

Project Number: **215219**
 Project Acronym: **SOA4ALL**
 Project Title: **Service Oriented Architectures for All**
 Instrument: **Integrated Project**
 Thematic Priority: **Information and Communication Technologies**

D8.3 Web21c Futures Design

Activity N:	3 – Use Case Activities	
Work Package:	8 – BT W21C	
Due Date:	M13	
Submission Date:	18/05/2009	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	BT	
Revision:	2.0	
Authors:	Marc Richardson BT John Davies BT Sandra Stincic BT Nikolay Mehandjiev UNIMAN Usman Wajid UNIMAN Freddy Lecue UNIMAN Guillermo Álvaro Rey ISOCO	

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
1.0	22.12.2008	ToC and initial structure outlined	Marc Richardson (BT)
1.1	12.01.2009	First version of draft	Marc Richardson (BT), John Davies (BT), Sandra Stincic (BT)
1.2	24.01.2009	Second version with updated design	Marc Richardson (BT), John Davies (BT), Sandra Stincic (BT)
1.3	11.02.2009	Third version including mock-ups	Marc Richardson (BT), John Davies (BT), Sandra Stincic (BT)
1.4	11.03.2009	Consumption section added	Guillermo Álvaro Rey (ISOCO)
1.6	22.03.2009	Evaluation section added	Nikolay Mehandjiev (UNIMAN), Usman Wajid (UNIMAN), Freddy Lecue (UNIMAN)
1.7	25.03.2009	Annex A and B added	Marc Richardson (BT)
1.8	14.04.2009	Internal corrections	John Davies (BT)
1.9	8.5.2009	Reviewers comments incorporated	Bernhard Schreder (Hanival) Carlos Pedrinaci (OU)
2.0	15.5.2009	Overall Format and final revision	Malena Donato (ATOS)

Table of Contents

EXECUTIVE SUMMARY	8
1. INTRODUCTION	9
1.1 INTRODUCTORY EXPLANATION OF THE DELIVERABLE	9
1.2 PURPOSE AND SCOPE	11
1.3 STRUCTURE OF THE DOCUMENT	11
1.4 METHODOLOGY	11
2. OVERVIEW OF USE CASES	13
2.1 SCENARIO 1 (S1)	13
2.2 SCENARIO 2 (S2)	14
2.3 SUMMARY OF REQUIREMENTS	15
3. ARCHITECTURE OF PROTOTYPE	18
3.1 SOA4ALL STUDIO AND PLATFORM SERVICES	18
4. DESIGN	22
4.1 UI AND FRONT-ENDS	22
4.1.1 <i>Standard Look and Feel</i>	22
4.1.2 <i>BT Ribbit Branding</i>	23
4.1.3 <i>Additional UI Components</i>	23
4.1.4 <i>Storyboard and mock-ups</i>	25
4.2 SERVICE DISCOVERY AND SELECTION	25
4.2.1 <i>Meta Data enhanced discovery</i>	25
4.2.2 <i>Full Semantic discovery, Using Goals</i>	26
4.3 SERVICE COMPOSITIONS	26
4.3.1 <i>Customizing the SOA4All Composer</i>	26
4.3.2 <i>Domain-specific Wizards for the SOA4All Composer</i>	27
4.3.3 <i>Domain-specific Process Templates</i>	27
4.4 ANNOTATIONS AND COMMUNITY SUPPORT	28
4.4.1 <i>Community Sharing of Compositions & sub-Compositions</i>	28
4.4.2 <i>Data mappings</i>	28
4.4.3 <i>Service Annotations</i>	29
4.4.4 <i>Ratings and Rankings</i>	29
4.5 SERVICE STORAGE	29
4.6 SERVICE DEPLOYMENT AND EXECUTION	30
4.6.1 <i>End user service consumption</i>	30
4.6.2 <i>The adapters</i>	31
4.7 MONITORING AND MANAGEMENT	32
5. EVALUATION WORKSHOPS	33
5.1 DESIGNS FOR COMPOSING WEB SERVICES	34
5.2 DESIGN RATIONALE	34
5.3 TOP-LEVEL ISSUES TO BE DISCUSSED	36
6. CONCLUSIONS	37
7. REFERENCES	38
ANNEX A. WALKTHROUGH FOR SCENARIO 1	39
ANNEX B. CURRENT RIBBIT SERVICES	47

ANNEX C. EXAMPLE SEMANTIC SERVICE DESCRIPTION _____ **50**

List of Figures

Figure 1. BT Ribbit website	10
Figure 2. Scenario 1 Example	14
Figure 3. Scenario 2 Example	15
Figure 4. Summary of Requirements	17
Figure 5. SOA4All Overall Architecture.....	18
Figure 6. The SOA4All standard look and feel.....	22
Figure 7. Standard Ribbit Logo.....	23
Figure 8. BT and Ribbit Logo.....	23
Figure 9. The current Ribbit community forum.....	24
Figure 10. Building a composition in the SOA4All Composer.....	27
Figure 11: Layers Involved in Service Consumption	31
Figure 12. gIBIS design rationale for Composition (Issue 1)	35
Figure 13. gIBIS design rationale for Composition (Issue 2)	35

List of Tables

Table 1. Summary of components used in prototypes V1 and V2.....	20
--	----

Glossary of Acronyms

Acronym	Definition
21CN	21 st Century Network
ANI	Application Network Interface
API	Application Programming Interface
BB	Broadband
BSS	Business Support Systems
BT	British Telecom
CRM	Customer Relationship Manager
D	Deliverable
DT	Deutsche Telekom
EC	European Commission
FT	France Telecom
GoS	Grade of Service
ICT	Information and Communications Technology
iDEN	Integrated Digital Enhanced Network
IM	Instant Messaging
IP	Internet Protocol
IPTV	Internet Protocol Television
ISP	Internet Service Provider
JAIN	Java APIs for Integrated Networks
LLU	Local Loop Unbundling
Ofcom	The Office of Communications
OS	Operating System
OSS	Operations (or Operational) Support Systems
OTT	Over-the-top
PCS	Personal Communications Service
PSTN	Public Switched Telephone Network
PTT	Postal Telephone and Telegraph
QoE	Quality of Experience
QoS	Quality of Service
REST	REpresentational State Transfer
SDK	Software Development Kit
SIP	Session Initiation Protocol
SLA	Service Level Agreement

Acronym	Definition
SOA	Service Oriented Architecture
SOA4All	Service Oriented Architecture for All
SP	Service Provider
VISP	Virtual Internet Service Provider
VoIP	Voice over IP (Internet Protocol)
WLAN	Wireless Local Area Network
WP	Work Package
XMPP	Extensible Messaging and Presence Protocol

Executive summary

Work package 8, the BT Web21C case study, will build on the current BT Ribbit infrastructure, which provides a set of Web accessible Telco services, and leverage SOA4All research and technology to allow end-users to access, use and create services based on BT's 'capabilities' (such as VOIP, SMS etc.). Web 2.0 principles, Semantics and Context will be used to create a powerful but easy to use platform for discovering, consuming and combining BT's capabilities, and for incorporating third party services.

Following a detailed analysis of the current Ribbit user base, the case study has developed two scenarios as the basis for developing prototypes based on SOA4All technology. The prototypes will aim to provide the next generation of Ribbit.

The two scenarios are described in detail in D8.1, and derive the following main requirements:

1) Business Requirements

- Encourage greater uptake of Ribbit services by providing tools to enable easier use of the services
- Provide tools with a focus on end users with limited technical experience
- Create a community of Ribbit users, to encourage collaboration and innovation in creating new Telco applications
- Create an infrastructure to allow a third party business to resell BT's Ribbit services, providing support in design and management of the third party services
- Increase overall use of the Ribbit services, and hence increase revenue

2) Technical Requirements

- Usable without detailed knowledge of ontologies, WSDL[1] or programming languages
- Web Browser based, drag and drop, easy to use interface
- Search on functional and non-functional aspects of service, and keywords
- Semi-automated assistance with creating service compositions
- Service ranking and help with design time selections based on user context
- Export/share composition with other users
- Fault handling of service failures and error reporting
- Import and link to Industry standard ontologies
- Import and mark-up third party Web services

This deliverable outlines the design of the prototypes. In many cases the design will follow the generic design of the SOA4All platform, so the deliverable mainly details where extensions or modifications are made to fit specific requirements of WP8. Where design details follow the generic design, this document refers to the specific design deliverables in the technical work packages (WP1-WP6)

1. Introduction

This section introduces the deliverable, and explains its purpose, scope and structure.

1.1 Introductory explanation of the deliverable

This WP8 case study will investigate creating the future Ribbit infrastructure based on SOA4All technology.

Ribbit is the name currently given to the platform over which BT will provide next generation services on top of its IP-based 21st Century Network (BT 21CN). Some of these services will be provided by BT and others will be provided by third parties. Ribbit is central to BT's transformation from a traditional telecommunications company to a converged software and services business. Ribbit will allow third parties to use BT's network as a platform for delivery of their services, for which BT will get revenue. These are not typically other network competitors, but new breeds of partner - software companies, developers and content providers.

Ribbit comprises of a set of Web services, and software development kits (SDKs) that provide external access to a number of BT capabilities, such as making a voice call and sending an SMS text message. The website for Ribbit is shown below.

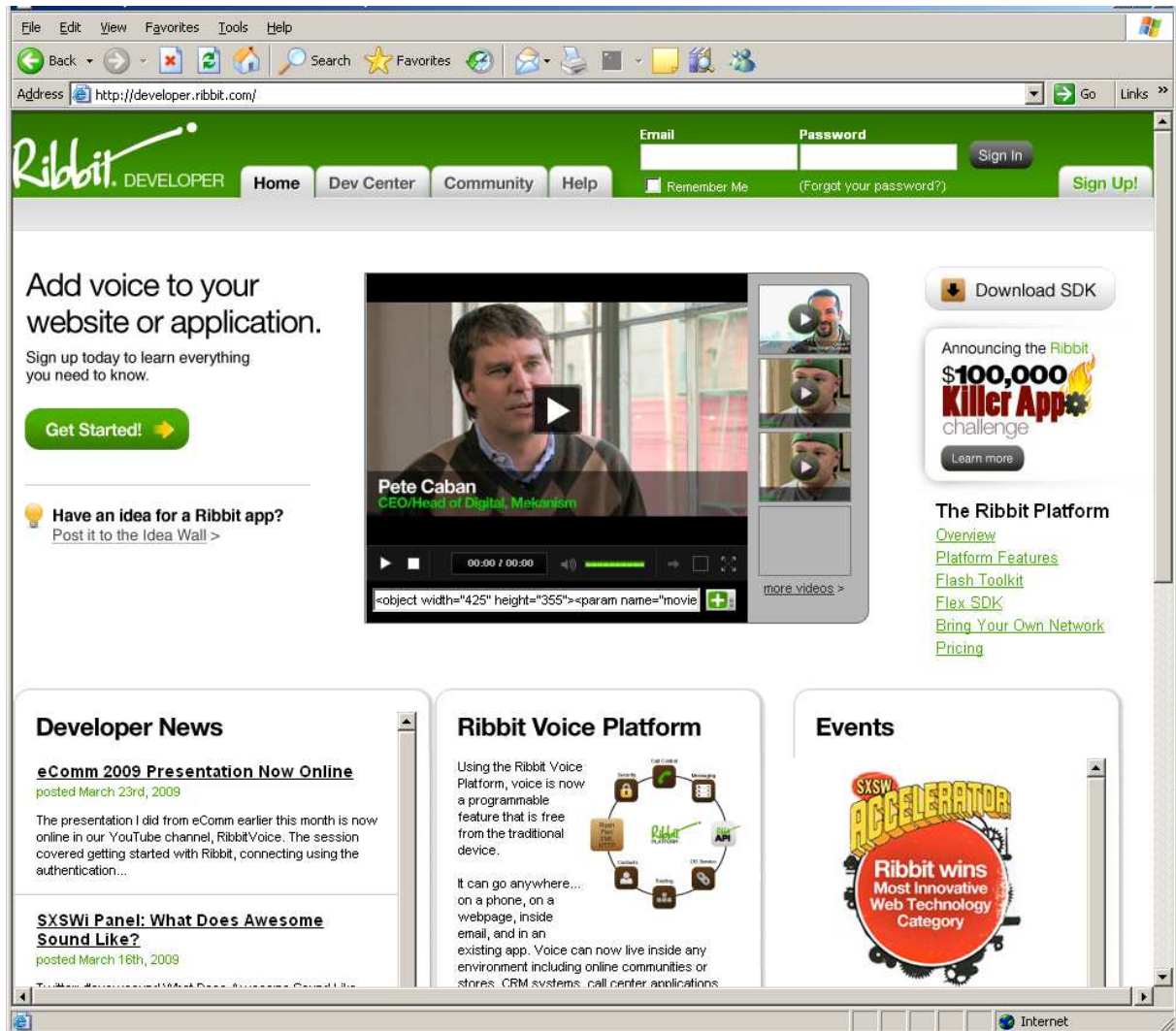


Figure 1. BT Ribbit website

Currently Ribbit requires a detailed technical knowledge of Web service languages and programming languages (e.g. Java & Adobe Flex) to be able to access, combine and use the services. The aim of the case study is to provide the next generation of Ribbit where the process of discovering, integrating, using and sharing BTs services can be done much more easily and effectively.

