

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

SOA4All Studio UI and Infrastructure Services D2.4.2 First Demonstrator & Interface Specification - Prototype Documentation -

Activity N:	Activity 1 – Fundamental and Integration Activities	
Work Package:	WP2 – SOA4All Studio	
Due Date:	M18	
Submission Date:	31/08/2009	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	TIE	
Revision:	1.0	
Author(s):	Sven Abels Jean-Philippe Lombardi Juergen Vogel Tomas Pariente Lobo Guillermo Álvaro Rey Iván Martínez Marin Dimitrov Alex Simov	TIE SAP SAP ATOS ISOCO ISOCO ONTOTEXT ONTOTEXT
Reviewers:	Freddy Lecue Philippe Merle	UNIMAN INRIA

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	28.06.2009	Kick-Off Version	TIE
0.2	19.07.2009	Additions in 3.2.4.	iSOCO
0.3	31.07.2009	Small changes; integration	TIE
0.4	03.08.2009	Updated content for section 3.2	iSOCO
0.5	04.08.2009	Changes in section 3.4	TIE
0.6	09.08.2009	Updates for section 3.1 and 3.2	Ontotext
0.7	09.08.2009	Layouting	TIE
0.8	10.08.2009	Updated section 3.2	iSOCO
0.9	10.08.2009	Updated section 3.3.1	ATOS
1.0	11.08.2009	Integration of various comments	ALL
1.1	11.08.2009	Integration of various comments	SAP
1.2	27.08.2009	Large number of updates for integrating internal review comments	ALL

Table of Contents

EXECUTIVE SUMMARY	7
1. INTRODUCTION	8
1.1 PURPOSE AND SCOPE	8
1.2 STRUCTURE OF THE DOCUMENT	9
2. REMINDER: FUNDAMENTALS AND ROLE WITHIN THE PROJECT	10
2.1 ROLE WITHIN THE SOA4ALL STUDIO	10
2.2 SCOPE	10
2.3 HIGH-LEVEL STRUCTURE	10
2.4 INFRASTRUCTURE SERVICES	11
2.5 UI LIBRARY	12
3. PROTOTYPE STATUS: RESULTS ACHIEVED IN PROTOTYPE I	13
3.1 STORAGE SERVICES	13
3.1.1 <i>Repository Management</i>	13
3.1.2 <i>Managing RDF</i>	14
3.1.3 <i>Querying RDF</i>	15
3.1.4 <i>Managing Files</i>	15
3.1.5 <i>Retrieve Files</i>	15
3.1.6 <i>Summary of Achievements</i>	16
3.2 MANAGEMENT SERVICES	16
3.2.1 <i>Identification + Authentication</i>	16
3.2.2 <i>Simple Profile Management</i>	17
3.2.3 <i>Authorisation</i>	18
3.2.4 <i>Auditing</i>	23
3.2.5 <i>Preference Management</i>	23
3.2.6 <i>Social Graph Support</i>	23
3.2.7 <i>Summary of Achievements</i>	23
3.3 COMMUNICATION SERVICES	24
3.3.1 <i>Long Polling</i>	24
3.3.2 <i>Http streaming</i>	24
3.3.3 <i>Summary or achievements</i>	25
3.4 DESIGN TEMPLATES	25
3.4.1 <i>Summary of Achievements</i>	25
3.5 UI COMPONENTS	25
3.5.1 <i>Charting Widget</i>	26
3.5.2 <i>Form Generation Widget</i>	26
3.5.3 <i>Advanced List View Widget</i>	27
3.5.4 <i>Timeline Widget</i>	27
3.5.5 <i>Taxonomy Selector Widget</i>	27
3.5.6 <i>Graph Visualization Widget</i>	27
3.5.7 <i>Fault Handler Widget</i>	27
3.5.8 <i>Help System Widget</i>	27
3.5.9 <i>Gauges Widget</i>	27
3.5.10 <i>Rating Widget</i>	28
3.5.11 <i>Search & Result Handling Widget</i>	28
3.5.12 <i>Client Side Filtering Widget</i>	28
3.5.13 <i>Drawing API Widget</i>	28

3.5.14	<i>History Widget</i>	29
3.5.15	<i>Tag Cloud Widget</i>	29
3.5.16	<i>Progress Bar Widget</i>	29
3.5.17	<i>Summary of Achievements</i>	29
3.6	DASHBOARD	30
3.6.1	<i>Summary of Achievements</i>	31
4.	INSTALLATION & USAGE	32
4.1	REQUIREMENTS & PREPARATIONS	32
4.1.1	<i>For End-Users (“4All” of SOA4All)</i>	32
4.1.2	<i>For Administrators</i>	32
4.2	INSTALLATION (DEPLOYMENT)	32
4.3	EXECUTION	33
5.	CONCLUSIONS AND NEXT STEPS	34
6.	ANNEX A	35
7.	REFERENCES	36
8.	WEBLINKS	37

List of Figures

Figure 1: Task 2.4 Structure	11
Figure 2 User Profile Management Module	17
Figure 3 Resource Authorisation Model.....	18
Figure 4: Charting Widget - Current Implementation.....	26
Figure 5: Form Generator Widget.....	26
Figure 6: Rating Widget Implementation.....	28
Figure 7: Current Implementation of the Tag Cloud Widget	29
Figure 8: Current Dashboard implementation	31
Figure 9: Downloading the 2.4 prototype	33

Glossary of Acronyms

Acronym	Definition
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
COMET	Reverse Ajax
D	Deliverable
DSB	Distributed Service Bus
EC	European Commission
Ext GWT	Extended Google Web Toolkit (Framework)
GWT	Google Web Toolkit
HCI	Human Computer Interaction
REST	REpresentational State Transfer
RDF	Resource Description Framework
RIA	Rich Internet Application
SPARQL	SPARQL Protocol and RDF Query Language
UI	User Interface
URI	Uniform Resource Identifier
UX	User Experience
WAR	Java Web Archive
WSDL	Web Service Description Language
WP	Work Package

Executive Summary

This document complements *D2.4.2 SOA4All Studio First Prototype* of Task 2.4 *SOA4All Studio UI and Infrastructure Services* and describes the software implementation of the first official T2.4 prototype including the current status of the implementation. This document is included as part of the zip file that contains the first prototype software including all necessary files and instructions to install and run the prototype on Windows, Linux, MacOS or any other operating system that supports Java based software.

Please note: While D2.4.2 is the first official prototype, the T2.4 partners have already submitted a first “alpha” version in conjunction with D2.4.1. This new prototype D2.4.2 is based on the M12 2.4.1 prototype and has been strongly extended. The current prototype reflects a very good progress that is even more advanced than the original time planning given in D2.4.1. This prototype covers all features mentioned in the time planning of D2.4.1 and also realized two additional graphical widgets and one unplanned update to a new version of the underlying framework (GWT).

Please note that the API specifications in this deliverable only give a brief overview in order to keep the document short. For a technical specification please have a look at deliverable 2.4.1 and at the code via <https://trac.sti2.at/trac-soa4all/browser/trunk/soa4all-studio/soa4all-dashboard>. Moreover, the 2.4 team has created the Widget Explorer which lists all implemented widgets and provides an “Example” button that allows developers to view the code that is used to create the widget and to use the 2.4 components. The Widget Explorer can be found in the SOA4All Dashboard start menu at <http://coconut.tie.nl:8080/soa4all>

1. Introduction

1.1 Purpose and Scope

In order to increase the usability of SOA4All, it is of significant importance to ensure a low entrance barrier for potential users. One way of achieving this is the provision of one holistic user interface instead of providing separate user interfaces for each work package. The SOA4All consortium therefore decided to provide a common core user interface (UI) framework which forms the base for UI implementations and which will be implemented by development activities in all work packages. This ensures a common look and feel among all development activities and will allow easy navigation between SOA4All components.

