**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
# D2.4.3 SOA4All Studio Second Prototype
# - Prototype Documentation -

| Activity N: | Activity 1 | |
|---|---|---|
| **Work Package:** | WP2 – SOA4All Studio | |
| | T2.4 – SOA4All Studio UI and Infrastructure Services | |
| **Due Date:** | M30 | |
| **Submission Date:** | 31/08/2010 | |
| **Start Date of Project:** | 01/03/2008 | |
| **Duration of Project:** | 36 Months | |
| **Organisation Responsible of Deliverable:** | TIE | |
| **Revision:** | 1.0 | |
| **Author(s):** | Sven Abels | TIE |
| | Juergen Vogel | SAP |
| | Guillermo Álvaro Rey | ISOCO |
| | Iván Martínez | ISOCO |
| | Tomas Pariente Lobo | ATOS |
| | Alex Simov | ONTOTEXT |
| **Review:** | Usman Wajid (UNIMAN) | |
| | Philippe Merle (INRIA) | |

| | Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013) | |
|---|---|---|
| | **Dissemination Level** | |
| **PU** | Public | **x** |
| **PP** | Restricted to other programme participants (including the Commission) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission) | |
| **CO** | Confidential, only for members of the consortium (including the Commission) | |

# Version History

| Version | Date | Comments, Changes, Status | Authors, contributors, reviewers |
|---|---|---|---|
| 0.1 | 10.07.2010 | Kick-Off Version | TIE |
| 0.2 | 12.07.2010 | Extended Version | All |
| 0.3 | 16.07.2010 | Extended Version | All |
| 0.4 | 20.07.2010 | Extended Version | All |
| 0.5 | 22.07.2010 | Internal Review Version | All |
| 0.6 | 05.08.2010 | Internal Review Comments | UniMan, Inria |
| 0.7 | 23.08.2010 | Final format check | Julia Wells (ATOS) |
| 1.0 | 11.08.2010 | Final Version | All |

# Table of Contents

# List of Figures

# 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 *D2.4.3 (SOA4All Studio Second Prototype)* of Task *2.4 (SOA4All Studio UI and Infrastructure Services)* and describes the software implementation of the second official T2.4 prototype including the current status of the implementation. This document is included in the zip file that also contains the second 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.

The T2.4 prototype implementation not only covers all features mentioned in the time planning of D2.4.1 but also realizes 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 the 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/dashboard

# 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 second prototype of the SOA4All Studio that has been complied 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 server. The WAR file therefore combines the T2.4 results and integrates results of various modules coming from other WP2 tasks. As such, prototype D2.4.3 is called the "SOA4All Studio Second Prototype".

A temporary version is available at:

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

This document complements the *D2.4.3 SOA4All Studio Second Prototype* of the Task *2.4 SOA4All Studio UI and Infrastructure Services* and describes the software implementation of the second official D2.4 prototype. This document is included as part of the zip file that contains the second 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.

- 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/her 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, these services 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, Task 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 users to 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.
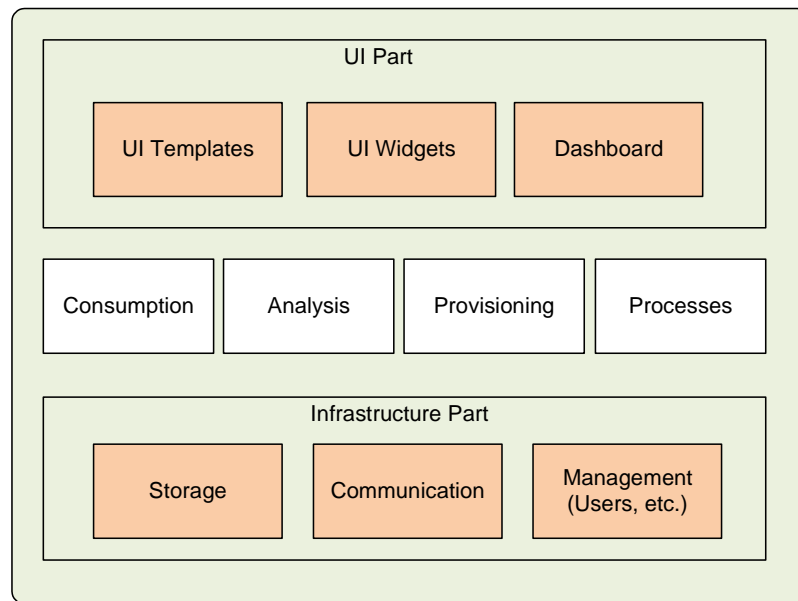


*Figure 1: Task 2.4 Structure*

## 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 Amazons S3 services[1] or to Google[2] 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 Task 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 pattern[3]. 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 built on top of the GWT[4] or EXT-GWT[5] 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 II

This section describes the prototype status as of month 30 of the project. It gives a short overview about what has been implemented in the second 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 RESTful[6] service, which facilitates functionality realization and reduces integration efforts significantly. The following sections describe in more details the APIs for managing repositories, files and RDF[7] data.

