Abstract— Although Service Oriented Architecture technologies are becoming widespread, user research to support the exploitation and composition of services by ordinary Internet users is still in its infancy. This paper presents three user studies conducted during the user-centered design of the ServFace Builder, a service composition tool for non-programmers. Results suggest that end users do not realize that services can be connected together and do not easily understand that information can flow between services. Further conceptual and usability problems of service composition for end-users and some guidelines to overcome them are discussed.

Keywords-component; lightweight service composition; user requirements; usability; design recommendations.

I. INTRODUCTION

According to Seekda1, a service monitoring and discovering company, the number of online services is growing steadily and has already exceeded 28,000 services from various providers. It is predicted that it will reach hundreds of thousands of services in the near future. The ability to create value-added services by extending their functionalities is one of the central contributions of Service Oriented Architecture (SOA). Yet, the composition process (i.e., the process of wiring services together to form augmented service assemblies) is still a complex, time consuming, and error prone task even for experienced programmers [24]. Despite some efforts into simplifying traditional methods of composition, which rely on XML editors and composition languages (e.g. BPML [3], BPEL4WS [5]), by enriching them with graphical and usable tools (e.g., ZenFlow [18]), ordinary Internet users still find the composition task challenging ([13], [21]). As a matter of fact, service research so far has mainly concentrated on engineering and technical issues of SOA [1]. Current solutions, although perfectly functional, are difficult to use by ordinary Internet users, who are unlikely to have the computation knowledge and skills required by them. The complexity of service development environments has to be reduced to empower end users and motivate them to build services just like they proactively develop the web (e.g., wikis, social network platforms). In order to transform Internet users from passive consumers into producers of services (i.e. prosumers), research should be directed towards understanding the requirements of these target users and identifying ways to simplify service composition.

This paper reports selected results from 3 user studies performed during the user-centered design of the ServFace Builder, a composition tool for non-programmers. Our research builds upon a large methodological and theoretical apparatus from Human Computer Interaction (HCI) and the emerging field of end-user development [15], with the aim of achieving a rich knowledge of ordinary end users’ mental models, opinions, and expectations about SOA. The paper contributes a methodology for user research in service research, an example of a composition tool for ordinary Internet users, and initial guidelines for the design of service composition tools. Section II reviews related work on service composition. Section III introduces the ServFace methodology and its rational. Section IV, V, and VI reports the user studies. Finally, section VII discusses the findings and introduces a set of design guidelines.

II. BACKGROUND AND RELATED WORK

In recent years, Internet users have demonstrated a keen interest in developing the web through different tools, such as wikis, blogs, video-sharing, and social-networking sites. Customisable web-portals, such as iGoogle and MyYahoo are also gaining momentum. They enable end-users to create web-pages where they can incorporate information feeds and gadgets (i.e. programs that provide services). These portals are easy to use but they do not support the creation of complex software applications because services cannot be combined to each other. The Web 2.0 enables users to develop the content of the web but it does not support the creation of advanced applications, which consist of web services interacting together [23].

Even though service composition is well-covered by existing approaches for technical developers (e.g., BPEL [3]), tools and methodologies for enabling end-user composition have been largely ignored [24]. Empowering ordinary people to create and customize their own applications can be an interesting way forward to accomplishing daily goals in a more efficient way. Namoune et al. [19] proved this point through a series of focus groups to discuss benefits and risks of end-user composition of service-based applications. Overall, people demonstrated an interest in the idea, but they were worried about the privacy and security of their personal data, as well as of the technical complexity of the task.

---

1 http://webservices.seekda.com
Promising approaches for a lightweight user-driven application design are Mashup platforms (for an overview see [10], [31]), like Yahoo! Pipes or Open Mashup. They enable the creation of sophisticated service-based applications by aggregating web feeds, web pages and web services from different sources. Their graphical composition style represents a first step towards user empowerment and shifts the creation of applications from programmers to domain experts. With these platforms, users can define relationships between modules by dragging and linking them together within a visual editor. Mashup platforms mainly focus on data aggregation; they require computing skills and good understanding of programming concepts such as that of message passing which most users do not have ([21], [25]).

