

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

D2.5.2. Summative Evaluation Report

Activity N:	Activity 1 - Fundamental & Integration Activities	
Work Package:	WP2 - Service Deployment and Use	
Due Date:	M24	
Submission Date:	28/02/2010	
Start Date of Project:	01/03/2008	
Duration of Project:	36 Months	
Organisation Responsible of Deliverable:	The University of Manchester	
Revision:	1.0	
Author(s):	Abdallah Namoune	UNIMAN
	Usman Wajid	UNIMAN
	Nikolay Mehandjiev	UNIMAN
Reviewers:	Sven Abels	TIE
	John Davies	BT

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination Level		
CO	Confidential, only for members of the consortium (including the Commission)	X

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	10/01/2010	First proposal and table of content	Abdallah Namoune (UniMan)
0.2	16/01/2010	Evaluation techniques and plan	Abdallah Namoune (UniMan)
0.3	01/02/2010	Evaluation rational and scenarios	Abdallah Namoune (UniMan)
0.4	04/02/2010	Usability problems probed by scenario one	Abdallah Namoune (UniMan)
0.5	07/02/2010	Usability problems probed by scenario two	Abdallah Namoune (UniMan)
0.6	08/02/2010	Usability problems probed by scenario three	Abdallah Namoune (UniMan)
0.7	09/02/2010	Upcoming evaluation studies and Conclusions	Abdallah Namoune (UniMan)
0.8	11/02/2010	Usability problems, evaluation and review of the document	Usman Wajid (UniMan)
0.9	15/02/2010	Further revision of the deliverable and inclusion of internal comments.	Abdallah Namoune (UniMan) Nikolay Mehandjiev (UniMan)
1.0	25/02/2010	Addressing reviewers' comments	Usman Wajid (UniMan)

Table of Contents

EXECUTIVE SUMMARY	5
1. INTRODUCTION	6
1.1 INTRODUCTORY EXPLANATION OF THE DELIVERABLE	6
1.2 PURPOSE AND SCOPE	6
1.3 STRUCTURE OF THE DOCUMENT	6
1.4 METHODOLOGY	6
2. USABILITY EVALUATION STRATEGY AND TECHNIQUES	7
2.1 EARLY EVALUATIONS	7
2.2 UPCOMING EVALUATIONS	7
2.3 UPDATED USABILITY EVALUATION PLAN	8
3. PHASE TWO EVALUATION METHODOLOGY	8
3.1 EVALUATION RATIONALE	8
3.2 PROCEDURES OF EXPERT-BASED EVALUATION OF SOA4ALL STUDIO	9
3.2.1 <i>Evaluation Steps</i>	9
3.2.2 <i>Usability Heuristics</i>	10
3.2.3 <i>Test Scenarios</i>	11
4. PHASE TWO EVALUATION RESULTS	11
4.1 USABILITY PROBLEMS PROBED BY WP 7 SCENARIO	11
4.2 USABILITY PROBLEMS PROBED BY WP8 SCENARIO	16
4.3 USABILITY PROBLEMS PROBED BY WP9 SCENARIO	19
5. CONCLUSIONS	23
6. REFERENCES	24
7. APPENDIX	25

List of Tables

Table 1: Updated Usability Evaluation Plan	8
Table 2: Nielsen Usability Heuristics	10
Table 3: Usability Problems in the SOA4All Studio and their Corresponding Design Recommendations, Probed by WP7 Scenario	16
Table 4: Usability Problems in the SOA4All Studio and their Corresponding Design Recommendations, Probed by WP8 Scenario	19
Table 5: Usability Problems in the SOA4All Studio and their Corresponding Design Recommendations, Probed by WP9 Scenario	23

Glossary of Acronyms

Acronym	Definition
D	Deliverable
EC	European Commission
WP	Work Package
M12	Milestone 12 of the SOA4All project
M1	Milestone 1 of the SOA4All project
QoS	Quality of Service
DL	Description Logic
IPs	Integrated Projects

Executive summary

The current document presents the results of the interim usability evaluations of SOA4All Studio up to month M22. This second iteration of expert-based evaluation is a continuation of the initial evaluation efforts reported in Deliverable D2.5.1, which proved to be invaluable to developers of the Studio as it quickly highlighted the main design issues and suggested design remedies at the early stages of the project. In this second phase of evaluation, experts walked through the current version (M22 version) of the SOA4All Studio and focused on inspecting new design flaws and proposing recommendations to counterbalance them. On the contrary to the initial evaluation (reported in D2.5.1), in which experts used only one scenario to steer the assessment, experts in this occasion employed the detailed scenarios of WP7 (deliverable D7.2), WP8 (deliverable D8.1), and WP9 (deliverable D9.2) to thoroughly investigate the Studio. The variety of user actions encapsulated by those three scenarios allowed identifying various issues and testing many important features of the Studio such as: the profile editor, composition editor, annotation editor, discovery platform, consumption platform, and monitoring platform.

In general, the results of the current extensive expert-based evaluations pinpointed issues of diversified severity in M22 version of the Studio. Some issues identified by the experts relate to light-weight user interaction, such as movement, deletion and modification of services/activities. Whilst, other relate to service composition (e.g. creation of bindings between ontology elements and service elements), service consumption (e.g. executing and interpreting the results of services), and service monitoring aspects (e.g. assessing the quality of a particular service using its monitoring data). Corrective measures to overcome both types of issue are highlighted for developers to address in the upcoming versions of the Studio.

1. Introduction

1.1 Introductory explanation of the deliverable

Continuous usability evaluation of software designs and artefacts is a crucial step in the development of effective interactive software applications since it enables the detection of design problems during the software life cycle and consequently proposes early solutions before products are released for public consumption.

In deliverable D2.5.1, we reported results summarising early focus group-based and expert-based evaluations of various end user development concepts, early prototypes, and software artefacts. The current deliverable D2.5.2 details the latest evaluation efforts undertaken to assess recent implementation and features added to the SOA4All Studio up to month M22 of the project. It primarily presents results of the most up to date expert-based usability evaluations carried out in the SOA4All project to assess different features and parts of the Studio. In addition, the document highlights and justifies the need to update the evaluation plan in accordance with the project deadline and effort spent to develop the SOA4All Studio and its components.

1.2 Purpose and Scope

In essence, this deliverable mainly describes the second iteration of usability evaluations, which we have carried out between month 18 and month 24 of the project, namely: an extensive expert-based usability evaluation of the SOA4All Studio and its components, guided by three different scenarios. At this stage of the project it is not appropriate to test the Studio with end users since much functionality is not yet available, therefore we preferred to perform cost-effective heuristics evaluation which checks conformity of the Studio to design guidelines. The recommendations by evaluation experts will be fed into the development process.

In this deliverable, we describe the evaluation approach and argue for the rationale behind our evaluation strategy at this stage of the project. The evaluation focuses on only those user activities specified in the work packages (WP7, WP8, WP9) scenarios which are supported by the current version of the Studio.

1.3 Structure of the Document

The rest of this document is organised into five main sections. Section 2 restates the evaluation strategy and techniques employed in SOA4All. Section 3 explains the evaluation methodology and procedures of phase two of the project, alongside the rationale behind the evaluation philosophy. Section 4 reports a comprehensive list of design flaws and proposes proactive measures for developers of the components of the Studio in order to resolve them in future versions. Finally, Section 5 summarises the major findings of this deliverable.

1.4 Methodology

At first, we had to assess the status of the Studio in order to decide which type of evaluation best suits the most recent version of the Studio. This step was necessary since many components in the studio heavily depend on the availability of other services. Subsequently, three scenarios from Work Package 7, Work Package 8, and Work Package 9 were selected owing to their diversity and practical relevance to guide and assist evaluators in the evaluation process.

Following this, two experts repeated their initial expert-based evaluation reported in D2.5.1,

stepped through the Studio and inspected the latest usability problems. The problems were then documented and design recommendations were proposed to overcome the detected problems.

2. Usability Evaluation Strategy and Techniques

2.1 Early Evaluations

During the early usability evaluations of M14, we employed two techniques - focus groups and expert-based evaluation - to firstly assess users' understanding of services and service composition, their perception of risks and benefits and willingness to take up end user development activities, and secondly measure the usability of the SOA4All prototypes and software artefacts.

Focus groups usually include a number of participants (6-10) who discuss different topics of interest under the supervision of a moderator [1]. Focus groups help to elicit user requirements and provide a better understanding of users' perceptions and attitudes towards end user development. The main findings of focus groups (M14) showed that ordinary end users have a poor understanding of the technical details of services and service composition, while they expressed great interest in undertaking development activities. In terms of risk, people were concerned about their personal privacy and security. In terms of benefits, people argued that enabling end users to develop specialised service-oriented applications is interesting, useful, and will save the time. Further details of the results were published in [2].