In short, the main objectives are:

- Reducing the cost and time of using and combining the services
- Reducing the barriers to entry of utilising services
- Increase innovation in the area for people to create exciting applications utilising BT services
- Increase Revenue for BT

There are also many non-technical issues in moving to a SOA4All based model for conducting business that the case study will investigate:

- Managing Quality of Service [2] and Service Level Agreements [3] in composed services
- Trust and Security
- Increasing the use of ontologies and semantic technology in the Telco sector
- Relationship between ontologies and other Telco models/standards
- Regulation, competition and legal issues
- Viable business models in an open services ecosystem

Many of the non-technical issues, specifically: market analysis, business models, regulation, competition and legal issues were investigated in Deliverable 8.2 (Semantic Telco analysis) and Deliverable 8.5 (Telco 2.0 recommendations).

WP8 has developed two scenarios using Ribbit that will form the basis for developing prototypes to test and showcase SOA4All technology, and fulfill the aims described above. The deliverable describes these scenarios and outlines the requirements for each.

1.2 Purpose and Scope

The purpose of this deliverable is to describe design of the prototypes to be developed in WP8. It is intended to be a reference for partners in WP8 and for the other technical work packages 1-6.

1.3 Structure of the document

Section 2 provides a short summary of the requirement defined in deliverable 8.1. Section 3 gives a brief description of the generic SOA4All Architecture and explains which components will be used in the WP8 prototype. It also outline the components that will be modified or extended to meet specific requirements of WP8. Section 4 gives details of the Design for the prototype, including some mock-ups for the UI, and information regarding the specific extensions to components. Section 5 outlines the Evaluation plan for WP8

1.4 Methodology

The initial analysis has involved a number of methods for gathering information which have contributed to the development of the scenarios described, and the associated requirements and design. These include surveying current users of the Ribbit, interviewing developers of the SDK, attending Telco industrial events and speaking to research experts in Web Services, Semantics and Web 2.0.

As the project progresses it is planned to involve real users in evaluation and testing of the prototypes developed in the case study. A formal evaluation will be undertaken in Task 8.7. Some details of the evaluation is already planned and is described in section 5. This will be taking place in the near future, with the first evaluation scheduled for May 2009. It is hoped to try and build a community of users by releasing early prototypes and encouraging

collaboration using the web 2.0 aspects of the project. As informal and formal feedback is received, it is envisaged that some requirements will evolve, and some new requirements will emerge. In addition the second scenario (S2) described in the document will not be developed until the second half of the project (M18+), so there may be some updates and further requirements derived as the case study progresses.

2. Overview of Use Cases

The following section provides an overview of the use cases, describing scenario 1 (S1) and scenario 2 (S2) in more detail.

2.1 Scenario 1 (S1)

S1 describes a situation in which SOA4All technology is used for creating relatively simple mash-ups of BT services and other popular services on the web. The aim is to make it easy for novice users to get access to the facilities of the Ribbit SDK and combine them with other services on the Web. SOA4All will be used to overcome some of the current problems that limit the uptake of the SDK, primarily the technical knowledge required and familiarity with a programming language such as Java.

The scenario involves building up Web Service composition to create a new web application incorporating Ribbit services. As the focus of S1 is on casual users and on building non-critical applications; the scenario will consequently involve minimal security or management infrastructure. An example service composition that a user would create is outlined below. In this example, a composition is created that allows you to organise a meet up with a group of friends at the last minute.

The main steps are:

- Get list of friends from social networking site (e.g. Facebook)
- Find out which ones are in the area using Ribbit location
- Find out weather and travel information for proposed meeting.
- Send out invite and directions using Ribbit SMS

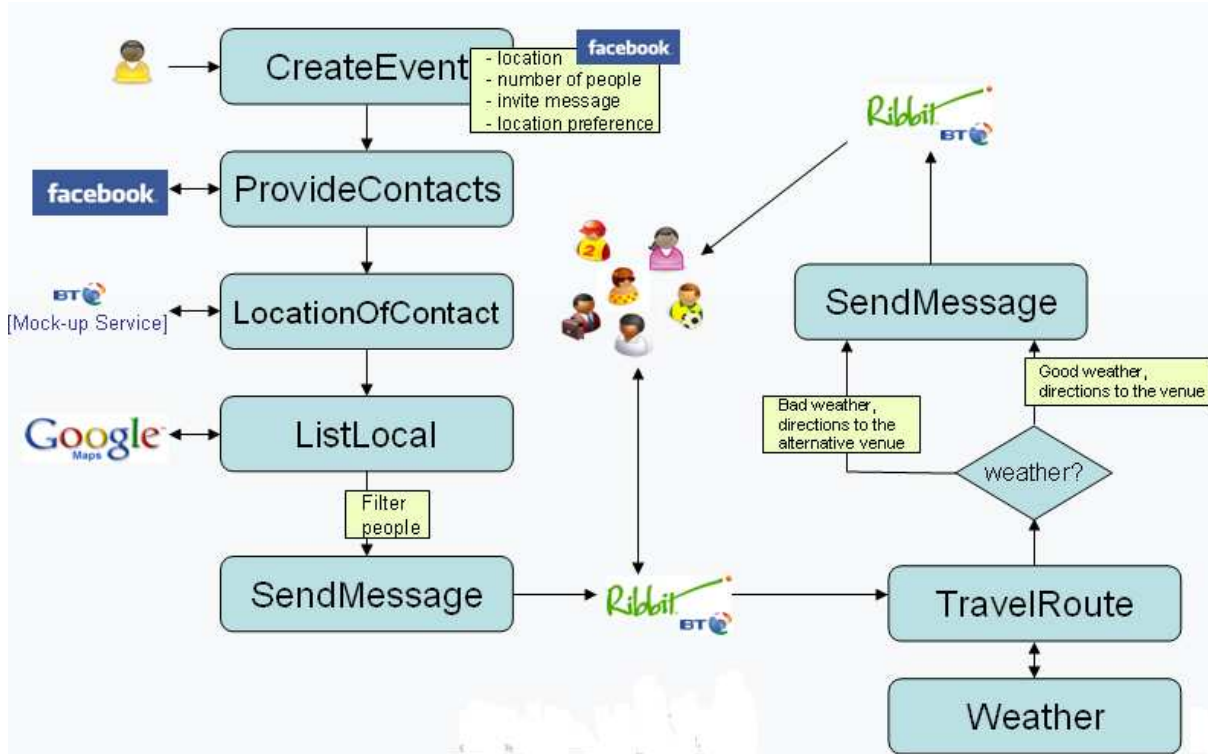


Figure 2. Scenario 1 Example

Users will undertake a similar sequence of steps each time they design a new application, although the specific services may differ.

2.2 Scenario 2 (S2)

S2 will be developed in the second half of the project, and aims to utilise all the technical results of the projects. As it is based on a business scenario, it has additional requirements that stem from the need to have a greater level of control over the execution of services, monitoring and fault handling.

In S2 businesses will use SOA4All technology to design and compose more complex end user applications to resell or use as part of their business, incorporating BT white label Ribbit services, their own services & OSS and some BT OSS services. This will enable people to create a business incorporating BT services, without complex face to face contract negotiations and manual work to integrate services. This will enable businesses to go from 'idea to product' in minimal time.

An example service composition that a user would create is outlined below. In this example, a composition is created which allows a company to build a bulk text messaging service using the BT Ribbit SMS service.

The scenario is based on a company which allows you to ring a number and leave a message. The message is converted to text and sent out as SMS messages to a to groups of people who are subscribed to that service. The Company buys bulk rate SMS service from Ribbit, but it is a free service to subscribers, subsidised with an advertising model that attaches advert to each SMS. The user profile and location are used to contextualise SMS advert. The service may be used for things such as a Traffic Alert SMS service, or Weather Warning Service

The composite service, built from a combination of Ribbit services and the companies SMS service is shown below.

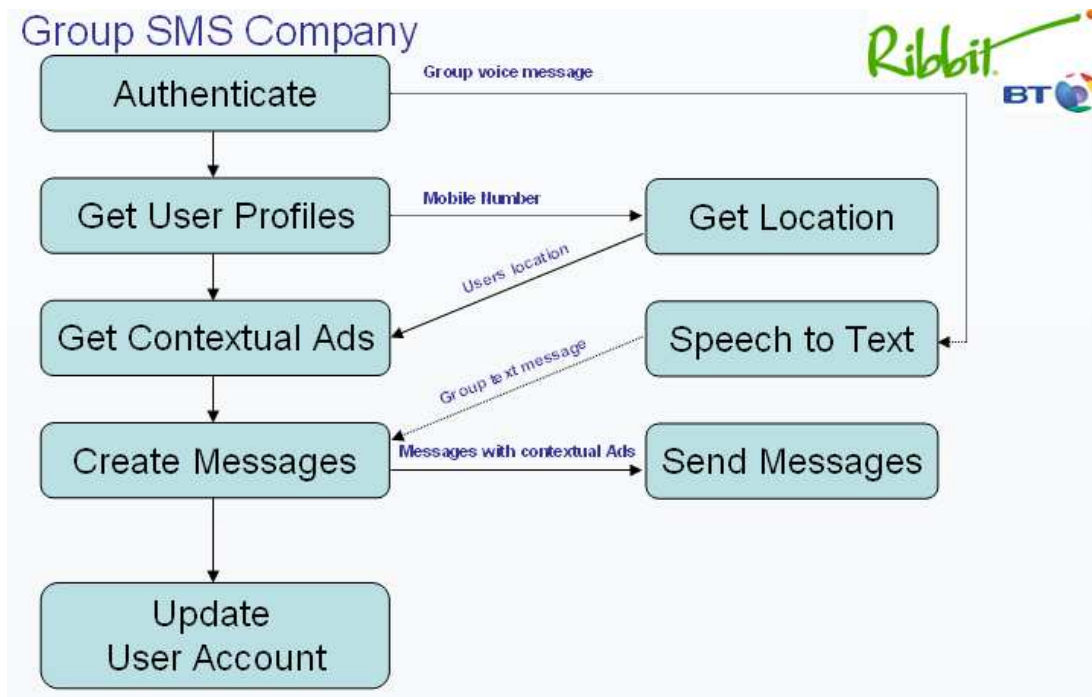


Figure 3. Scenario 2 Example

2.3 Summary of Requirements

The detailed requirements for WP8 are defined in detail in deliverable D8.1 and the scenarios outlined in section 5.1 of this deliverable. The table below summarises the requirements

Requirement ID	Description	Key Technologies	Tasks
S1.R1	Usable without detailed knowledge of Ontologies, WSDL or programming languages	SOA4All (WP2), Studio Construction (WP6)	T2.1, T2.2, T6.1, T6.2
S1.R2	Web Browser Based, Drag and drop, easy to use interface	SOA4All (WP2)	T2.1, T2.2
S1.R3	Search on functional and non-functional aspects of service, and keywords	Service Annotations and Reasoning	T3.1, T3.2, T5.3,

		(WP3), Service Location (WP5)	T5.4, T5.5
S1.R4	Service ranking and help with design time selections based on user context	SOA4All Studio (WP2), Service Location (WP5)	T2.1, T2.2, T6.1, T6.2
S1.R5	Suggestions for compatible services in a web service compositions	SOA4All Studio (WP2), Service Construction (WP6)	T2.1, T2.2, T6.1, T6.2
S1.R6	Automated or semi automated assistance with linking services in composition	Service Constructions (WP6)	T6.1, T6.2
S1.R7	Link GUI to web service compositions	SOA4All Studio (WP2), Service Construction (WP6)	T2.1, T2.2, T6.1, T6.2
S1.R8	Export/share composition with other users	Service Deployment and Use (WP2), Service Construction (WP6)	T2.1, T2.2, T6.1, T6.2
S2.R1	Monitoring of process state and Quality of Services (QoS) for service compositions	Service Deployment and Use (WP2)	T2.3
S2.R2	Fault handling of Service Failures and Error reporting	Service Deployment and Use (WP2)	T2.3
S2.R3	Import and link to Industry standard ontologies	Service Deployment and Use (WP2), Service Annotations and Reasoning (WP3), Service Construction (WP6)	T2.1, T3.1, T3.2, T6.1, T6.2
S2.R4	Import and mark-up third party Web Services	Service Deployment and Use (WP2)	T2.1, T3.1, T3.2

Use (WP2), T6.1, T6.2
Service
Annotations and
Reasoning (WP3)
Service
Construction
(WP6)

Figure 4. Summary of Requirements

3. Architecture of Prototype

The architecture of the prototype is based on the overall architecture defined in WP1. A detailed description of this can be found in deliverable 1.4.1 [4]. The full architecture for SOA4All defines all components and their interactions (see fig 5 below). WP8 will develop 2 prototypes (V1 at M18 and V2 at M30) which will focus on a subset of the full functionality outlined in WP1. This is due to the specific requirements of the WP8 case study and also taking into account the availability of some of the components developed in the project. As V1 of the prototype is developed by M18 there will only be some components available at this stage. V2 will use a greater number of components. This is reflected in the complexity of the two scenarios defined (S1 and S2) which form the basis for prototypes V1 and V2 respectively.