Usage examples of all functionalities of the storage services API are available as simple Java applications in the project's SVN[8] 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 identifiers. The (RESTful) API for managing repositories is described below:

| Resource/Operation | Method | Description |
|---|---|---|
| /repositories | GET | Lists all available repositories<br>**params**: none<br>**result**: XML (list of repository IDs – see Annex A) |
| /repositories/*<repository-id>* | PUT | Creates a new repository<br>**params**: none<br>**result**: none |
| /repositories/*<repository-id>* | DELETE | Deletes an existing repository<br>**params**: none<br>**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<br>**params:**<br>'*subj*' (optional) - subject restriction<br>'*pred*' (optional) - predicate restriction<br>'*obj*' (optional) - object restriction<br>'*context*' (optional) - named graph URI<br>**result:** RDF |
| /repositories/<*repository-id*>/statements | PUT | Updates statements in a repository (removing previous content and adding new statements)<br>**param:**<br>'*context*' (optional) - named graph URI<br>**header param:**<br>'*rdf-data*' - RDF triples data<br>**result:** none |
| /repositories/<*repository-id*>/statements | POST | Adds statements to a repository<br>**param:**<br>'*context*' (optional) - named graph URI<br>**header param:**<br>'*rdf-data*' - RDF triples data<br>**result:** none |
| /repositories/<*repository-id*>/statements | DELETE | Deletes statements from a repository<br>**params:**<br>'*subj*' (optional) - subject restriction<br>'*pred*' (optional) - predicate restriction<br>'*obj*' (optional) - object restriction<br>'*context*' (optional) - named graph URI<br>**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 SPARQL[9] query interface, which allows query evaluation on the data from a certain repository.

| Resource/Operation | Method | Description |
|---|---|---|
| /repositories/<*repository-id*> | GET | SPARQL query evaluation over a repository<br>**params:**<br>'*q*' (or alternatively header param '*sparql-q*') - SPARQL query expression<br>**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 that either cannot be serialised as RDF or where the serialisation may be inefficient (e.g. managing WSDL[10] files).

| Resource/Operation | Method | Description |
|---|---|---|
| /repositories/<*repository-id*>/files | GET | Lists all files in a repository<br>**params:** none<br>**result:** XML (list of file names – see Annex A) |
| /repositories/<*repository-id*>/files/<*file-name*> | PUT | Creates/Updates file in a repository<br>**param:** none / file content<br>**result:** none |
| /repositories/<*repository-id*>/files/<*file-name*> | DELETE | Deletes file from a repository<br>**params:** none<br>**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<br>**params**: none<br>**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 30:

| ID | Name | Priority<br>(1=high, 10=low) | Status |
|---|---|---|---|
| **ST-1** | Managing RDF | 1 | Implemented |
| **ST-2** | Querying RDF | 1 | Implemented |
| **ST-3** | Managing Files | 2 | Implemented |
| **ST-4** | Receiving Files | 2 | 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, In the SOA4All Studio OpenID[11] has been included as an 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.

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

---

[11] http://openid.net/

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.

**Security Aspects**

User authentication is naturally a field that requires high security. Within the T2.4 approach, those security aspects are two-fold:

Firstly, the approach relies on security of the OpenID service provider such as https://www.myopenid.com, which is used by T2.4.1 in the course of the project and which follows the specifications of the OpenID Foundation (http://openid.net/). MyOpenID is used by many different users and companies and is widely accepted as a trustful and secure OpenID provider. OpenID uses https encryption for all sensitive websites.

Secondly, the overall security also depends on the SOA4All implementation and user management. For this purpose, SOA4All never exposes any user information with the exception for OpenID integration, which is used to exchange authentication tokens. Therefore, private/sensitive user information never leaves the SOA4All system and cannot be manipulated by users.