In heuristic evaluation, on the other hand, usability experts go through the purported system and check whether it conforms to well-known usability heuristics such as those proposed by Jacob Nielsen who developed it from analysing 249 usability problems [3]. End users are not involved in this type of evaluation. The result of this assessment is summarised as a record of usability problems in the SOA4All Studio user interface and a list of accompanying recommendations to design teams. In the first heuristic evaluation, experts used a realistic scenario (One Stop Cloud Shop, D2.5.1) that contains a set of potential tasks to be performed by a specified user whilst carrying out their daily job. At the end of the evaluation, evaluators pinpointed the underlying problems in the Studio, related them to design principles and most importantly suggested appropriate counteractive solutions to SOA4All design teams. Further details of the initial evaluation results have been reported in D2.5.1.

2.2 Upcoming Evaluations

In future evaluations, especially when the Studio is fully functional and most of its features are completely implemented, we plan to conduct a series of user-based evaluations that will assess the usability of the final products of SOA4All. This type of evaluation is most convenient for the third stage of the project (M24 – M36) because by then it is possible to involve real end users in the evaluation process. Usually user testing provides rich user interaction data and more insights about the actual problems users face when using a particular interactive system, but it requires a certain degree of stability, maturity, and integration of the software artefacts under investigation and SOA4All Studio has not yet reached this stage at the time of evaluation. The researcher measures user performance while carrying out typical tasks, after recording user interaction behaviour via video, audio, and log recording programs. It is also possible to capture user opinions and satisfaction via questionnaires and debriefing interviews. The researcher can later analyse the number and type of problems users encountered, and calculate various objective performance measures such as the time spent and number of errors to perform the tasks. For a deeper understanding of the inspected problems, results of the usability testing will be analysed using a usability post analysis process (i.e. Model Mismatch Analysis (MMA)) [4].

2.3 Updated Usability Evaluation Plan

Table 1 summarises the most suitable usability evaluation techniques for each Use Case in WP7, WP8, WP9, and WP 2, and redefines a new evaluation deadline for the third stage of the project, as M33 instead of M36. This change is motivated by the fact that user testing together with the analysis of data consume a considerable amount of time. Moreover the final evaluation report is due in M36, which is the end date of the SOA4All project. Hence, it is only reasonable to carry out the user-based evaluations 3 months before M36 to be able to analyse the data and submit the final deliverable on its due date, in order that an extension to finish the project is not required.

Work Package	Target end users	First stage	Second stage	Third stage
		Initial mock-ups, low-fidelity prototypes, power point presentations,	Initial prototypes, high-fidelity prototypes	End-user products
WP7	End users from public sector	Focus groups Heuristics evaluation	Heuristics evaluation	User testing
WP8	BT customers	Focus groups Heuristics evaluation	Heuristics evaluation	User testing
WP9	E-Commerce User (Buyers, Sellers, Resellers)	Strategic priorities interviews Analysis of existing user data	Heuristics evaluation	User testing
WP2	General users (e.g. students, staff at University)	Focus groups Heuristics evaluation	Heuristics evaluation	User testing
Deadline of evaluation		M14: 05 / 2009	M22: 01 / 2010	M33: 11 / 2011
Deliverable due date		M18	M24	M36

Table 1: Updated Usability Evaluation Plan

3. Phase Two Evaluation Methodology

3.1 Evaluation Rationale

In contrast to the first stage of the usability evaluation, in which both focus groups and

heuristics evaluations were undertaken, only expert-based evaluations have been performed in the second stage of the evaluation, which is aligned with the original usability evaluation plan. The present evaluation aims to discover existing issues in various components of the Studio and provide feedback and recommendations to designers in order to make further improvements in the design of the Studio before the final user-based evaluations scheduled for M33.

Our approach to evaluating SOA4All prototypes and software artefacts at this stage of development mainly concentrate on expert-based inspection for various reasons. Firstly, many parts of the Studio are still under development and it is not suitable to test their applicability with end users. Secondly, the cost of user-based testing at this phase will outweigh the benefits and scientifically it is not worthwhile testing features that are not completed yet. Essentially, user-based evaluations mainly assess the interaction behaviour between potential end users and interactive systems whilst trying to execute realistic tasks. Unfortunately, at present it is not possible to realise pragmatic user interactions, which envisage actual activities within specified contexts. Thirdly, for those Studio parts that are implemented, it is not possible to create a typical user story for testing purposes. The formulation of a coherent test scenario that embodies realistic user activities is not feasible owing to the interdependency between many aspects of the Studio that are yet under development. For the above reasons, it is more appropriate that heuristic evaluations are performed to obtain the most useful results, whilst user testing should be held back until the Studio is fully functional.

3.2 Procedures of Expert-based Evaluation of SOA4ALL Studio

Usability experts endeavoured to identify as many design issues as possible in the current version of SOA4All Studio, especially those that relate to direct user interface manipulation and end user development activities by checking Studio behaviour against the Nielsen heuristics. Another motivating objective was to provide quick feedback to Studio developers and generate recommendations to improve the user interface in the next development iterations. In what follows, we present the evaluation procedure, heuristics, and scenarios used to fulfil the objectives.

3.2.1 Evaluation Steps

Expert-based evaluation comprises six basic steps that were followed by the evaluators:

- Define the aim of the evaluation, the target end users, and the context of use for SOA4All Studio: the intended end users of SOA4All Studio are general web users who frequently use web 2.0 applications such as: Facebook¹, Twitter², and Wikis. The SOA4All Studio will be used to create personal applications for general leisure, and also for business purposes to generate revenue (such as: reselling services).
- Select heuristics: for the purpose of this evaluation, we selected the most used heuristics of Jacob Nielsen [3, 5] for user interface design (section 3.2.2). There are also other usability heuristics available, such as Jill Gerhardt-Powals cognitive principles [6].
- Brief the evaluators about SOA4All Studio and how it is intended to be used: developers of the Studio outlined the purpose of the SOA4All Studio and explained

¹ <http://www.facebook.com>

² <http://www.twitter.com>

how it can be used to usability evaluators.

- Each evaluator independently makes a first pass through the Studio to obtain an overall impression about the general features and look and feel of the design.
- Each evaluator independently examines the aspects of the design in detail, working through typical scenarios. In this second iteration of usability evaluation, three diverse and comprehensive scenarios, which address different features of the Studio (e.g. composition, consumption, monitoring, etc), were used to steer the usability inspection process.
- Produce a record of problems, link each design problem to appropriate heuristics, rate their severity on a 1-3 rating scale (1 = not severe at all, 3 = very severe), and suggest solutions to counterbalance these problems.

3.2.2 Usability Heuristics

To perform our heuristic evaluation, we have selected the widespread and general-purpose heuristics of Nielsen [5]. Table 2 lists and explains the ten heuristics defined by Nielsen.

ID	Heuristic
H1	Visibility of system status: is the system continuously informing the users what is going on using appropriate feedback? Are all things visible to the user?
H2	Match between system and real world: does the system use familiar words, phrases, and concepts to the users? Is information presented in a natural and logical order? Are metaphors used effectively?
H3	User control and freedom: does the system support the undo and redo actions? Are there clearly marked exits in case of a mistake? Can the user easily go back to the initial stage?
H4	Consistency and standards: is the use of different components consistent throughout the system? Have the platform conventions been followed?
H5	Error prevention: does the system eliminate error-prone conditions? Does the system ask for confirmation before executing a dangerous action?
H6	Recognition rather than recall: Are the objects, actions and options visible to the user? Does the system offer visible instructions of how to use the system?
H7	Flexibility and efficiency of use: does the system support both novice and expert users? Does the system allow the users to skip unnecessary actions?
H8	Aesthetic and minimalist design: does the system contain the relevant elements only? Is it free from distractive elements?
H9	Help users recognize, diagnose, and recover from errors: does the system clearly describe the problem and suggest a way of recovery?
H10	Help and documentation: does the system provide clear and focused help and documentation?

Table 2: Nielsen Usability Heuristics

3.2.3 Test Scenarios

Expert evaluators were supplied with three distinct scenarios that were used to identify design issues in the current version of the Studio (M22 version). The dissimilarity between the three scenarios was very beneficial since it allowed (1) the testing of various critical parts of the Studio, particularly the profile editor, discovery platform, annotation editor, composition editor, consumption platform, and monitoring platform, as well as (2) the identification of different problems in that relate to interaction and look and feel of the Studio.

Two evaluators of SOA4All Studio used the WP7, WP8, and WP9 scenarios described in deliverable D7.2, D8.1, and D9.2 respectively to step through the Studio and focus their assessment on particular Studio aspects and features. WP7 scenario comprises user tasks that relate to the use of profile editor, discovery platform, consumption platform, and composition editor of the SOA4All Studio. WP8 scenario comprises user tasks that relate mainly to the use of profile editor, discovery platform, and composition editor. Lastly, WP9 scenario comprises user tasks that relate to the use of composition editor, consumption platform, monitoring platform, and WSMO-Lite editor. For further details about the scenarios refer to D7.2, D8.1, and D9.2.