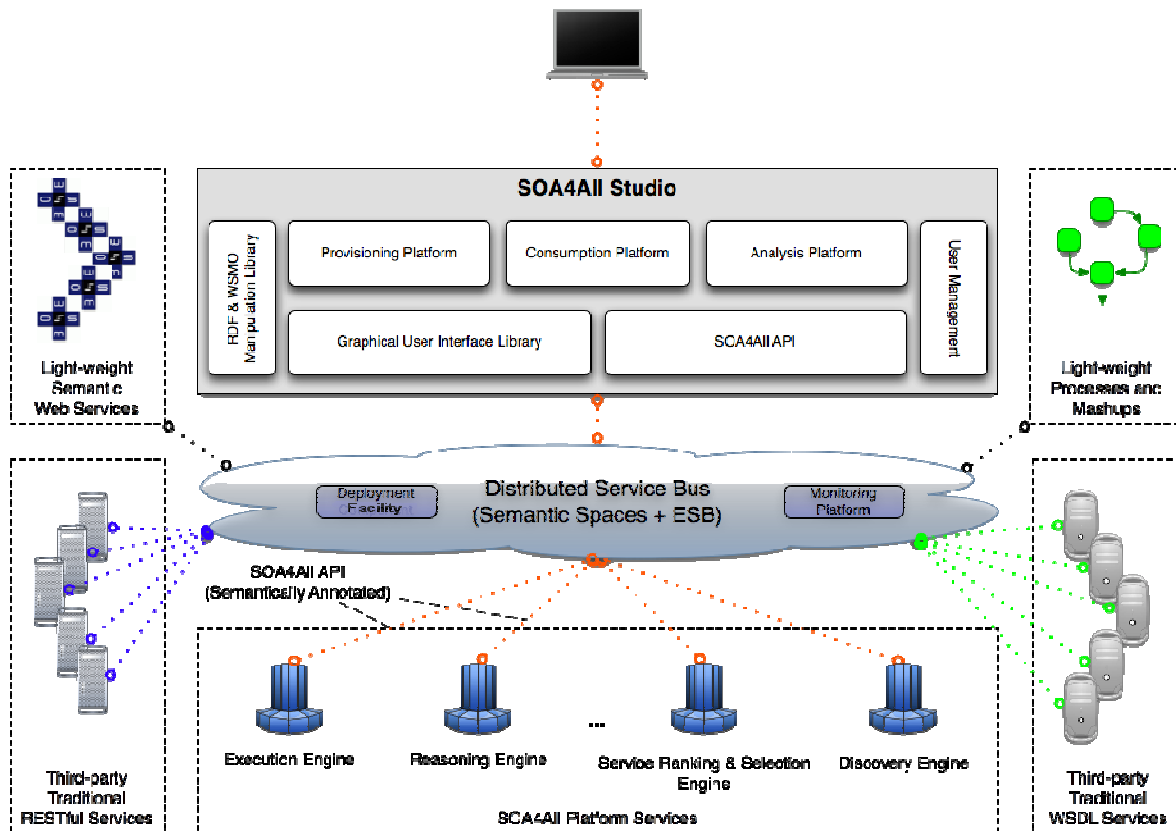


Figure 5. SOA4All Overall Architecture

3.1 SOA4ALL Studio and Platform Services

As can be seen in Fig. 5, around the integration platform, there are at the top the SOA4All

Studio and at the bottom the SOA4All Platform services. These are the components that are delivered by the various research and development work packages. The SOA4All Studio is developed in WP2 and delivers the user front-ends that enable the creation, provisioning, consumption and analysis of the platform services and various third party business services that are published to SOA4All. The studio supports different types of users at different times of interaction. SOA4All defines two major groups of users: i) a large group of service consumers that use SOA4All for the consumption of functionality that is provided by either platform services or mainly business services, and ii) a significantly smaller group of users the exploit the capabilities of the platform services (via SOA4All Studio) to annotate, select and compose Web services. The WP8 case study is aimed at both groups of users, the main work will be to enable users to create interesting compositions with Ribbit services, and once created is it is hoped that these compositions will be used by a many others over the web.

The SOA4All studio will be the most important aspect for WP8 as it form the basis for all interactions (design time and runtime) that users will have with the SOA4All platform

The platform services are products of WP3, 5 and 6 and deliver the various functionalities needed for service discovery, ranking and selection, composition and invocation. These components are exposed to the SOA4All Distributed Service Bus as Web services and hence consumable as any other published service. Their functionalities are used by the SOA4All Studio to offer clients the best possible service, while their combined activities are coordinated via DSB.

The components used by WP8 in V1 and V2 of prototypes are detailed in the table below, according to the following key:




	<p>Yes Work package outcome will heavily be used</p>
	<p>Partly/Maybe Work package outcome will partly be applied depending on the actual progress of implementation</p>
	<p>No Work package outcome will not be used directly within this scenario</p>

Table 1. Summary of components used in prototypes V1 and V2

WP	Task	Deliverable / Prototype	V1	V2
WP1	T1.2	D1.2.2 - WSMO Data Grounding Tool		
Service Web Architecture	T1.3	D1.3.3 - Semantic Spaces: A Unified Semantic Data Coordination Infrastructure		
	T1.4	D1.4.4 - SOA4All Runtime		
WP2	T2.1	D2.1.4 - Service Provisioning Platform Prototype		
	T2.2	D2.2.3 - Consumption Platform Prototype		
	T2.4	D2.4.1 - SOA4All Studio Prototype		
	T2.7	D2.7.1 - Recommendation System Prototype		
	T2.6	D2.6.2 - SOA4All Process Editor Prototype		
	T2.3	D2.3.3 - Monitoring & Management Tool Suite Platform Prototype		
WP3	T3.2	D3.2.5 - Repository Reasoner for WSML		
Service Annotations and Reasoning	T3.2	D3.2.6 - Rule Reasoner for WSML		
	T3.2	D3.2.7 - Description Logic Reasoner for WSML		
	T3.3	D3.3.2 - Established Ontology Tag Clouds		
WP5	T5.3	D5.3.2 - Service Discovery Prototype		
Service Location	T5.4	D5.4.1 - Service Selection And Ranking Prototype		
	T5.5	D5.5.2 - Service Adaptation Prototype		
WP6 Service	T6.1, T6.2	D6.2.2 - Prototypes, Tool Integration (containing the integration of the two advanced prototypes of both Lightweight and Adaptive service composition)		

Constructions				
----------------------	--	--	--	--

4. Design

The design section gives design details of the components and UIs that are specific to WP8.

4.1 UI and Front-ends

The user Interface for WP8 will follow the standardised UI defined in WP2 (see deliverable D2.4.1). Where appropriate, additional UI components will be developed (or standard ones customised) for some specialised tasks that are not provided by the standard SOA4ALL studio. All additional UIs will adhere to the standard ‘look and feel’ defined in D2.4.1 [5].

4.1.1 Standard Look and Feel

The Dashboard will provide a starting point for SOA4All users (shown below). It will show all components and elements of SOA4All and it will provide a menu structure to access them.

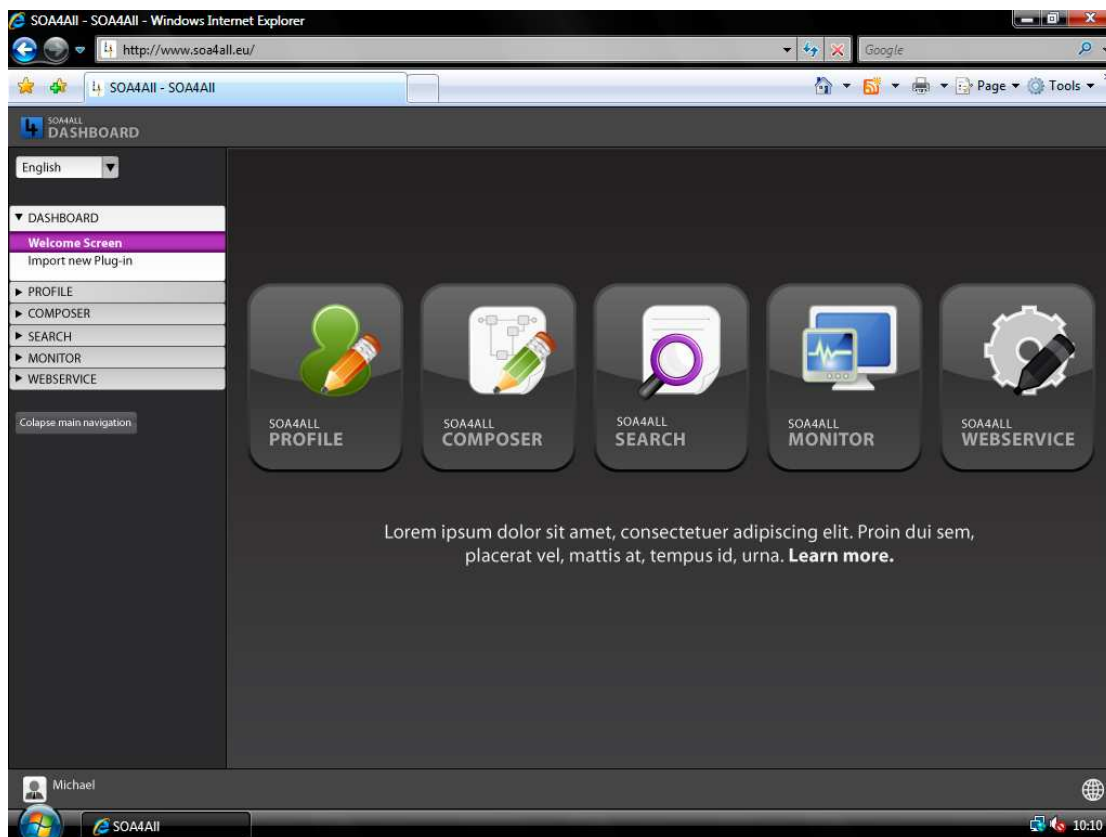


Figure 6. The SOA4All standard look and feel

For each component, the Dashboard will display a component specific text and a graphic to guide the user. Figure 6 shows five initial starting points of the SOA4All Studio allowing the user to jump to any of them.

Users may personalize the Dashboard depending on their preferences. This allows users to decide where to place information and to configure the Dashboard to hide functionalities that

they do not want to see on the start page.

In the Dashboard, new components can be included on the start page by adding a so called “Dashboard Widget”. Doing so will allow them to specify a name, an image and a description that will be displayed to the user. For a full specification of the Dashboard section, see D2.4.1 [5] section 4.6.

4.1.2 BT Ribbit Branding

Where appropriate the SOA4All UI will be branded as BT Ribbit, but will retain the standard look and feel developed by SOA4All. In most cases this will simply be the inclusion of the Ribbit logo, but might also include some other aspects of the Ribbit colour scheme.



Figure 7. Standard Ribbit Logo



Figure 8. BT and Ribbit Logo

4.1.3 Additional UI Components

WP8 will use the standard UI components where possible, but for some additional or modified features, it will be necessary to develop custom UIs. The following section outlines these

Integrated Ribbit Forum (scenario 1 & 2)

There is currently an active community and online forum for Ribbit developers. The WP8 prototype will take advantage of this, by integrating the current community functionality with the community and web 2.0 functionality offered by SOA4All. It is hoped that this will encourage current Ribbit developers to start using the SOA4All tools. Additional work will be required by WP8 to develop these integrated community aspects and will require working with the current Ribbit forum team to develop a solution that fit the purpose for both the old community and the new tools offered by SOA4All. Details of the functionality provided for community support can be found in section 5.5

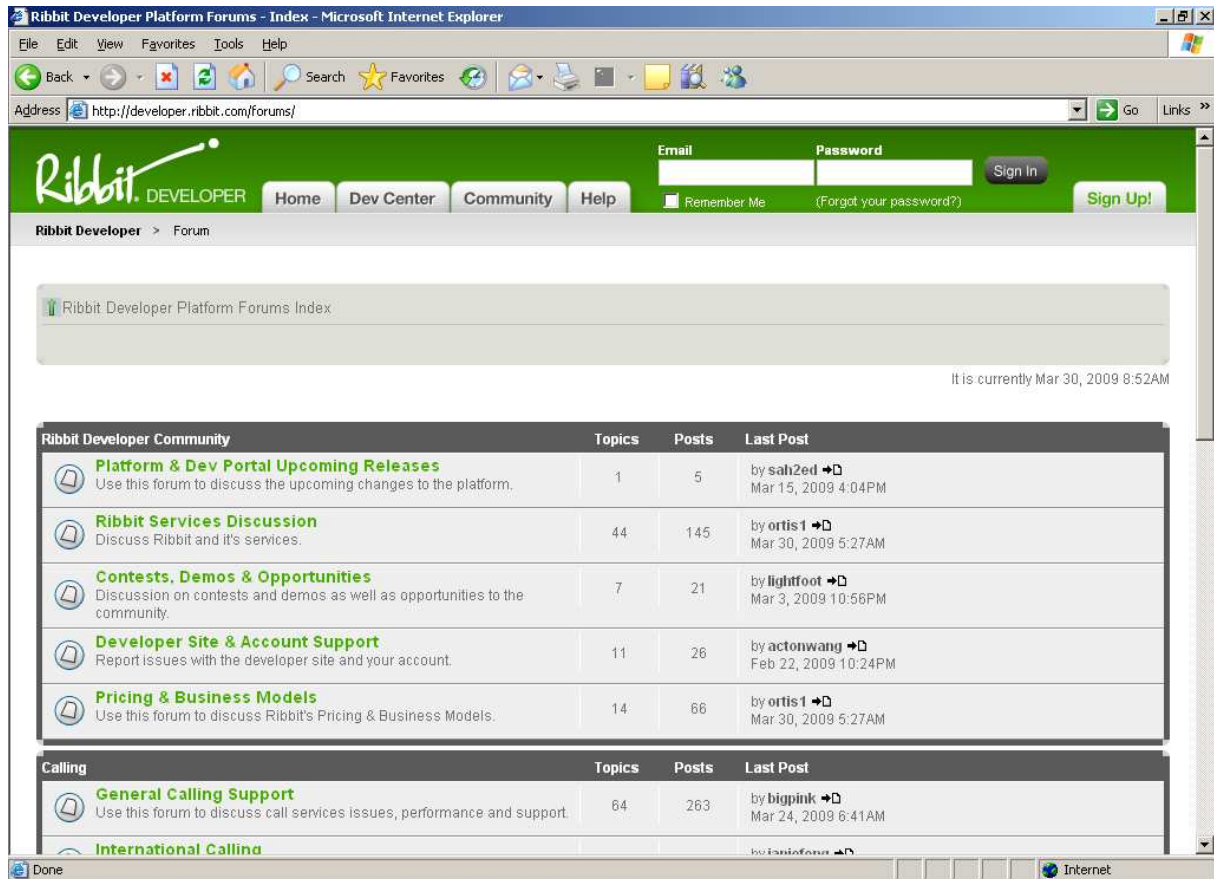


Figure 9. The current Ribbit community forum

Import OSS/BSS services wizard (scenario 2)