It needs, however, be mentioned that SOA4All is an open system and will be published under and open source license. It can therefore be installed on any third party server and it can be changed by any developer to adapt to the own needs. This obviously means that the overall system security of a SOA4All installation strongly depends on the security of the underlying server. There is, nothing that can prevent this danger as it directly originates from the open nature of the system. However, it would be possible to allow only certified or verified servers to host official SOA4All installations to minimize this risk.

### 3.2.2   Simple Profile Management

The *User Profile Management* (see Figure 2) module is part of the SOA4All Studio[12]. 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 components of the studio.

---

[12] Snapshot version available and regularly updated at: http://coconut.tie.nl:8080/soa4all
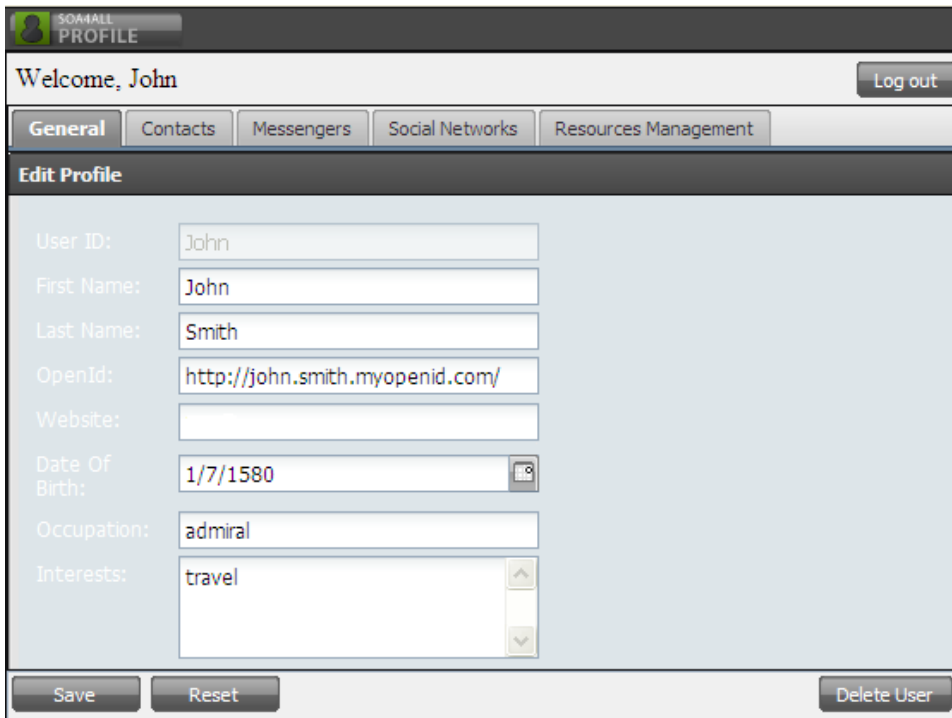
*Figure 2 User Profile Management Module*

### 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.

---

[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.
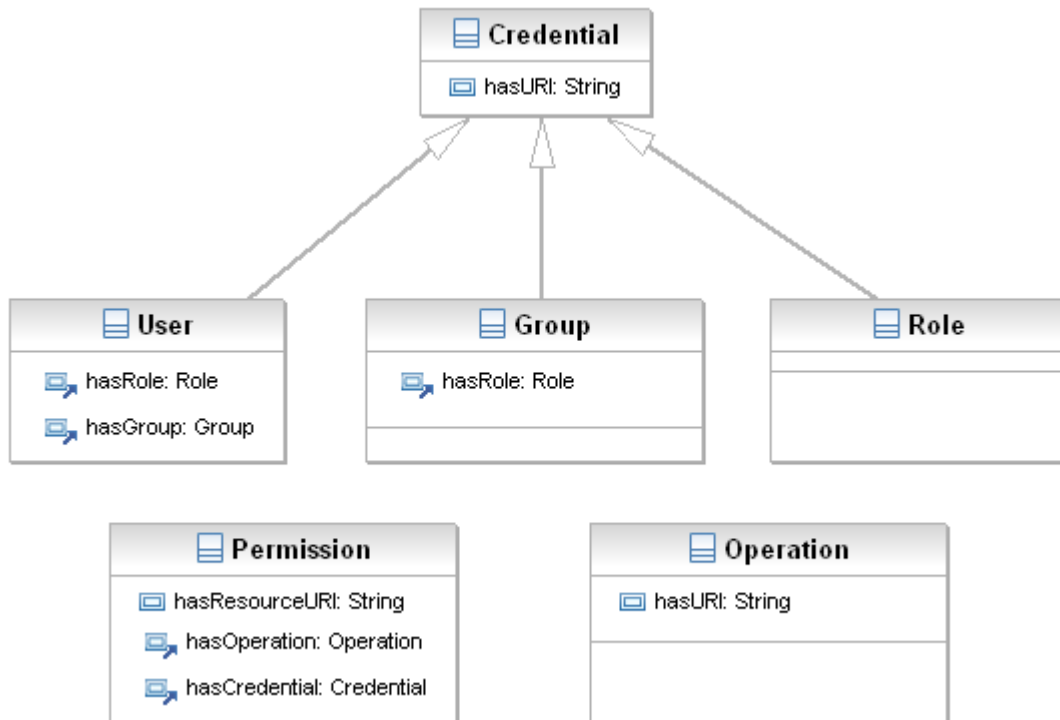
*Figure 3* Resource Authorisation Model