This common framework is covered by Task 2.4 “SOA4All Studio UI and Infrastructure Services” and is documented in Deliverable 2.4.1. Task 2.4 has been started in Month 9 of the project after restructuring the work packages and subtasks. It covers those parts of the SOA4All Studio that have formerly been described in an extra deliverable, called “DX-UI: Holistic User Interface”.

The idea of Task 2.4 is to provide core elements that will help other work packages to provide their results in a holistic look & feel. This means that Task 2.4 provides a set of services that can be used by other work packages or tasks. Those services cover two essential parts:

- The first one is related to infrastructure services (e.g. common management mechanisms for objects such as storing preferences/settings)
- The second one addresses graphical elements such as defining a holistic design or providing graphical widgets (e.g. charting components).

This document is accompanied by the first prototype of the SOA4All Studio that has been compiled into a single deployment file (i.e., a WAR file) that can be found in the /bin directory and that can be deployed on an Apache Tomcat web application service. The WAR file therefore combining the T2.4 results and integrates results of various modules coming from other WP2 tasks. As such, prototype D2.4.2 is called the “SOA4All Studio First Prototype”.

A temporary version is available at:

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

This document complements the *D2.4.2 SOA4All Studio First Prototype* of the Task 2.4 *SOA4All Studio UI and Infrastructure Services* and describes the software implementation of the first official D2.4 prototype. This document is included as part of the zip file that contains the first prototype software including all necessary files and instructions to install and run the prototype on Windows, Linux, MacOS or any other operating system that supports Java-based software.

1.2 Structure of the Document

The remainder of the document is structured as follows:

- Section 2 gives a short recapitulation of Task 2.4 including a description of its role in the project.
- Section 3 gives a brief description about what has been implemented in the last 6 months by the 2.4 team. It presents a status table for each sub-task showing the current status and showing which parts have been implemented in a first version already.
- Section 4 describes the usage of the prototype including the installation and startup instructions.

Please note that this document is kept as short as possible on purpose. It's only purpose is to guide readers though the installation of the prototype and to give him an update of the status and the different components. For any functional descriptions and technical descriptions, please refer to the D2.4.1 deliverable

2. Reminder: Fundamentals and Role within the project

2.1 Role within the SOA4All Studio

The SOA4All Studio acts as an important building block when bringing SOA4All results to users. As such it provides a bridge from the technological parts to the world of the user interface. For example, in the SOA4All Process Composer (Task 2.6), the SOA4All Studio builds the base for creating a user interface with the help of the high-level and infrastructure services.

The SOA4All Studio consists of several subtasks covering different aspects of SOA4All such as provisioning, consumption, process modelling, and analysis functionalities. These components are developed within the Tasks 2.1, 2.2, 2.3 and 2.6. They altogether form the so-called SOA4All Studio that is the main entry point for SOA4All users.

The components that are provided by the SOA4All Studio are brought together by a holistic graphical user interface and Task 2.4 provides the base for this graphical user interface. It provides a set of services that are offered to the other WP 2 tasks but that might also be used by other SOA4All work packages or even by third party components that are integrated into SOA4All.

2.2 Scope

Task 2.4 does not implement core activities of the concrete Tasks 2.x and does not force tasks to use specific services. Instead, it provides services to ease the development and the integration of tasks that have a need for user-interactions.

To make this more concrete, let us focus on the SOA4All Process Composer Task 2.6. The Process Composer allows process modeling for non-technical users and provides a set of functionalities for this. The 2.4 components do not replace this work but they make the development process easier. The Process Composer task will still need to create a graphical Process Composer but Task 2.4 will provide a UI and an infrastructure framework that makes it easier to e.g. create web 2.0 user interfaces, to store and manage data or to use communication functionalities.

For example, 2.4 will provide the basic functionalities to realize drag & drop, while 2.6 will use those functionalities to actually implement a Process Composer UI allowing people drag & drop process elements. In addition to this, a Dashboard view seamlessly integrates the Process Composer UI into the overall SOA4All Studio environment.

Similar to this, task 2.4 also provides services for other implementations such as the WSMO Lite and MicroWSMO editor and the consumption platform.

2.3 High-Level Structure

The SOA4All Studio provides two different levels of services:

- **Infrastructure Services**
On the one hand Infrastructure Services will provide support for Tasks 2.x and potentially for all other web applications that could be based on SOA4All such as the WP9 Facebook application which will access processes that have been defined in SOA4All.
- **UI Library**
On the other hand UI Library provides services that are located on top of the platforms. They provide UI-related elements, a Dashboard view, etc. While the use of those

Services will not be strictly necessary in order to interact with SOA4All (as nothing will prevent third parties to develop their interfaces on top of the different platforms APIs) it will be the easiest way to implement the connection between users and the SOA4All runtime.

As shown in Figure 1, Task 2.4 consists of a set of components, which are classified to belong to either the UI Services Part or the Infrastructure Services Part.

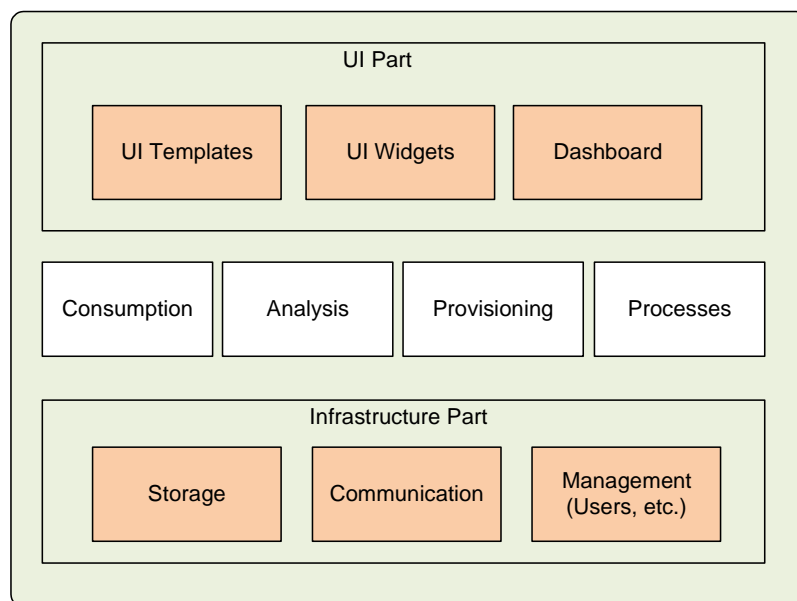


Figure 1: Task 2.4 Structure

2.4 Infrastructure Services

The SOA4All Studio provides basic functionalities that may be used to rapidly create applications. Those services may be seen as a “basic SOA4All API”. For example, the storage services may be compared to Amazon S3 services¹ or to Google² Base allowing applications to store and retrieve data in a data store. Each of the services may be accessed independently with a Web service interface.

The following infrastructure services are provided by 2.4:

- **Storage Services**
Storing and retrieving information similar to Google Base / Amazon S3 but focused on the SOA domain and service requirements (e.g. for storing preferences or user profiles). The results of WP1 will be used for realizing this while 2.4 will provide API access to it.
- **Management Services**
Managing Users, Roles and Access Rights; Providing Logging and Auditing functionalities

¹ <http://aws.amazon.com/s3/>

² <http://www.google.com>

- **Communication Services**
Basic messaging services for allowing components to exchange information based on the COMET pattern³. COMET describes an approach in which a long-held HTTP request allows a web server to push data to a browser instead of having the web browser to ask the server. This allows SOA4All components to implement event-driven communication patterns.