Only a limited number of studies have involved actual end users in the evaluation of mashup platforms such as [13], [21], [27], [29], [30]. However, design guidelines which are informed by empirical evidence to support the design of effective mashup platforms and service development environments are yet to be realized. The challenge to service engineering methodology for an integrated development of user interfaces for a composition of services. It investigates this process from two perspectives: (1) the development of services with corresponding user interface descriptions, and; (2) the development of user interfaces for a composition of services. The methodology articulates on three sequential steps.

Service annotation: existing services are annotated with user interface descriptions, which specify the presentation of their elements (e.g. human readable labels for fields, grouping of parameters), behavior (e.g. client-side validation rules, suggestion of input values), and the relationship between elements (e.g. semantic data type relations). These annotations are based on a formally defined meta-model, can be edited via a specialized graphical tool, are saved into a separate XML file and stored in a repository [11].

Service composition: the annotated services are combined using a visual composition editor, by which relationships between user interface elements of different services are defined “at the presentation layer” [9], [11]. The output of the service composition phase is a model containing a consolidated user interface representation and the business logic of the application to be designed.

Runtime generation: an executable application is generated automatically from the Composite Application Model [26].

A. From Service Annotations to Service Frontends

The vision of lightweight service composition emerged through the development of the ServFace Builder, a web-based authoring tool for ordinary Internet users. This graphical tool follows the approach of service composition at the presentation layer [12], [11]. Applications are built by composing web services based on service frontends [22]. A service frontend represents a single operation of a WSDL-described web service in the form of a generated UI component. The frontend comprises a nested container structure, which includes UI-elements (e.g. labels and text fields) that are bound to the relevant service operation parameters. During the design process, each web service operation is visualized as a UI component and can be connected to other web service operations following a graphical direct manipulation approach. The end user creates applications in a WYSIWYG (What you see is what you get) environment without writing any code.

In the ServFace Builder, single services are represented through a service model which is derived from the WSDL file and the corresponding annotation model [26]. The service model stores a reference (URI) to the WSDL file and service information (e.g. operations and parameters). This information serves as the foundation for the visualization of the frontends. The ServFace Builder is composed of three main modules. The Service Browser (Figure 1, Area A) lists the available services and their operations. The Composition Area (Figure 1, Area B) allows users to create an application and displays its layout in the form of a web page. Service frontends or form-based UI components (Figure 1, Area C) represent service operations and are used to populate pages of the resulting application.

III. THE SERVFACE METHODOLOGY

ServFace endeavors to create a model-driven service engineering methodology for an integrated development process of service-based applications. The project is concerned with composing user interfaces of services. It investigates this process from two perspectives: (1) the development of services with corresponding user interface descriptions, and; (2) the development of user interfaces for a composition of services. The methodology articulates on three sequential steps.

1. Service annotation: existing services are annotated with user interface descriptions, which specify the presentation of their elements (e.g. human readable labels for fields, grouping of parameters), behavior (e.g. client-side validation rules, suggestion of input values), and the relationship between elements (e.g. semantic data type relations). These annotations are based on a formally defined meta-model, can be edited via a specialized graphical tool, are saved into a separate XML file and stored in a repository [11].

2. Service composition: the annotated services are combined using a visual composition editor, by which relationships between user interface elements of different services are defined “at the presentation layer” [9], [11]. The output of the service composition phase is a model containing a consolidated user interface representation and the business logic of the application to be designed.

3. Runtime generation: an executable application is generated automatically from the Composite Application Model [26].

B. The Service Composition Process

Our concept of service composition at the presentation layer focuses on the UI level abstracting from the underlying technical complexity. Users directly interact with single UI-elements, entire service frontends, and pages at the design time in order to model the desired application. No other abstraction layer is required to define data or control flows.

The graphical development process includes the following steps:

- **Select a platform:** the user starts by selecting the target platform from a set of templates which will adjust the size of composition area accordingly.

- **Add a service frontend:** a blank page is created and the authoring process begins. The user can drag a
service operation from the Service Browser and drop it into the composition area. The tool automatically creates the corresponding service frontend, which can be positioned on the page.

- **Combine service frontends:** to define a data-flow between two service frontends, the user connects the target UI-element to be filled with data and the source UI-element from which data has to be taken. The relationship is visualized by a one way arrow showing from where data is passed to where.

- **Define a transition:** frontends can be placed on one page or distributed over several pages. All pages are shown in a graph-like overview and can be connected to define page flow in a graphical way.

