGoals (user_id) :`

It returns a list of suggested goals for a given user

4.2.3 Recommendation Manager Component

Description

The Recommendation Manager component is in charge to bring into the platform the results from the Recommendation System that will be developed in Task 2.7. The goal of this component is to provide recommendations (or suggestions) to users by exploiting similarities *among users* (i.e., suggest services that have been used by similar users) and by exploiting similarities *among users and services* (i.e., suggest services that have been used by similar users and that are similar to the current service).

The following picture (Figure 9) provides an overview on how the two platforms will exchange information (from a logical point of view):

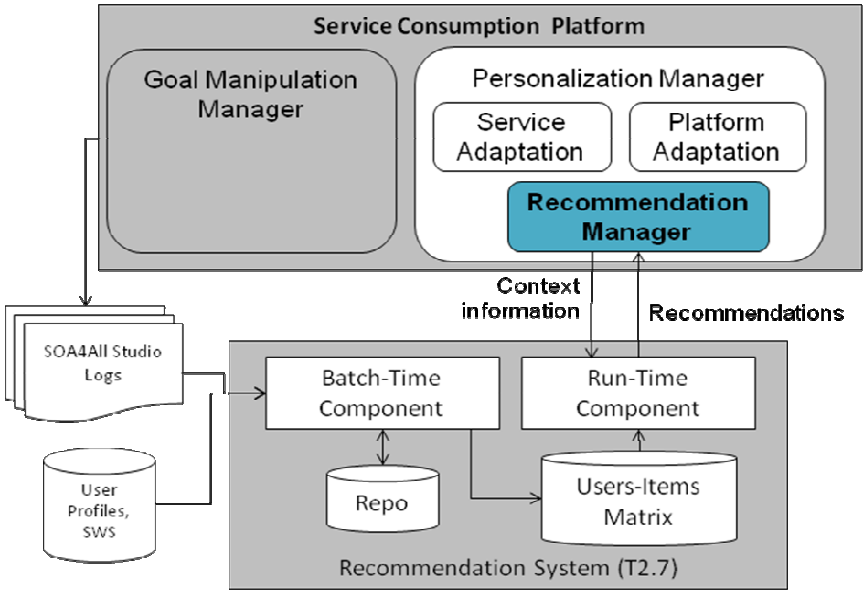

Figure 9: Relationship between the Consumption Platform and the Recommendation System

As we can see, the Recommendation System requires three types of information:

1. Logs produced by Users in interacting with the Consumer Platform (like TimeStamp, User-ID, Action-ID, Additional info describing the action), necessary to analyse users' behaviour at batch time
2. Relationships amongst users and relationships amongst services, still necessary at batch time
3. User run-time context information (user ID), necessary to provide the correct information

The first type of input is generated by the Consumption Platform and it is logged by the SOA4All Studio Infrastructure Services in the Semantic Spaces. The second type of information is available in the semantic repositories. The third type of information should be provided by the Recommendation Manager every time it requires some recommendations. The Recommendation Manager can obtain this information from the SOA4All Studio Management Services.

The Recommendation Manager represents thus a gateway for the Consumption Platform, in order to obtain recommendations on services and goals every time these are required (run-time).

Component Id	Comp#06
Functionalities covered	F#11
Related use cases	Uc#05
Components used	Service Adaptation Component Platform Adaptation Component SOA4All Studio Infrastructure Services (DSB) → Recommendation System SOA4All Studio Infrastructure Services (Storage services) → User profiles, Goals repositories

Table 10: Recommendation Manager component summary

Technical Specification

Functions:

1) `GetRecommentedGoals(user_id)` and `GetRecommentedGoals(user_id, goal_id):`

It returns a list of recommended goals for a given user. Additionally, a Goal Id can be passed as an argument in order to get recommended Goals similar to a particular one.

2) `GetRecommendedServices(user_id)` and `GetRecommendedServices(user_id, service_id):`

It returns a list of recommended services for a given user. Additionally, a Service Id can be passed as an argument in order to get recommended Goals similar to a particular one.

3) `GetRecommendedGoalCategories (user_id)` and `GetRecommendedGoalCategories (user_id):`

It returns a list of recommended goal categories for a given user.

4.3 Service Discovery and Invocation Manager

While the previous two logical sets of components address some of the most important characteristics of the Service Consumption Platform (i.e., handling Goals and dealing with Personalisation), it is important to note that the objective the users will have when interacting with the platform –as its name already points out– will be to *consume services*. Hence, the ability to manipulate Goals and the way the context will be able to enrich the experience of the user, are covered by a layer closer to the user, where they will be able to discover services and invoke them (and thus consume them), making use of the underlying components.

We will consider three different interrelated components:

- **Service Discovery** component, which takes care of finding suitable, making use of the SOA4All Discovery services.
- **Service Invocation** component, which is used to actually consume the service.
- **Wizards** component. The burden involved in the whole process of interacting with a service by creating Goals can be lightened by the use of pre-defined Wizards that

help with this task, which are possible thanks to this component.

It is worth noting here that even though we foresee these components as the entry points for the users, we will make use of the specific UI Components developed in T2.4 [20][9] in order to present a coherent and holistic graphical user interface. We place special emphasis here on the functionality that these components provide, and we will detail how they will expose it.

4.3.1 Service Discovery Component

Description

This component will enable ways for a user to discover services based on a Goal where he has defined his desires thanks to the Goal Manipulation components. We will align this component with the relevant Discovery and Location component from WP5, where both a full text based discovery (i.e., using keywords) and a semantic discovery (i.e., using Goals) are envisaged. Also, the Ranking and Selection component from WP5 will be used, permitting us to have a more suitable and ordered selection of services.

Component Id	Comp#07
Functionalities covered	F#12
Related use cases	Uc#08
Components used	WP5 Discovery and Location WP5 Ranking and Selection

Table 11: Service Discovery component summary

Technical Specification

This function will be exposed in order to allow users perform a Goal-based discovery of services.

1) `DiscoverServices(Goal) and DiscoverServices(keyword[]): Service[]`

   It uses the WP5 facilities for discovering services, based on semantic definitions or keyword-based, in the simpler case.

4.3.2 Service Invocation Component

Description

This component will enable ways for a user to invoke services based on a Goal where he has defined his desires thanks to the Goal Manipulation components.

Component Id	Comp#08
Functionalities covered	F#13, F#14
Related use cases	Uc#01, Uc#02
Components used	Service Discovery component WP6 Execution Engine component

Table 12: Service Invocation component summary

Technical Specification

The following function will be available to invoke services.

1) `LinkGoalService(Goal, Service): Selected service`

It will permit associating a particular service with a Goal that a user is dealing with. It implies having used the Service Discovery component before, and selecting a particular service from the resulting set.

1) `Invoke(selected service) and Invoke(Goal): Service execution`

The invocation of a service will be possible in two different ways: Either by having selected the service beforehand, or by using a Goal and delegating the choice of the service to the discovery engine, which will select the most relevant from the characteristics of the Goal.

4.3.3 Wizards Component

Description

The Wizards component will help non-technical users to define Goals and use them to discover and invoke services without the need of having the knowledge to define them from scratch or using templates. In this case, pre-designed software assistants will help service consumers to specify their objectives by completing a sequence of helpful steps. In fact, the wizards enabled by this component will be on top of the previously described components, guiding the user on how to use them successfully, reducing errors and ensuring their correct use. By making lighter ways of finding the proper services, more users will be able to interact with the service world envisaged by SOA4All.

The Wizards component will enable wizards as an instructional help to guide the user through a series of multiple-choice questions to define formal Goals, and then use them. In order to have efficient wizards, an intelligent sequence of steps and clear questions will be defined and stored in a structured way.