4. Phase Two Evaluation Results

During the first evaluation study (reported in D2.5.1) the SOA4All studio offered very simple features. The look and the feel of the studio was at preliminary level at that time and the functionality provided by the studio was very basic, for instance users were not able to add activities and goals to the process model and users cannot execute services in the composition platform. In summary, during the last evaluation the studio did not support development activities. However, significant improvements have been made in the studio since the first evaluation study. For instance, in the current version you can execute services, compose services and annotate services.


In this section of the deliverable, we report the usability design problems probed by the WP7, WP8 and WP9 scenarios, link the problems to Nielsen’s usability heuristics, rate their severity on a 3-point rating scale (where 1 = low severity, 2 = medium severity, and 3= high severity), and finally propose design resolutions to remedy these problems.

4.1 Usability problems probed by WP 7 Scenario

Usability problem	Severity rating (1-3)	Design recommendation
The profile creation option is hidden in a tree menu on the top-left corner of the screen, which is difficult to find by a first time user	3	Either toggle on the menu at all times clearly showing the profile editor options or provide an up-front button for profile creation for ease of use.
No comprehensive system support is provided to users to illustrate how a profile	3	Provide clear instructions and hints to users about how

<p>can be created, especially with the necessity to have an OpenID account. What is OpenID? No instructions are provided in the Studio.</p>		<p>to create a SOA4All profile. This is very important to novice users and beginners.</p>
<p>The three rectangular menu items (Create, Consume, and Analyse) on the home page are not very elaborative of what functionality they offer. A novice user might easily be asking herself, create what? an application, a model, a process, an annotation, an interface, etc?</p>	<p>1</p>	<p>Add descriptive text snippets, that appear upon mouse-hovering to explain what functions these menu options offer if a user decides to commit to any of them.</p>
<p>During the logging in process, users are directed to the OpenID website which is confusing. Users then have to click on “continue to Studio” in order to be diverted back to the Studio. The benefits of this transit are questionable, as it adds no value.</p>	<p>2</p>	<p>There is no need to transit users through intermediary web sites such as: OpenID website, hence users should be kept in the Studio unless it is absolutely necessary.</p>
<p>Once users are logged into the Studio, no feedback is presented to indicate their status. Therefore, users would not know if they are still logged in the system. At certain times, this was confusing and errors of the Studio could be mistakenly linked to the possibility of “not being logged in”.</p>	<p>3</p>	<p>At all times, the system should notify users whether they are logged in the system or not via a message which could be displayed on the Studio to show their status.</p>
<p>No functionality is provided for users to log off the Studio. If users visit the profile editor, no greeting message or information regarding their status (logged in / logged out) is presented. User can not create personal profiles, which could be shared with other users. This may be relevant in the case of collaborative design activities.</p>	<p>2</p>	<p>Empower end users to log off from the Studio any time they want. It might also be worthwhile to enable them to create personal profiles which contain contact details, interests, a photo... etc. Such information becomes handy in collaborative development activities or could be used in the process editor.</p>
<p>In case a user forgets his log-in information (ID, password), there is no way to retrieve it .</p>	<p>1</p>	<p>In this situation, the Studio must ask users to supply their email address, to which their log-in details will be sent.</p>
<p>The Studio does not offer quick and easy way to navigate back to the home page, or navigate between pages of the Studio.</p>	<p>2</p>	<p>Add a link/logo that enables users to easily access the main/start page of the Studio.</p>
<p>Although the Studio is hosted within a web browser, back arrows of the browser are not working properly.</p>	<p>1</p>	<p>Ensure browser features such as back and forward arrows are fully operational.</p>

		Since SOA4All Studio is hosted within a web browser, it is expected that users will very much depend on browser navigational features.
Extending the menu of classes, in the discovery platform, results in a long menu hindered by pre-condition / effect windows on the left hand side. Hence, menu items at the bottom become invisible and inaccessible to users.	2	Make sure the menu of classes is viewable always to facilitate user navigation. Either relocate the pre-condition / effect windows to the right hand side or add a vertical scroll bar to the menu of classes.
Meaningless names are given to services such as: service696, service32, etc. Users will not be able to understand the purpose or functionality of those services.	3	Use self-explanatory names whenever possible to represent services and development-related concepts.
Service discovery sometimes does not work i.e. after pressing the 'Search' button no results are displayed.	3	This can be a problem with the server but in this case there should be a message displayed with the appropriate content.
Too many technical details are shown to users in the right hand windows (pre-condition, effect, input message, output message), in the discovery platform. Users' understanding of such technical jargon is very much dependent on their background and knowledge. Whilst technical users are likely to understand the content of these windows, ordinary people will have no clue of their meaning.	3	Either remove the pre-condition, effect, input message, and output message windows if unimportant or use natural language to convey the embodied technical details to ordinary users.
Not sure how the pre-condition and effect search windows can be used to find particular services. No explanation or instructions are offered to Studio users.	3	Provide examples and help to show how to use the pre-condition and effect search windows on left side of the discovery platform.
Searching for particular services in the discovery platform, for example: Amazon service, is not supported via a typical search box. The functionality-based search is based on the assumption that users will browse the service classification on the left side to find their preferred service.	3	Add a typical keyword-based search box to enable easy and quick search of services. This is more efficient than browsing the service classification.

<p>It is not obvious how services that are found using the discovery platform can be used, since at the moment only information about service classification, pre-condition, effect, input message, and output message are shown there and no link to other editors of the Studio is exposed.</p>	<p>3</p>	<p>Ensure that users clearly see how to use their target services by adding options that enable different functionalities such as: execution, composition, consumption ... etc.</p>
<p>Upon invoking the consumption platform, several unmeaningful messages (info: suggestions retrieved etc) are shown at the bottom right corner of the Studio.</p>	<p>1</p>	<p>Remove such messages, which might confuse and distract user attention.</p>
<p>The consumption platform shows error messages at the bottom right of the Studio which might not be easily noticeable by users. It is also hard to make associations to the occurring problems.</p>	<p>2</p>	<p>The system feedbacks should be associated to the place where the problem occurs.</p>
<p>Dialog windows in the consumption platform, contain the close symbol (x) to left which is unnatural. </p>	<p>3</p>	<p>Place the close symbol (x) on the top right corner.</p>
<p>The maximize button changes its position from centre to right after a window is maximized and then minimized.</p>	<p>1</p>	<p>The maximize button should always appear at one position, better if it is in the centre.</p>
<p>Every time a category is selected from the categories tree in the consumption platform, a new search box is added to the main window below previous search windows. Therefore, selecting (n) categories would produce (n) search windows, which creates an unpleasant design and badly manages design space.</p>	<p>3</p>	<p>Instead, use only one search box to display search results. If the user selects a new category, simply update the existing search box. There is no need to create a new one.</p>
<p>In case users select a particular category, which has no services, an empty search window is shown to the user with no information.</p>	<p>1</p>	<p>Add meaningful messages / feedbacks to inform users about the results of their actions and give advice in case no services are found.</p>
<p>Service windows in the consumption platform, are not resizable. Users can only maximise them to full size. This hinders users from organising services in the manner they find most convenient.</p>	<p>1</p>	<p>Enable users to resize service windows to the size they desire. This will allow them to arrange services in the design space and make most use of it.</p>
<p>At the time of testing it was not possible to comment on, rate, or add found services to the list of favourites in the consumption</p>	<p>2</p>	<p>It is crucial for the Studio to enable users to share their experience by commenting, rating, and adding services to</p>

platform.		their list of favourites. This gives them the feeling of engagement and control.
Multiple selection of a particular service from the search window adds the selected service many times to the main window (i.e creates duplicate services).	3	The Studio should ensure no duplicate services are added to the main window of the consumption platform.
In the process editor, all top left icons, which are used for quick access to most used options, are labelled with the term “compose”. This does not signify their true functionality.	3	Annotate top left icons with text that reflect their true functionality.
In File->Open menu, the “Load” button in the ‘Load Model’ dialog window seems like static text, which might be confusing.	1	Ensure buttons have clickable characteristics and are easily differentiated from static text.
Entering a wrong file name and pressing the “Load” button in the ‘Load Model’ dialog window does not bounce back any error messages.	3	The Studio must notify users of any errors they make and offer clear ways of dealings with them.
In order to open a process model, users are requested to both select the file to open and enter its name. This is time consuming and error prone.	3	To open a particular process model, users should only need to select the target file.
It is not possible to close a particular process model.	3	Add a close menu item to the main menu to enable users to close unwanted process models
The menu “view” on the top left seems to serve no purpose.	3	All unnecessary elements should be removed from the interface.
If users want to open a process model, whilst another one is already open, no error message is shown to warn users. Moreover, the new process model is displayed on top of the existing process model.	3	The Studio must warn users about the possibility of losing their work and ask them to save it. If the user opens a new process model, the system should close the existing one to avoid confusing the user.
After opening a desired process model, it is not possible to move or edit its elements (such as: activities).	3	Empower users to manipulate loaded process models by editing, deleting or moving their elements.
Deleting activities and goals works	3	The use of delete button on

sometimes but not always.		the keyboard will be much easier way to implement this functionality
The processes in 'Favorites' tab do not expand in the centre window. Nothing happens when you click on them or drag them onto the centre window	2	The functionality should be implemented to allow users to re-use the processes saved in the 'Favourites' section.
'Help' button do not have any associated functionality	2	Relevant functionality should be implemented.
Simple editing like cut, copy and paste cannot be performed from the 'Edit' menu on the command bar	3	The functionality behind editing operations should be implemented
An incoming connector will only connect to the left hook of 'Parallel Split' but not with the top one.	2	The incoming connector should be allowed connection with the top hook of 'Parallel Split'