2.5 UI Library

While the infrastructure services can be seen to be located “below” the specific T2.x outputs, the UI Library provides functionalities that will be based on top of them. They can be split into different parts:

- **UI Design Templates**
A UI template and designs as well as examples are provided to be used as recommendations for WPs. WPs are highly encouraged to follow them in order to achieve a common look & feel in the project.
- **UI Components**
This subtask provides various basic services related to allowing the creation of a holistic user interface. Because of their nature, these services are referred to as “UI Widgets”. They consist of a set of useful API methods build on top of the GWT⁴ or EXT-GWT⁵ framework for allowing a rapid development of Web 2.0-like UIs.
- **Dashboard View**
The Dashboard provides an entry point to SOA4All components.
- The project will also integrate the Recommendation component developed in T2.7 in the SOA4All Studio so that users will receive recommendations while interacting with the GUI.

³ see [COMET1]

⁴ <http://code.google.com/intl/de-DE/webtoolkit/>

⁵ <http://extjs.com/products/gxt/>

3. Prototype Status: Results Achieved in Prototype I

This section describes the prototype status as of month 18 of the project. It gives a short overview about what has been implemented in the first official prototype. This summary is broken down based on the different components.

3.1 Storage Services

The storage services provide an easy to use API that provides simple data storage facilities consisting of fundamental Create, Update, Delete and Query functionalities. The implementation of the storage service is realized as a RESTful⁶ service, which facilitates functionality realization and reduces integration efforts significantly. The following sections describe in more details the APIs for managing repositories, files and RDF⁷ data.

Usage examples of all functionalities of the storage services API are available as simple Java applications in the project's SVN⁸ repository.

3.1.1 Repository Management

One crucial aspect of storing large amounts of heterogeneous data is the organization of the artefacts in such a way that any subsequent operations like searching, browsing, and management is done with minimal effort. To meet such requirements, the Storage services introduce the notion of *repository*, which stands for a placeholder for storing files and RDF data, identified by unique repository identifier. The (RESTful) API for managing repositories is described below:

Resource/Operation	Method	Description
/repositories	GET	Lists all available repositories params: none result: XML (list of repository IDs – see Annex A)
/repositories/<repository-id>	PUT	Creates a new repository params: none result: none
/repositories/<repository-id>	DELETE	Deletes an existing repository params: none result: none

⁶ http://en.wikipedia.org/wiki/Representational_State_Transfer

⁷ <http://www.w3.org/RDF/>

⁸ <https://trac.sti2.at/trac-soa4all/browser/trunk/soa4all-studio/soa4all-infrastructure/soa4all-studio-storage/src/main/java/eu/soa4all/studio/infrastructure/storage/example/StorageServiceExample.java>

3.1.2 Managing RDF

The storage services allow managing (adding, updating, deleting) RDF information to a selected repository. These services are in line with the typical functionality that RDF repositories usually allow. The table below reveals the basic management operations, which allow working with RDF data in a simple and unified way.

Resource/Operation	Method	Description
/repositories/<repository-id>/statements	GET	Lists (all) statements from a repository params: 'subj' (optional) - subject restriction 'pred' (optional) - predicate restriction 'obj' (optional) - object restriction 'context' (optional) - named graph URI result: RDF
/repositories/<repository-id>/statements	PUT	Updates statements in a repository (removing previous content and adding new statements) param: 'context' (optional) - named graph URI header param: 'rdf-data' - RDF triples data result: none
/repositories/<repository-id>/statements	POST	Adds statements to a repository param: 'context' (optional) - named graph URI header param: 'rdf-data' - RDF triples data result: none
/repositories/<repository-id>/statements	DELETE	Delete statements from a repository params: 'subj' (optional) - subject restriction 'pred' (optional) - predicate restriction 'obj' (optional) - object restriction 'context' (optional) - named graph URI result: none

3.1.3 Querying RDF

One very important feature for each system offering RDF storage is the ability to perform querying on the data. Such operations must be uniform and efficient, regardless of the underlying implementation. Therefore, the Storage services support the SPARQL⁹ query interface, which allows query evaluation on the data from certain repository.

Resource/Operation	Method	Description
/repositories/<repository-id>	GET	SPARQL query evaluation over a repository params: 'q' (or alternatively header param 'sparql-q') - SPARQL query expression result: RDF

3.1.4 Managing Files

Apart from managing and querying RDF data, the storage services support storing, updating or deleting files. This functionality is needed in order to manage data either that cannot be serialised as RDF or where the serialisation may be inefficient (e.g. managing WSDL¹⁰ files).

Resource/Operation	Method	Description
/repositories/<repository-id>/files	GET	Lists all files in a repository params: none result: XML (list of file names – see Annex A)
/repositories/<repository-id>/files/<file-name>	PUT	Creates/Updates file in a repository param: none result: none
/repositories/<repository-id>/files/<file-name>	DELETE	Deletes file from a repository params: none result: none

3.1.5 Retrieve Files

The storage services also allow retrieving files from a certain repository. This operation is done in a simple and implementation transparent manner.

⁹ www.w3.org/TR/rdf-sparql-query

¹⁰ <http://www.w3.org/TR/wsdl>

Resource/Operation	Method	Description
/repositories/<repository-id>/files/<file-name>	GET	Retrieves a file from a repository params: none result: file content

3.1.6 Summary of Achievements

The following table summarizes the tasks that have been identified in D2.4.1 and their status as of month 18:

ID	Name	Priority (1=high, 10=low)	Status
ST-1	Managing RDF	1	Version 1 implemented
ST-2	Querying RDF	1	Version 1 implemented
ST-3	Managing Files	2	Version 1 implemented
ST-4	Receiving Files	2	Version 1 implemented

3.2 Management Services

Management Services provide means for user identification and authentication, resource authorisation, auditing/logging and some simple key/value based user preference storage.

3.2.1 Identification + Authentication

Identification makes it possible for different users to be identified according to their own profiles/accounts. *Anonymous access* is also possible in such collaborative systems (although not mandatory). Password based authentication of the identified user accounts is required. Ownership of resources (such as services, ontologies, tag clouds, comments and ratings, etc.) is associated with a specific user or group profile. Thus, it is required that user profile management is properly implemented in the system.

On the other hand, it has been included in the SOA4All Studio OpenID¹¹ as authentication mechanism. OpenID is an open, decentralized standard for user authentication and access control, allowing users to log on to different services with the same digital identity. OpenID replaces the common login process that uses a login-name and a password, by allowing a user to log in once and gain access to the resources of multiple software systems.

¹¹ <http://openid.net/>

An OpenID is in the form of a unique URL, and is authenticated by the user's "OpenID provider" (that is, the entity hosting their OpenID URL). The OpenID protocol does not rely on a central authority to authenticate a user's identity. Since neither the OpenID protocol nor Web sites requiring identification may mandate a specific type of authentication, non-standard forms of authentication can be used, such as smart cards, biometrics, or ordinary passwords.

The logon flow basically works like this: A user lands on the OpenID-enabled logon SOA4All Studio entry point. The user enters a Uniform Resource Identifier (URI) that points to a Web page that contains information about the OpenID provider that handles the user's credentials. The SOA4All Studio then requests authentication from that provider. In our case, the authentication process works by redirecting the user to the provider's logon page to do the logon. Upon authentication, the Studio can proceed as necessary based on the result. For example, if authentication was successful, it can grant access to the user. If authentication failed, it can prevent access and proceed accordingly.