Component Id	Comp#09
Functionalities covered	F#15
Related use cases	Uc#09
Components used	Service Discovery component Service Invocation component Natural Language component Goal Completion component

Table 13: Wizards component summary

Technical Specification

The following function will enable the wizards.

1) `InteractWizard(): Goal configuration, service discovery and invocation`

A wizard will consist of a series of steps coded into structured ways in XML format. They will be stored into XML documents (XML files). These files will be parsed by this

component and present the relevant instructions to the user, on a draggable box on top of the other elements in the platform, in a similar way to the Service Provisioning Platform wizards (D2.1.1 [11] , Section 4.3.4)

4.4 User Feedback Manager

Within the Service Consumption Platform, we put a strong emphasis on Goals, and thus it makes sense that our users give feedback on them. After all, “Service consumers” in our platform will also be “Goal *prosumers*”, and it will be quite beneficial that they are able to enrich the information on them, in order to organise them more conveniently. We will distinguish between three kinds of feedback that are typical in the Web 2.0:

- Tagging (assigning the Goals relevant keywords)
- Rating (assigning the Goals values that express their usefulness)
- Evaluations (writing reviews or comments about the Goals)

We will make use of the Feedback Management Framework from the Provisioning Platform, which is described in D2.1.1 [11] (Section 4.3.5). In turn, the components from that Framework will interact with the repository of user feedback in Semantic Spaces, through the use of the SOA4All Studio Infrastructure Services for Storage (T2.4). However, we will abstract here from those details and will just make call to the relevant components of the other platform.

In the following subsections, we will briefly describe the three components that will take care of the different kinds of feedback.

4.4.1 Goal Tagging Component

Description

This component will enable the tagging of Goals and also their retrieval.

Component Id	Comp#11
Functionalities covered	F#18, F#19
Related use cases	Uc#03, Uc#06
Components used	Service Provisioning Platform: Tagging component Goal Templates Manager component

Table 14: Goal Tagging component summary

Technical Specification

1) `tagGoal(idGoal, tag, user)`

It permits a user assign a keyword (tag) to a Goal, calling the `tagItem` function.

2) `getGoalTags(idGoal) and getGoalTags(idGoal, user): Tag[]`

For a given Goal, it retrieves its tags, calling the `getItemTags` function.

3) `getGoals(tag[]) and getGoals(tag[], user): GoalId[]`

For a given tag (or set of tags), this function returns the list of relevant Goals, calling the `getGoals` function from the Provisioning platform, and making use of the Goal Templates Manager component to access their relevant information.

4) `getGoalTagCloud(number): tagCloud`

It creates a tag cloud with the specified number of tags, calling the `getGoalTagCloud` from the other platform.

4.4.2 Goal Evaluations Component

Description

This component will enable the evaluation of Goals with textual comments, and their retrieval.

Component Id	Comp#12
Functionalities covered	F#20, F#21
Related use cases	Uc#06
Components used	Service Provisioning Platform: Evaluations component

Table 15: Goal Evaluations component summary

Technical Specification

This component will enable the evaluation of Goals with textual comments, and their retrieval.

1) `commentGoal(idGoal, comment, user)`

It stores a comment by a user on a Goal, calling the `commentItem` function.

2) `getGoalComments(idGoal) and getGoalComments(idGoal, user): comment[]`

For a particular Goal, it returns a list of comments, calling the `getComments` function.

4.4.3 Goal Ratings Component

Description

This component deals with the user-generated ratings of Goals, both allowing people to rate Goals, assigning a numeric value amongst a predefined set to them, and retrieving those ratings conveniently.

Component Id	Comp#13
Functionalities covered	F#22, F#23
Related use cases	Uc#06
Components used	Service Provisioning Platform: Ratings component

Table 16: Goal Ratings component summary

Technical Specification

1) `rateGoal(idGoal, rating, user)`

It stores a new rating on a Goal by a user, calling the `rateItem` function.

2) `getGoalRating(idGoal): [1-5 decimal number], number of ratings and getGoalRating(idGoal, user): [1-5 integer number]`

It returns the average rating of a service, as well as the total number of ratings, calling the `getRating` function.

4.5 Taxonomy Support Manager

We will support a way to deal with taxonomies that will be used within the Service Consumption Platform. The Taxonomy Support Manager groups the two components that support their use:

- A Taxonomy Browser component, technically a wrapper over a SOA4All Studio component, adapted for our particular case, allowing users to browse through a particular taxonomy.
- An Import Facilities component, which allows users to import any other desired taxonomy into the system, enabling a later interaction with them trough the platform.

We cover these two components below.

4.5.1 Taxonomy Browser Component

Description

The Service Consumption Platform permits users browse through different taxonomies in order to find Goals From the browsing result, some relevant Goals can be retrieved and the end-user can then refine or personalize them by means of parameterization, and sub-classing.

For this we imagine a faceted based search to be included:

i) on the one hand, the categorizations will be one set of facets,

ii) on the other hand, both the input and output parameters of Goals will be two other sets of facets.

From this, the end-user will able to efficiently retrieve Goals and then refine them according to their parameters.

In more detail, this component is a wrapper over the Taxonomy Selector Widget provided by the SOA4All Studio UI Components. We will incorporate a generic default taxonomy, UNSPSC[6] (United Nations Standard Products and Services Code), that is used to classify products and services, but when other taxonomies are loaded (through the next component) it will be possible to select them too.

It is important to note here that while we expect users browsing a taxonomy of Goals (in line with the stress on the use of Goals within the platform), in the simpler case we will also enable browsing a taxonomy of (semantically enhanced) services, thus facilitating their retrieval without the need of defining a Goal previously. As stated before, this won't be appropriate in a "world of billions of services"; but while we reach that scenario, it will be a useful way to permit users easily find what they need.

Component Id	Comp#13
Functionalities covered	F#01
Related use cases	Uc#01

[6] http://www.unspsc.org/

Components used	SOA4All Studio UI Components: Taxonomy Selector Widget

Table 17: Taxonomy Browser component summary

Technical Specification

The following functions will be available to permit the functionalities associated with taxonomy browsing:

1) `SelectTaxonomy(taxonomy[]): Taxonomy`

From the list of available taxonomies, it selects one to be used.

2) `changeFacetedView(taxonomy): Taxonomy`

From a given taxonomy, it changes its view (categorization or Input/Output parameters view).

3) `browseTaxonomy()`

It allows navigating through different levels in the tree, wrapping the functionality of the Taxonomy Selector Widget.

4.5.2 Import Facilities Component

Description

This component allows users to use any kind of taxonomy they want when interacting with the Service Consumption Platform. As the platform is Web-based (accessed on-line), users that want to handle different taxonomies need to upload them first.

The imported taxonomies will be stored in Semantic Spaces through the Storage Services.

Component Id	Comp#14
Functionalities covered	F#24
Related use cases	Uc#10
Components used	SOA4All Studio UI Components: Taxonomy Selector Widget

Table 18: Import Facilities component summary

Technical Specification

The functionality of this component will be exposed through the following function:

1) `ImportTaxonomy(taxonomy)`

It permits the user upload a taxonomy to the system, which will be stored and conveniently made available.

5. Conclusions

In this deliverable, we have defined the main design characteristics of the Service Consumption Platform of SOA4All. While doing this, we have kept in mind two main global concerns:

1. The platform needs to allow all kind of users, namely experts and non-experts, configure Goals in several fashions, both in a heavyweight and in a lightweight fashion. We have envisaged the configuration of Goals as the best way to interact with services in the service world paradigm proposed by SOA4All, taking into account the vast number of them that may be expected.
2. Personalisation is key to satisfy the users' interactions with services in our platform. We will use contextual information in order to enable the desired personalisation of services and the platform itself.