Scenario 2 is aimed at businesses wishing to build applications that include Telco services offered by Ribbit. The main task for these businesses will be to compose the Ribbit services with their own services to create the composite service which will form the overall application. These services will likely be categorised into the two areas

- Unique Business Services
These services are unique functionality that the business is composing with Ribbit services to add value and create a novel application. In the example given in section 2.2 the Bulk SMS service company has the unique services that manage user profiles, and work out the contextual advertisements based on user profile and location.
- Common OSS/BSS Services
These are the services that provide the Operation and Business support, and deal with common functions such as Authentication, Billing and other management type operations.

As it will be a common task for businesses to incorporate the OSS/BSS services, WP8 will provide the functionality and a specialised GUI to Import and annotate a limited set of the most common services. These will extend the general service annotation functionalities

provided by SOA4All

Domain ontology manager (scenario 2)

Over the last few years, the Telecommunications industry has made progress towards embracing standards relating to Telco data and process. A detailed analysis of ontologies for the Telco domain can be found in deliverable D8.1 [6] section 4.3. There are already many Telco services that support the data and process standards described in D8.1, and WP8 will support and encourage the use of these in the SOA4All platform.

A custom GUI will be developed to manage interactions with the standard Telco domain ontologies, and will assist business users to import the services that are compatible with the them. It will also assist and encourage users to annotate services using these domain ontologies when a compatible service is identified.

QoS monitoring and management wizard (scenario 2)

Scenario 2 involves businesses creating applications that will need to have a degree of monitoring and management (rather than fun non-critical applications in scenario 1). Telco services have a specific set of parameters for measuring the Quality of Service (QoS). WP8 will provide a GUI to define and manage these in a way consistent with other QoS management tools in the Telco domain. The GUI will be an extension of the general monitoring and management tool described in D2.3.1 [7]

4.1.4 Storyboard and mock-ups

The sequence described in Annex A gives a detailed walk-through of the steps taken in scenario S1 with accompanying screenshots.

4.2 Service Discovery and Selection

Service discovery is the process of searching for a service that meets the user's required functionality. Selection is the final step where a user picks an actual concrete service based on a number of services that meet the initial search criteria.

Discovery and selection functions will be accessed by the user, either in the dedicated SOA4All search tool of the SOA4All Studio (see step 2 in the walkthrough) or as embedded function of the other Studio tools like the SOA4All Composer.

For the most part discovery and selection will make use of the generic functionality provided by the SOA4All tools. Users will be able to define simple keyword based searches, or define more complicated semantic queries, by defining Goals. In addition to full semantic queries using Goal templates, WP8 will also extend the basic keyword search to allow the specification of some Telco specific meta-data about the service (such as it's author, basic type, QoS or performance etc.). This will allow a search more expressive than the standard keyword search but less than a full semantic search.

4.2.1 Meta Data enhanced discovery

The Client-Side filtering widget (as described in D2.4.1) will be extended to include domain

specific meta-data, that can be used to refine the search. This is more relevant to Scenario 2 where users will want to search on Telco related Meta-Data such as Quality of Service (QoS), Service Level Agreements (SLAs) and performance parameters.

Moreover, all users will be provided with filters that display (1) the services and compositions they use most frequently, (2) the services and compositions they have used most recently.

4.2.2 Full Semantic discovery, Using Goals

The SOA4All Studio will provide designated tools for supporting the user to select a goal that is then automatically linked to one or more concrete services, as described in D2.2.1 [8]. The concrete services can then be invoked via the consumption platform, described in section 4.6

The prototypes will provide a number of goals templates that will guide the user in defining semantic queries for the most common types of requirements that users of the SOA4All Ribbit platform will perform. The Goal templates provide the skeletons for a semantic query and enable the user to fill in details in an easy to use manner with forms and drop down boxes. Goal templates will allow users to perform much more targeted queries and provide an enhanced discovery process compared with keyword based searches. It will make use of the Discovery and Reasoning tools developed in WP 3. The Goal Completion component will help the user in finding an appropriate goal and in specifying missing information, the natural language processing component allows the user to find goals based on descriptions typed in textually using natural language. Concrete method interfaces to these components will be defined by T2.2 at a later stage of the project.

4.3 Service Compositions

As described in D8.1, users who build service compositions are not expected to have a detailed knowledge of the underlying technology such as WSDL, ontologies languages or process modeling formalisms usually used in BPM tools. To support these users in building compositions, SOA4All will provide a lightweight modeling language (WP6), the use of semantics to support the discovery (WP5 and WP3) and composition of services and processes (WP6) and to hide details via goals (WP2 and WP6) as well as a user-friendly GUI that also offers wizards on demand to guide the user in modeling tasks (WP2). Some customization to the generic functionality is required for WP8, which is described below.

4.3.1 Customizing the SOA4All Composer

The starting point for the WP8 user choosing a modeling activity is the Dashboard, which is the entry point of the SOA4All Studio (see D2.4.1). The user chooses the SOA4All Composer functionality as described in D2.6.1 [9]. Now, the user has different options to build service compositions – e.g., search, browse bookmarks, start modeling wizards, use patterns and templates etc. Figure 10 shows a screenshot of the SOA4All Composer UI during modeling. Users with a degree of technical knowledge may model a new service composition in the so-called “free mode” by placing the graphical elements of the modeling language (start process, stop process, activity, and connector) via drag and drop operations on the canvas. Details of an activity (e.g., service parameters) can be specified in a details box displayed for the selected activity. For novice users there will be the option to only use services with full semantic descriptions or wizards for integration. This will mean they will not have to deal with any details of data mappings between services, as the composer will have enough information to work this out automatically, or semi-automatically

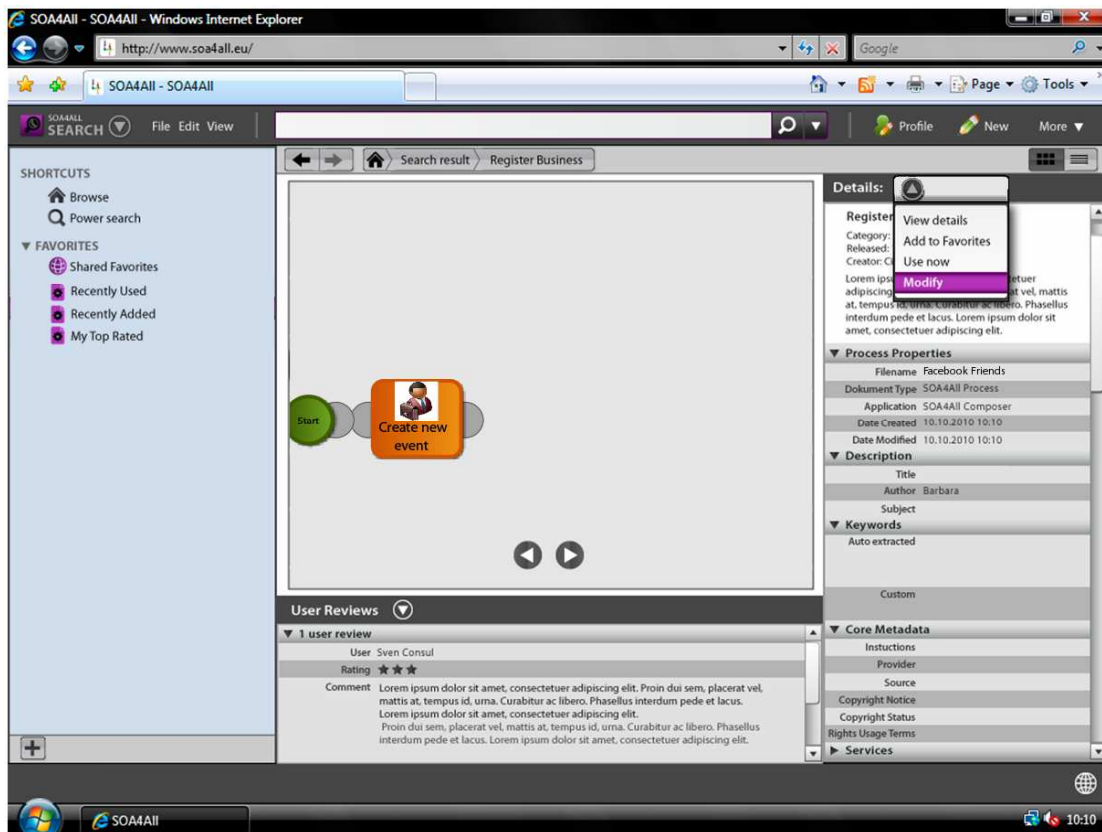


Figure 10. Building a composition in the SOA4All Composer

Activities can either be concrete services or goals in case where the user is not able to provide the information required to specify a concrete service, or does not wish to at this stage (e.g. late binding of goals to services may be desired).

4.3.2 Domain-specific Wizards for the SOA4All Composer

WP8 will implement extensions to the SOA4All Composer performed to include wizards to assist with adding services into Telco based compositions.

This will assist the user in completing common tasks associating with creating Telco based service compositions. The wizards will allow users to easily add in Telco functionality into a composition such as Voice, SMS or Text to Speech. For the list of current services offered by Ribbit, see Annex B.

For instance, as described in Step 5 of the scenario 1 (see Annex A), a smart, domain-specific wizard *BT Get Location* is implemented to allow this functionality to be easily added.

4.3.3 Domain-specific Process Templates

As described in D6.3.1, there is agreement that process patterns and templates can accelerate the process of designing a solution and reduce modeling time. The templates hereby can be generally applicable or domain-specific. The SOA4All Composer will allow for the creation of both kinds of templates. In the context of WP8, we will develop several process templates for Telco Services. The actual modeling of a template does not differ from

the modeling of a regular service composition.

4.4 Annotations and Community Support

The aim of the WP8 case study is to encourage people to create new and innovative applications with BT and Ribbit services, enabled by the SOA4All platform developed in the project. One of the ways of encouraging innovation is to provide tools that encourage and support community working. It is hoped that users will work together to create new service compositions and help each other with specific tasks involved in building compositions (such as discovering, annotating, and joining services). The section below details the specific community functions that WP8 will investigate in addition to the generic community support.

4.4.1 Community Sharing of Compositions & sub-Compositions

Sharing of compositions and sub-compositions will enable more than one user to work together to complete a task. When a user has started work on a composition, they will have the option to share their work to the rest of the community. They can choose to share the whole thing, so that other users can edit and update any aspect of the composition, or only specific things (such as the semantic annotations). User profiles will contain details of specific interests and community settings, and will inform interested users of new compositions that are relevant to them. The following additional service will be implemented:

```
ShareCompostion(bool Services, bool DataMappings, bool Tags, bool  
SemanticDescription)
```

A sub-composition is defined a composition of more than one service that is part of a larger composition, but is useful in its own right. For example in Step 5 of the walkthrough of Scenario 1 presented in Annex A, the user has composed two services together (get friends & find location). The full composition goes on to join other services, but the composition of these two services is a useful function (i.e. get the location of my friends) which would likely be desired by other users. Where this occurs, it will be possible to define and share these sub-compositions, so that a library of useful one is built up. As with individual services and full compositions, sub-compositions can be annotated with semantic service descriptions to enable their discovery. The following additional service will be implemented:

```
ShareSubCompostion (List ServiceList, bool Services, bool  
DataMappings, bool Tags, bool SemanticDescription)
```

4.4.2 Data mappings

When a user builds a service composition, it will often be necessary to pass data from the output of one service into the input of another. In a situation where there are two fully annotated services with semantic descriptions, and using the same underlying ontologies, it is possible for the system to work out the mappings between input and outputs automatically. In a situation where there are not full semantic descriptions, or different ontologies are used it may be necessary for the user to work out some mappings manually. Once these mappings have been defined, this data is valuable for other users who need to join the same services in a composition. In the WP8 prototypes there will be the ability to capture the information about

data mappings and share it with other users if the same mapping task occurs. In a situation where different users have performed different mappings with the same services, the user will be presented with the option to choose between them (or ignore and do his own if he so wishes). When multiple mappings are available ranking of the mappings will also be determined by similarity of users' profiles and information about the service composition they are creating. The following additional services will be implemented

```
CreateDataMapping(Data ServiceAOutput, Data ServiceBInput)
```

```
SuggestDataMapping (Data ServiceAOutput, Data ServiceBInput)
```

```
SuggestRankedDataMapping(Profile UserProfile, Data ServiceAOutput,  
Data ServiceBInput)
```

4.4.3 Service Annotations

The Service descriptions are stored in a shared repository (see Section 5.6). In SOA4All, the shared repository is implemented using a distributed Semantic Space (see D1.3.2A [10]).