3.2.2 Simple Profile Management

The *User Profile Management* (see Figure 2) module is part of the SOA4All Studio¹². It is responsible for managing essential information associated with each user, such as *user id*, *name*, *contact e-mail*, *OpenID account*, etc. Each user can provide such information after registering successfully into the system. Subsequently the user can modify/erase the profile information after log-in in the system using his/her OpenID account.

The profile component stores the user data in RDF format using the *Storage Services* described above, thus making it accessible for other interested components of the studio.

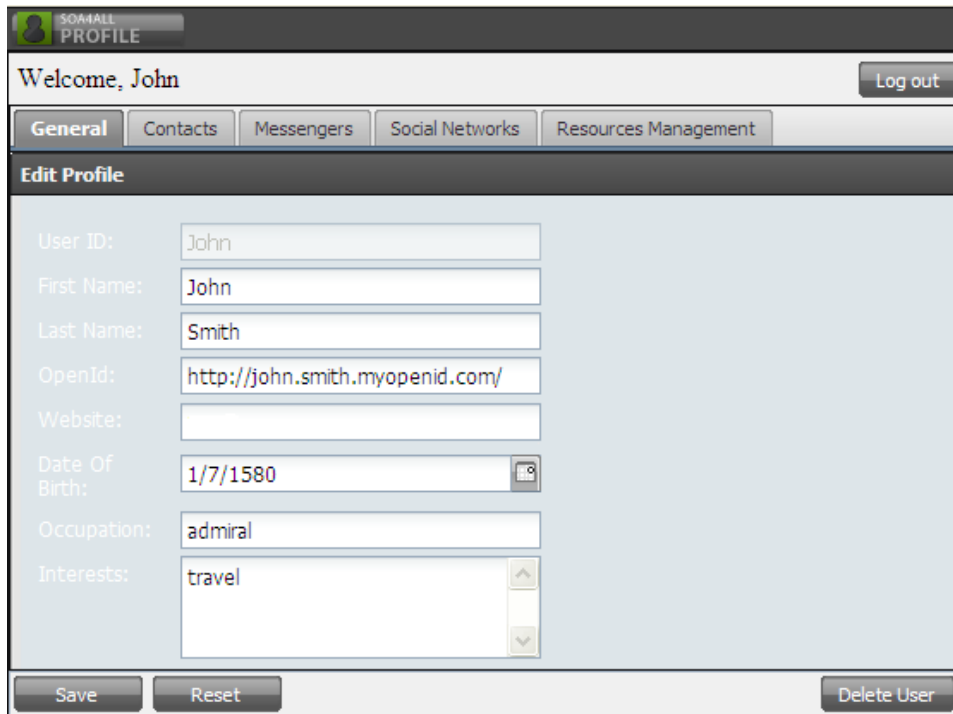


Figure 2 User Profile Management Module

¹² Snapshot version available and regularly updated at: <http://coconut.tie.nl:8080/soa4all>

3.2.3 Authorisation

Authorisation provides a scheme for defining the permission/operations applicable to a particular resource per an identified user. Since the number of resources and users in the system is potentially unlimited, *Access Control List*¹³ (ACL) based schemes may be inefficient and a *Role-Based Access Control*¹⁴ (RBAC) one may be more suitable. That is the reason we adopt a hybrid approach combining user roles with access lists.

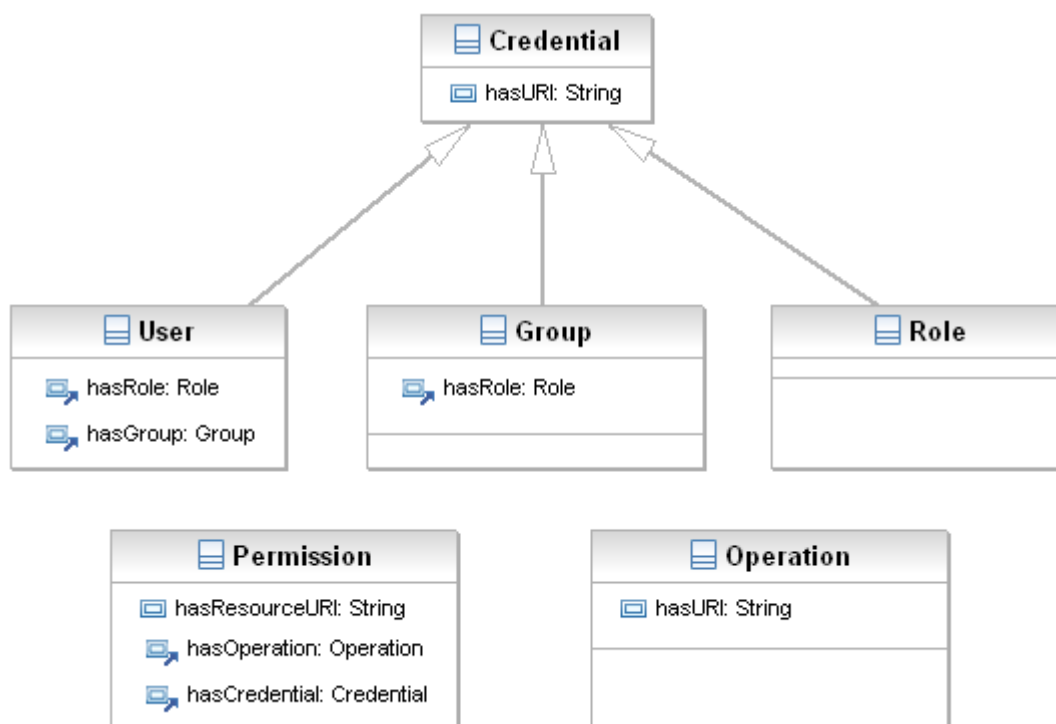


Figure 3 Resource Authorisation Model

The key is to bring together *Groups*, *Roles* and *Users* in a single concept that is *Credential*, permissions over resources are based on credentials, every user has a set of credentials, by default its user himself is a *Credential*, a user can be associated to *Groups* and *Roles*, a *Group* may have a set of *Roles*.

On the other hand, we have the resources, which can be protected by permissions, which can be expressed as associations between resources, credentials and operations. Since a credential can be a group, role and user, we can set permissions for a resource at three levels.

The implementation of the model is realized (and accessible) by the following three RESTful services, supporting different aspects of the authorisation functionalities.

¹³ An ACL specifies which users are allowed to access a particular resource and which operations are allowed to be performed.

