End User Development of Service-based Applications

Abdallah Namoune  
Manchester Business School  
Booth Street West  
Manchester, M15 6PB  
0044 161 306 63095  
abdallah.namoune@mbs.ac.uk

Tobias Nestler  
SAP Research CEC Dresden, Germany  
0049 351 4811 6148  
tobias.nestler@sap.com

Antonella De Angeli  
Manchester Business School  
Booth Street West  
Manchester, M15 6PB  
0044 161 306 63383  
antonella.de-angeli@mbs.ac.uk

ABSTRACT
This paper reports some reflections on the users’ mental models of web services and service composition. This knowledge was elicited from two studies involving individual semi-structured interviews and group discussion. In particular, we focus on what end-users, with no formal programming background, understand about web-services and how they view service composition at the presentation layer. The results showed a high user acceptance towards a simple authoring tool dedicated to building their own service-based applications. However, several difficulties emerged when trying to simplify service orientated programming to make them available to a non-technical audience. Recommendations to resolve them are discussed in the conclusion.

Categories and Subject Descriptors
H.5.2 [Information Interfaces and Presentation]: User Interfaces – Graphical user interfaces, Interaction styles, Prototyping, Standardization, User interface management systems.

General Terms
Design, Human Factors.

Keywords
Services, service composition, end user development, service based applications, presentation layer, task taxonomy.

1. INTRODUCTION
The recent evolution from traditional services (e.g. a hair cut service) to software and online services (e.g. a flight booking service) has helped users to fulfill their daily needs more efficiently and conveniently than ever before. Web-services are formally defined as a software system designed to support interoperable machine-to-machine interaction over a network [19]. As Service Oriented Architectures (SOA’s) are becoming increasingly common on the Internet, there is a need to facilitate the use and re-use of web-services, especially by people who have no programming knowledge. The major issue is how to simplify the composition of different services in a single application [1].

Two main approaches to end-user design of service-based applications have been proposed so far. The simplest option, implemented for example in iGoogle, myYahoo, and FaceBook [4, 7, 11], follows a widget-based model, whereby individual services are visually represented in separate interface windows. The user can select their preferred services and locate them on a page. This approach has the advantage of being simple and intuitive, but it does not support complex tasks, such as service composition (i.e. the process of combining existing atomic and composite services together to form new services). Mashups are produced from aggregating existing services and web feeds from different sources using special builders [6, 14, 20]. This approach is dedicated to data aggregation and highly relies on computing knowledge and skills of the end-user.

This paper presents a methodology developed within the EU project ServFace to support UI composition for web-services. As part of this project we are investigating new approaches and tools to design simple authoring environments allowing non-technical users to design composite web-service applications. This work also summarises non-technical end users’ views about services and highlights the potential issues that regular users could face while assembling services through a set of semi-structured interviews and a focus group. The results of this evaluation will be fed, as a set of requirements, into the development of a user oriented service integration authoring tool.

The general outline of this paper is as follows: section 2 introduces the problem of service composition. Section 3 describes the ServFace project [17], methodology, and results of the elicitation studies. Section 4 highlights the problems that could be encountered whilst end users create composite applications. Finally, section 5 and 6 discusses the concept of services and the implications of service composition.

2. SERVICE COMPOSITION
The integration of software components has attracted a large body of research in the area of software engineering as it enables the creation of powerful applications and increases reusability. Most of the previous research efforts focused on the integration at the data and application (also known as business logic) layers while giving less importance to integration at the presentation layer [3]. Recent advances in web services and SOA’s made it possible to develop and integrate loosely-coupled components for the creation of service-based applications [1]. Service composition is supported by several languages (such as: BPEL [2]) and tools (such as: Oracle Process Manager [13]), which requires
considerable computing knowledge and training. In most cases, users need to understand the concepts behind web services and learn specific programming languages to compose service. User-oriented mashup tools have been proposed to meet the needs of end-users with minimum programming efforts [9, 14]. Although, data aggregation can be useful, it is limited in terms of what it can achieve. Users would benefit more if they were able to combine business functionalities of web services within applications that serve their daily needs.