Table 3: Usability Problems in the SOA4All Studio and their Corresponding Design Recommendations, Probed by WP7 Scenario

4.2 Usability problems probed by WP8 Scenario

Usability problem	Severity rating (1-3)	Design recommendation
The profile creation option is hidden in a tree menu on the top-left corner of the screen, which is difficult to find by a first time user	3	Either toggle on the menu at all times clearly showing the profile editor options or provide an up-front button for profile creation for ease of use.
No comprehensive system support is provided to users to illustrate how a profile can be created, especially with the necessity to have an OpenID account. What is OpenID? No instructions are provided in the Studio.	3	Provide clear instructions and hints to users about how to create a SOA4All profile. This is very important to novice users and beginners.
Once users are logged into the Studio, no feedback is presented to indicate their status. Therefore, users would not know if they are still logged in the system. At certain times, this was confusing and errors of the Studio could be mistakenly linked to the possibility of "not being logged in".	3	At all times, the system should notify users whether they are logged into the system or not via a message which could be displayed on the Studio to show their status.

<p>No functionality is provided for users to log off the Studio.</p> <p>If users visit the profile editor, no greeting message or information regarding their status (logged in / logged out) is presented.</p> <p>User can not create personal profiles, which could be shared with other users.</p>	<p>2</p>	<p>Empower end users to log off from the Studio any time they want. It might also be worthwhile to enable them to create personal profiles which contain contact details, interests, a photo, ... etc. Such information becomes handy in collaborative development activities or could be used in the process editor.</p>
<p>In case users forget their log-in information (ID, password), there is no way to retrieve it.</p>	<p>3</p>	<p>In this situation, the Studio must ask users to supply their email address, to which their log-in details will be sent.</p>
<p>The Studio does not offer quick and easy way to navigate back to the home page, or navigate between pages of the Studio.</p>	<p>2</p>	<p>Add a link/logo that enables users to easily access the main/start page of the Studio.</p>
<p>Although the Studio is hosted within a web browser, back arrows of the browser are not working properly.</p>	<p>1</p>	<p>Ensure browser features such as back and forward arrows are fully operational. Since SOA4All Studio is hosted within a web browser, it is expected that users will very much depend on browser navigational features.</p>
<p>Extending the menu of classes, in the discovery platform, results in a long menu hindered by pre-condition / effect windows on the left hand side. Hence, menu items at the bottom become invisible and inaccessible to users.</p>	<p>2</p>	<p>Make sure the menu of classes is viewable always to facilitate user navigation. Either relocate the pre-condition / effect windows to the right hand side or add a vertical scroll bar to the menu of classes.</p>
<p>Meaningless names are given to services such as: service696, service32, etc. Users will not be able to understand the purpose or functionality of those services.</p>	<p>3</p>	<p>Use self-explanatory names whenever possible to represent services and development-related concepts. Small descriptions of the services that can be activated via specialised options may also be added.</p>
<p>Service discovery sometimes does not work i.e. after pressing the 'Search' button no results are displayed.</p>	<p>3</p>	<p>This can be a problem with the server but in this case there should be a message displayed with the appropriate content.</p>

<p>Too many technical details are shown to users in the right hand windows (pre-condition, effect, input message, output message), in the discovery platform. Users' understanding of such technical jargon is very much dependent on their background and knowledge. Whilst technical users are likely to understand the content of these windows, ordinary people will have no clue of their meaning.</p>	<p>3</p>	<p>Either remove the pre-condition, effect, input message, and output message windows if unimportant or use natural language to convey the embodied technical details to ordinary users.</p>
<p>Not sure how the pre-condition and effect search windows can be used to find particular services. No explanation or instructions are offered to Studio users.</p>	<p>3</p>	<p>Provide examples and help to show how to use the pre-condition and effect search windows on left side of the discovery platform.</p>
<p>It is not obvious how services that are found using the discovery platform can be used, since at the moment only information about service classification, pre-condition, effect, input message, and output message are shown there and no link to other editors of the Studio is exposed.</p>	<p>3</p>	<p>Ensure that users clearly see how to use their target services by adding options that enable different functionalities such as: execution, composition, consumption ... ect. A button for selecting a service would be helpful</p>
<p>It is not clear how users can start a composition of services; no support is provided (e.g. tutorials, help topics, etc).</p>	<p>3</p>	<p>Appropriate mechanisms should be implemented to enable system support for service composition</p>
<p>No wizards are supplied to assist users in matching or composing suitable services</p>	<p>3</p>	<p>Define wizards to assist users in finding and composing suitable and compatible services</p>
<p>In the process editor, all top left icons, which are used for quick access to most used options, are labelled with the term "compose". This does not signify their true functionality.</p>	<p>3</p>	<p>Annotate top left icons with text that reflect their true functionality.</p>
<p>The menu "view" on the top left seems to serve no purpose.</p>	<p>3</p>	<p>All unnecessary elements should be removed from the interface.</p>
<p>Deleting activities and goals works sometimes but not always.</p>	<p>3</p>	<p>Add prominent navigational options and activate mouse / keyboard to allow users to delete unwanted process model elements. The use of delete button on the keyboard will be much easier way to</p>

		implement this functionality
The purpose of the “data flow” / “control flow” buttons at the bottom of the editor is not clear.	3	Provide help text to clarify the purpose of these buttons and how they can be used by end users.
The functionality underneath ‘Binding’ button in not clear	3	A brief description of the functionality offered by this button would be helpful
When connecting activities in the process editor, it is only possible to connect the right hook of one activity with the left hook of another. Right hooks cannot be connected to right hooks of other activities located below them.	1	Use different symbols / colours to signify the start and finish points of process model elements (activities, parallel split ... etc).
It is not possible to change the names of activities and connectors.	3	The Studio should allow users to directly manipulate and edit details of process model elements.
It is not possible to close a particular process model.	3	Add a close menu item to the main menu to enable users to close unwanted process models
It is not possible to execute services in the process editor. Hence, users will not be able to test the results of their composition.	3	Implement this feature to empower users to constantly check their progress.


Table 4: Usability Problems in the SOA4All Studio and their Corresponding Design Recommendations, Probed by WP8 Scenario

4.3 Usability problems probed by WP9 Scenario

Usability problem	Severity rating (1-3)	Design recommendation
In the process model, the purpose of “data flow” / “control flow” buttons at the bottom of the editor is not clear.	3	Provide help text to clarify the purpose of these buttons and how they can be used by end users.
The functionality underneath ‘Binding’ button in not clear	3	A brief description of the functionality offered by this button would be helpful
When connecting activities in the process editor, it is only possible to connect the right hook of one activity with the left hook of	1	Use different symbols / colours to signify the start and finish points of process

another. Right hooks cannot be connected to right hooks of other activities located below them.		model elements (activities, parallel split ... etc).
In the process editor, it is not possible to change the names of activities and connectors.	3	The Studio should allow users to directly manipulate and edit details of process model elements.
At the current time, it is not possible to delete unwanted elements of process models (such as activities) from the design area.	3	Add prominent navigational options and activate mouse / keyboard to allow users to delete unwanted process model elements.
Some information, such as author and Date created, on the left hand side of the process editor have to be filled in by the user every time a new model is created.	1	These attributes can be automatically filled in by the Studio using data from the profile editor.
Icons at the top centre and top left of the process editor are annotated with the word "compose" which does not reflect their true functionality.	3	Re-annotate each icon with descriptions that reflect its job.
It is unclear what connections between activities mean. So, if two activities are connected together via a connector, what sort of relationship is there between the two entities?	3	The system should enable users to precisely describe the type of connections between different process model elements, being data flow, control flow ... etc.
When connectors are selected, the system does not clearly highlight the selection making it difficult to manipulate the connectors.	1	Clearly highlight user selection of process model elements using, for instance, a high contrast colour, so that further actions can be made easily.
The "Profile" button at the top right corner of the process editor does not invoke any system action.	3	If this button does not provide any added-value, consider removing it. Otherwise, implement its functionality.
Right clicks on particular items (e.g. activity, parallel split, etc) of the process model invoke browser's options. This design style is not a problem in itself but can be improved, especially with the fact that expert users often use right clicks to quickly access various functionalities.	1	It is more convenient if right mouse clicks invoke Studio options such as: "delete, rename, ... etc" to accommodate the needs of different users (i.e. novice and experts users).
If a user wants to save a particular process	3	For each process model,