¹⁴ Contrary to the ACL approach, an RBAC specifies permissions not for resources but for specific operations within the system

Please note that the following list gives a brief overview in order to keep the document short. For a technical specification please have a look at the widget examples and at the code at <https://trac.sti2.at/trac-soa4all/browser/trunk/soa4all-studio/soa4all-dashboard>

3.2.3.1 Resource Authorisation Service

Resource/Operation	Method	Description
/authorize	GET	Check resource authorization for user params: 'user' - user ID 'resource' - resource URI 'op' - operation URI result: "true"/"false"

3.2.3.2 Resource Permissions Management Service

Resource/Operation	Method	Description
/resources	GET	Lists all resources params: none result: list of resource URIs
/resources	PUT	Creates a resource param: 'user' - owner ID 'res' - resource URI result: none
/resources	DELETE	Deletes a resource param: 'user' - owner ID 'res' - resource URI result: none
/resources/owner	PUT	Changes resource owner params: 'owner' - owner ID 'newOwner' - new owner ID 'res' - resource URI

/resources/perm	GET	<p>Lists resource access permissions</p> <p>params:</p> <p>'res' - resource URI</p> <p>'cred' - credential URI (user, group or role)</p> <p>result: list of operation URIs permissible for the given credential with respect to the resource</p>
/resources/perm	PUT	<p>Adds a new resource access permission</p> <p>params:</p> <p>'user' - user ID, adding the new permission (must be the owner of the resource)</p> <p>'res' - resource URI</p> <p>'cred' - credential URI (user, group or role) given the permission</p> <p>'op' - operation URI</p> <p>result: none</p>
/resources/perm	DELETE	<p>Deletes a resource access permission</p> <p>params:</p> <p>'user' - user ID, deleting the permission (must be the owner of the resource)</p> <p>'res' - resource URI</p> <p>'cred' - credential URI (user, group or role) given the permission</p> <p>'op' - operation URI</p> <p>result: none</p>
/repositories/creds	GET	<p>Lists all credentials which at least one permission is granted</p> <p>param: 'res' - resource URI</p> <p>result: list of credential URIs which are granted at least one permission</p>

3.2.3.3 User Management Service

Resource/Operation	Method	Description
/users/groups	GET	<p>Lists all user groups</p> <p>params: none</p> <p>result: list of URIs of all user groups defined</p>

/users/groups	PUT	Creates a new user group param: 'id' - group URI result: none
/users/groups	DELETE	Deletes a user group param: 'id' - group URI result: none
/users/roles	GET	Lists all roles params: none result: list of URIs of all roles defined
/users/roles	PUT	Creates a new role param: 'id' - role URI result: none
/users/roles	DELETE	Deletes a role param: 'id' - role URI result: none
/users/assignGroup	PUT	Assigns a user to a group params: 'user' - user ID 'group' - group URI result: none
/users/assignGroup	DELETE	Excludes a user from a group params: 'user' - user ID 'group' - group URI result: none
/users/assignGroup	GET	Lists all group assignments for a user param: 'user' – user ID result: list of user group URIs
/users/assignUserRole	PUT	Assigns a role to a user params: 'user' - user ID 'role' - role URI

		result: none
/users/assignUserRole	DELETE	Divests a user of a role params: 'user' - user ID 'role' – role URI result: none
/users/assignUserRole	GET	Lists all roles of a user param: 'user' – user ID result: list of role URIs
/users/assignGroupRole	PUT	Assigns a role to a user group params: 'group' - user URI 'role' - role URI result: none
/users/assignGroupRole	DELETE	Divests a user group of a role params: 'group' - user group URI 'role' - role URI result: none
/users/assignGroupRole	GET	Lists all roles of a user group param: 'group' – user group ID result: list of role URIs
/operations	GET	Lists all operations over resources params: none result: list of operation URIs
/operations	PUT	Creates an operation param: 'id' - operation URI result: none
/operations	DELETE	Deletes an operation param: 'id' - operation URI result: none

3.2.4 Auditing

The Auditing Service has already been created as a server-side module (`soa4all-dashboard-auditing-service`) inside the SOA4All Studio Dashboard. It permits client-side modules within the SOA4All Studio to log some interesting actions that the users perform when interacting with the tool. The logged actions are exploited by the Analysis platform and the Recommendation System, which are able to infer conclusions about users' interactions within the platform thanks to the stored logs.

The Auditing Service stores the actions performed by the users, following the RDF schema defined in the Recommendation System deliverable (see D2.7.1), into the Semantic Spaces using the Storage Services described above.

There is only one relevant action for this service, which is `logAction`. It receives the action identifier, a persistent session Id that is used to correlate actions performed by a user, even if a user is not logged into the platform, and extra parameters, depending on the type of action. For example, if the action being stored is "Log in", the user identifier has to be passed in the invocation:

```
public String logAction(String actionId, String persistentSessionID,
    Map<String,String> params);
```

Client-side modules within the platform are able to log users' actions by invoking this method, as the Consumption Platform is already doing.

3.2.5 Preference Management

Preferences management allows a simple storage and retrieval of user and/or application based settings (key-value pairs). It is desirable that the system provides sufficient means for customisation according to the specific end user preferences, so that the end user experience is maximised.

3.2.6 Social Graph Support

The Social Graph related functionality will make it possible for users to find and add other SOA4All users to their social graph by means of either finding users based on name / e-mail, or import contacts from existing social networks such as Facebook, MySpace, LinkedIn, OpenSocial, etc.

Harnessing the power of the social graph provides great advantages to Web 2.0 applications when it comes to recommendations, ratings and collaborative authoring. SOA4All will not aim at creating and maintaining yet another social network but will instead leverage the social graphs related to its users in various existing social networks.

3.2.7 Summary of Achievements

The following table summarizes the tasks that have been identified in D2.4.1 and their status as of month 18:

ID	Name	Priority (1=high, 10=low)	Status
MS-1	OpenID authentication	1	Version 1 implemented
MS-2	Multiple OpenID accounts per user	4	Not started yet
MS-3	Resource authorisation	1	Version 1 implemented
MS-4	Social graph import	6	Not started yet
MS-5	Auditing/logging	3	Version 1 implemented
MS-6	User preferences	2	
MS-7	Anonymous access	5	Not started yet
MS-8	Simple profile management	1	Version 1 implemented
MS-9	Social graph support	2	Not started yet

3.3 Communication Services

Communication services are basic messaging services aiming to give extra functionality for the communication the server and the client side.

3.3.1 Long Polling

Long polling would enable WP2 GUI components to maintain a connection open with the server and receive data when the server pushes it, taking into account that these updates should not happen very often.

A high-level API providing the long polling (or server push) technique is a task that was initially planned to be implemented by WP2 T.4 because no other library could offer the same functionality by that time. However, as of M18, the only component that makes use of the long polling approach is the monitoring service. That service uses the library GWTEventService, which leverages the GWT RPC mechanisms to deliver the COMET functionality. Unless other components need a further abstraction from the functionality provided by the GWTEventService library, there is no real need to reinvent the wheel.

3.3.2 Http streaming

HTTP Streaming is similar to the long polling technique except the connection is never closed, even after the server push data. With this technique, the AJAX client will only send a single request and receive partial responses as they come, re-using the same connection forever.

Web applications with relatively infrequent updates can use long polling without significant overhead (opening a new connection with the server would be not as costly as maintaining it all the time). On the other hand, high polling frequencies can waste server resources and bandwidth. Use then Http streaming when your AJAX application requires frequent updates.

This technique significantly reduces the network latency as the browsers and server do not need to open/close the connection (i.e. Google Mail¹⁵ is using that technique to update the mail interface in a real time fashion).

In any case, the implementation of this technique is not mandatory, because long polling may be enough for WP2.

3.3.3 Summary or achievements

The following table summarizes the tasks that have been identified in D2.4.1 and their status as of month 18: .

ID	Name	Priority (1=high, 10=low)	Status
CS-1	Long polling	4	No need to implement
CS-2	Http streaming	8	Not implemented yet

3.4 Design Templates

Following the design principles and design studies presented in D2.4.1, the overall design of the SOA4All Studio and its embedded Tools has been implemented using the possibilities of GWT and Ext GWT. For this purpose, the general layout, colours, widgets, and icons have been designed to create a close match to the mock-ups presented before. The resulting look and feel of the Studio is platform-independent.

3.4.1 Summary of Achievements

The following table summarizes the tasks that have been identified in D2.4.1 and their status as of month 18:

ID	Name	Priority (1=high, 10=low)	Status
DTC-1	Rich Internet Application Metaphor	1	Implemented
DTC-2	Rich User Experience (UX)	1	Implemented
DTC-3	Universal Design and High Usability	1	Implemented

3.5 UI Components

Reusable UI components simplify the development of consistent and ergonomic user interfaces. In D2.4.1, a number of UI components had been identified and specified, which can all be leveraged by several Studio tools and therefore are provided by Task 2.4 in order to avoid duplicate work and to achieve a common look and feel of the entire Studio.