- **Deploy the application:** finally the application can be deployed according to the initial target platform selected.

### IV. USER STUDY ONE

A first user study was organized at the beginning of the design process to collect user requirements and gain a clear vision of the mental models of ordinary Internet users about web services and the composition task. The study took the form of a contextual interview [8], a research and design approach based on the knowledge that users are best able to articulate their opinions, attitudes and needs about a product while they are actually engaged in using that product. In this study, we used mockups and a low fidelity prototype as examples to describe our composition approach (Figure 2). The basic idea of the prototype was to facilitate service composition by graphical user interfaces representing service operations via a service browser menu, additional UI-elements (i.e. widgets), canvas for designing multipage applications, page flow to show pages of the application, and a database storage for storing and retrieving data items.

![Mockup (A -Top) and early prototype (B -Bottom) of the ServFace Builder](image)

**Figure 2.** Mockup (A -Top) and early prototype (B -Bottom) of the ServFace Builder

**A. Procedure**

A total of 15 students from the University of Manchester took part in the evaluation which lasted approximately one hour. None of them had a computer science background. The evaluation was facilitated by two researchers and included the following steps.

1. Users were invited to provide a definition of web services, widgets, and web applications, and explain how these software artefacts operate.
2. Users were invited to view and comment on a mock-up of the composition tool.
3. Users were demonstrated a simple service composition example related to “student course enrolment” in which the purpose and main aspects of the tool were explained (e.g. Figure 2 A).
4. Users were prompted to give their views regarding service composition and to evaluate the mock-ups.
5. Users were requested to build a composite service using an early interactive prototype of the tool (Figure 2 B) based on the following task description: “…build a software tool that facilitates your future travel bookings”. The evaluation was facilitated by the technique of the thinking aloud, an approach to user research whereby people are constantly stimulated to verbalise their thoughts and motivations during interaction with a system [14]. This method is well suited to understand the user mental models of operating systems, by providing a view on their action rationale.
6. Students were prompted to report their final views about the composition tool and the composition approach.

**B. Results**

Users provided various definitions of web services, which tended to encompass their practical functionality with little technical understanding. A total of 3 users defined services as **services that are delivered through the web**. Other 3 users perceived services as the **provision of information or knowledge over the Internet**. The others directly identified services with some form of software, such as a ‘tool to build the web and applications’, ‘online communities’, ‘search engines’, and ‘interactive elements’. A total of 6 people argued that unlike traditional web pages which are static, services are **interactive elements** which enable them to perform tasks. When prompted to report examples of web services, 7 users mentioned search engines (e.g. Google, Yahoo) and E-commerce sites (e.g. Amazon). Others mentioned social-networking systems (such as Facebook), website tools, and the University portal, which is called ‘Student Services’.

All participants showed a high interest towards the idea of ‘developing service-based applications’ that are tailorable to their needs with the help of an authoring tool. Users argued that service composition could empower them to perform complex and time-consuming tasks more easily and rapidly and fulfill their activities more efficiently. Further details of the results
are reported in [19]. The usability testing of the tool showed user frustration and difficulty to understand specialized terminology, such as operations, parameters and data flows. However, other concepts such as that of input and output were easily understood by users. Overall, participants perceived the composition of heterogeneous services as a difficult process since it required the accomplishment of several steps. Therefore, they claimed to need a clear guidance and automatic-support through the process.

TABLE I. summarizes the main problems evinced in the evaluation alongside their severity rating, on a three point scale (high, medium, and low) reported in bracket. The development of the ServFace Builder followed an iterative process and was informed by the evaluation results. The main recommendations and changes to the ServFace Builder from the first study were:

- avoid and remove technical terms like widgets, operations, or mashups and instead use user-friendly terminology such as ‘building blocks, component’;
- computer-oriented concepts such as that of a temporary data storage or of additional UI control elements confused most of the participants and thus were discontinued;
- integrate a graph-based overview for all pages to create connection between pages and visualize the corresponding dependencies;
- attach icon-based menu to each UI element to offer various functionalities like renaming labels, removing UI elements, creating frontend connections.