The service compositions created by the user are also semantically described. The user therefore creates some annotations for the composition as a whole using the annotation editor developed in the service provisioning Task 2.1 (see D2.1.2 [11]). The standard annotations within SOA4All cover input and output data, preconditions, effects, and capabilities.

Within WP8, we will define specific annotations for common services and functions. For the common Ribbit services that will be used frequently in WP8 compositions, the prototype will provide fully annotated service descriptions. In the case of scenario 2, the prototype will provide annotations for common OSS functions (such as billing, authentication, error reporting) that a business is likely to include in service compositions they create. These will be linked to standard Telco domain ontologies and accessed with a specialized UI described in section 4.1.3

4.4.4 Ratings and Rankings

WP8 will extend the Rating widget (as described in D2.4.1 [5]) in order to describe how useful a service is for a certain category. Users will be able to tag services to provide category information for service compositions (e.g. Locate Friends). WP8 will provide an extra option for the rating result according to the category. Furthermore, we will use the standard SOA4All rankings based on security, availability, reliability, provider trust etc. The recommender system developed in T2.7 will be used to sort result lists generated in a search according to the estimated relevance for the user.

In order to support the sharing and reuse of service compositions, WP8 will provide annotation mechanisms for comments and ratings that can then be used by the search and filtering mechanisms developed in T5.3 and the recommender system developed in T2.7.

4.5 Service Storage

In the scope of SOA4All there are two layers relating to storage of services:

- WP1 provides the Semantic Spaces, which are used as a shared memory to build repositories (see D1.3.2A [10])
- WP2 provides the Storage Services, an easy to use API built on top of the Semantic

Space that provides simple storage functionalities (see D2.4.1 [Error! Reference source not found.]). Those services consist of fundamental functionalities that enable RDF creation, update, deletion and querying (SPARQL) functionalities. Also the WP2 storage services allow the storage of files such as WSDL description files.

The SOA4All Composer builds upon the WP2 storage services (see D2.6.1 [9]) to store the service compositions themselves as well as the semantic annotations.

For most cases WP8 will use the generic storage facilities. In Scenario 2 there may be the requirement for businesses to keep some services private (i.e. not access by all), in which case they will host these on a private accessible space, that will interact with the other public spaces to communicate with public services.

4.6 Service Deployment and Execution

The SOA4All Studio includes a Service Consumption Platform (see D2.2.1 [8]) that enables the lightweight consumption of services in an easy-to-use Web 2.0 fashion. This platform is envisaged as an area where end-users can interact with services as consumers in a highly personalised environment.

From the technical point of view, the platform uses the concept of Goals as the underlying semantic artefact that users will use to discover and consume services, but at the same time it is worth mentioning that the non-technical user is not expected to know about those technical details when interacting with the platform.

Regarding the case studies of SOA4All, such as the one described in this deliverable, there are some issues that should be addressed in order to explain how the Service Consumption Platform could be helpful for real world scenarios, as discussed below.

4.6.1 End user service consumption

The first concern that arises when mentioning the concept of “platform” is about the place where the end-users of BT’s facilities will consume services. While we talk about the Service Consumption Platform as a Web-based application that the user will connect to, enabling interaction with every (semantically enriched) Web service created within the use case, it is important to note that different scenarios regarding the location of services are possible.

The two foreseen options are:

- Users consume services **inside** the Service Consumption Platform: This implies that the platform will act as a marketplace for the users to consume every kind of service within the platform, even the composed scenarios addressed in Section 2.
- Users consume services **outside** the Service Consumption Platform: This alternative way means that the users will not need to connect to this particular website in order to consume services, but they will interact with them in different endpoints. In this case, the platform itself will offer the ability to generate widgets that can be incorporated in other pages, so users will still be able to consume services from different places, satisfying the (platform-enabled) consumption of services from outside the platform.

This way, the case study will be able to benefit from the Service Consumption Platform even though the envisaged consumption does not happen directly in the platform.

4.6.2 The adapters

It is worth noting that the consumption of services envisaged by the project implies several layers of operation.

- At the lowest level, the consumption of a service will be achieved by interchanging data, typically XML for the invocation of WSDL-based services, and potentially for the responses of REST-based services as well.
- Those data structures are covered by a Conceptual Layer, where ontological instances and concepts will be used to treat the data at a semantic level, with lifting and lowering mechanisms available to move between the data structure and semantic levels.
- At the top level, there is an Application Layer, which uses some adapters to transform the inputs of the user into the necessary ontological terms that can trigger the whole consumption process, and can also translate the outputs of the process into the desired presentation.

These layers and the lifting and lowering mechanisms and adapters addressed are shown in Figure 11.

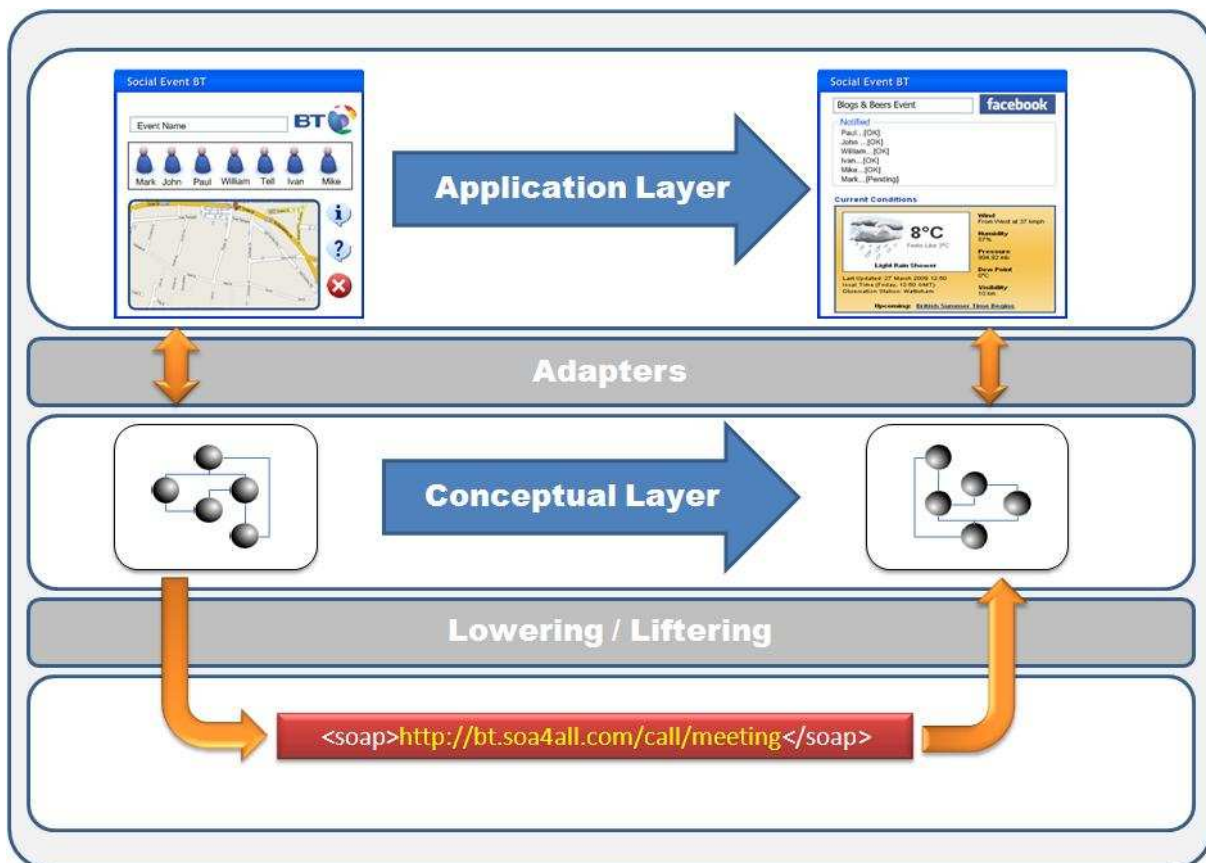


Figure 11: Layers Involved in Service Consumption

What we foresee as necessary in the case studies, and in particular in the one described herein, is the development of the aforementioned adapters that can map the ontological concepts of the Conceptual Layer into presentation elements of the Application Layer, in order to satisfy the necessities of BT's platform.

Particularly, the two foreseen adapters are:

- Input adapter: Displays the relevant information to the users letting them define the characteristics of the invocation to be performed, transforming their selections into the semantic definitions to be used.
- Output adapter: Once the execution has taken place, the output adapter gets the semantic information from the level below in order to present the user the relevant results in a suitable format.

It is important to note that these adapters don't just deal with CSS-style transformations, but also cover the elements that the user will be presented in order to conform the semantic information of the level below, or what kind of results are relevant for the end-user, once the service has been executed.

The development of these adapters will permit BT's end-users to abstract from the underlying technical details when interacting with services.

4.7 Monitoring and Management

The monitoring and management tool suite of the SOA4All Studio provides information about executed services and processes for service consumers, composers, annotators, and providers (see D2.3.1 [7]). Service consumers can check the status of all running process instances.

Scenario 2 will make the greatest use of the monitoring management since in business critical applications it will be important to ensure that the processes are performing to an acceptable level. In scenario 2 a third party company might have a Service Level Agreement (SLA) in place. An SLA is a part of a service contract where the level of service is formally defined. In the case of SOA4All services it might define things such as maximum downtime, response time and other factors relating to performance. The monitoring and management tool will be used to define and monitor the SLA parameters. When a parameter is breached, a specific action can be triggered, such as emailing the BT support team to look into the problem.

As well as runtime monitoring of individual processes, the tool will also be used to aggregate information about total usage of services and other statistics (such as location and type of users) that may be useful to Ribbit and third party companies (in scenario 2)

5. Evaluation Workshops

Two focus group half-days will be organised, one at BT for an internal BT audience, and one at the Being Digital Mashups event in London (<http://www.being-digital.com/mashups/>). The aim is to encourage participants to share and discuss their opinions and experiences related to the SOA4All vision as instantiated by the WP8 case study. The specific design choices underpinning the *Mashup/Widget discovery, composer and personalisation component of the SOA4All studio* will also be discussed. The result of the focus group will be used as input to the software development of the first SOA4All prototype.

In this respect, the workshop offers an important opportunity for system designers to carry out preliminary studies and ensure all end user requirements are collected and understood at the software design phase. On the other hand, it offers an opportunity for BT staff and target end users from the telecom domain to learn about recent developments in the service-based technologies that can help in providing flexible services.

In summary, the main objectives of this workshop are to:

1. Obtain general opinions of the end-users about end user development of service-based software in the telecoms domain;
2. Evaluate the current mock-ups of the discover, composer and personalisation editor as customised by WP8 within a participatory design process;
3. Capture as many discovery, composition and personalisation editor requirements as possible.

We aim to have at least 15 participants for each of the two planned sessions, to be divided into 3 groups of 5 people in each group, plus one moderator. The user profiles should match the profiles of the target users envisioned by WP8. The session structure is as follows. A 30-minutes introductory talk will be followed by a 20-minutes discussion on the perceptions about risks and benefits of the envisioned mode of user-driven service composition, and on existing practices and proposed supporting actions. A short notational study will discover how participants understand core proposed representations such as boxes and lines.

The discussion will then focus on alternative designs for an end user tool for web service discovery, composition and personalisation in the telecoms domain.

The discussions will be conducted in groups of 4 to 7 people, depending on the number of participants. Each group will have a facilitator to steer the discussion. Questionnaires and audio tapes will be used to record the participants' responses for later analysis.

Our optimal number of groups will be 3 groups of 5 participants, where each group discusses two of the three alternative designs. The moderator of the workshop should encourage and facilitate free-flowing discussion by:

1. Going through the design rationale behind each alternative and explain the related mock-ups;

2. Asking the participants to answer questions related to the current alternative by filling in questionnaires;
3. Iteratively repeating step 1 and 2 for the other alternative;
4. Discussing the collected answers by all participants.

5.1 Designs for Composing Web Services

The workshop will focus on three alternative approaches for composing web services and their potential problems within the SOA4ALL design studio.