We have identified a set of functionalities the Service Consumption Platform has to offer to end-users in order to cope with such concerns, and we have defined the different components that are necessary in the architecture and how they should communicate in order to cover the desired functionalities. Additionally, being SOA4All a project where many different platforms and architectural components are envisaged, it is important to ensure a good integration of the Service Consumption Platform within the global architecture, and we have addressed the connections where necessary.

6. References

[1] World Wide Web Consortium, W3C: Simple Object Access Protocol, SOAP, Version 1.2 Part 0: Primer, (2003). Web site: http://www.w3.org/TR/soap12-part0/).

[2] WSMO Working Group, D2v1.0: Web Service Modeling Ontology (WSMO). WSMO Working Draft, 2004. Web site: http://www.wsmo.org/2004/d2/v1.0/

[3] World Wide Web Consortium, W3C: Semantic Annotations for WSDL and XML Schema. W3C Recommendation (August 2007). Web site: http://www.w3.org/TR/sawsdl/

[4] T. Vitvar, J. Kopecký, M. Zaremba, and D. Fensel. WSMO-Lite: Lightweight Descriptions of Services on the Web. In Proceedings of the IEEE European Conference on Web Services, Halle (Saale), Germany, 11 2007. IEEE Computer Society.

[5] J. Kopecký, T. Vitvar, and D. Fensel. MicroWSMO: Semantic Description of RESTful Services. WSMO Working Draft (February 2008).

[6] R.T. Fielding. Architectural styles and the design of network-based software architectures. PhD thesis, University of California, Irvine, 2000. Chair-Richard N. Taylor.

[7] A.K. Dey,, Understanding and Using Context. Personal Ubiquitous Computing, 2001. 5(1): p. 4-7.

[8] G. Álvaro, L. Chekov, S. Abels, N. Mehandjiev, L. Xu: Service Modelling Tools Design, Project Deliverable D2.1.2, EU FP7 SOA4All project, August 2008.

[9] S.Abels, L. Chekov, N. Mehandjiev, G. Álvaro, C. Ruiz, Holistic User Interface Design, Project Deliverable DX-UI, EU FP7 SOA4All project, August 2008.

[10] C. Pedrinaci, C. Ruiz, N. Mehandjiev: Contextual Service Adaptation Framework, Project Deliverable D4.1.1, EU FP7 SOA4All project, August 2008.

[11] S. Dietze, C. Pedrinaci, A. Gugliotta, P. Merle, I. Martínez, G. Álvaro, C. Ruiz, M. Villa, S. Abels: Service Provisioning Platform Design, Project Deliverable D2.1.1, EU FP7 SOA4All project, August 2008.

[12] E. M. Maximilien, M. P. Singh, Conceptual Model of Web Service Reputation. SIGMOD Record 31 (2002).

[13] H. Cunningham. Information Extraction, Automatic. Encyclopedia of Language and Linguistics, 2nd Edition (2005).

[14] R. González-Cabero, C. Pedrinaci, J.M. Gómez, C. Ruiz, C. Hamerling, A. Mos, S. Abel: Service Monitoring and Management Tool Suite Design, Project Deliverable D2.3.1, EU FP7 SOA4All project, August 2008.

[15] H. Cunningham, D. Maynard, K. Bontcheva and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02), Philadelphia, US. (2006)

[16] E. Motta, Reusable Components for Knowledge Modelling. Case Studies in Parametric Design Problem Solving. Frontiers in Artificial Intelligence and Applications. Vol. 53. 1999: IOS Press.

[17] A. ten Teije, F. van Harmelen, and B.J. Wielinga. Configuration of Web Services as Parametric Design.in EKAW. 2004. Whittlebury Hall, UK: Springer.

[18] R. Studer, R. Benjamins, and D. Fensel, Knowledge Engineering: Principles and Methods. Data Knowledge Engineering, 1998. 25(1-2): p. 161-197.

[19] G. Schreiber, et al., Knowledge Engineering and Management: The CommonKADS Methodology, ed. G. Schreiber. 1999: MIT Press.

[20] S. Abels et al. SOA4All Studio First demonstrator + Interface Specification, D2.4.1, EU FP7 SOA4All project, February 2009
[21] R. Krummenacher, F. Huet, F. Baude, and I. Filali: Distributed Semantic Spaces: A Scalable Approach To Coordination, D1.3.2A , EU FP7 SOA4All project, February 2009
[22] British Telecom, “Web21C SDK.” http://web21c.bt.com/
[23] O. Shafiq et al. Semantic Spaces A Unified Semantic Data Coordination Infrastructure, D1.3.1, EU FP7 SOA4All project, August 2008
[24] M. d'Aquin, M. Sabou, E. Motta, S. Angeletou, L. Gridinoc, V. Lopez, F. Zablith: What Can be Done with the Semantic Web? An Overview Watson-based Applications. In: Gangemi, A., Keizer, J., Presutti, V., Stoermer, H. (eds.): SWAP, Vol. 426. CEUR-WS.org, Rome, Italy (2008)
[25] Ontotext Lab,”WSMO4J”,http://wsmo4j.sourceforge.net/

Annex A.

The present Annex shows two diagrams for each of the ten envisaged use cases within the platform: (i) a use case diagram and (ii) a sequence diagram.