¹⁵ <http://mail.google.com>

3.5.1 Charting Widget

The UI Framework provides a charting widget that allows other work packages to easily create and display charts. Types that are supported include bar charts, Pie charts and line charts. This functionality is required for Task 2.3 to display results of Web services that have been monitored and analyzed.

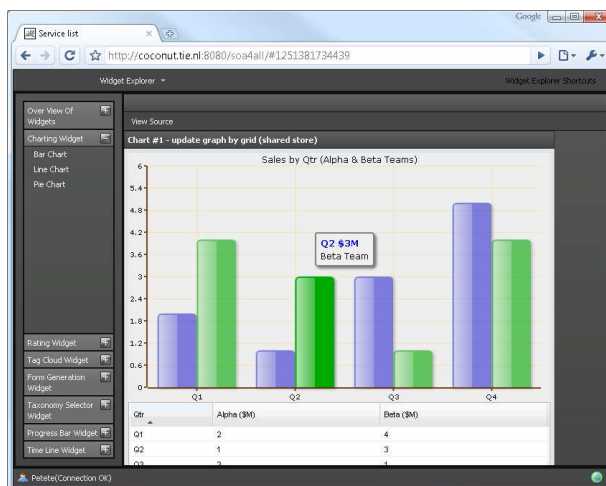


Figure 4: Charting Widget - Current Implementation

3.5.2 Form Generation Widget

This widget can be used to either edit an object or to create a new object for a specific class. It is based on a data structure that allows the developer to easily define the content of the form. The form may be used to e.g. request user information.

The widget is based on a generic data structure. As an example, it provides a way to use it to create a form for a given WSDL specification. Results are passed as a return value from the widget after the user has clicked on a submit button. The widget is required for tasks 1.2 / 1.3 / 1.4.

User Form

User Name:

Password:

Email Date:

Address:

User Selected:

Quarter:

Gender:

Figure 5: Form Generator Widget

3.5.3 Advanced List View Widget

This widget displays a set of data elements. It is connectable with the Form Generation Widget in order to allow a detailed view of an element. In contrast to the standard list views of Ext GWT, this widget is based on introspection and annotation to automatically create a layout. - The widget is required for tasks 1.2 / 1.3 / 1.4.

3.5.4 Timeline Widget

This widget visualizes events in a timeline view. This will allow SOA4All to assign events to a specific point of time and allow the user to see the events in a graphic view. Events are clickable in order to receive more information about them. The widget is required for tasks 2.3 and 2.6 in order to be connected with the SOA4All process editor.

3.5.5 Taxonomy Selector Widget

The aim of this widget is to provide a flexible way of presenting taxonomies to the user in a tree-like structure. Taxonomies can be large and will therefore be loaded 'on demand' when opening a sub-tree using the Ajax approach. Each taxonomy entry can be annotated with some HTML formatted explanation (e.g. description), which is displayed to the users. Users are able to select a specific entry of the taxonomy. The result can be used for creating a concept selector for RDF information. It is required by tasks 2.1 / 2.2 / 2.3 and could also be used as a base for a WSMO selector.

3.5.6 Graph Visualization Widget

For discovery of services a graph can be shown to the user (nodes = classes/categories and edges = relations, in the first version may be only subsumption). In order to do this, a widget was required that allows the display of graphs, namely nodes and edges. With this widget, the user is able to select specific nodes. It is therefore required to send an event after the user has selected an element. - The widget is required for 5.3 to display search graphs as described above.

3.5.7 Fault Handler Widget

This widget displays an error window and will allow the users to report this error to the SOA4All team. Users may specify their mail address in order to receive feedback by the SOA4All developers.

This is a general functionality that will help to improve the usability of SOA4All by providing error reporting as described in DX-UI. This comfortable way of reporting errors will also allow developers to catch errors in a very convenient and fast way and will therefore lead to a better stability of the SOA4All studio components, too.

3.5.8 Help System Widget

This widget displays help topics. It renders a help button next to another widget and allows users to click on this button. When being clicked, a window opens that will display help information to the user.

3.5.9 Gauges Widget

This widget creates gauges and displays them to the user. In addition to a tacho-like view, a traffic light view will be implemented as well in a future version, allowing to show a red,

yellow or green status. This function will be of use for displaying the health status in the Service Monitoring task and for the 2.4 Dashboard view.

3.5.10 Rating Widget

This widget allows people to rate an element and to view a specific rating.

Please note that this widget does not need to care about the logic (e.g. user management, duplicate rankings, etc.) as it only cares about rendering the ranking UI (e.g. displaying a 4/5 star rating). - This is a key function for enabling the Web 2.0 functionalities and will be used in several tasks such as e.g. 5.4.

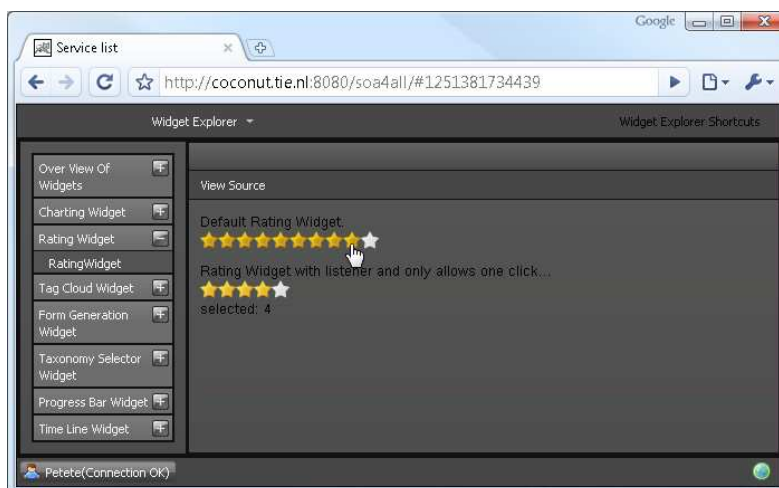


Figure 6: Rating Widget Implementation

3.5.11 Search & Result Handling Widget

This widget provides a dynamic search component. This might be used to search in a list of elements based on specific criteria. For example, it might be used to gather search input from users when they are searching for Web services. The widget also allows the display of the result list. It reuses the outcomes of the *Advanced List View Widget* for this. This is a required function for WP9 in order to display C2C eCommerce Services to the user.

3.5.12 Client Side Filtering Widget

This widget shows a client-side filter that can be applied by the user after the server-side search component has given back its results (since there might be a lot of results). Basically this is a combination of local sorting and filtering but the semantics for the actual search remain with the search server. This widget will help to improve the usability of the WP7 outcomes.

3.5.13 Drawing API Widget

The drawing API builds functionality into GWT/GXT that allows to create vector graphics -- lines, curves, shapes, fills, and gradients -- and to display them on the screen using Java (GXT/GWT)

Common drawing API tasks are

- Defining line styles and fill styles for drawing shapes
- Drawing straight lines and curves
- Using methods for drawing shapes such as circles, ellipses, and rectangles

- Using trigonometry with the drawing API
- This needs to be interactive (user draws predefined shapes)

This functionality is strongly required for realizing the Process Composer and it will also be the base for the Graph Visualization Widget.