1. Data Flow (a stateless mash-up)
2. Control Flow (a simplified BPMN)
3. Assisted composition (user just brings services together, computer tries to put them in order)

Each of these design approaches will be introduced by a scenario that will show the creation of simple Telco applications using Ribbit Services. Each scenario will be covered by one group of participants with the aim of capturing their ideas, feelings, and thoughts. Further details of the design approaches can be found in the generic workshop proposal in D2.5.1.

5.2 Design Rationale

For effective design exploration, we suggest the use of a combination of design rationale, use case scenarios and if possible early prototypes. Design rationale aims to support system designers by representing the argumentation and reasoning behind the design process. This justification of design decisions is important to understand, change, or recreate a design. In this workshop, Issue-Based Information System (gIBIS), an argumentative notation, will be used to represent issues, arguments, and resolutions. The graphical representations in Figures 12 and 13 show pathways starting from the root node (corresponding to the issue), to children nodes (corresponding to the solutions), and to leaf nodes (corresponding to the arguments). These design rationale diagrams enable comparing and establishing relationships between the various design solutions. For each issue, a gIBIS design rationale is produced as follows (here we focus on composition but discovery and personalization need to be considered in the same way):

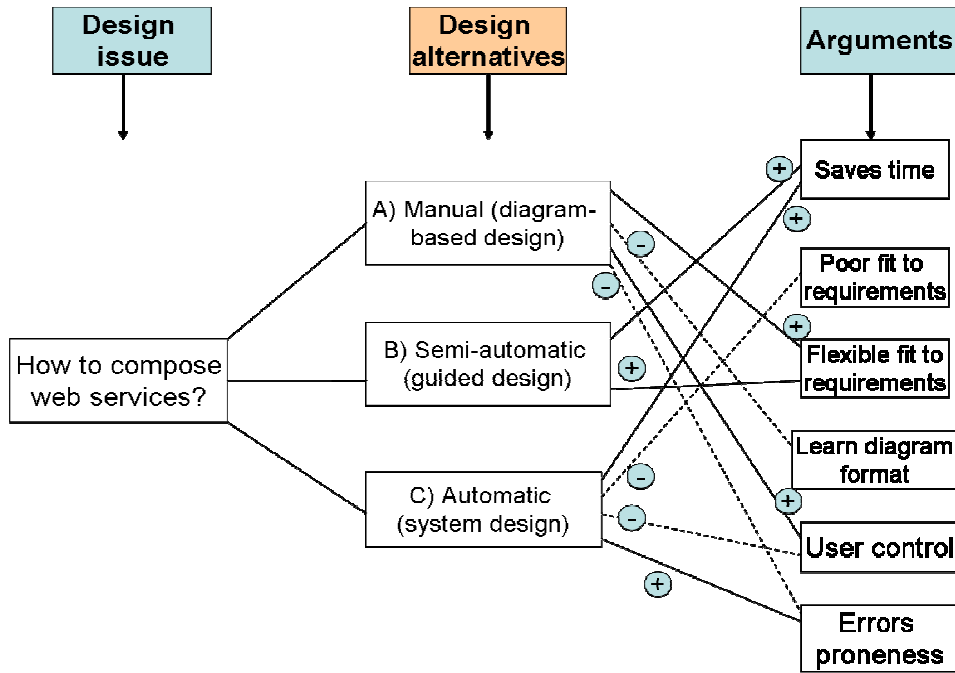


Figure 12. gIBIS design rationale for Composition (Issue 1)

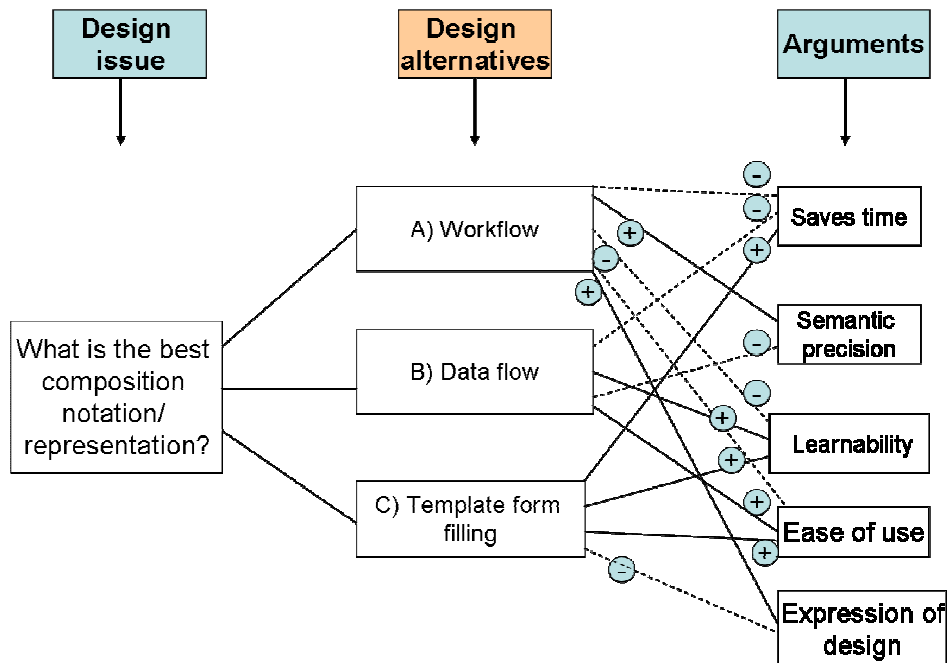


Figure 13. gIBIS design rationale for Composition (Issue 2)

5.3 Top-level issues to be discussed

The workshop will cover several topics related to general end user development in the telecommunication domain and the design choices underpinning the SOA4ALL composition editor. In this respect, information will be collected about the software tools used by target end users in their daily jobs. Of particular interest would be information about software tools used for service composition in the telecommunication domain and their frequency of use.

An important aspect of this workshop will be to acquire information about the background of end users, their general ideas and thoughts about software-based service composition. This type of information will be acquired by asking specific questions about the type of tools that end users consider important for their jobs and the specific features that they like or dislike about the current tools that they use in their daily jobs. Apart from covering the technical aspects and software related issues, some questions will be focused on end user development, such as:

- What are the benefits of end-user development?
- What are the risks of end-user development?
- What strategies / approaches do end users follow when developing applications?

After discussing the general aspects of software tools and end-user development, the workshop will specifically focus on the design choices underpinning the SOA4All web service composition editor. In this respect, screenshots or mock-ups of the currently developed tools will be shown on the wall or distributed in hand-outs. Workshop participants will be asked to provide their feedback on the initial design of SOA4ALL tools. The feedback of workshop participants will help in determining whether the currently developed tool (represented by screen shots or mock-ups) will be able to address the potential problems in their daily jobs.

The detailed list of issues that will be discussed in the workshop is specified in the generic workshop proposal (for WP2, WP7 and WP8) in D2.5.1. The later proposal also contains the workshop agenda that describes the sequence of activities in the proposed workshop.

The feedback and recommendations of workshop participants will be recorded during the workshop. Once the workshop has been carried out, the participants' responses will be collected and analysed. Recorded material and unstructured questionnaire responses will be analysed using Grounded Theory approach, quantitative questionnaire answers will be analysed using conventional statistical methods.

6. Conclusions

This deliverable has outlined the design of additional components that will be developed for WP8 in order to fulfil the requirements specified in Deliverable 8.1. The overall aim of the case study is to provide the next generation of Ribbit where the process of discovering, integrating, using and sharing BTs services can be done much more effectively. In most cases the generic functionality provided by the technical work packages WP1-WP6 meet the requirements (as detailed in Table 1), however some extensions are necessary and have been discussed in this deliverable.

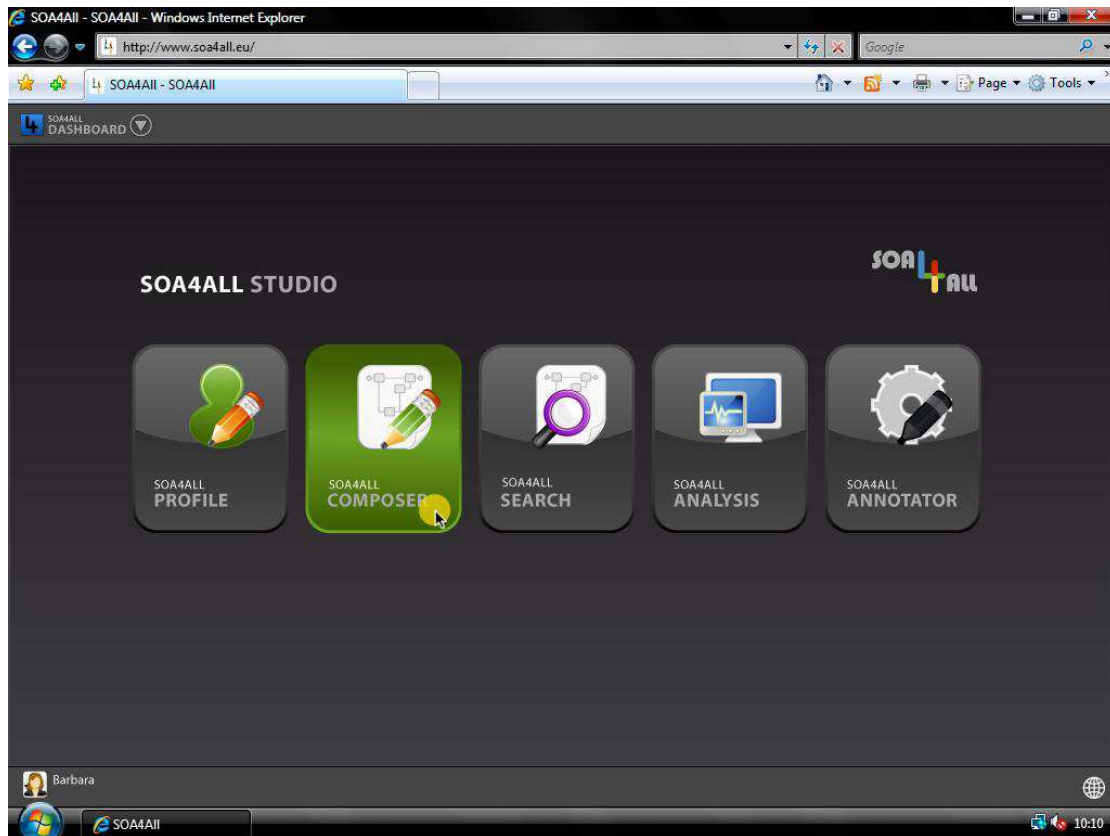
The next step in this WP will be to implement a first prototype due for August 2009 (M18) which is based on scenario 1, described in section 2.1.

7. References

1. Web Services Description Language (WSDL): version 1.1, W3C Note, March 2001. Available at <http://www.w3.org/TR/wsdl>, last accessed on 19/03/200921c
2. Quality of Service, Wikipedia entry, http://en.wikipedia.org/wiki/Quality_of_service
3. Service Level Agreement, Wikipedia entry, http://en.wikipedia.org/wiki/Service_level_agreement
4. D1.4.1 SOA4All Reference Architecture Specification, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>
5. D2.4.1 SOA4All Studio First demonstrator + Interface Specification, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>
6. D8.1 Web21c Requirements, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>
7. D2.3.1 Service Monitoring and Management Tool Suite Design, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>
8. D2.2.1 Service Consumption Platform Design, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>
9. D2.6.1 Specification of the SOA4All Process Editor, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>
10. D1.3.2A Distributed Semantic Spaces: A Scalable Approach To Coordination, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>
11. D2.1.2 Service Modelling Tools Design, Available at <http://www.soa4all.eu/file-upload.html?func=select&id=2>

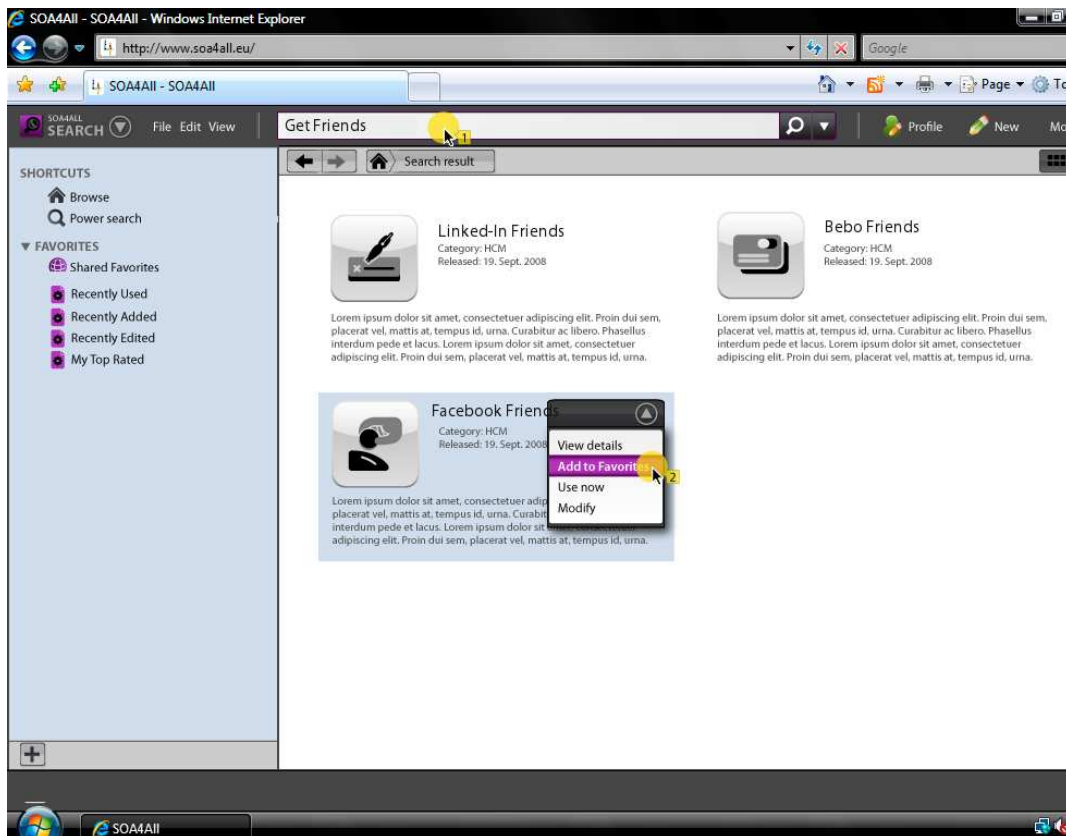
Annex A. Walkthrough for Scenario 1

Step 1.) The User logs into the system and Selects option to create new Telco application