Another major shortcoming of current approaches to service composition refers to the UI layer. Despite the provisioning of modern web user-interface technologies, such as Java Applets, ActiveX and AJAX, to create rich and interactive user interfaces, designers still generate the interfaces of service-based applications manually. This process is carried out at the design-time phase and was acknowledged as the most demanding task in terms of time and efforts [10]. Consequently, supporting service interface designers with tools to facilitate the creation and reuse of interfaces could empower end users in general and designers in particular. Problems related to UI integration may include: specifying a composition language, discovery and binding (i.e. identify and bind to the relevant components), and UI visualisation [3].

The challenge to service and Human-Computer Interaction (HCI) research lays in finding new methods to open up service composition to a larger population supplying non technical users with an intuitive development environment, that hides the complexity of services to create composite applications. This goal although desirable, opens other interesting challenges to HCI. Service-based applications should offer the same usability as traditional desktop applications. In that sense, users should be able to execute tasks easily and efficiently in order to accomplish their goals. End-user development of service based applications may negatively affect the final usability of an interface due to the end-user lack of interaction-design knowledge.

Recent research has attempted to automate the generation of user interfaces of web services [8, 18]. Some of these state of the art approaches exploit information related to the user interfaces incorporated within the service scheme (i.e. WSDL files). However, the resulting user interfaces are simple and their quality is poor due to the weak expressiveness of current UI description languages which limits the amount of UI related information that can be stored within WSDL files. Other limitations include supporting particular types of UI technologies only [15, 16] and the impossibility of customising the user interface during the design phase which is often a necessity [16]. Moreover, none of the existing approaches support the creation of composite service-based applications at the presentation layer.

A possibility to overcome these drawbacks, is to facilitate and expedite the composition of user interfaces of web services within a model-driven service engineering methodology [5, 17]. Ultimately, non technical users would be able to build service-based applications by combining services’ front-ends. This is the objective of the EU project, ServFace.

3. ServFace Approach
ServFace aims to create a model-driven service engineering methodology for an integrated development process of service-based applications. ServFace investigates this process from two perspectives: firstly, the development of services with corresponding user interface descriptions, and, secondly, the development of user interfaces for a composition of services. Thus, the typical service development paradigm will be extended to user interfaces. Essentially, ServFace is concerned with developing and composing user interfaces of services. For this purpose, three steps are taken into consideration, figure 1:

![Figure 1. ServFace Methodology [17]](image)

Service annotation: in this initial step of the methodology, existing services are annotated with user interface descriptions (Step 1, figure 1) which specify the presentation of elements, behaviour of elements, and relationship between service elements (i.e. inputs, outputs, and operations). These annotations are based on a formally defined meta-model and can be edited via a separate annotation tool. The resulting annotations are saved into a separate XML file and stored in an annotation repository for later reuse.

Service composition: in this step, the annotated services (resulting from step 1) are combined using a visual composition editor by which relationships between user interface elements of different services are defined at the presentation layer [12]. The output of service composition is a model containing a consolidated user interface representation and business logic of the application to be designed.

Runtime generation: in this final step, an executable application is generated automatically from the composite application model produced from the previous step.

3.1 User Understanding
To gain a clearer vision about what non-technical users think about web services and how they understand the process of service composition, we performed a set of contextual interviews with 10 postgraduate students at the Manchester Business School. A focus group involving a separate sample of 5 students was also conducted. Each individual interview and the focus group took approximately 1 hour, in which participants’ responses were recorded using video and audio recorders. In both settings, participants were encouraged to share their opinions in relation to the following topics:
1. Indicate their understanding of web services, widgets, web applications, and databases. These concepts were covered as they are instrumental to the design of an authoring tool for web-service composition.
2. View examples of web services, widgets, web applications, and databases and make further comments explaining their functionality.
3. Indicate their views regarding service composition and discuss potential related problems. This part was facilitated by the evaluation of an early prototype of a service composition tool.