The key is to bring together *Group*s, *Role*s and *User*s 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 *Group*s and *Role*s, a *Group* may have a set of *Role*s.

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.

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<br>**params**:<br> '*user*' - user ID<br> '*resource*' - resource URI |

| Resource/Operation | Method | Description |
|---|---|---|
| | | '*op*' - operation URI<br>**result**: "true"/"false" |

### 3.2.3.2 Resource Permissions Management Service

| Resource/Operation | Method | Description |
|---|---|---|
| /resources | GET | Lists all resources<br>**params:** none<br>**result:** list of resource URIs |
| /resources | PUT | Creates a resource<br>**param:**<br> '*user*' - owner ID<br> '*res*' - resource URI<br>**result:** none |
| /resources | DELETE | Deletes a resource<br>**param:**<br> '*user*' - owner ID<br> '*res*' - resource URI<br>**result:** none |
| /resources/owner | PUT | Changes resource owner<br>**params**:<br> '*owner*' - owner ID<br> '*newOwner*' - new owner ID<br> '*res*' - resource URI |
| /resources/perm | GET | Lists resource access permissions<br>**params**:<br> '*res*' - resource URI<br> '*cred*' - credential URI (user, group or role)<br>**result:** list of operation URIs permissible for the given credential with respect to the resource |
| /resources/perm | PUT | Adds a new resource access permission<br>**params**:<br> '*user*' - user ID, adding the new permission (must |

| Resource/Operation | Method | Description |
|---|---|---|
| | | be the owner of the resource) |
| | | '*res*' - resource URI |
| | | '*cred*' - credential URI (user, group or role) given the permission |
| | | '*op*' - operation URI |
| | | **result**: none |
| /resources/perm | DELETE | Deletes a resource access permission |
| | | **params**: |
| | | '*user*' - user ID, deleting the permission (must be the owner of the resource) |
| | | '*res*' - resource URI |
| | | '*cred*' - credential URI (user, group or role) given the permission |
| | | '*op*' - operation URI |
| | | **result**: none |
| /repositories/creds | GET | Lists all credentials which at least one permission is granted |
| | | **param**: '*res*' - resource URI |
| | | **result**: list of credential URIs which are granted at least one permission |

### 3.2.3.3  User Management Service

| Resource/Operation | Method | Description |
|---|---|---|
| /users/groups | GET | Lists all user groups |
| | | **params:** none |
| | | **result:** list of URIs of all user groups defined |
| /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<br>**param:** '*id*' - role URI<br>**result:** none |
| /users/roles | DELETE | Deletes a role<br>**param:** '*id*' - role URI<br>**result:** none |
| /users/assignGroup | PUT | Assigns a user to a group<br>**params**:<br>'*user*' - user ID<br>'*group*' - group URI<br>**result**: none |
| /users/assignGroup | DELETE | Excludes a user from a group<br>**params**:<br>'*user*' - user ID<br>'*group*' - group URI<br>**result**: none |
| /users/assignGroup | GET | Lists all group assignments for a user<br>**param:** 'user' – user ID<br>**result:** list of user group URIs |
| /users/assignUserRole | PUT | Assigns a role to a user<br>**params:**<br>'*user*' - user ID<br>'*role*' - role URI<br>**result**: none |
| /users/assignUserRole | DELETE | Divests a user of a role<br>**params**:<br>'*user*' - user ID<br>'*role*' – role URI<br>**result**: none |
| /users/assignUserRole | GET | Lists all roles of a user<br>**param:** 'user' – user ID<br>**result:** list of role URIs |

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

### 3.2.4 Auditing

Following the decupled approach of most of the infrastructure services within the SOA4All Studio, the Auditing Service, which permits client-side modules within different modules to log interesting actions that the users perform when interacting with the tools, is now exposed as a RESTful service as well.