3.5.14 History Widget

In some cases, SOA4All WPs will provide users to do multiple undo's. In this case a History Widget can be used. It allows users to control those steps. The History Widget is also able to control the Back-Button functionality of the web browser. However, the logical undo-functionality will still remain in control of the component that uses this widget. - The history widget is relevant for the SOA4All Process Composer during editing.

3.5.15 Tag Cloud Widget

This widget provides a tag cloud view to allow people to quickly jump to specific links. This functionality will form the base for many Web 2.0 elements and is therefore an essential part of the UI components.



Figure 7: Current Implementation of the Tag Cloud Widget

3.5.16 Progress Bar Widget

This widget creates a simple progress bar view showing a percentage between 0% and 100%. This function will be of use for displaying the health status in the Service Monitoring task.

3.5.17 Summary of Achievements

The following table summarizes the tasks that have been identified in D2.4.1 and their status as of month 18:

ID	Name	Priority (1=high, 10=low)	Status
UIC-1	Charting	1	Implemented
UIC-2	Form Generation Widget	1	Implemented in version 1
UIC-3	AdvListView	2	Not started
UIC-4	TimeLine	3	Implemented in version 1
UIC-5	Taxonomy Selector Widget	1	Implemented in version 1
UIC-6	GraphVisualization	2	Implemented in version 1
UIC-7	FaultHandler	1	Not started
UIC-8	HelpSystem	5	Not started
UIC-9	Gauges	6	Not started
UIC-10	Rating	3	Implemented in version 1
UIC-11	Search & Result	3	Not started
UIC-12	Client Side Filtering	6	Not started
UIC-13	Drawing API	1	Implemented in version 1
UIC-14	History Widget	5	Not started
UIC-15	TagClouds	1	Implemented in version 1
UIC-16	Progress Bar	1	Implemented in version 1

3.6 Dashboard

The Dashboard provides a starting point for SOA4All users. It shows all components and elements of SOA4All and it provides a menu structure to access them.

For each component, the Dashboard displays a component specific text and a graphic to guide the user. For example, the partners participating in task 2.3 (Analysis) are able to display an overview chart with statistics on the Studio Dashboard entry page. Users may click on this chart to directly jump to the 2.3 results.

An entry page is the main access and starting point for SOA4All users. It avoids that users get lost in information overload and it will allow SOA4All to present itself as a coherent set of solutions. Users will see the entry page as some kind of portal that will allow them to quickly jump to the different areas of SOA4All.

The following screenshot shows the current version of the Dashboard implementation:

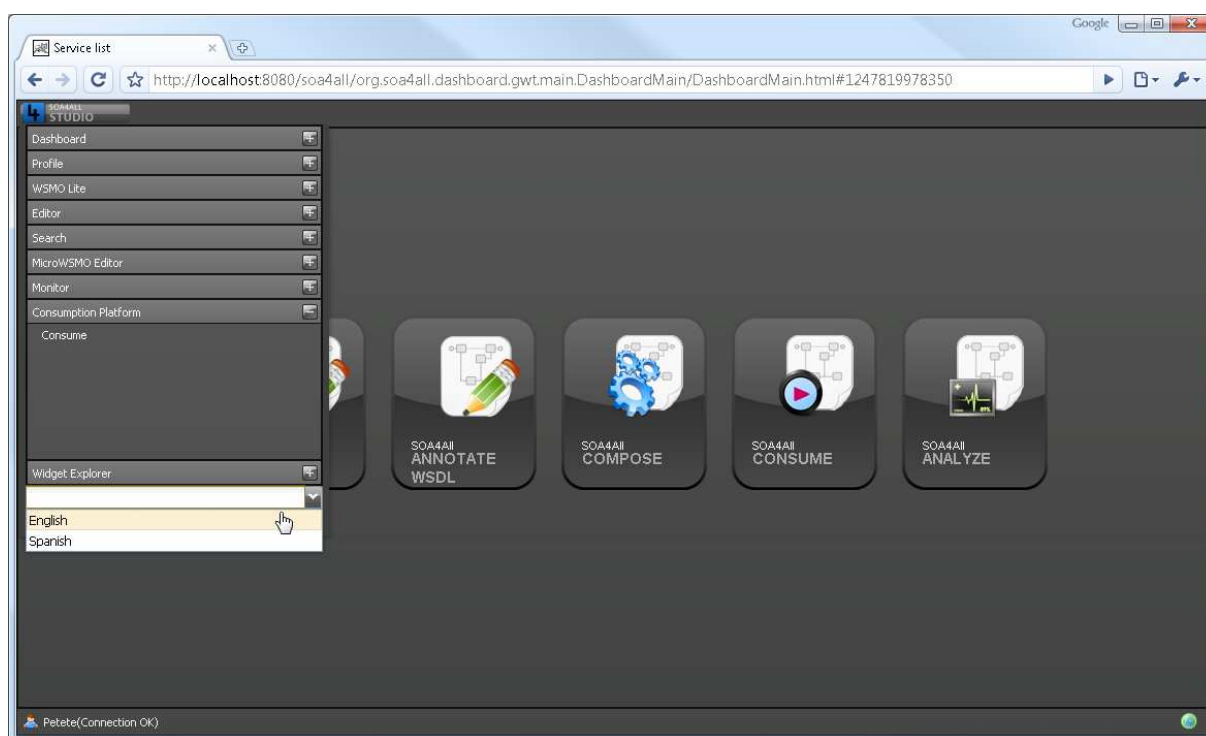


Figure 8: Current Dashboard implementation

3.6.1 Summary of Achievements

The following table summarizes the tasks that have been identified in D2.4.1 and their status as of month 18:

ID	Name	Priority (1=high, 10=low)	Status
DB-1	Entry Point	1	Implemented in verison 1
DB -2	Dynamic Coupling	1	Implemented in verison 1
DB -3	QuickStart	8	Implemented in verison 1
DB -4	Personalization	6	Not started
DB -5	Core Elements Integration	1	Implemented in verison 1

4. Installation & Usage

Within this section, a short overview is given on how to install and use the current 2.4 prototype.

4.1 Requirements & Preparations

4.1.1 For End-Users (“4All” of SOA4All)

Users do not need to install anything to use SOA4All. The only thing that they need is a modern Web Browser. Currently, the prototype implementation supports the current versions of Firefox, Chrome and the Internet Explorer. Based on this, they may simply invoke the SOA4All application by calling the web address. It is not necessary to install any plugins.

The D2.4.2 results are temporary available for testing purposes at:

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

However, they will be moved to the official SOA4All website in the process of the development.

4.1.2 For Administrators

4.1.2.1 Java

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

The 2.4.2 prototype requires Java **1.6** or newer. The latest version may be downloaded at <http://java.sun.com>

4.1.2.2 Tomcat

As SOA4All is a web based solution, the D2.4.2 prototype is available as a web application. As such, the Tomcat server (6.0 or newer) installation is required in order to setup the prototype. Tomcat is available at the following website:

<http://tomcat.apache.org>

4.2 Installation (Deployment)

Installation of the 2.4.2 prototype is very easy. Please ensure that Java 1.6 and Tomcat 6.0 have been downloaded and installed on your system.

Afterwards, copy the `soa4all-dashboard.war` file into the folder `webapps` of your Tomcat installation. You can download the latest file directly from the SOA4All build system which is described in deliverable D1.5.1. Once you have done this, please start Tomcat. This will automatically install all SOA4All files for you.