TABLE I. SERVICE COMPOSITION PROBLEMS ELICTED FROM THE INITIAL USER STUDY

<table>
<thead>
<tr>
<th>Problem, severity rating</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weak awareness of service composition (High)</td>
<td>End users had difficulty understanding the concept of connecting services together, even though we tried to explain it as clearly as possible. They expected that service connection should be defined at the backend of the system.</td>
</tr>
<tr>
<td>Difficulty in defining the execution steps of services, i.e. flow of services (High)</td>
<td>It was difficult for the users to define the steps and execution flow of services. This activity was found to require analytical skills and understanding of the application to be designed.</td>
</tr>
<tr>
<td>Worries about service security and information privacy (High)</td>
<td>Users declared to be very concerned about the privacy and security of their personal information which might be compromised by hackers or service providers.</td>
</tr>
<tr>
<td>Low control and manipulation of services (Medium)</td>
<td>Some users found difficulties when interacting with the UI representation of the services, e.g. moving and deleting services from the design space.</td>
</tr>
</tbody>
</table>

V. USER STUDY TWO

A second formative evaluation was conducted using a fully functional prototype of the ServFace Builder (Figure 3 A). The goals were to assess the quality of the implemented concepts (e.g. service browser, graph-based overview to see all pages, navigation between pages), the usability of the tool, and the amount of training required to operate the tool comfortably. This version of the tool implemented all the recommendations extracted from the initial study, mainly the ability to create and visualize connections between services and pages.

A. Procedure

A total of 12 students ranging from 19 to 25 years old participated in this evaluation. Their background knowledge was not related to Computer Science or Information Systems. An initial pre-test questionnaire revealed that only 25% of them had created blogs or websites. All users reported no knowledge of the concept of Mashups. This evaluation comprised two parts:

1. An exploratory part lasting approximately 15 minutes where participants freely explored various parts of the tool, and they could ask questions. At the end of this part, we asked users to rate some of the UI features of the tool (e.g. navigation).
2. A task-driven part where each participant had to develop a simple application based on a specific use case (e.g. “…use the given web services to create your business travel booking application”) and simultaneously verbalize their thoughts. A video-recorder was used to record user comments and their interaction behavior.

Figure 3. Second prototype of the ServFace Builder (A - Top) and its graph-based overview of all pages (B - Bottom)

B. Results

User’s comments and questionnaire answers of the second prototype were quite positive, yet they still had some problems getting started right after they reached the first tool screen. They explicitly criticized the lack of guidance and inadequate help. This leaves room for further optimizations regarding the start-up process and user guidance. Users praised the page
navigation which mimics the Microsoft Power Point Slide view; it is activated upon mouse-hovering (not shown in the above figure).

Users were neutral about “Drag&Drop” concept for adding service operations to the design space and some users suggested also enabling “double clicking” for adding service frontends. Problems regarding the connection of two service frontends occurred mainly in terms of defining the data flow direction, i.e. the source and destination of data items. In addition, users complained that the process to connect frontends between pages consisted of too many steps.

The page flow view –i.e. of that which mimics the Microsoft Power Point Slide view- was used as the central view to show the pages during the user study. Only 25% of the participants asked for the possibility to influence the order or to define alternative flows between pages, which emphasizes our preliminary findings that end users do not think about service connection. Interestingly, most participants just added new pages and integrated service front-ends in the linear order they had created the pages before. Some people directly questioned the rationale behind creating links between pages of the application. Another fundamental concern was that users were not aware of how the final application will look like. We also evinced a substantial lack of understanding of the difference between design- and run-time.

TABLE II. summarizes the main problems that emerged from the second evaluation alongside their severity rating, on a three point scale (high, medium, and low). The main recommendations and changes from the second study were:

- integrate a comprehensive video tutorial to explain how the tool can be used and to illustrate various composition concepts (e.g. design and runtime);
- increase user awareness of the final application by improving the layout of the pages e.g. UI elements like a browser bar for each page of the web application;
- add an assistant tool which provides hints to users during the service and page connection steps;
- support two styles of service frontends: freely movable widgets and forms that are integrated in the background;
- rename menu entries that created confusion (e.g. direction of data flow can now be clearly defined)

TABLE II. SERVICE COMPOSITION PROBLEMS ELICITED FROM THE SECOND USER STUDY

<table>
<thead>
<tr>
<th>Problem, severity rating</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex service and page connection process (High)</td>
<td>Users complained that the frontends' page connection process contained too many steps and should be simplified.</td>
</tr>
<tr>
<td>Weak system support and guidance (High)</td>
<td>Users suggested that it would be helpful if the system provides hints during development activities.</td>
</tr>
<tr>
<td>Weak awareness of the resulting application (Medium)</td>
<td>Users found it difficult to imagine how the final application will look like during runtime, even though they have interacted with UI-components.</td>
</tr>
</tbody>
</table>

VI. USER STUDY THREE

In the third user study, we tested an advanced prototype of the ServFace Building (Figure 1), which included a set of examples and a video tutorial to demonstrate the composition process and the resulting application. We have also added an assistant tool which provides help to users during the connection process, and improved the layout of the pages (Figure 1).