Uc#01 – Direct Goal Invocation, found via Goal taxonomy browsing

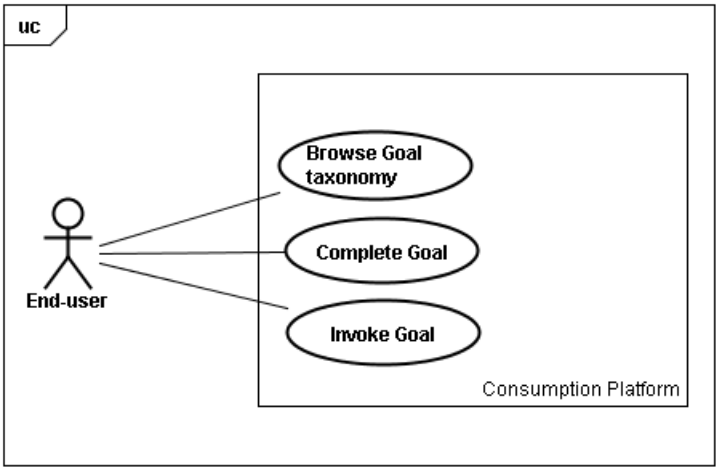

Figure 10: Use case #1 use case diagram

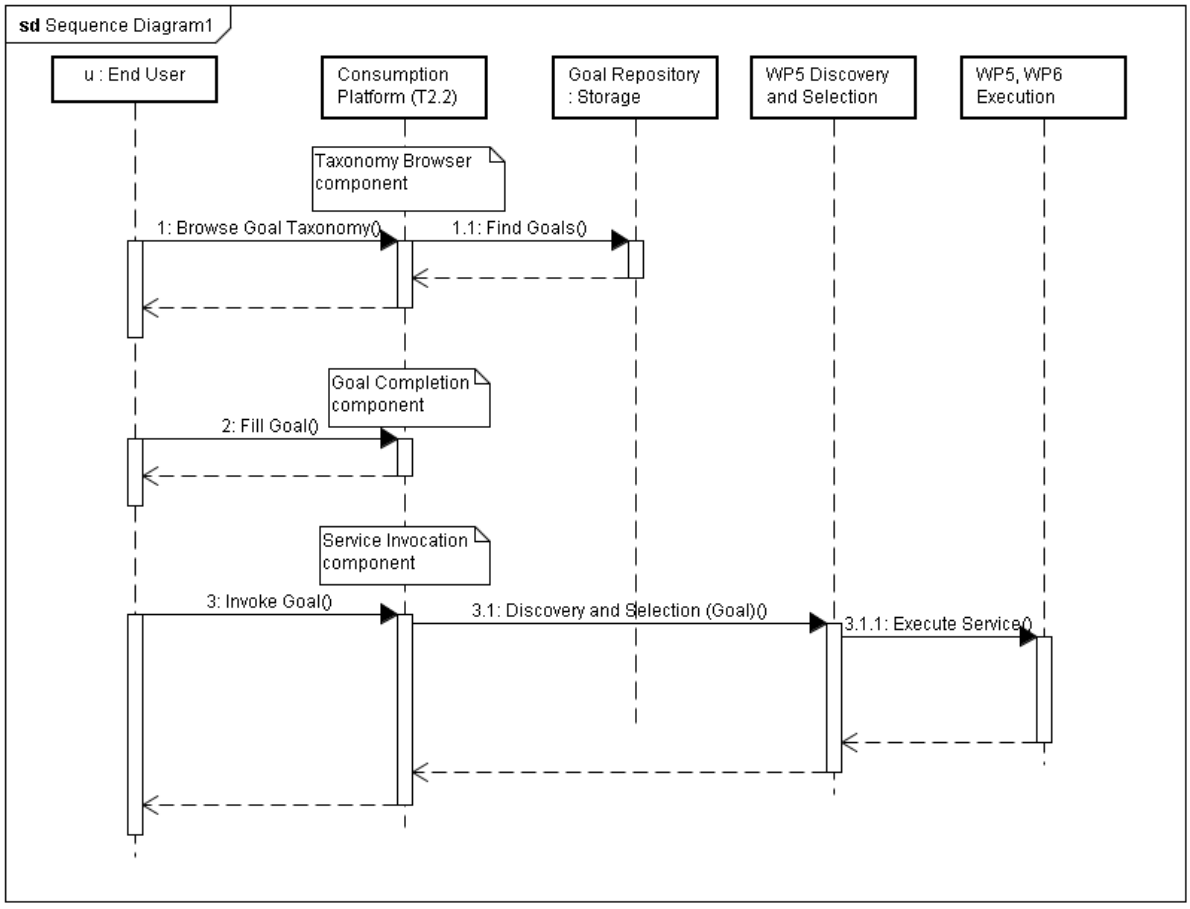

Figure 11: Use case #1 sequence diagram

Uc#02 – Direct Goal Invocation, found via NLP query

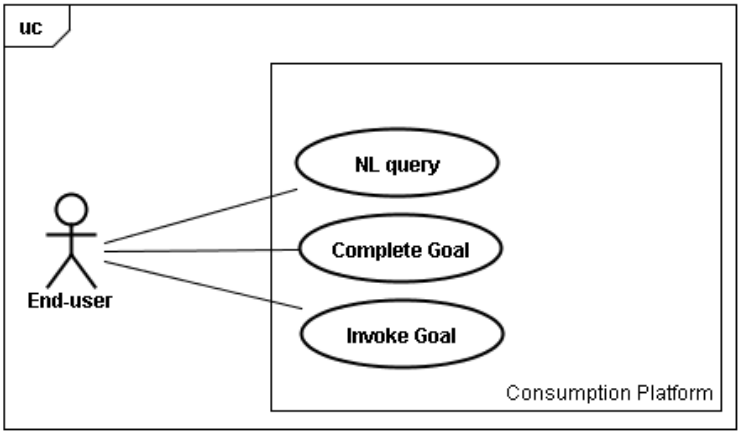

Figure 12: Use case #2 use case diagram

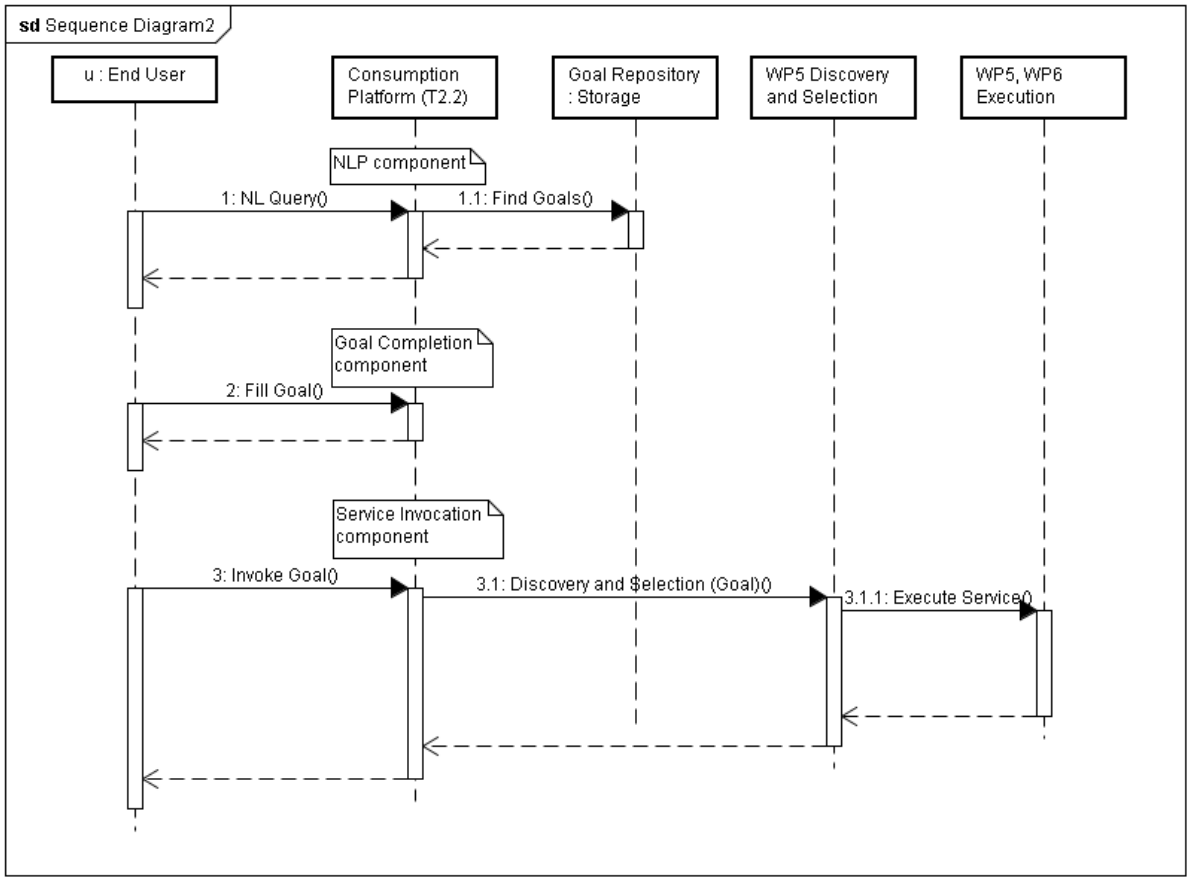