3.2 Results
Users provided diverse definitions of web services. 3 users defined services as services that are available on the web. 3 other users indicated the provision of information or knowledge as services. The remaining users referred to services as: a tool to build the web and applications, online communities, search engines, and interactive elements. A total of 6 users 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 services 7 users mentioned search engines (Google, Yahoo) and E-commerce sites (Amazon). Others mentioned social-networking systems (such as: Facebook), website tools, and learning environments (the University portal).

<table>
<thead>
<tr>
<th>Type</th>
<th>Example</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search engines</td>
<td>Google, Yahoo</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Search engine</td>
<td>2</td>
</tr>
<tr>
<td>E-commerce sites</td>
<td>Amazon</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Retailers</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Company</td>
<td>1</td>
</tr>
<tr>
<td>Social networking systems</td>
<td>Facebook</td>
<td>2</td>
</tr>
<tr>
<td>Website tools</td>
<td>FrontPage</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Dreamweaver</td>
<td>1</td>
</tr>
<tr>
<td>Learning environment</td>
<td>University portal</td>
<td>1</td>
</tr>
<tr>
<td>Other</td>
<td>Currency converter</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Booking system</td>
<td>1</td>
</tr>
</tbody>
</table>

Once the concept of services was introduced to the participants and the Google Map was presented as an example, they were all able to explain how it can be used. They had a very clear idea of how to interact with it, data can be entered by typing in their “search query or term” in the text field and clicking the “Search Maps” button. The service then returns the results back to the users in the form of an image (i.e. a map). Some users (3 users) also indicated that they can interact with the service by editing its options (e.g. show satellite imagery and show traffic). All users referred to the information they supplied as “input” and to the results returned by the service as “output”, showing that this terminology is commonly shared amongst users with no technical background.

A good understanding of the concept of database also emerged. All participants defined it as a repository used to store and retrieve data and records. The most common examples included Microsoft Access, Oracle, and SQL. On the contrary, the concept of widget appeared to be more difficult. Although some users acknowledged to have heard the term, no one was able to define or guess what a widget is. 5 users defined web applications as applications that run on the web, for example iGoogle, Google docs, and Hotmail. Once the “Google suite” was introduced to the participants, they all commented that it is useful to have one application with many services bundled together as this is more convenient, saves time, reduces workload, and prevents errors.

All users liked the idea of developing their own applications that suit their needs and interests with the help of authoring tools. They argued that this will allow them to perform complex tasks more easily and rapidly. They also pointed out that assembling many services within a single application is a powerful feature missing in today’s applications which are usually built for one purpose, for example booking a flight, finding a hotel and booking a car. In normal circumstances, users have to access different online services to accomplish their goals. Users appreciated that the introduction of this feature reduces the amount of work required to perform tasks (i.e. logging into one service instead of many services/applications), reduces the chance of making mistakes, and it is more convenient to have external services grouped together in one application. Furthermore, integrating services using their front-ends only without worrying about the integration of data and business logic reduces the complexity of application development.

4. ANTICIPATED COMPOSITION CHALLENGES
Since the purpose of this research is to concentrate on the composition of services’ front-ends, problems related to data and business logic integration were not investigated. The participants anticipated that, during the composition of services to build service-based software, they may encounter the following issues at the presentation level:

Although users might have some experience in building mashups or adding external services to their personal and networking sites, such as: iGoogle and Facebook, integrating heterogeneous services exposes a totally different complexity. Service composition at the presentation layer is a novel idea; hence, the design environment has to clearly communicate its purpose to end users for maximum usability.

Services are represented via their functional operations, inputs, and outputs, which is often not understandable by regular users. Hence, it is necessary to abstract services from their technical details and present service elements in a friendlier and easy to understand manner.