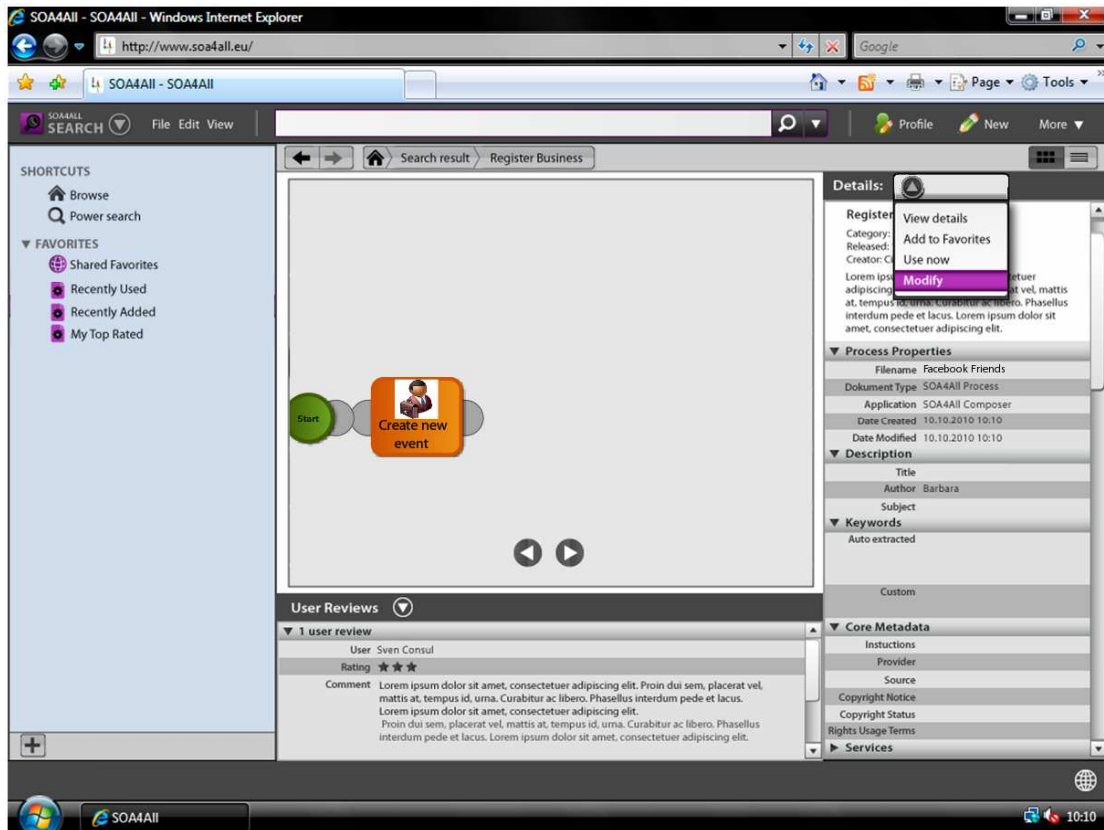


Step 2.) The User searches for appropriate service by using one of a number of methods

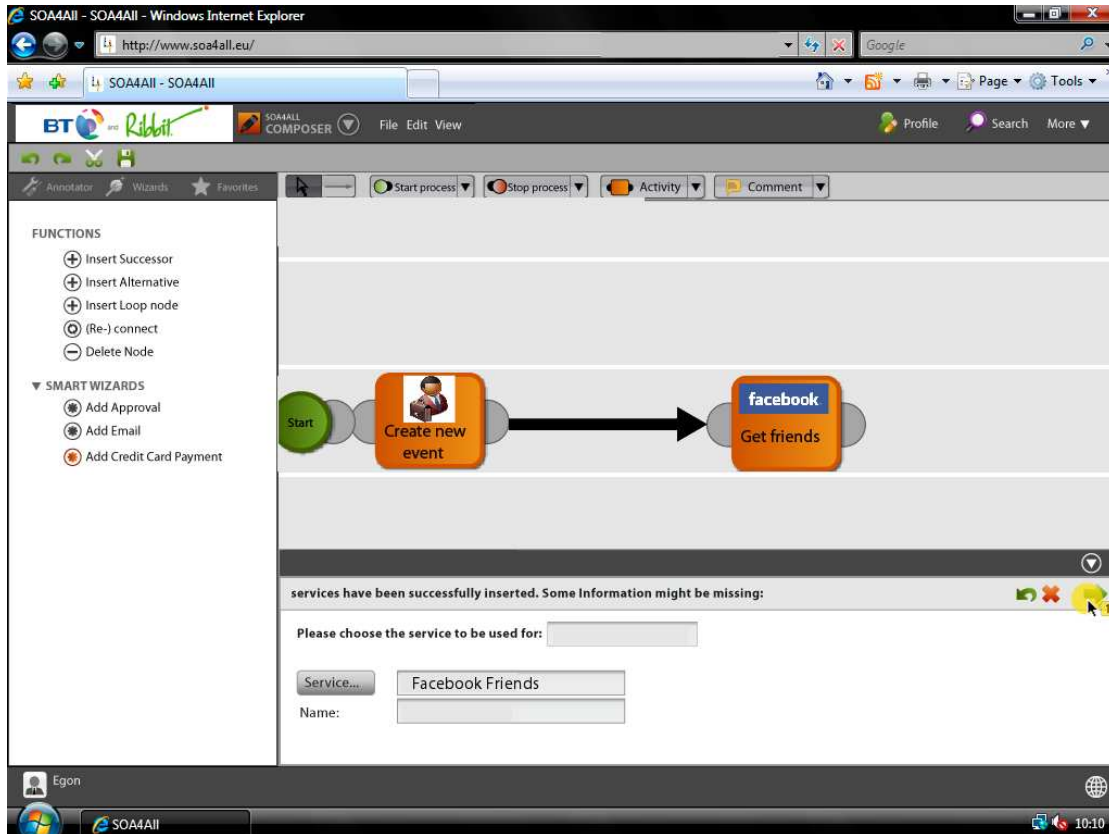
- a. searching via keywords or more complicated semantic queries regarding goal of service and input/output.
- b. browsing a hierarchy of services sorted by topic



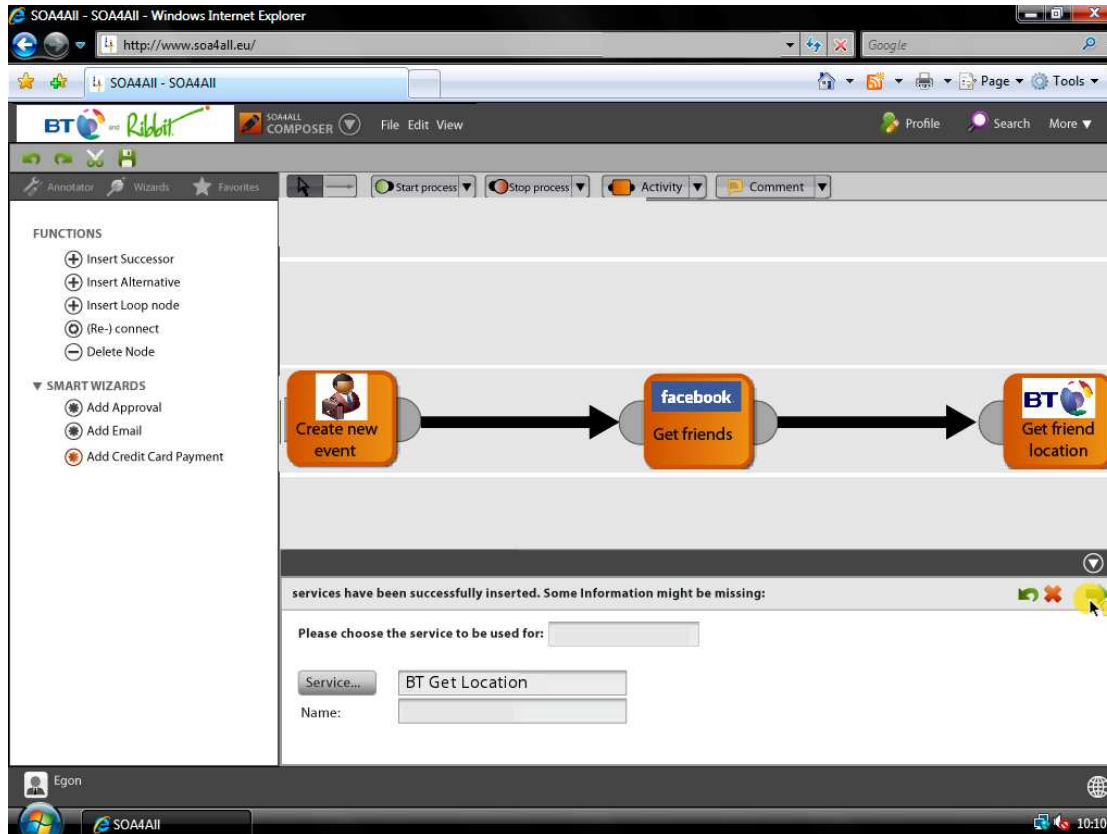
Step 3.) User checks details and community reviews for selected service



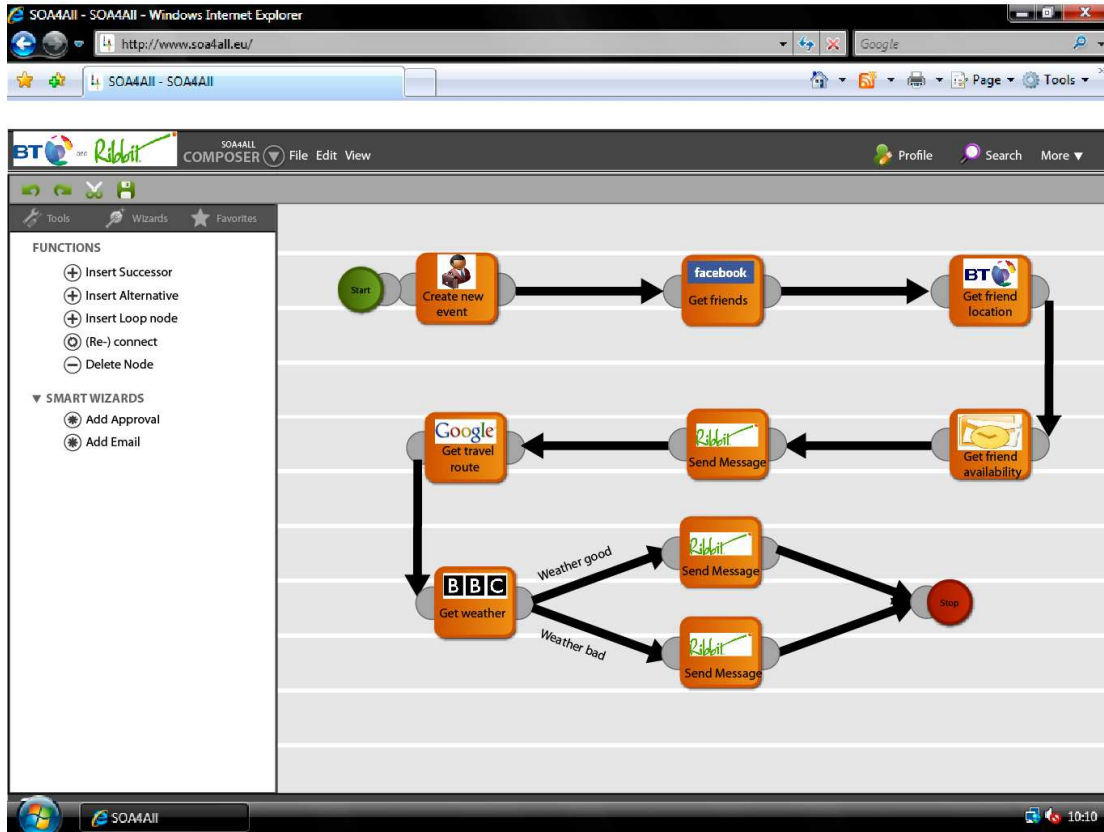
Step 4.) The system helps user to ‘wire’ service into composition, by offering matches to input and outputs (or possibly via some mediation service. Where system cannot help, the user may have to manually define data mappings between the services. These are captured by the system for re-use in future compositions.