Figure 13: Use case #2 sequence diagram

Uc#03 – Finding Goals through browsing tags

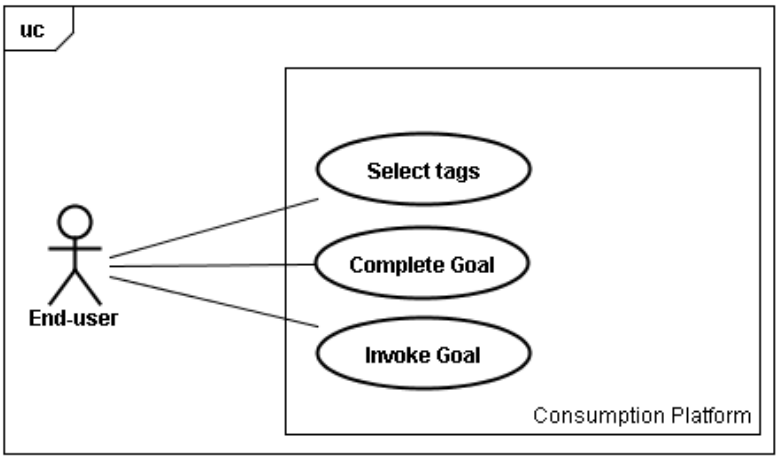

Figure 14: Use case #3 use case diagram

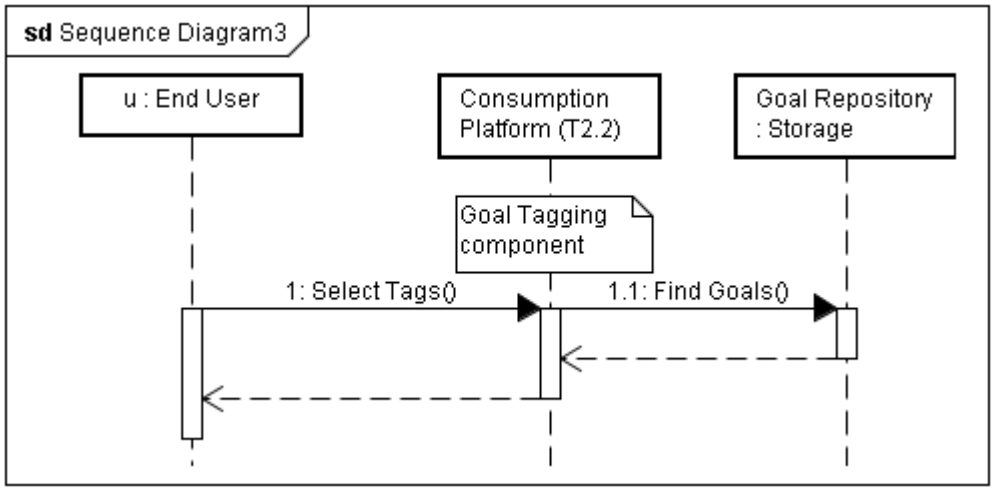

Figure 15: Use case #3 sequence diagram

Uc#04 – Modification of Goal, and Invocation

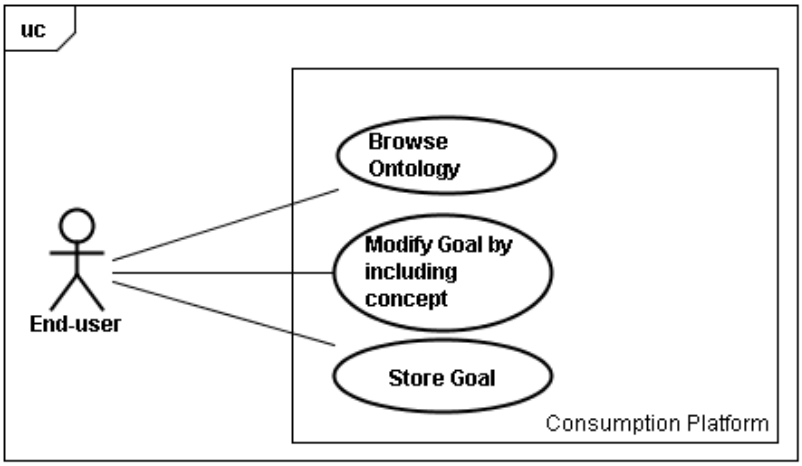

Figure 16: Use case #4 use case diagram

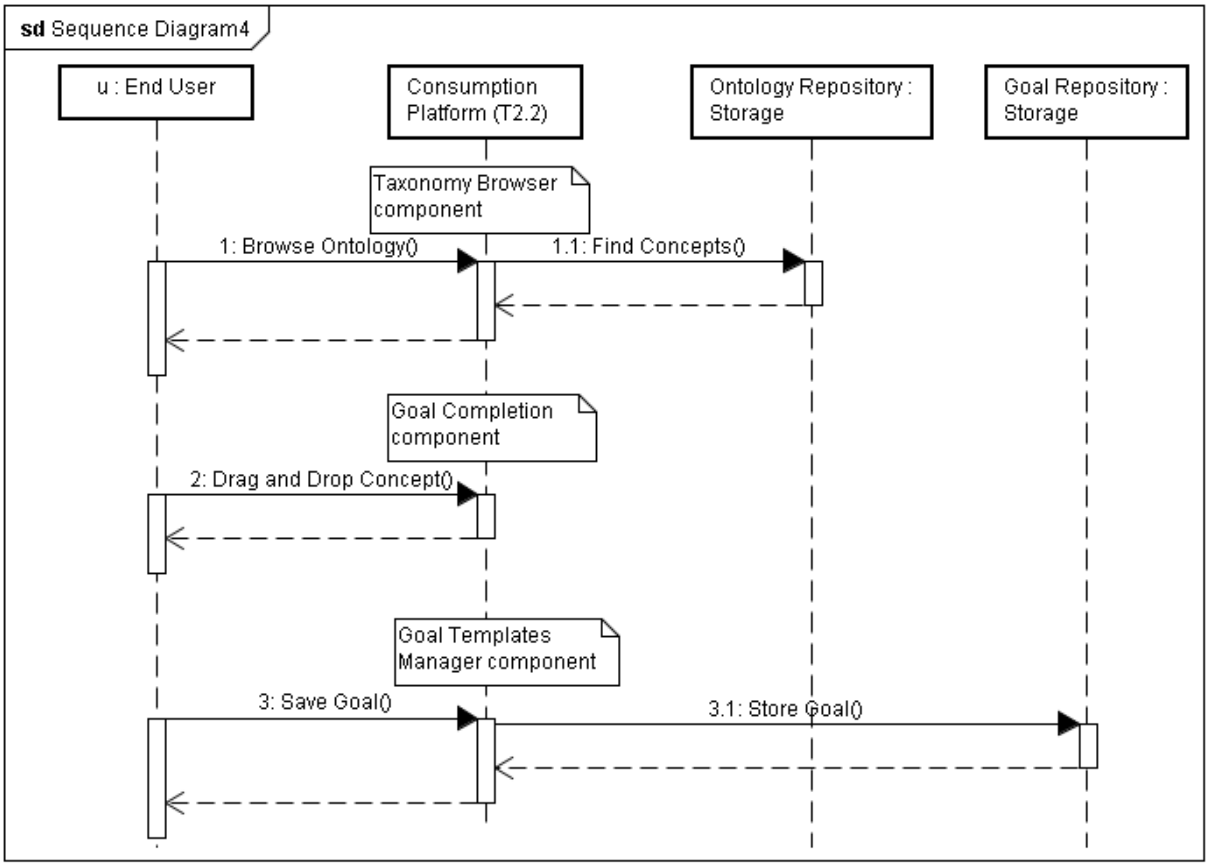