<p>model, she is constantly asked to retype the desired name every time a save is requested.</p>		<p>users should be asked to provide a desired name one time only. A “save as” option should be added to the process editor to enable saving the process model under different file names if required.</p>
<p>If process model elements are added to the design space, followed by clicking the “close” button no error message is shown to warn users. This sort of system behaviour is dangerous since it may lead to losing existing development work.</p>	<p>3</p>	<p>Under no circumstances, users must always be warned in case they are facing to lose their development work.</p>
<p>Every time a category is selected from the categories tree in the consumption platform, a new search box is added to the main window below previously added search windows. Therefore, selecting (n) categories will produce (n) search windows, which produces an unpleasant design and badly manages design space.</p>	<p>3</p>	<p>Instead, use only one search box to display search results. If the user selects a new category, simply update the existing search box.</p>
<p>Clicking on the same category twice in the consumption platform, creates two redundant search windows in the main window.</p>	<p>3</p>	<p>Ensure that the search results are not shown twice to users.</p>
<p>In case users select a particular category, which has no services, an empty search window is shown to the user with no information.</p>	<p>2</p>	<p>Add meaningful messages / feedbacks to inform users about the results of their actions and give advice in case no services are found.</p>
<p>Service windows in the consumption platform, are not extendible. Users can only maximise them to full size (the size of the main window).</p>	<p>1</p>	<p>Enable users to resize service windows to the size they desire. This will allow them to arrange services in the manner they want.</p>
<p>It is not possible to comment on, rate, or add found services to the list of favourites in the consumption platform.</p>	<p>3</p>	<p>Allow users to comment on, rate, and add retrieved services to their list of favourites. Such feedback about services are invaluable to other services consumers.</p>
<p>Every time, a particular service is selected from the search window, it is added to the main window, creating duplicate services.</p>	<p>3</p>	<p>Ensure no duplicate services are added to the main window of the consumption platform.</p>

<p>In the monitoring platform, it is unclear how to assess the quality of the specific service, especially for novice users. Some attributes in service description, such as number of failed requests and Number of requests, could be meaningless to ordinary users.</p>	<p>3</p>	<p>The system should use understandable measures / ratings to enable users understand whether a service is of a high quality or not.</p>
<p>Close symbol of the monitoring widgets is on the left, which is uncommon. </p>	<p>2</p>	<p>Follow design conventions, where close symbols are on the most right corner.</p>
<p>In the WSMO-Lite editor, selecting the open “Service Description” or “Ontology” option from the File menu, followed by “List” button lists all potential directories. The user then has to navigate, using his knowledge, to the right directory and find the desired file. This is inefficient, wastes time, and error-prone (user may open the wrong file).</p>	<p>2</p>	<p>The system should show only relevant directories and files. For instance: if the user clicks on open “ontology” option and clicks “List” button, the system should report the available ontologies only, no other directories or files should be visible. Constraining user options by omitting non-relevant directories and files decreases the chance of making mistakes.</p>
<p>It is possible to open an ontology using the “open service description” dialog window, which is incorrect.</p>	<p>3</p>	<p>Ontologies should be opened only using “open ontology” dialog window.</p>
<p>“Quick Find” button and Ontologies Search section at the bottom left of the WSMO-Lite editor serve no purpose at the moment.</p>	<p>2</p>	<p>Either implement these features, or remove them if they are not meant to offer any functionality.</p>
<p>It is possible to create multiple bindings between same ontology element and service element.</p>	<p>2</p>	<p>The Studio should disallow duplicate bindings of the same elements.</p>
<p>Whilst it is possible to bind ontology concepts to service elements (e.g. parameters of a message), there is no way of knowing whether the created bindings are practical and useful or not.</p>	<p>3</p>	<p>The system should support the creation of meaningful bindings by providing examples, hints, an automatic mappings whenever possible. This will minimise user efforts and reduce the number of potential wrong bindings.</p>
<p>Closing an existing service description or opening a new one whilst another one already exists, does not warn users of the</p>	<p>3</p>	<p>Whenever there is a chance users might lose their development work, the</p>

WSMO-Lite editor. Thus,there is a possibility of losing work.		Studio must warn them and confirm whether they want to proceed with their actions.
The “close” menu option clears only the content of the main window. It is not possible for users to clear the content of the Semantic Models section. It is not possible to close an existing ontology.	3	Ensure users can close an existing ontology if they desire to through navigational options or a close symbol. Add an option to empower users to delete/close the content of the Semantic Models section.

Table 5: Usability Problems in the SOA4All Studio and their Corresponding Design Recommendations, Probed by WP9 Scenario

5. Conclusions

The current heuristic evaluations have focused on inspecting usability issues within M22 prototypes and software artefacts of the SOA4All project. For this purpose, three different scenarios embodying typical user actions were used to guide the evaluation procedure. It is worth noting that not all actions of the test scenarios (of WP7, WP8, and WP9) were investigated since many features and functionalities of the Studio still need to be completely implemented. Hence, the evaluation efforts primarily concentrated on the available functionality of the profile editor, consumption platform, composition editor, monitoring platform, and annotation editor. The evaluation results presented in this document highlight a number of issues related to the usability and functionality aspects of the Studio and in order to address these issues, the evaluation experts have made appropriate recommendations which have been included with the evaluation results.

6. References

- [1] Morgan, D. L. 1997. Focus Groups as Qualitative Research. California, Sage Publications
- [2] Namoune. A, Wajid, U. Mehandjiev, N. Composition of Interactive Service-based Applications by End Users. UGS2009 - 1st International Workshop on User-generated Services
- [3] Nielsen, J. 1993. Usability Engineering pp.214-216, Academic Press
- [4] Sutcliffe, A. G. And Ryan, M. 2000. Model mismatch analysis: Towards a deeper explanation of users' usability problems. Behav. Inf. Tech. 19, 1, 43-55
- [5] Jakob Nielsen. Ten Usability, from http://www.useit.com/papers/heuristic/heuristic_list.html
- [6] Gerhardt-Powals, J. 1996. Cognitive Engineering Principles for Enhancing Human-Computer Performance.s International Journal of Human-Computer Interaction 8 (2): 189–211

7. Appendix

The enclosed paper was presented in the 1st International Workshop on User-Generated Services (ICSOC 2009), Stockholm, Sweden.

Composition of Interactive Service-based Applications by End Users

Abdallah Namoune¹, Usman Wajid¹, and Nikolay Mahendjiev¹,

¹ Manchester Business School, Booth Street West,
Manchester, M15, United Kingdom
{abdallah.namoune, nikolay.mahendjiev, usman.wajid }@mbs.ac.uk

Abstract. In this paper, we investigate web users' mental models of services, the underlying risks and benefits of service composition, and the problems anticipated while combining web services into final interactive applications. The study comprised three focus groups integrating group discussions and questionnaires, with a total of 35 participants, the majority without specialist programming skills. The results of the focus groups revealed a high degree of optimism towards service composition and consumption. However, several concerns, primarily related to personal privacy, trust, and technical difficulty, were highlighted during the focus groups. This paper discusses these concerns and proposes some ideas about how to address them.

Keywords: Web services, service composition, end user development, service-based applications.

1 Introduction

Service Oriented Architecture (SOA) technologies are becoming very popular on the Internet, especially in the form of independent services [1]. Their key benefit is reuse, indeed existing web services can be loosely coupled to produce new composite web services through the so called process of "service composition". Whilst only a small proportion of users, often with considerable computing knowledge and programming skills, can construct complex service based applications, the majority of online users are unable to exploit the advantages offered by SOA technologies and develop service-oriented applications tailored to one's needs. This difficulty can be linked to the complexity of the composition process which is carried out using advanced composition languages, and to the limited technical knowledge of ordinary users. In this respect, the research challenge lays in simplifying the composition process so that various services can be combined into interactive applications, and abstracting this process from unnecessary technical complexity. Such research promises to promote the consumption and reuse of web services, especially by ordinary web users. When creating such user-friendly service composition interface, we also need to consider user expectations regarding the trade-off between the costs of learning new tools and the benefits they expect to get from using them. For example, the spreadsheet interface hides aspects such as order of calculations and propagating updates, and minimises learning costs by using familiar metaphor of calculation tables and accounting books. The balance between costs and benefits is likely to differ for different groups of users and different target domains (e.g. [8,10]), yet we believe that identifying user attitudes and expectations towards service composition is a key to predicting successful uptake [8,10,11], hence it is the focus of the study reported in this paper.

Currently, end users can add web services as widgets/gadgets to their personal pages in a lightweight manner; this is particularly relevant to networking websites such as: Facebook [2] and personalized homepages such as: iGoogle [3] and myYahoo [4]. Users of these websites can select from a list of services and position them on their personal pages. The services are visually represented as independent windows and the users can interact with these services and customize their look and that of their personal pages. Although the widget-based model is simple and enables hosting different services together, it does not support service composition. Indeed, the web services, represented as widgets, are autonomous and do not interact with each other, thus restricting their usefulness for creating more complex assemblies. For instance, given a flight service, a car service, a hotel service, a card payment service, and an insurance service, users should be able integrate them to form a mini holiday organizer application. Service composition not only fulfils users' needs but also allows easy extension and customization of applications; thus, saving considerable time and resources.