A. Procedure

A total of 12 people participated in the study, 9 were students and 3 administrative staff at the Manchester Business School. Their age ranged from 25 to 34 years (mean 30) and nobody had programming or modeling experience. This evaluation lasted around 2 hours and comprised the following steps.

1. An initial contextual interview was performed to elicit users’ opinions about, mental models of, and experience with current software development environments. Microsoft Publisher and Yahoo! Pipes were used as examples in this phase. Users also tried to interact with these applications.

2. A 15-minute training of the ServFace Builder was provided by a video tutorial and face-to-face explanations provided by the experimenter.

3. Users were given a scenario (e.g. given a set of web services, build a student admission management system that processes student applications) and invited to perform 12 tasks using the ServFace Builder (Figure 1). Example tasks include: add a service component which retrieves all student applications to the design space; connect input of ‘Application ID’ of service component 1 to output of ‘App ID’ of service component 2.

4. Participants were then given a questionnaire developed for the purpose of this study and invited to rate the quality of the interaction with the ServFace Builder on a number of dimensions.

5. A final interview to elicit users’ opinions and comments was conducted.

B. Results

The definition of software services provided by the users closely mirrored the results of study 1. Participants tended to define software services based on the action they perform with very little (if any) technical details. Half of the sample defined software services as a form of assistance to perform on-line activities. The other half identified web services with specific software applications such as websites (e.g. Google and iGoogle), or tools to build websites and blogs. Nobody mentioned any computing properties of services. Their definitions were very basic and far from any technical jargon, reflecting a shallow mental model of web services. The participants perceived themselves as pure consumers of services, no one spontaneously mentioned the possibility to create or customize services. However, once introduced to the concept they were rather positive about it. They agreed that service composition is useful (mean = 4.17, std = 1.11),
interesting (mean = 4.17, std = 0.71), and brings about efficient ways to conduct online activities (mean = 4.58, std 0.66). All evaluations were collected on a 1 to 5 scale, where 1 reflected a disagreement with the sentence and 5 an agreement. However, participants rated the “easiness to achieve service composition” lower (mean = 3.58, std = 1.44), but they disagreed with the statement that “service composition is unfeasible” (mean = 1.75, std = 0.7).

User comments during the interaction with MS Publisher and Yahoo! Pipes were analyzed by thematic analysis, an inductive technique by which themes emerge from the data by looking at patterns and similarities [2]. The analysis revealed 6 main categories, mainly reflecting worries and concerns of the users. People were concerned about difficulties in learning how to use these environments (a theme evident in 6 comments); insufficient help and limited templates (6 comments), and difficulty in understanding the terminology and concepts used within the environments (5 comments). Other themes addressed fears of system failures (crashes), the difficulty of the task even after the training phase, and the time consuming nature of the task.

As regards the evaluation of the ServFace Builder, the analysis supplemented performance data with qualitative data (interviews) and quantitative data (questionnaire) of the user-experience. The 12 tasks performed with ServFace Builder were clustered into 5 categories: adding service components to the canvas (4 tasks), adding pages to the application and renaming them (3 tasks), connecting service components (2 tasks), connecting pages of the application (2 tasks), and customizing service components (1 task). For each category, TABLE III. reports the average time spent by the users to successfully complete the task and the percentage of failure, i.e. uncompleted or wrongly performed tasks. The last column report the evaluation of the user on the complexity of the task as addressed by a 5-point Likert scale (i.e. “it was easy to achieve task X”; 1= disagree, 5= agree). NA means that the evaluation was not collected on that dimension. The performance analysis revealed that connecting service components and pages triggered the highest percentage of failures (>37%). However, once discovered the right strategy, users were able to complete the “page connection” tasks faster than other types of tasks. A notice of caution is necessary when considering the variable execution time in this study. Indeed, tasks were always performed in the same order with no counterbalancing, and a learning effect was evident on these measures. Execution time was also affected by different types of attitudes. For instance, 3 users spent a large amount of time (>180 sec) to complete the first task (e.g. adding a service) using this time to familiarize themselves with the system. The large standard deviation proves this point.