Figure 17: Use case #4 sequence diagram

Uc#05 – Platform adaptation

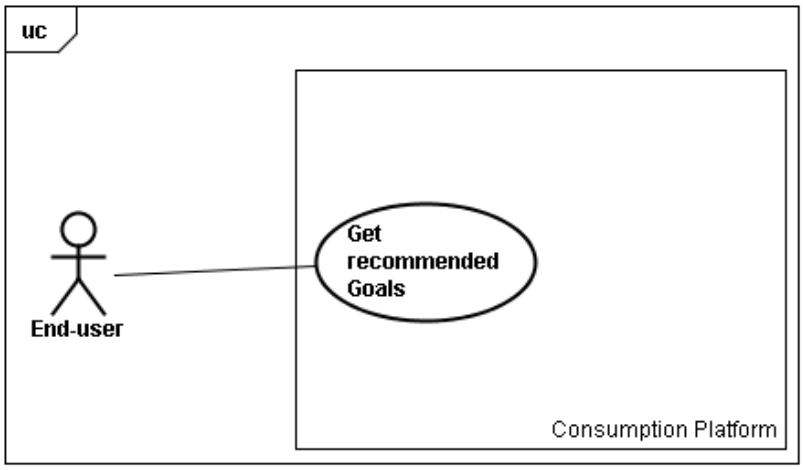

Figure 18: Use case #5 use case diagram

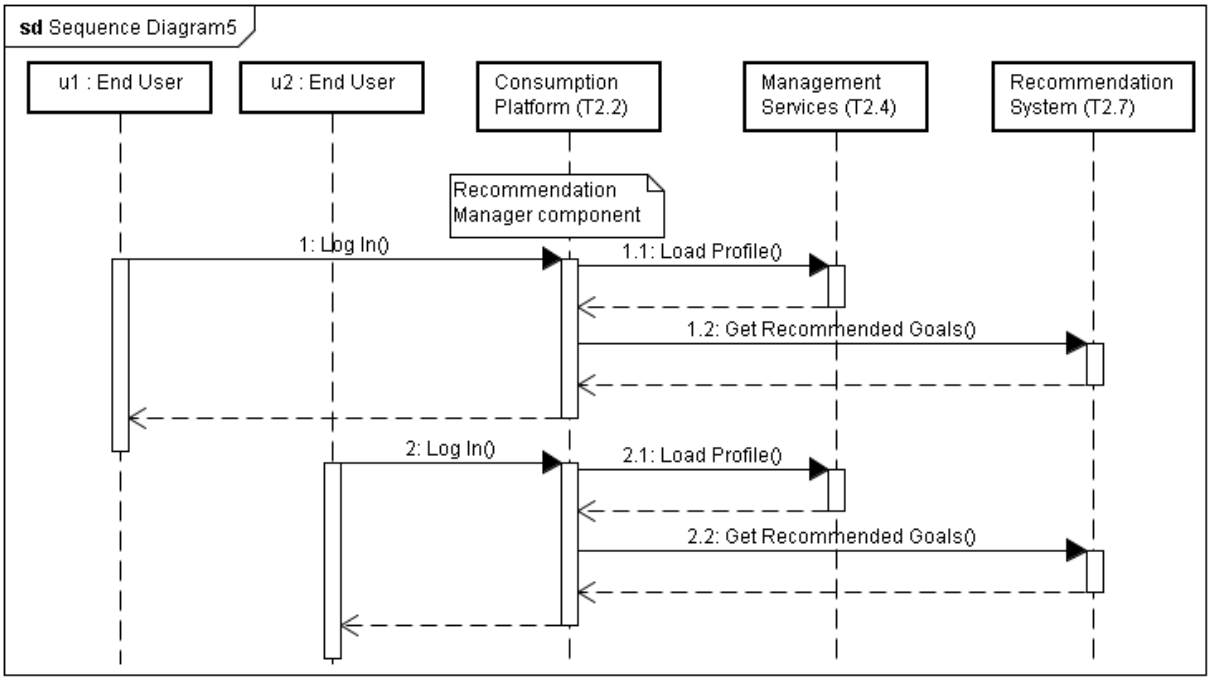

Figure 19: Use case #5 sequence diagram

Uc#06 – Feedback on Goals

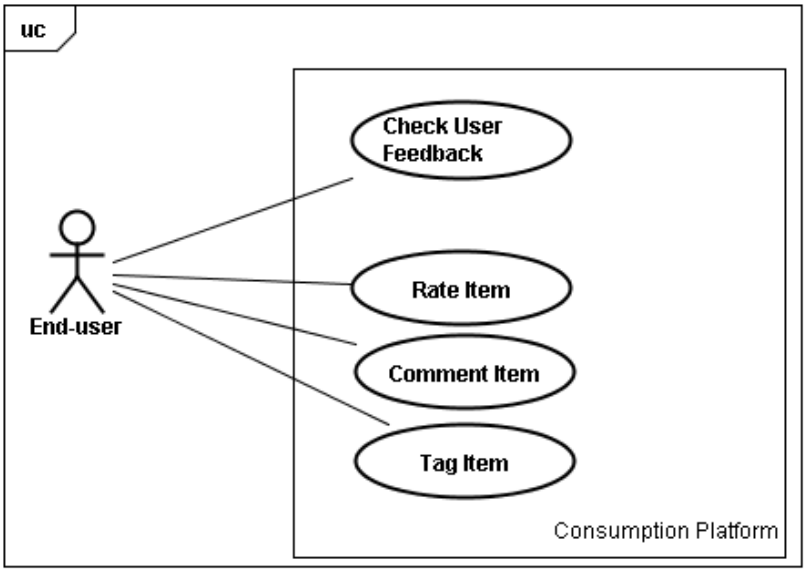

Figure 20: Use case #6 use case diagram

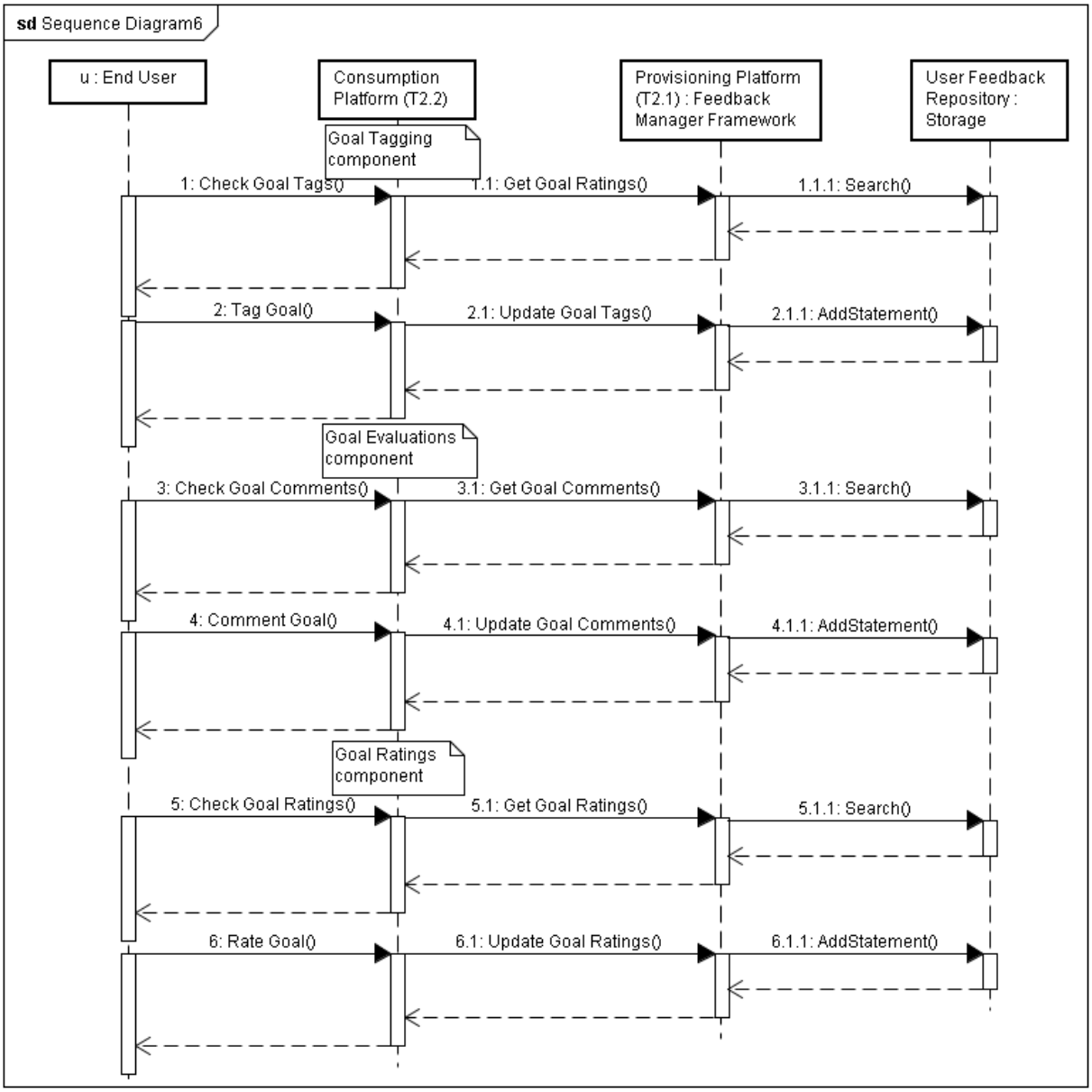

Figure 21: Use case #6 sequence diagram