Another advanced and rich approach to end user development of applications follows the mash-up based model. In this particular case, end users combine existing services and web feeds from multiple sources into a single web-based application using specialized mash-up editors, such as: Open Mashups Studio [6] and Yahoo!Pipes [5]. The major drawbacks of this approach relies in, firstly, the modelling skills needed to understand the data flow between services and secondly the strong emphasis on data aggregation while giving less importance to functionality aggregation.

Whilst the mash-up based model is complex and lacks flexibility, the widget-based model does not support any interaction between services offered by different service providers, This motivates the pressing need for more effective approaches to compose low-level services into interactive service-oriented applications by non-programmers. Easy to use and flexible service composition authoring tools that simplify the composition process should be offered. This is the main objective of the EC funded project, SOA4All [7].

Here we report on a study which aims to identify the balance between user expectations about costs and benefits of the SOA4All vision, and to chart users' concerns and background as relevant to this vision. It is worthwhile to note that this paper focuses on service composition and consumption by human actors and not by software agents. Focus groups were used as a self-contained method to conduct this study since no suitable prototype was available to evaluate at that stage. Focus group is an efficient technique used to collect qualitative data and generate concentrated information on a specific topic. It is argued to be better than user observation and individual interviews owing to the group interaction which provides detailed insights into opinions and experiences of participants [18].

This paper is organized into the following sections: Section 2 reviews the latest work on service composition. Section 3 provides a short description of the SOA4All project. Section 4 details the procedures carried out in the focus groups. Section 5 reports the findings of this research study. Section 6 presents a discussion about the findings and suggests various solutions to encounter the highlighted problems. Finally, Section 7 summarizes the paper.

2 Service Composition by End Users

Service Composition is broadly supported by two main approaches: workflow-based scripting of service components, and AI-based automatic composition of service components, reasoning with pre- and post-conditions. Further details are available elsewhere [13, 22,233].

A large number of visual representations for service composition and interaction have been proposed with the purported aim to make the composition more user-friendly (e.g. Zenflow [14]). However, most of them are *ad hoc*, i.e. they use technology-led representations and metaphors, which are not derived from user studies. Only a few of them have been evaluated in terms of usability and cognitive effectiveness. For example, Lets Dance [15] has been evaluated using the framework of Cognitive Dimensions [**Error! Reference source not found.**], but iterative testing and enhancement have not been documented in the related references. The framework of cognitive dimensions contains 14 principles describing aspects that are relevant to cognition [17]. It aims to evaluate the usability of interactive information artefacts (e.g. software applications) and non-interactive information artefacts (e.g. notations, programming languages) by non-specialists. Vitabal WS [16] is a version of an earlier visual language tuned to the needs of web service composition. It has been evaluated using the cognitive dimensions framework, yet it targets experienced web service developers and hence would have different characteristics from the service composition representations to be developed by SOA4All.

We believe that technology-led *ad hoc* visualizations will not work. Indeed opening up service use and development to people who are not professional programmers (we call them end users) requires the delivery of user interfaces that are task-oriented rather than technology-oriented, that is they should be tuned to the expected skills and foreseen tasks of our target users. Activities such as service construction and composition will involve non-trivial problem-solving in a context called *End User Development (EUD)* [12]. EUD research results provide an insight into the type of software interfaces

and motivational factors likely to support end user activities.

Sutcliffe *et. al.* [11] see the trade-off between expected benefits and learning costs as a main determinant of uptake of an End User Development tool by its users. This has been extended to organizational context by Mehandjiev *et. al.* [10], who identify a number of risks and benefits for end users being involved with the development of software, including the construction of software services. These factors have then been used to underpin a number of quantitative studies in concrete domains, aiming to elicit the likelihood of uptake for end user development ideas in the specific context of that domain (e.g. [8]). The workshops reported here are an example of one such application of this approach to the target domains of SOA4All.

Several research studies have attempted to explore end user perception of software development, for example: McGill and Klisc [19] argue that end user developers of web development are aware of the associated risks and benefits and it is crucial to involve them in the development of approaches to minimise risks. Due to the difficulty of learning traditional programming languages, Myers *et. al* [20] report a number of studies aiming to elicit understanding of how people think about a particular task and design natural programming languages and environments that support the way end user developers are thinking. The generated data about user behaviour is used to build intuitive and usable programming environments. More recently, Namoune *et. al* [21] report on a user study in which potential problems of service composition are extracted when using a visual composition tool (although at its early stages of development). The main findings show that end users have difficulty connecting services together and understanding specialised service- related terms such as: operations, parameters, data types. Overall, review of available literature demonstrates that research in end user development of service based applications is very rare and most studies are in their infancy.

3 SOA4All

The research presented in this paper is part of the ongoing work on SOA4All, which is an EC-funded project that focuses on acquiring end user perception of web-services, and then using this vital information to develop sophisticated tools and techniques that can enable end-users from a variety of background to use web-services. In this respect, SOA4ALL aims at opening up services to the scale and accessibility typical for the WWW. On the technical front it includes the use of Web2.0 principles and state-of-art techniques for semantically tagging, retrieving and composing services. The developments on the technological front will result in addressing the specific needs of end users and allow them to implement innovative business models in order to address niche markets.

In order to support the entire service lifecycle (service discovery to service consumption) SOA4ALL intends to provide a coherent and domain independent platform where a massive number of parties can expose and consume services. To facilitate in the development of such a platform, research within SOA4ALL involves clarifying the requirements as to how end users from a variety of backgrounds can not only interact with individual services but also compose different services to achieve their desired objectives. The requirement gathering process is realized by several end user studies (focus groups) and the results of some of these studies have been reported here.

The results obtained from the focus groups give a holistic view about the perception of target end-users. These results will be fed into to the design of SOA4ALL studio. SOA4ALL studio is envisioned as a rich web-based platform that will provide users with a unified view covering the whole lifecycle of services, including design-time, run-time and post-mortem analysis. It will provide the starting point for end-users that get in touch with SOA4ALL. In essence, the SOA4ALL studio represents a set of components to facilitate the composition of web-service based applications for novice users. The functionality offered by the studio will automatically help the end-users with the selection and placement of related web-services within the user interface.

The high-level view of SOA4All architecture is shown in Figure 1 (below):

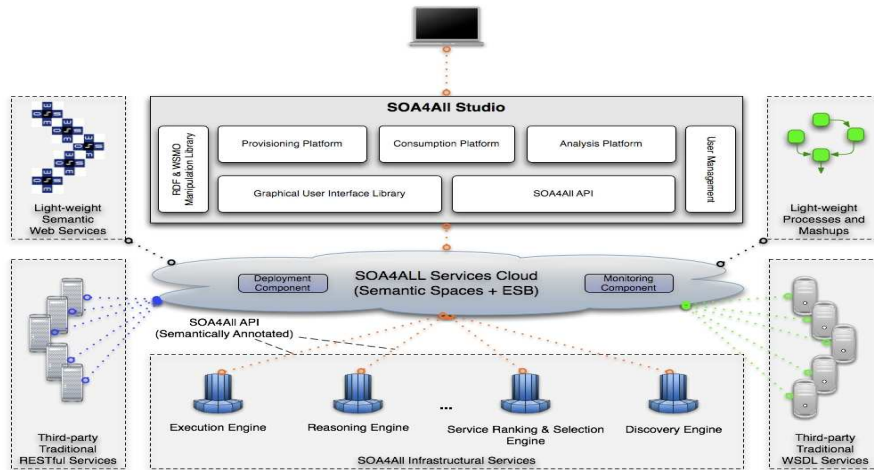


Figure 1: High level description of the SOA4All architecture.

4 Methodology

Three separate focus groups, involving 35 participants without programming skills (25 students and 10 academic and research staff) (range 19 to 40 years with a mean of 26 years) were undertaken within the Centre for Service Research at the Manchester Business School to acquire a better understanding of end users perception about web services, and the likelihood of uptake of user development. Each focus group lasted for approximately one hour; participant responses were recorded using audio recorders and questionnaires. The overall strategy was to first introduce participants to the topic of web services composition by end users through a presentation, followed by capturing their subjective judgment about the topic through a questionnaire, and finally discuss several issues in small groups. All participants were invited to perform these tasks:

- 1- Provide a definition of web services
- 2- Listen to a 20 minute presentation in which they were familiarized with web services and the concept of service composition; this was facilitated by examples
- 3- Fill in a service composition questionnaire
- 4- Discuss the potential risks and benefits of service composition and anticipate the composition-related problems; this was carried out in small discussion groups containing 5 participants each
- 5- Propose solutions to resolve the highlighted problems

4.1 Service Composition Questionnaire

The service composition questionnaire used in our study contains three main parts, as follows:

Part 1.

- My experience with Service Composition is (none 1-2-3-4-5 expert)
- I find web service composition interesting (disagree 1-2-3-4-5 agree)
- Please list the Service Composition languages and systems you are familiar with (or circle these examples: iGoogle, Facebook, Yahoo!Pipes, BPEL4WS, BPML, BPSS, OWL-S, WSCI, WSCL, WSFL, Semantic Pipes)
- How often do you compose services or build service based applications (daily – weekly – monthly – less often - never)
- What are your favourite service composition languages or systems?