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.

The REST interface has a unique resource `action` with the POST verb allowed to track new actions, and different parameters depending on the action, as depicted in the following table:

| Resource/Operation | Method | Description |
|---|---|---|
| /auditing/action | POST | Logs a new action<br><br>**params**:<br> '*action*' – ID of the action<br> '*persistentSessionID*' – cookie information<br>'*item*' – URI of the item (optional)<br>'*user*' – ID of the user (optional)<br>'*review*' – URI of the review (optional)<br>'*tag*' – URI of the tagging (optional)<br>'*category*' – name of the category (optional)<br>'*agent*' – ID of the agent (optional)<br>'*method'* – ID of the method (optional)<br>**result**: "ok"/"nook" |

The following table represents the actions that can be tracked through these method, and the expected parameters for each of them:

| Action | Parameters | Action | Parameters |
|---|---|---|---|
| LoginAction | user | ItemReview | item, review |
| LogoutAction | - | ItemInvocation | Item |
| ItemSelection | item | ItemCreation | item, agent, method |
| ItemBookmarking | item | ItemRetrieval | item, agent, method |
| ItemTagging | item, tag | itemUpdating | item, agent, method |
| ItemEditing | item | itemDeleting | item, agent, method |
| ItemCategorization | item, category | | |

Web modules within the Studio are able to log users' actions by invoking this REST method, as SPICES (the Consumption Platform) does.

### 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 30:

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

## 3.3   Communication Services

Communication services are basic messaging services aiming to give extra functionality for the communication among 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 M30, 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. The T2.4 team therefore stopped this activity in M18 (see deliverable D2.4.2) and concentrated on other parts of T2.4 instead.

### 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. In this respect, the use of Http streaming is suggested when an 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 (e.g. Google Mail[15] is using that technique to update the mail interface in a real time fashion).

### 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 30:

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

## 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.

---

[15] http://mail.google.com

### 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 30:

| 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 duplication of work and to achieve a common look and feel of the entire Studio.

### 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.
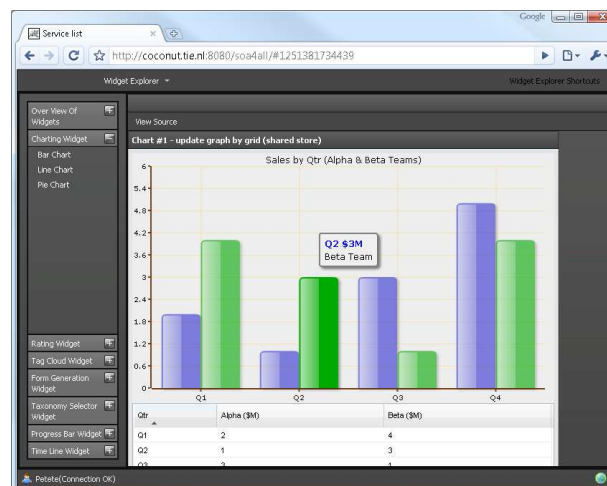


*Figure 4: Charting Widget - Current Implementation*

### 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

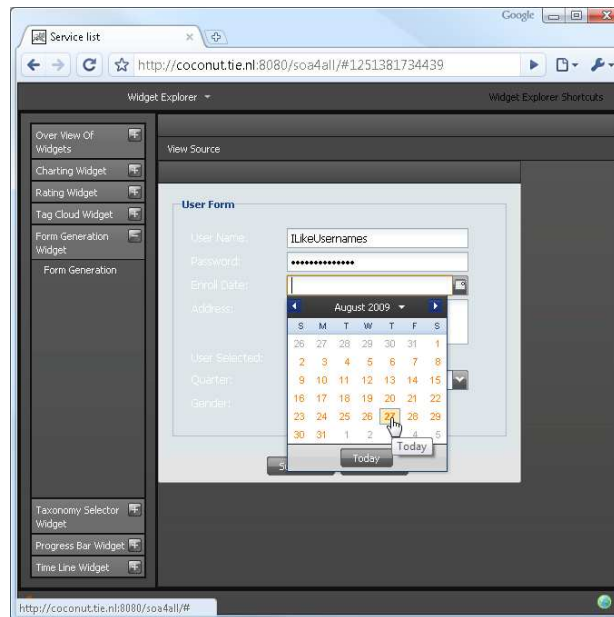the widget after the user has clicked on a submit button. The widget is required for tasks 2.2 / 2.3 / 2.4.