Uc#07 – Service Adaptation

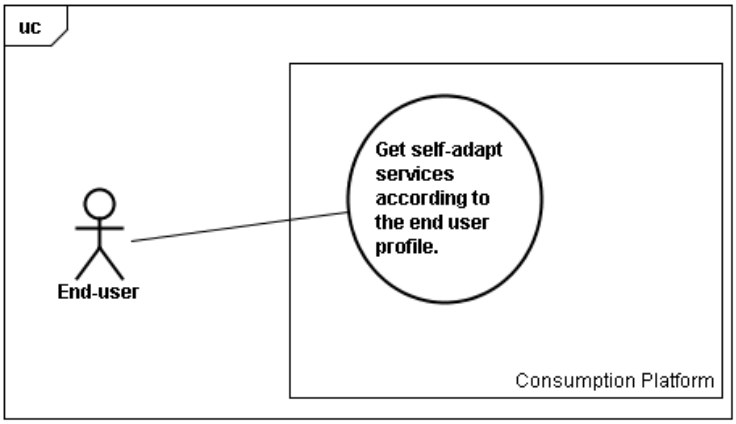

Figure 22: Use case #7 use case diagram

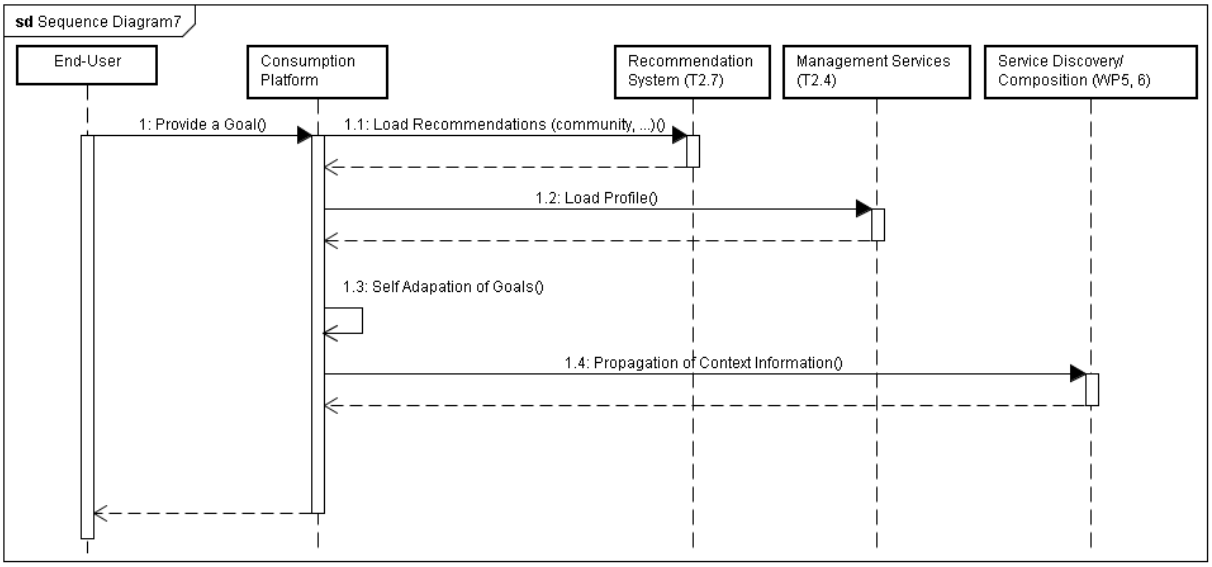

Figure 23: Use case #7 sequence diagram

Uc#08 – Discovery of services and choosing which one to Invoke

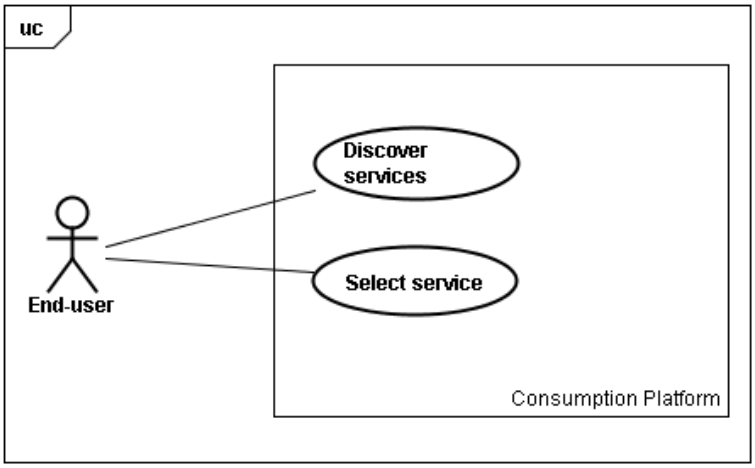

Figure 24: Use case #8 use case diagram

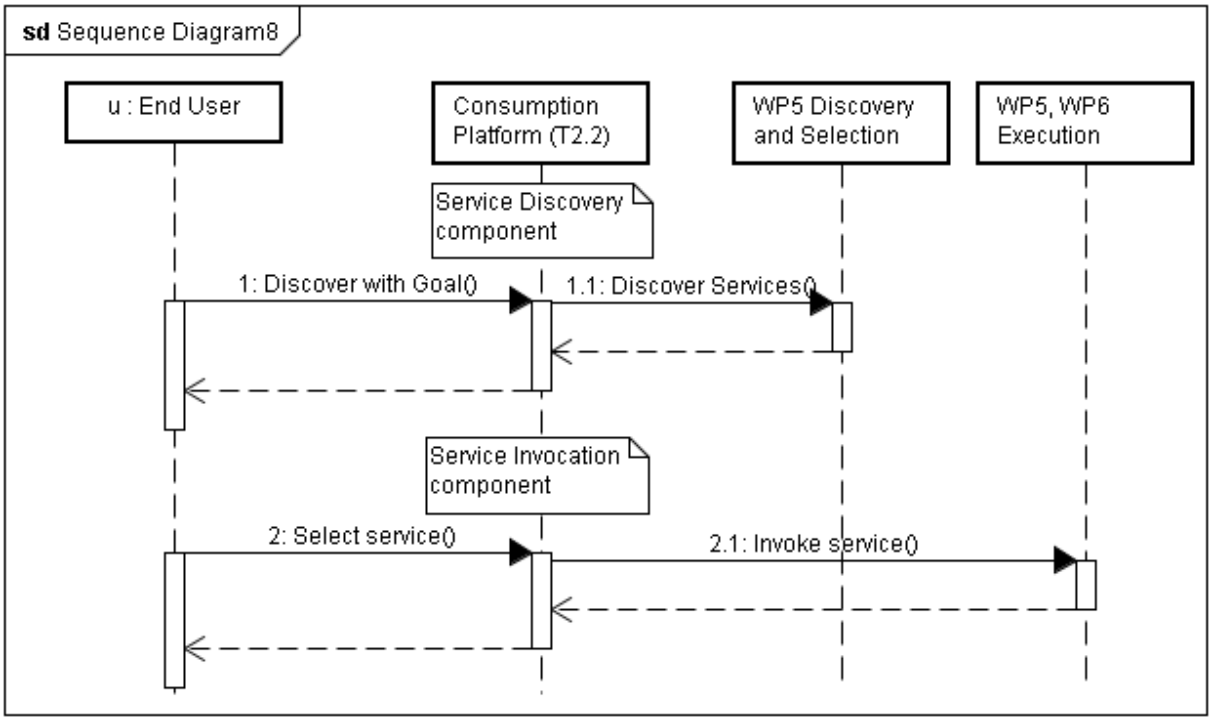

Figure 25: Use case #8 sequence diagram