The subjective evaluations revealed that understanding the relationship between pages and between services was not easy (in the order mean = 3.33, std = 1.43 and mean = 3.58, std = 0.90). Users rated the task of connecting services between different pages as more difficult (mean = 2.83, std = 1.46) than connecting services on the same page (mean = 3.75, std 1.54). The difference approached significance as demonstrated by a repeated measure t-test ($t_{\text{diff}} = 1.83$, $p = .09$). Similarly, participants considered that connecting services between different pages was significantly more difficult that connecting different pages (mean = 3.83, std = 1.46, $t_{\text{diff}} = 2.70$, $p < .05$).

Navigating through the ServFace Builder, adding new pages to the application, and creating applications of multiple pages were perceived as pretty easy tasks (in the order, mean = 4.25, std = 1.35, mean = 4.92, std = 0.29, and mean = 4.33, std 0.98).

<table>
<thead>
<tr>
<th>Task type</th>
<th>% of failure</th>
<th>Average time per correct task (in seconds)</th>
<th>Subjective rating (easiness on a 5-point rating scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect services</td>
<td>42</td>
<td>52.25 (std 24.75)</td>
<td>3.29 (std 1.54)</td>
</tr>
<tr>
<td>Connect page</td>
<td>37</td>
<td>26.87 (std 16.78)</td>
<td>3.83 (std 1.46)</td>
</tr>
<tr>
<td>Add and rename a page</td>
<td>28</td>
<td>47.43 (std 34.56)</td>
<td>4.92 (std 0.29)</td>
</tr>
<tr>
<td>Customize a service</td>
<td>25</td>
<td>51.33 (std 55.60)</td>
<td>NA</td>
</tr>
<tr>
<td>Add a service</td>
<td>15</td>
<td>76.71 (std 77.28)</td>
<td>NA</td>
</tr>
</tbody>
</table>

In the interviews, users often complained about the lack of help in the system, as they often felt to lack control on the interaction. These points also emerged from the questionnaire items. Participants were unsatisfied with the level of help and support provided by the tool (mean = 2.58, std = 1.16), did not feel in control when using the ServFace Builder (mean = 2.75, std = 1.28) and felt slightly confused during the service development process (mean 3.67 = sd 1.37). Users’ awareness of how to begin service development and how the final application will look like was average (mean = 3.67, std = 1.30, mean = 3.50, std = 1.44 respectively). Overall, users agreed that the tool was rather easy to use (mean = 3.40, std = 1.35).

When enquired about the most positive features of the ServFace Builder in the post-test interview, participants spontaneously mentioned simplicity and ease of use (6 comments), direct-manipulation interaction (5 comments), and service/core components (4 comments) and the help (3 comments). The most negative features were (23 comments in total) connecting service components and pages (5 comments) which agrees with the user performance results, followed by difficulty in understanding tutorial and insufficient help (5 comments), aesthetics of the ServFace Builder (4 comments), navigation through the tool (2 comments), and awareness of actions (2 comment). Some users expressed worries about the security of their personal information.

The main recommendations and planned changes –i.e. work in progress- from the third study are the follows.

- offer several composition patterns (a set of single connection steps that are always performed in the same way) to the users to guide them through the most common composition steps in order to alleviate the problem related to lack of understanding of this task;
- integrate a suggestion algorithm to identify suitable connections between frontends;
• implement a two page Mode to facilitate the connection of frontends which are located in different pages. The two page mode shows two pages of the application with their frontends side by side;
• incorporate a Master Page Layout Manager to define and customize the look and feel of the application, e.g. background and frontends colors, logo, etc;
• support two different navigation types of the application pages (free (default) and sequential (wizard or guided procedure);
• remove all design time control elements (icon-based menus) from the pages and frontends to achieve a real WYSIWYG.