*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 2.2 / 2.3 / 2.4.

### 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.  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 in section 3.5.1.

### 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 email 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 easy 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.

### 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 consider 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.
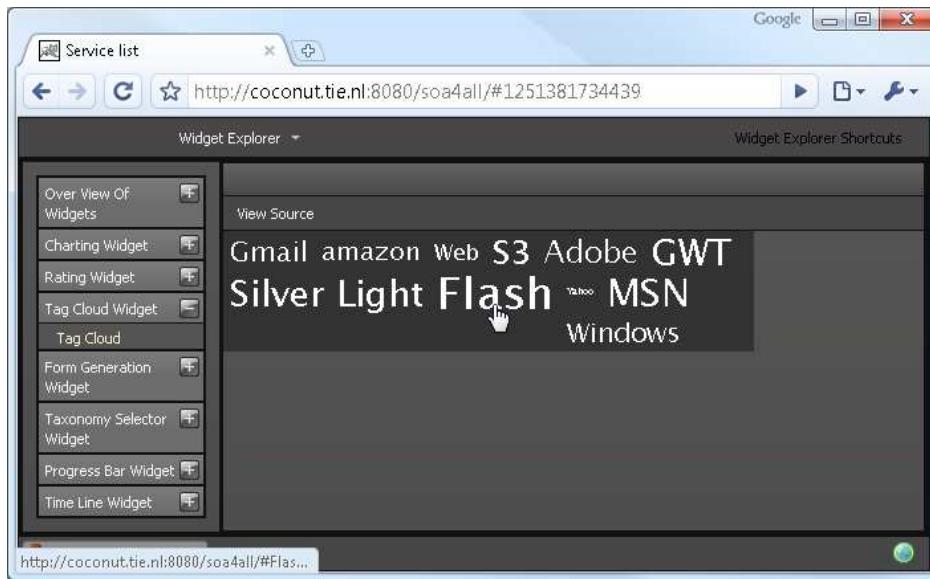
*Figure 7: Current Implementation of the Tag Cloud Widget*

### 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 30:

| ID | Name | Priority (1=high, 10=low) | Status |
|----|------|---------------------------|--------|
| UIC-1 | Charting | 1 | Implemented |
| UIC-2 | Form Generation Widget | 1 | Implemented |
| UIC-3 | AdvListView | 2 | Implemented |
| UIC-4 | TimeLine | 3 | Implemented |
| UIC-5 | Taxonomy Selector Widget | 1 | Implemented |
| UIC-6 | GraphVisualization | 2 | Implemented |
| UIC-7 | FaultHandler | 1 | Implemented |
| UIC-8 | Optional HelpSystem | 5 | Not implemented due to low priority |
| UIC-9 | Gauges | 6 | No implementation needed (part of new |

| | | | GWT version) |
|---|---|---|---|
| UIC-10 | Rating | 3 | Implemented |
| UIC-11 | Search & Result | 3 | Implemented |
| UIC-12 | Optional Client Side Filtering | 6 | Not implemented due to low priority |
| UIC-13 | Drawing API | 1 | Implemented |
| UIC-14 | History Widget | 5 | Implemented |
| UIC-15 | TagClouds | 1 | Implemented |
| UIC-16 | Progress Bar | 1 | Implemented |