You can download the prototype WAR file from the SOA4All build server via

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

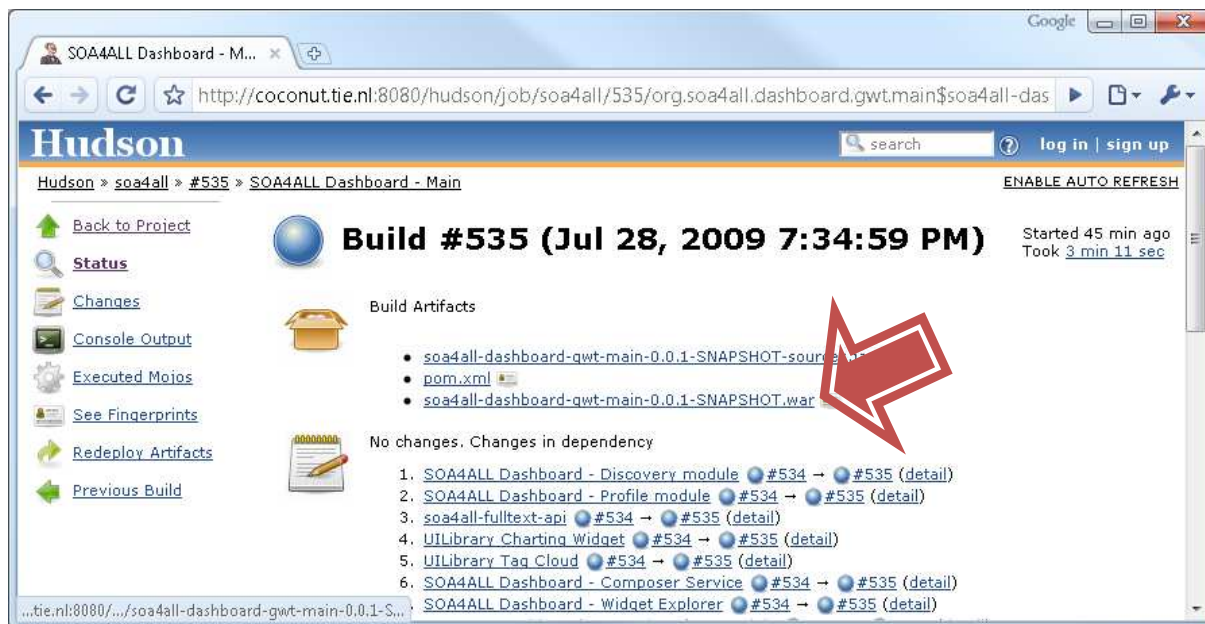


Figure 9: Downloading the 2.4 prototype

4.3 Execution

After installing the SOA4All 2.4.2 prototype, open a web browser and navigate to the following URL:

<http://localhost:8080/soa4all-dashboard>

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

Please note that this WAR file only contains the D2.4.2 results which means that some services that are coming from other work packages will not be working as they rely on other servers to be installed on your system.


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

5. Conclusions and Next Steps

This document has described the outcome and the developments of task 2.4. It represents the first version of the prototype installation. From the current viewpoint, the former deliverable D2.4.1 has helped the task 2.4 team a lot during the implementation of the current prototype:

- The requirements have helped the 2.4 team as a base for defining the functional specification.
- The functional specification has been used by the 2.4 team as a guideline in the implementation phase. Its technology selection section is the base for all development efforts in the next period as well.
- The architecture specifications and the interface specifications have been used by the 2.4 team in order to allow a parallel development of the components and in order to connect them with each other in a consistent way

The following table shows the scheduled activities of the different subtasks in 2009 and 2010. The red arrow indicates the current position in the time plan:



		2008		2009											
		Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
		9	10	11	12 (M2)	13	14	15	16	17	18 (M3)	19	20	21	22
T2.4.1.1	Storage Services														
T2.4.1.2	Management Services														
T2.4.1.3	Communication Services														
T2.4.2.1	Design Templates														
T2.4.2.2	UI Components + Examples														
T2.4.2.3	Dashboard														

D2.4.1: First Demonstrator + Interf.Spec.

- Final Specs of everything
- First version of 2.4.2.1
- First alpha version of 2.4.1.1

Specs = Selection, Functional, Technic.

D2.4.2: First Prototype

- 2.4.1.x, 2.4.2.2, 2.4.2.3:
- Version 1 implementation
- First Demo
- How-To for developers

		2010							
		Jan	Feb	Mar	Apr	May	Jun	Jul	Aug
		23	24	25	26	27	28	29	30 (M4)
T2.4.1.1	Storage Services								
T2.4.1.2	Management Services								
T2.4.1.3	Communication Services								
T2.4.2.1	Design Templates								
T2.4.2.2	UI Components + Examples								
T2.4.2.3	Dashboard								

D2.4.3: Second Prototype

- 2.4.x Final Prototype
- 2.4.x Stunning Demo
- 2.4.x Detailed Documentation
- 2.4.x Completely Bug-Free

It needs to be emphasized that task 2.4 is currently in a very good shape in terms of the development and the progress. All deadlines have been kept and the task is even about 4-6 weeks in advance compared to the original time planning. As such, no problems are foreseen for the next iteration.

6. Annex A

Storage Services - repository IDs list structure definition (DTD):

```
<!DOCTYPE repositories [  
<!ELEMENT repositories (repository*)>  
<!ELEMENT repository (#PCDATA)>  
<!ATTLIST repository  
  urlEncoded CDATA #IMPLIED>  

```

Storage Services - file IDs list structure definition (DTD):

```
<!DOCTYPE files [  
<!ELEMENT files (file*)>  
<!ELEMENT file (#PCDATA)>  
<!ATTLIST file  
  urlEncoded CDATA #IMPLIED>  

```

7. References

- [APP08] Apple: Apple Human Interface Guidelines, Addison-Wesley, 2008
- [CEN09] Center for Universal Design, School of Design, North Carolina State University, Raleigh, USA: The 7 Principles of Universal Design, 2009
- [DX-UI] SOA4All Project Team: DX UI – Holistic User Interface, Extra Deliverable, 2008
- [JOH08] Johnson, J.: GUI bloopers 2.0: common user interface design don'ts and dos. Amsterdam; Boston, Elsevier/Morgan Kaufmann Publishers, 2008
- [KBa06] Koyani, S. J., R. W. Bailey, et al.: Research-Based Web Design & Usability Guidelines, U.S. Dept. of Health and Human Services., 2006
- [MIC95] Microsoft: The Windows interface guidelines for software design. Redmond, Wash, Microsoft Press., 1995
- [ROG07] Rogowski, R: The Business Case For Rich Internet Applications, 2007
- [SUN01] Java look and feel design guidelines. Boston, Addison-Wesley, 2001
- [TRI08] TRIPCOM: Specification of the Store Architecture and Interfaces, 2008
- [USA06] UsabilityNet: Design Guidelines for The Web., 2006. Retrieved 03/03, 2008, from <http://www.usabilitynet.org/tools/webdesign.htm>.
- [Vanduyne03] Duyne, D. K., Landay, J., and Hong, J. I. 2002 *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Addison-Wesley Longman Publishing Co., Inc.

8. Weblinks

[AJAX]	http://en.wikipedia.org/wiki/Ajax_%28programming%29
[COMET1]	http://gtoonstra.googlepages.com/cometwithgwtandtomcat
[EXT_GWT]	http://extjs.com/products/gxt/
[GWT]	http://code.google.com/webtoolkit/
[JOID]	http://code.google.com/p/joid/
[OPEN_ID]	http://www.openid.net
[OPJAVA]	http://code.sxip.com/openid4java/
[ROCKET]	http://code.google.com/p/rocket-gwt/wiki/Comet
[WSDL]	http://www.w3.org/TR/2005/WD-sprot11-20051024/
[REST]	http://en.wikipedia.org/wiki/Representational_State_Transfer