Part 2.

Service composition by users (SCU)

- Is useful (disagree 1-2-3-4-5 agree)
- Is easy to achieve (disagree 1-2-3-4-5 agree)
- Brings about a more efficient way of conducting on-line activities (disagree 1-2-3-4-5 agree)
- Is unfeasible (disagree 1-2-3-4-5 agree)
- Is error-prone (disagree 1-2-3-4-5 agree)
- Can be used to break organisational rules and policies (disagree 1-2-3-4-5 agree)

Part 3.

Please tell us your opinion about the following ways of encouraging and supporting Service composition by users (SCU)

- Examples of successful SCU can stimulate one to try it (disagree 1-2-3-4-5 agree)
- Recognising and rewarding SCU effort will make people more willing to try it (disagree 1-2-3-4-5 agree)
- Attending a training course could help people to start SCU (disagree 1-2-3-4-5 agree)
- SCU quality standards and testing will decrease risks (disagree 1-2-3-4-5 agree)

Although the questionnaire contains some questions which are difficult to assess at this stage, for example, it is practically hard to assess whether “composition is easy to achieve” without actually trying it, the principal aim was to drive first impressions about service composition and most importantly to check users’ acceptability of this innovative idea. In addition, the results will provide a reference point to advanced evaluation stages when end users perform composition using our composition authoring tool.

4.2 Introductory Presentation

The introductory presentation “The Internet of Services”, presented by one of the authors, aimed to introduce the concept of service and provide examples of service composition. It explained the difference between conventional services, software services and hybrid services, where human-performed services are enabled through software interfaces and services, such as buying a book through Amazon.com. The influence of current Web2.0 technologies was argued to enable end users to take part in the development of the web, and the idea is to move this influence to the internet of services. Following this, Yahoo! Pipes was used as a motivating example (Figure 2). Figures about the number of web services found were also reported (27.684 services and 7284 providers during the last 2 years), as suggested by the SEEDKA service crawler. Next, the motivation behind SOA4All was introduced to the attendees, with the project aiming to transform the current web of information into a web of services through which users of services could also become producers of applications, or what we call “Prosumers”.

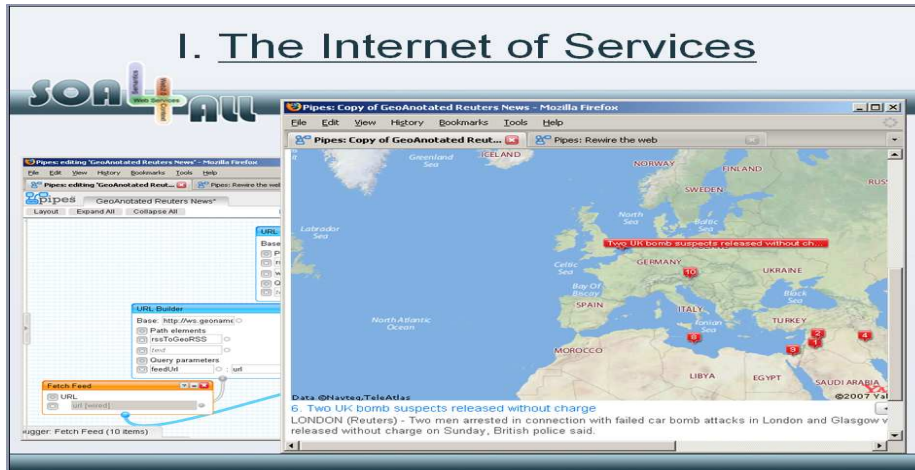


Figure 2: Yahoo! Pipes as a Stimulating Example

Then the scenario driving further discussions was introduced, the creation of a *Meet Friends* composite service. This hypothetical composite service allows a particular user to organise a meeting with friends at short notice. The Meet Friends composite service contains four services; service one fetches the address of friends from social networking sites (e.g. Facebook), service two finds out which friends are in the vicinity of the target venue, service three finds out weather and travel information for proposed meeting venue from a 3rd party, and service four sends out invites and directions using an SMS service. Finally, the presenter showed some mockups of a future authoring service composition tool (Figure 3). Participants were invited to ask questions related to aspects of the presentation before starting the focus groups.

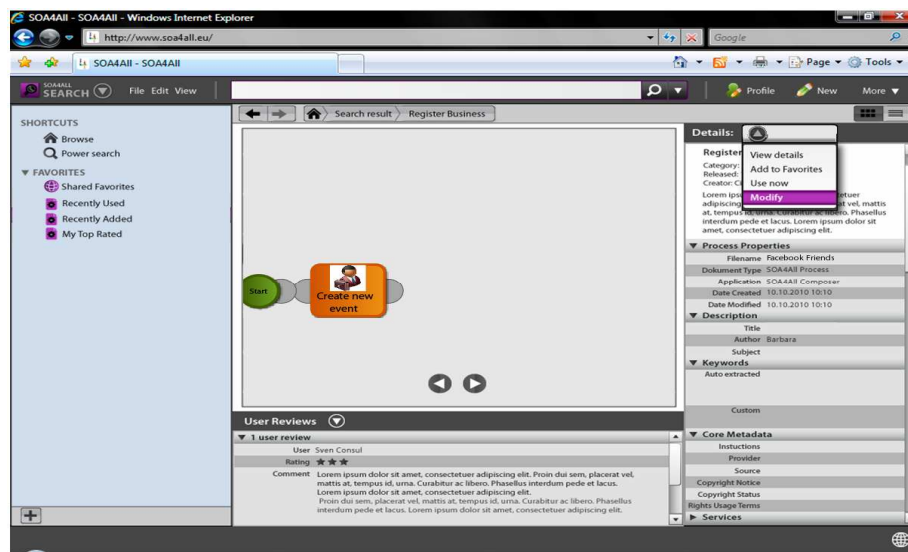


Figure 3: A Mockup of the SOA4All Studio – a user-friendly composition tool under development in SOA4All

5 Results

The results of the three focus groups undertaken are divided into three main themes, as follows:

5.1 Web Services and Service Composition Perception

The pre-test questionnaires revealed that more than 85% of the participants considered themselves as

not experts in terms of software and service development. 60% of the users specified that they have “never or less often” composed services or built service based applications. The qualitative analysis of the responses gathered in the focus groups showed that 25 user comments relate to service understanding. The results demonstrated diverse user understanding/definitions of services; these definitions varied between: features assisting users, solutions to issues, components of business process, offerings to customers, information provision, and execution of transactions. In general, users’ definitions concentrated on two main aspects, (1) describing attributes/features of services such as: services are intangible and they have a back end, (2) describing specific interactions with users in the form of service consumption, such as: providing users with information, helping users, and delivering expertise.

Table 1. Service composition questions, rated between (1= disagree and 5= agree)

Service composition by users	Mean answer	SD
... I find web service composition interesting	4.20	0.76
... is useful	4.44	0.82
...brings about a more efficient way of conducting on-line activities	4.12	0.96
...is easy to achieve	3.32	1.19
... is unfeasible	2.26	1.18
... is error-prone	2.54	0.87
... can be used to break organisational rules and policies	3.50	1.08
Ways of encouraging and supporting Service composition by users		
Examples of successful SCU can stimulate one to try it	4.69	0.52
Recognising and rewarding SCU effort will make people more willing to try it	4.15	0.90
Attending a training course could help people to start SCU	4.38	0.77
SCU quality standards and testing will decrease risks	4.32	0.76

When asked whether service composition is interesting, 80% of users showed a high level of interest (mean = 4.20 /5, questions were rated on a five-point Likert scale where 1 corresponds to disagree and 5 corresponds to agree). Users also rated the usefulness of service composition high (mean = 4.44 /5), as well as the efficiency of service composition in promoting the accomplishment of online activities (mean = 4.12 /5). However, service composition by end users was regarded neither easy nor difficult (mean = 3.32 /5). In terms of error-proneness, fears were evident about the possibility of creating errors by ordinary web users (mean = 2.54 /5). Users concerns that relate to disruptive use of service composition (i.e. service composition can be used to break organizational rules and policies) were rated high (mean = 3.5 /5). Finally, 77% of the users disagreed or remained neutral in regards to the question: “service composition by users is unfeasible” (mean = 2.26 /5).

In regard to user support, users agreed that successful examples (mean = 4.69) and training courses (mean = 4.38) could encourage people to be actively involved in the composition of services and development of service based applications. In summary, end users demonstrated a high level of interest and strongly agreed that service composition is useful and possible, but expressed uncertainty about the difficulty and potential misuse of service composition by the general public (Table 1).

5.2 Risks and Benefits

The discussion about the balance between risks and benefits is based on work [8-11] explaining the uptake of software development by end users (known as End User Development) as a rational economic decision based on the balance of perceived costs and perceived benefits of each user. The ongoing program of research in this area aims to analyse the factors which impact this perceived balance, and to discover organizational and technical strategies which aim to tip the balance in favour of the benefits, thus supporting the uptake of such technologies.