## 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:
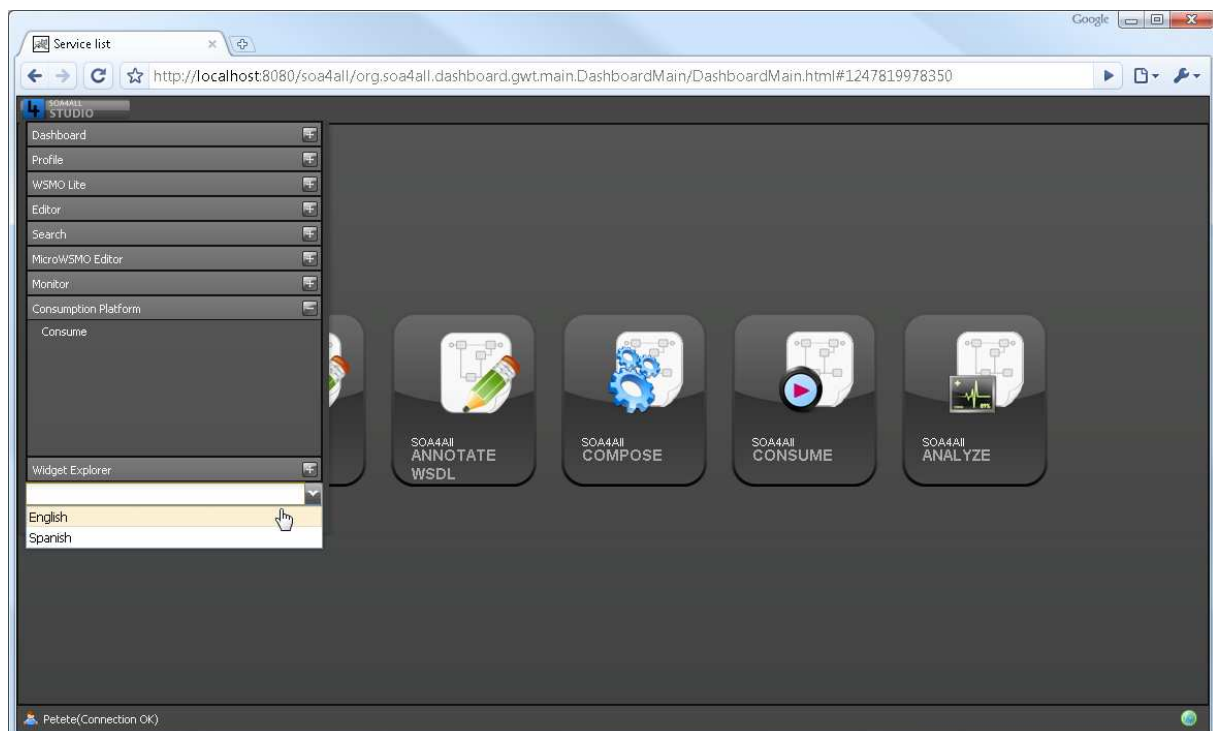


*Figure 8: Current Dashboard implementation*

### 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 30:

| ID | Name | Priority (1=high, 10=low) | Status |
|---|---|---|---|
| DB-1 | Entry Point | 1 | Implemented |
| DB -2 | Dynamic Coupling | 1 | Implemented |
| DB -3 | QuickStart | 8 | Implemented |
| DB -4 | Personalization | 6 | Implemented |
| DB -5 | Core Elements Integration | 1 | Implemented |

# 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 results are temporary available for testing purposes at:

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

However, they will be moved to the official SOA4All website in the course of the open source publishing activities of the SOA4All project.

### 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 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 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 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
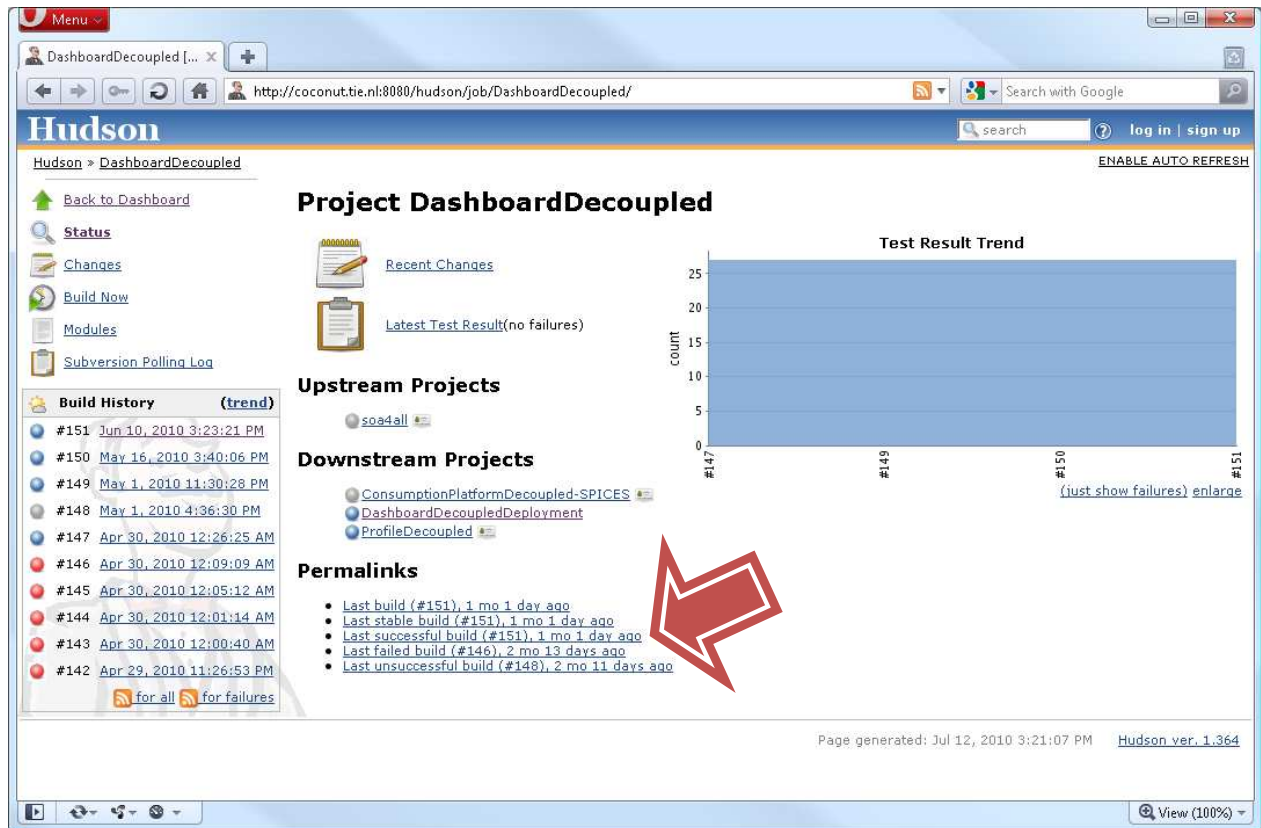
http://coconut.tie.nl:8080/hudson/