Uc#09 – Use of Wizards to consume services

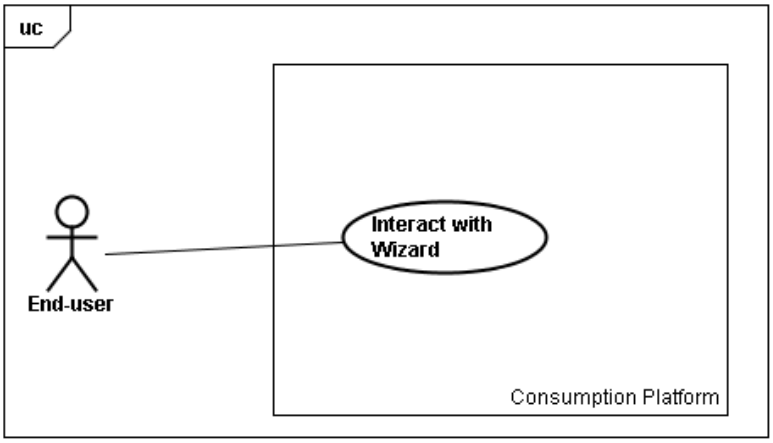

Figure 26: Use case #9 use case diagram

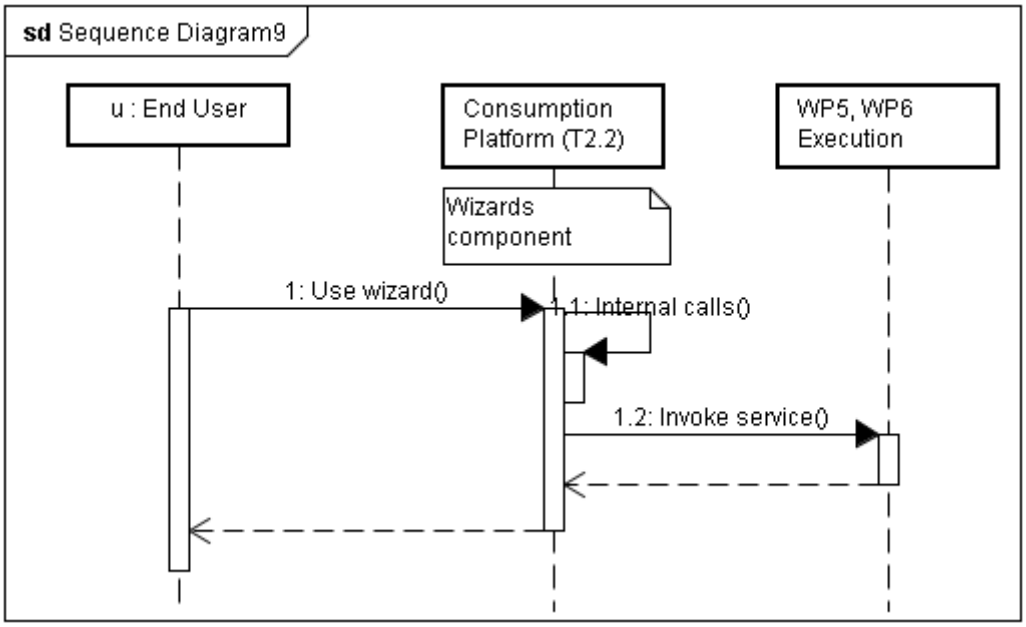

Figure 27: Use case #9 sequence diagram

Uc#10 – Import taxonomy and use it to define a Goal

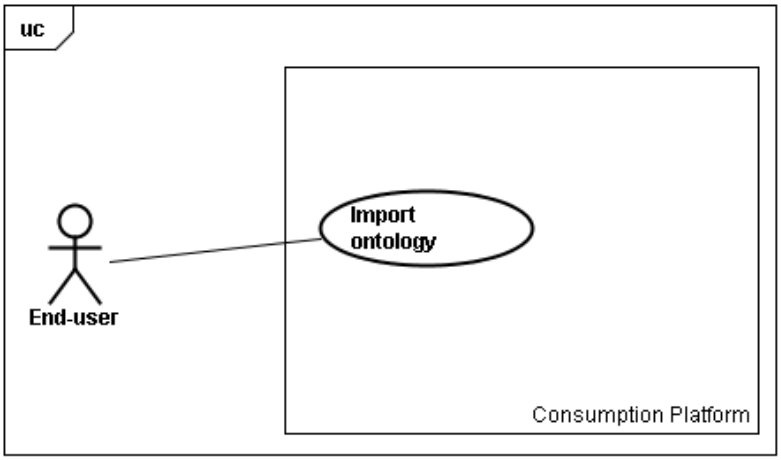

Figure 28: Use case #10 use case diagram

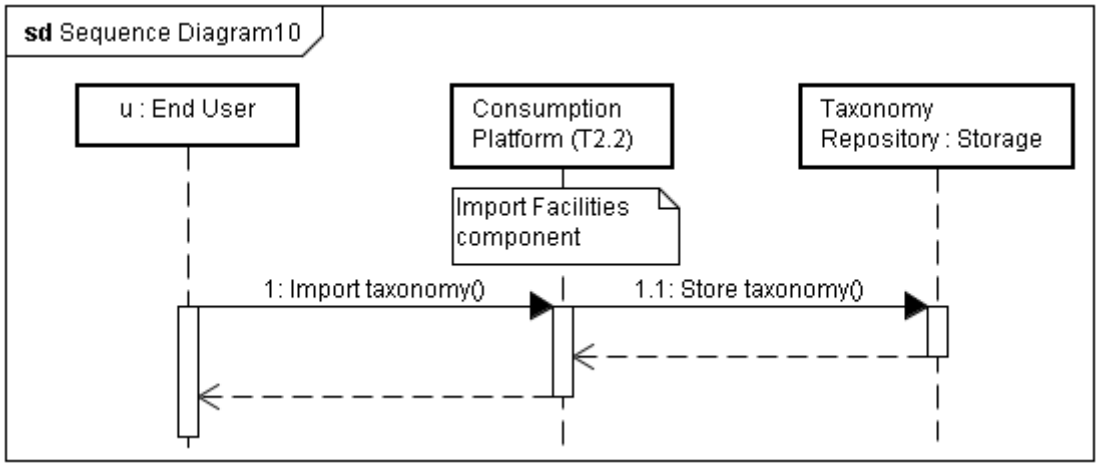

Figure 29: Use case #10 sequence diagram