In terms of benefits, discussions in the focus groups mainly focused on the usefulness of reusing composition knowledge (40% out of all benefit responses), and the time users can save as a result of this (30% out of all benefit responses). Giving ordinary users control over service composition would empower them to produce various service oriented applications that can be tailored to their needs (15% out of all benefit responses), such as meta-search engines, thus saving them time and enabling them to obtain rich results.

In terms of risks, the biggest fear was about losing control over personal information (8% out of all risk responses), especially when the effect is mediated through the effect of social interactions (e.g. your friends exposing information about you), or through the service provider (information aggregator), which may pass your personal information (e.g. phone number) to other sub-contracting services, which may or may not be bound to the data protection principles. Technical difficulty imposed by service compose was also amongst the biggest fears of end users (8% out of all risk responses). Errors in putting information together were also possible, especially when the composition is performed by inexperienced users and un-trusted third parties.

Moreover, users felt that services may no longer be there when they need them, and that any recommendation support for services may be biased to a set of services.

The participants also discussed what could be the social and organisational support for user-based service development. The following ideas emerged:

- “Go with the flow” – once everybody is doing it, people will join, mirroring success in other technologies;
- Non-trivial examples of successful use will also help (to sell benefits), this was felt quite strongly;
- Community-level control mechanisms such as feedback, etc. would ensure validation of services and, together with a validating body/watchdog may help to ensure the trust, which is considered vital for uptake of user-driven service composition.

5.3 Composition Problems

Although users favoured the idea of assembling services to formulate interactive applications that fulfil their daily needs, several service composition-related issues were raised, in particular:

- Services complexity: services are usually represented using their functional elements (operations and parameters) which are often not understood by ordinary web users.
- Services compatibility: users expressed frustration in regards to aggregating heterogeneous services from different service providers. How do they ensure the business services they are trying to combine together are technically compatible with each other?
- Composition steps: users agreed that it might be problematic to define the single steps required to combine services together and the order in which these services should be executed due to their lack of technical knowledge and skills. This issue becomes more complicated in the case of many services (for example: 100 atomic services).
- Other less aggravated user interface-related concerns evolved around the use of the service composition editor, for example: direct manipulation of web services (i.e. selection, deletion, etc) within the design space could be the main source of frustration.

In terms of technical support which can be provided by the composition editor, the following themes emerged:

- The difference between naïve and professional users was felt to lie partially in the awareness about the consequences of one's actions; this awareness should be supported;
- Full automation such as Google search results will frustrate owing to lack of control by the end users, a balance should be maintained;
- Tools should offer clarity of process in respect to building and using;
 - Context and personalization;
 - Reuse of designs.

6 Discussion

End users with no or little computing knowledge showed either no or basic knowledge of the technical aspects of services, i.e. they could not provide a technical definition of services. This result is expected as our target group has no specialist technical skills. Essentially, they perceived services as elements which deliver services (be it information, help, solutions ... etc) to accomplish specified users goals. This view emphasises that services need to be abstracted from their technical complexity and presented in a way that efficiently describes their purpose/functionality, especially for ordinary web users.

Users showed a high likeability towards the idea of composing services into personalised interactive applications. This agrees with the current trends that end users are becoming proactive about developing the web. Users argued that service composition will save them time and enable them to develop applications on the fly and without the need to acquire considerable technical knowledge. Hence, it is important that end users are able to develop service-based applications without the need to learn programming languages and modelling notations.

To overcome the aforementioned problems, various tentative remedies that will form the functional requirements of a future visual service composition authoring tool –currently under development - are proposed in this section:

Promote service composition awareness: even though web users have experience adding autonomous services to their networking or personalised sites, the composition of services imposes a totally new and different challenge. Therefore, the composition editor should clearly communicate “the composition aspect” of services. Users’ awareness of the possibility to develop service-based applications should be elevated via the right amount of publicity to familiarize ordinary people with SOA technologies.

Simple service composition: this research aims to increase service reuse by ordinary users, it is therefore crucial to simplify service composition by hiding the technical aspects of services from users. Composition should be as easy as dragging and dropping a service into a design space, followed by creating connections between the selected services. No programming knowledge or expensive training should be required.

Guided service composition: users should be supplied with wizards, tutorials, and help messages to guide them through the composition process within an easy to use composition tool. This is particularly important to overcome the services compatibility and composition steps definition problems.

7 Conclusion

This paper reports on the results of three focus groups aiming to gauge end users’ perception of web services and their acceptability of service composition. Generally, users showed a high willingness to develop interactive service-oriented applications, but expressed fears that relate to the complexity

underlying the composition process and to the knowledge required to build software applications. In future research, various composition design approaches of different complexity levels will be offered to accommodate end users with various skills and backgrounds within an easy to use online authoring tool, formally known as SOA4All studio.

Acknowledgments. This research work is supported by the EC funded project SOA4All. We would like to thank the students and academic and research staff of the University of Manchester for taking part in this study.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures, and Applications. Springer Verlag (2004)
2. FaceBook, <http://www.facebook.com/>
3. iGoogle, <http://www.google.com/ig>
4. MyYahoo, <http://my.yahoo.com/>
5. Yahoo! Pipes, <http://pipes.yahoo.com/pipes/>
6. Orange Labs, Open Mashups Studio, <http://www.open-mashups.org/>
7. SOA4All, <http://www.soa4all.eu/>
8. Mehandjiev, N., Stoitsev, T., Grebner, O., Scheidl, S., Riss, U.: End User Development for Task Management: Survey of Attitudes and Practices. In Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing. Herrsching am Ammersee, Germany. IEEE Press. ISBN : 978-1-4244-2528-0 (2008).
9. Fischer G., Giaccardi E., Ye, Y., Sutcliffe A.G., and Mehandjiev N.: Meta-Design: A Manifesto for End-User Development. Communications of ACM, a Special Issue on End User Development (2004).
10. Mehandjiev, N., Sutcliffe, A., Lee, D.: Organisational View Of End-User Development, in H Lieberman, F Paterno, and V Wulf, eds, End User Development, Human-Computer Interaction Series , Vol. 9, XVI, 492 p., Hardcover ISBN: 1-4020-4220-5 (2006)
11. Sutcliffe, A., Lee, D., Mehandjiev, N.: Contributions, Costs and Prospects for End-User Development, Proceedings of HCI International, Lawrence Erlbaum Associates, Inc. New Jersey, USA (2003)
12. Sutcliffe, A. and Mehandjiev, N. 2004. Introduction: Special Issue on End User Development. The Communications of ACM, 47, 9, 31-32. (2004)
13. Jinghai R, Xiaomeng Su. A Survey of Automated Web Service Composition Methods by: Semantic Web Services and Web Process Composition, Vol. 3387/2005, pp. 43-54 (2005)
14. Martinez, A., Patino-Martinez, M., Jimenez-Peris, R., and Perez-Sorrosal, F. ZenFlow: A Visual Web Service Composition Tool for BPEL4WS. In Proceedings of the IEEE Symposium on Visual Languages and Human-Centric Computing. VLHCC. IEEE Computer Society, Washington, DC, 181-188 (2005)
15. Johannes Maria Zaha, Alistair P. Barros, Marlon Dumas, Arthur H. M. ter Hofstede: Let's Dance: A Language for Service Behavior Modeling. OTM Conferences (1) 145-162 (2006)
16. Li, Karen Na-Liu. Visual Languages for Event Integration Specification. PhD Thesis, University of Auckland, Department of Computer Science (2008)
17. Thomas Green, T., Blackwell, A.: Cognitive Dimensions of Information Artefacts: A Tutorial, <http://www.cl.cam.ac.uk/~afb21/CognitiveDimensions/CDtutorial.pdf> (1998)
18. Morgan, D. L.: Focus Groups as Qualitative Research. California, Sage Publications, (1997)
19. McGill, T. and C. Klisc. End User Perceptions of the Benefits and Risks of End User Web Development. Journal of Organizational and End User Computing 18(4): 22-42 (2006)
20. Brad A. Myers, John F. Pane and Andy Ko, Natural Programming Languages and Environments. Communications of the ACM. (special issue on End-User Development). Vol. 47, no. 9. pp. 47-52 (2004)
21. Namoune, A., Nestler, T., and Angeli, A.D. End User Development of Service-based Applications. 2nd Workshop on HCI and Services at HCI 2009 Cambridge, (2009)
22. Hoffmann, J., Bertoli, P., and Pistore, M. Web service composition as planning, revisited: in between background theories and initial state uncertainty. In Proceedings of the 22nd National Conference on Artificial intelligence - Volume 2. A. Cohn, Ed. Aaai Conference On Artificial Intelligence. AAAI Press, 1013-1018 (2007)

-
23. Papazoglou, M. P., Traverso, P., Dustdar, S., and Leymann, F. Service-Oriented Computing: State of the Art and Research Challenges. *Computer 40 (11)*, 38-45. DOI= <http://dx.doi.org/10.1109/MC.2007.400> (2007)