*Figure 9: Downloading the 2.4 prototype*

## 4.3    Execution

After installing the SOA4All 2.4 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 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 second 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 and D2.4.2 helped as an intermediate step to the final prototype:

- The requirements have served the 2.4 team as a base for defining the functional specification.

- The functional specification has been used by the Task 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

With this deliverable, Task 2.4 has ended. It should be emphasized that the task has fulfilled all planned activities within time and has even achieved additional extra tasks such as the widget explorer and additional build maintenance activities.

# 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>
]>
```


Add DTD for complex XML results of the management REST services.

# 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

# 9. SOA4All Publications related to task 2.4

Within the course of the project, several publications have been created. Those have been published as articles in journals or within conference and workshop proceedings. Since Task 2.4 provides a core set of components for the SOA4All Studio, it is indirectly described in all Studio related publications. The following list shows those SOA4All publications that have relevance for task 2.4:

Domingue, J., Fensel D., González-Cabero, R. SOA4All, Enabling the SOA Revolution on a World Wide Scale. In Proceedings of the 2nd IEEE International Conference on Semantic Computing (ICSC 2008), Santa Clara, CA, USA, August 2008.

Lecue, F., Gonzalez, R., Gorronogoitia, Y., Radzimski, M., Villa, M. SOA4All: An Innovative Integrated Approach to Services Composition. In Proceedings of the IEEE International Conference on Web Services (ICWS 2010), July 5-10, 2010, Miami, FL, USA. IEEE Computer Society 2010 (to appear).

Krummenacher, R., Norton, B., Simperl, E., Pedrinaci, C. SOA4All: Enabling Web-scale Service Economies. In Proceedings of the 3rd IEEE International Conference on Semantic Computing (ICSC) 2009, IEEE Computer Society.

Abels, S. SOA4All: Enabling a Web of Billions of Services. In International Journal of Interoperability in Business Information Systems (IBIS), Issue 3 (1), 2009, pp. 35-38.

Pedrinaci, C., Domingue, J. Toward the Next Wave of Services: Linked Services for the Web of Data. In Journal of Universal Computer Science, 2010 (to appear).

Pedrinaci, C., Simperl, E., Krummenacher, R., Norton, B. Towards Web-scale SOA. In The Architecture Journal, 21, Microsoft, 2009.

Adrian Mos, Carlos Pedrinaci, Guillermo Alvaro Rey, Jose Manuel Gomez, Dong Liu, Guillaume Vaudaux-Ruth and Samuel Quaireau, "Multi-Level Monitoring and Analysis of Web-Scale Service Based Applications". In Proceedings of the 2nd Workshop on Monitoring, Adaptation and Beyond (MONA+), ICSOC/ServiceWave 2009, Stockholm, Sweden, November 2009.