VII. DESIGN RECOMMENDATIONS FOR LIGHTWEIGHT SERVICE COMPOSITION

The studies presented in this paper aimed to discover common conceptual and usability problems ordinary Internet users would face when trying to build composite services. It is worth noting that the below design guidelines were all inspired by the user feedback and problems we observed and identified from the user interaction with our composition tool.

Task-based composition process and system guidance. The most complex conceptual issue that emerged from the three studies was connecting services and understanding the meaning of service composition. The issue is complex since users did not realize that services need to be connected, and struggled to understand what composition mean (e.g. data flow, control flow, both, or nothing –black box-).

To simplify the composition process, this task should be driven by the system whilst giving users the right level of control to modify the semi-automatic solutions. The composition platform should help users to think about service connection by making them realize how services should be executed to fulfill a particular goal. We envisage a solution through a platform which allows end users to perform a task analysis by first specifying the general goal of the service-based application (e.g. calculate promotions and send instructions to guide users. Proactive help and advice by the system will also help users to locate problems [6].

Secure service composition platforms and their resulting applications. All users were concerned about creating service-based applications (e.g. those including bank software services) that might be penetrated by hackers and intruders which agrees with our previous findings [20]; thus guarantees and secure policies should be provided to raise users’ confidence and trust.

Graphical development and level of abstraction. With the term graphical development we refer to the ability to manipulate services in a visual manner, a feature which was strongly appreciated by the users; this finding is consistent with [27]. Representing services by their user interfaces made it easier for the user to understand the goal of the service, than the information contained in the programming code. Consequently, development environments targeting non-programmers should leverage on visual representations and WYSIWYG paradigms to simplify development and offer high levels of abstraction which hides technical details.

Preview of application (runtime view). Besides enabling users to create their application easily, the composition platform should provide a preview of the potential application and empower users to flexibly switch between design and runtime views.

User language and terminology. Technical terms like function, operation, or data flow should be avoided and replaced by more abstract and common terms that are easier to understand by Internet users. New metaphors need to be developed to allow Internet users to deal with the complexity of service composition.

Realistic examples and reusable templates. Use real life examples, demos, and tutorials to motivate users and boost their level of confidence. It is not enough to provide a tool that is easy to use, it is important to supplement it with examples and use cases that clearly communicate the value of service composition for ordinary users. Moreover, the tool should offer a set of reusable service composition templates of varying difficulty to accommodate different types of end users.

System help. Development environments should offer comprehensive and easy to access help and documentation about each of the composition steps, as well as tooltips and instructions to guide users. Proactive help and advice by the system will also help users to locate problems [6].

Aesthetics. Our results showed that the aesthetics of a service composition platform is important to ordinary Internet users, an aspect that is usually underestimated by tool designers. This is consistent with [16] who showed that aesthetics affects first impressions and perceived usability of a product. Therefore, the working space should be convenient to manage the development activities but also eye-pleasing.

VIII. CONCLUSION AND FUTURE WORK

This paper presented the user-centered design of the ServFace Builder. Three user studies were performed to shed some light on the mental models that ordinary Internet users held of services and on their attitudes towards the possibility of composing software services into personalized applications.
These studies allowed gathering a set of end-user requirements for lightweight service composition tools, and informed the iterative design of the ServFace Builder. Results also informed a set of design guidelines, which can be used in similar design challenges.

On the average, ordinary Internet users have a very simplified view of web-services, ranging from the idea of work done by a company through the Internet on behalf of a customer, to applications supporting the delivery of information on the web. Users were positive as regards the potentials that a lightweight service composition tool can offer them, particularly as regards the ability to tailor software applications to their needs without requiring computing expertise. However, major obstacles were found in creating data-flow and control-flow dependencies, tasks which showed to be cognitively demanding for ordinary Internet users.

An initial set of design guidelines for end-user composition tool were proposed based on the problems users encountered during composition tasks. Although these guidelines were extracted from the analysis of just one service development environment, we are confident that they can be applied to similar service composition platforms. We are planning to conduct further empirical studies using a diverse set of mashup and service composition tools (e.g. Yahoo! Pipes, SOA4All Studio [20], MashMaker [28]) which, to the best of our knowledge, have so far been evaluated using mainly on-line surveys and focus groups.

ACKNOWLEDGMENT

The work presented herein is supported by the EU-funded project ServFace. We would like to thank the students and staff of the University of Manchester for taking part in this research.

REFERENCES