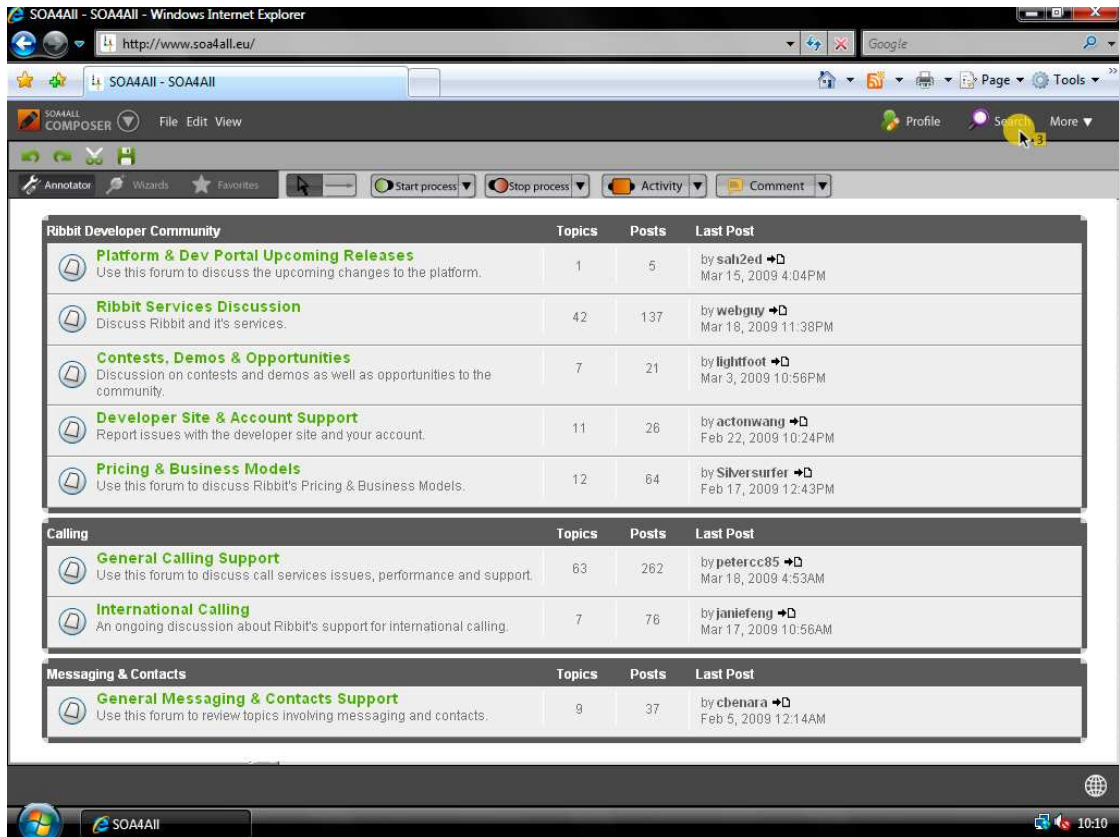
Step 5.) From combination of context and selected services, system decides matching services suitable to compose. Services can be individual atomic services from BT and third parties, or can be composed services already created and exported with the SOA4All architecture by other users. Where available wizards will help the user to 'wire' the service into the composition.



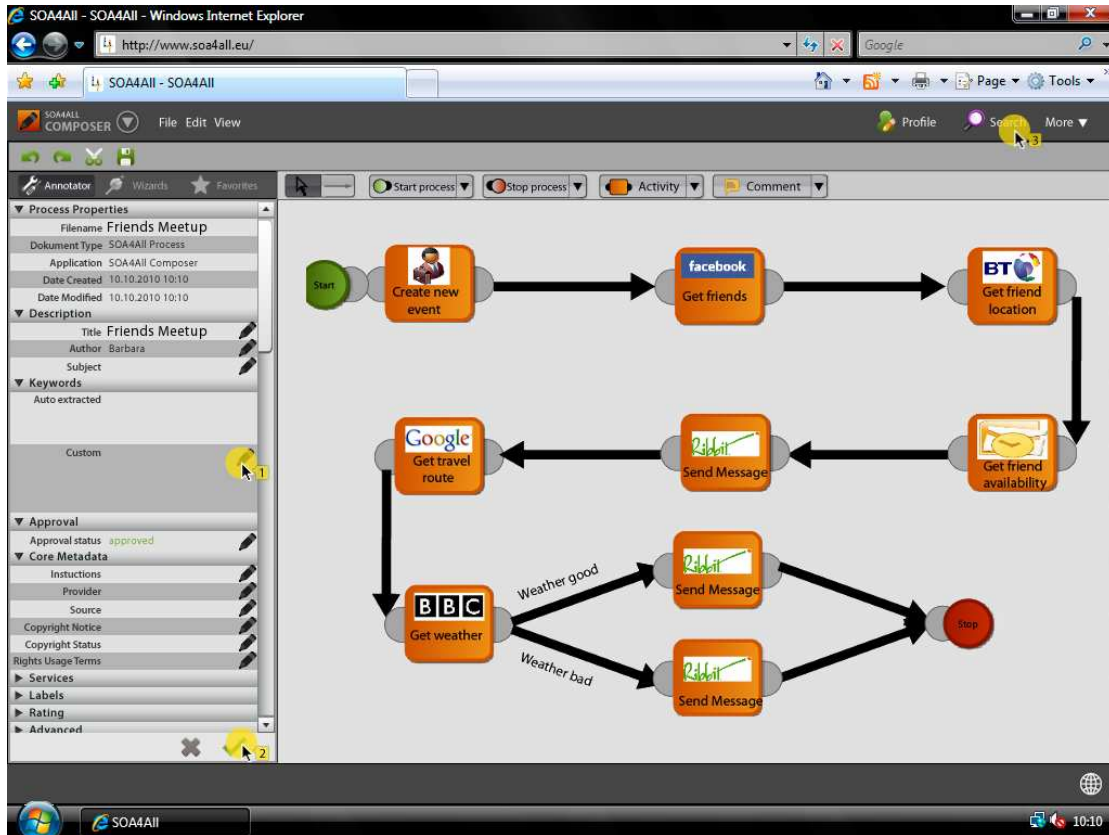
Step 6.) These sequences of steps are repeated until the composition is complete. The user can choose to add basic tags to describe the service composition, or annotate with a full semantic description (using the annotation wizard). The completed application can then be saved and shared with other users of the community.



Step 7.) Based on the users different profile interests and settings, they are informed of the new service composition.



Step 8.) A Second user looks at the service composition and adds a more detailed semantic annotation to the service description.



Annex B. Current Ribbit Services

This Section gives a list of the current Ribbit Services. Further details on individual services can be found at <http://developer.ribbit.com/>

Ribbit services are currently offered via Flash Toolkit, Flex SDK and subset of services is exposed as REST (can be accessed via JavaScript or PHP SDK).

The 1.0.3 Beta release of the RESTful API supports the basic operations such as:

- **Users management** – creation and management of Users of the Application
- **Devices management** – registration and configuration of Users' Devices
- **Making Calls** – creating Calls between fixed, mobile and VoIP numbers, browsing of Call Log and details
- **Text Messaging** – sending of Text Messages and browsing of Messages Log and details
- **Voice messaging box w/ player** – management of Voicemail records and transcription of Voicemail (English only)
- **App distribution of multi-user apps (token authentication)** – token-based authentication for building of applications with multiple users.

Further releases of the RESTful API will increase the number of ribbit services offered.

The Ribbit Flash Toolkit includes drag-and-drop controls, which Flash Developer or Designer can use to quickly compile and deploy everything from prototypes to full applications. The Ribbit Flex SDK is the programmatic API layer to the Ribbit global phone network.

Flash/Flex Ribbit offering includes services such as:

- **Embedded flash (web) phone** – possibility of extending any web page and turning a web browser into a two-way communications device. Users can have online conversations with others (connecting to another Flash phone or any other type of phone worldwide). In addition, AIR Flash phones can be converted into downloadable local applications that run right on the desktop.
- **Inbound calling** – extending the application to accept incoming phone calls, turning it into a two-way phone or executing a predefined programmatic event (which can be based on rules).
- **Outbound calling** – allowing application to place calls to any phone number worldwide or any online application. By using Flash on the client, Ribbit is able to access the microphone and speaker to enable a 2-way live conversation
- **Text messaging** – text messaging can be embedded into the application and send text messages to one or more recipients

- **Voice messaging box w/ player** – message box that lists member messages, presenting metadata (such as message state, origination caller name, origination caller number, time date stamp, and message transcription) and allows streaming of the audio message to an end user.
- **Group voice message sharing** – enables sending of voice messages to one or more recipients (via Ribbit phone number or any email address) from the application.
- **Notification services** – email and text message notifications for received messages. It is possible to include the message transcription and the message envelope information (such as the caller name, number, and the time of the call) and a direct callback number to listen to the specific message. If an email is sent, then the audio message can be attached in either a WAV or MP3 file format.
- **Transcription** – transcription engine that has the ability to process each new voice message and the audio is transcribed to text (English only).
- **Phone numbers or purpose numbers** – Ribbit-assigned phone numbers that application can use to accept incoming calls. The standard call flow is to route calls to the member's flash phone. If the phone is not answered, the call will route to voicemail.
- **Mobile phone linking** – mobile phones can be linked to Ribbit by using conditional call forwarding (CCF). Once a mobile phone is linked, incoming calls are routed to Ribbit if the mobile phone is not answered (instead of the mobile phone voicemail).
- **Telephone User Interface (TUI)** – possibility for the developers to dial-in to voicemail. The Ribbit TUI phone number is (619) 916-2500. The dial-in interface allows end-users to check voicemail, make phone calls, send messages, and configure account settings.
- **Address Book** – personal address book available for every user. This address book holds business and personal contact information. Developer applications can add, edit, and delete entries in the address book.
- **Two-legged "Bridge" Calling** – enabling 3rd party developers to initiate phone calls between two regular or everyday phones using the Ribbit service but not requiring the use of the Ribbit Flash phone. This enables developers to create 3rd party call control (3pcc) establishing a "two-legged call" where they determine the phone numbers for each leg of the call.
- **App distribution of multi-user apps (token authentication)** – building and deployment of applications for multiple users. An unlimited number of users are able to interact in guest mode with token-authenticated Ribbit applications.
- **Bring your own network (SIP Interoperability)** – Developers can apply to connect their network with Ribbit. SIP interoperability supports IP

connections over the open internet, VPN, and private point-to-point solutions.

- **Application Administration** – for published (Live) applications it is possible to track application's usage, manage billing and administer user accounts. Developers can track the communications activity on "their network." Ribbit provides usage reports on everything from how many calls have been made, to the number of minutes, text message activity and all other communications that application users have generated

Annex C. Example Semantic Service Description

For the prototype demo described in Section 2.1 semantic service descriptions will be produced for all the services. The example below show the MicroWSMO description for the Facebook getFriends service.

```
@prefix ex: <http://example.com/onto#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix wsl: <http://www.wsmo.org/ns/wsmo-lite#> .

# functional taxonomy of services
ex:ContainerService      rdfs:subClassOf      wsl:FunctionalClassificationRoot .
ex:GroupList             rdfs:subClassOf      ex:ContainerService .
ex:PhotoAlbumSite       rdfs:subClassOf      ex:ContainerService .
ex:SocialContactList    rdfs:subClassOf      ex:ContainerService .
ex:SocialEventListst    rdfs:subClassOf      ex:ContainerService .

# this is an NFP for free services
ex:free                  a                    wsl:NonFunctionalParameter .

# these are NFPs for indicating supported formats
ex:FormatSupport        a                    rdfs:Class .
ex:jsonSupport          a                    wsl:NonFunctionalParameter, ex:FormatSupport .
ex:xmlSupport           a                    wsl:NonFunctionalParameter, ex:FormatSupport .

# these are NFPs for indicating the authentication mechanism
ex:AuthenticationMechanism rdfs:subClassOf wsl:NonFunctionalParameter .
ex:httpBasic            a                    ex:AuthenticationMechanism .

# this comes from WSDL RDF mapping
ex:SafeInteraction      a                    rdf:Class .

# this is used for call id in facebook calls
# it's used in the ontology for inputs/outputs
ex:CurrentTime          a                    rdf:Class .
```

```
@prefix fb: <http://example.com/fb#> .
@prefix ex: <http://example.com/onto#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix wsl: <http://www.wsmo.org/ns/wsmo-lite#> .
@prefix foaf: <foaf-sorry-don't-remember> .

# ontology for data
fb:FacebookApiKey a rdfs:Class .
fb:FacebookUser rdfs:subClassOf foaf:Person .
# in OWL, this would be a subclass of some List class, constraining the members to be
# FacebookUser instances
fb:FacebookUserList a rdfs:Class .

# nonfunctional property specific to FB - signifies the FB authentication mechanism
fb:auth a wsl:NonFunctionalParameter, ex:AuthenticationMechanism
.
```