Usually creating a composite application requires the execution of several steps. Let us, for example, consider a holiday booking system which requires four services: a hotel service, a flight service, a car service, and a resort service. The way in which these services can be assembled together to form a standalone application can vary. This problem becomes more complicated in the presence of thousands of web services. Therefore, the tool should clearly guide the users through the composition steps of the to-be application.
5. RECOMMENDATIONS
To counterpoint some of the above problems and thus enhance end-user development of service-based applications, we propose the following recommendations:

5.1 A Service Composition Strategy
Since our intended target audience is the non technical users, ServFace aims to hide the complexity and technicality of service-based application development by supplying them with a chain of easy to use authoring tools. For the composition of services’ front-ends, ServFace is implementing a simple composition tool tailored towards non programmers. The results of this evaluation will influence the development and improve the usability of the tool. Some recommendations regarding changes in design and functionality of the current version can already be made at this point of time. The composition tool should allow users to select and add services from a left hand-sided taxonomy of services to the main canvas / design space. As soon as a service (e.g. Serv 9, figure 2) is dropped into the design page, the system will automatically highlight any potential links (dashed lines, figure 2) with the existing services (i.e. Serv 1, 2, and 3). The user can then establish the relevant links between services. This approach not only highlights the composition aspect of the tool but also solves the services’ compatibility problem. The integration development process will be further facilitated using instructions, hints, and wizards. Another novel aspect of the tool is the categorization of services based on user tasks. In this sense, the service browser (left hand menu, figure 2) will organise services according to the user tasks they support while hiding their technical descriptions.

5.2 Interaction Tasks
For an application to be highly usable it should support the execution of all necessary user actions. Therefore, our objective is to shed light on and gain more understanding about the commonly encountered user tasks while using service-based applications. This understanding will contribute towards the design of effective user interfaces. Hence, in another ongoing research study, we propose a strategy to employ an interaction task taxonomy intended to promote the creation of service-based applications. The proposed taxonomy incorporates tasks related to search, user to user communication / interaction, control of software, user interface interaction, and cognitive elaboration. Essentially, the interaction task taxonomy is introduced to achieve the following objectives:

- Improve the design of service-based software interfaces
- Compare and evaluate different interfaces of service-based software
- Enhance the semi-automatic / automatic generation of interfaces

6. DISCUSSION
The participants demonstrated a good understanding of services, web applications, and databases and explained well their purpose and how they function without the use of technical terms. The participants in this study had no IT or programming background; hence, it is justifiable why they were not able to provide technical definitions of these terms. They also showed a high likability towards the idea of building service based applications, suited to their own needs and interest. This agrees with the current trend that end users are becoming proactive about developing the web.

The ability to assemble many services within a single application was favoured by the participants because it saves time, reduces efforts, prevents errors especially amongst non technical users, and promotes the reuse of software components (i.e. services). This view emphasises the requirement that service integration should be simplified to enable end users to take advantage of them, i.e. users should be able to aggregate services without the need to learn programming concepts or understand service elements (input, output, and operations).

One approach to facilitate the creation of service-based applications is through the employment of an interaction task taxonomy which represents the common tasks users perform when using composite applications. This will alleviate the technical complexity implied by service technology. Ultimately, elements of a service will be grouped under a specific user task which they accomplish as opposed to the conventional way of representing service elements which is challenging to understand by non programmers.

7. CONCLUSION
This paper presented an initial evaluation of users’ perception about services and their acceptance towards a design tool tailored to composing services with minimum programming efforts. Users showed a good understanding of the term services and expressed their likability towards service composition. In future works, we plan to carry out further interviews to consolidate users’ mental models of services, evaluate a potential service composition tool that is currently under development, and address all the anticipated problems to better reflect the purpose of the tool and simplify the process of building applications. Subsequently, a user study will be conducted to evaluate the usability of the composition tool.
8. ACKNOWLEDGMENTS

This work is supported by the EU project ServFace. We would like to thank the students of the University of Manchester for participating in this research study.

9. REFERENCES