End User Requirements for the Composable Web

Abdallah Namoun[1], Tobias Nestler[2], Antonella De Angeli[1]

[1] Manchester Business School, Booth Street East, Manchester,
M13 9SS, United Kingdom
{abdallah.namoune, antonella.de-angeli}@mbs.ac.uk
[2] SAP Research Center Dresden,
Chemnitzer Str. 48, 01187 Dresden, Germany
tobias.nestler@sap.com

Abstract. With the enormous auspices and resources EU research projects are receiving to enable the diffusion of lightweight service composition approaches among end users, it is imperative for these projects to understand and establish the correct user requirements that lead to development of easy to use and effective software platforms. To this end, a user-centric study which includes 15 participants is carried out to unravel users' perception of software services and service composition, their working ways, and identify users' expectations and usability problems of a future service composition tool. Several examples and prototypes are used to steer this elicitation study, among which is a simple composition tool designed to support non-programmers to create interactive service-based applications in a lightweight and visual manner. Although a high user acceptance emerged in regard to "developing service-based applications by users", there is evidence of a fundamental issue concerning conceptual understanding of service composition (i.e. end users do not think about connecting services). This paper discusses various conceptual and usability problems of service composition and proposes recommendations to resolve them.

Keywords: requirements, end user development, light weight composition, web services, presentation layer, usability.

1 Introduction

Europe is continuously spending tens of millions of Euros funding service research projects that aim at developing user-friendly software platforms to facilitate the development of interactive service-based systems. Much of this research effort is dedicated to solving technical complexities and implementation problems rather than understanding what end users really need, how they think about service composition, and their natural working ways to support their activities and enhance service consumption and production. Unfortunately, some of the funded projects decide on implementation strategies and solutions for their target user group before even starting the project either because they want to pursue their research interest and agenda or claim they understand the needs of their users.. This justifies why, despite the rapid

advancement in Service Oriented Architecture, user research in that area is still in its early infancy. User studies that focus on understanding user development abilities, needs, and mental models about service composition are evidently required to establish correct and well-informed requirements.

Web 2.0 enables Internet users to add content to the web, interact with and customize web pages, share their thoughts, collaborate, and develop the WWW. In general, these activities do not necessitate acquisition of IT specialist knowledge since, for example, the act of modifying a wiki page is fairly simple for anyone who knows how to use a computer and browse the Internet. A promising approach to empower users beyond web content development is to use and reuse loosely coupled software components like web services in order to create composite applications tailorable to their diverse needs. However, there is no substantial proof that naïve users (i.e. users with no computer science/IT background) can combine functionalities together to produce augmented software services. The question as to whether "do users perceive or think about uniting/connecting independent functionalities to form greater assemblies?" is a key answer to the success of many service composition research projects.

This motivation becomes more interesting when taking into consideration the fact that creating computer programs requires strong modeling abilities and problem-solving skills which most Internet users do not acquire, a fact that always intimidates end users. Changing and customizing web content is different from composing software services since the latter is more complex and challenging. Transforming ordinary consumers of services into actual producers of services invites other unanswered research questions: what do users understand by web services? are they just black boxes, interfaces they interact with, snippets of code, etc? are end users willing to take the burden of connecting services together sacrificing time and effort? how do we assist users to better understand service composition and encourage them to uptake development activities? which metaphors should be used to enable easy development of service-based applications?. Such understanding is crucial to the representation of services and service composition in authoring and modeling environments.

To sum up, this paper endeavors to:

- Capture users' true understanding of web services and composition of service-based applications; this finding will impact the way services will be represented to developers in development environments.
- Identify conceptual and usability problems that relate to service composition, as probed by realistic examples and prototypes of a visual composition tool.
- Improve the design of service development environments through a set of guidelines and recommendations; thus enhancing the diffusion of services among Internet user.

2 Existing Work on Data Mashups and Service Composition

At present the web offers users the capability to build personal pages through customizable web portals, such as iGoogle[1] and MyYahoo![2], whereby they add web feeds and gadgets (i.e. programs that provide services) to their personalized pages. In principle, users browse a list of services and add the desired ones to their pages such as: Weather, Wikipedia, Google Map, and Day and Time services. They can also edit these services and modify the look and feel of the pages by applying a desired theme or moving the gadgets within the page outline. Customizable web portals are easy to use and interact with but do not support the creation of complex software applications because services can not be combined with each other, in other words users can not create or manage links between services. Widget-based applications are merely a collection of independent services that do not communicate to each other or have very limited communication. It is more useful and interesting if ordinary users are enabled to produce rich and complex service-based systems that fulfill their specific needs, but also allowed to easily extend and customize applications.

Web 2.0 makes the formation of web-based communities and consumption of services, such as: wikis, blogs, video-sharing, and social-networking sites, more feasible. However, it does not allow the formation of powerful applications that consist of web services interacting together ([8], [11]). Even though service composition is well-understood and covered by existing approaches for technical developers using composition languages (e.g. BPML, BPEL4WS, WSCDL … etc), tools and methodologies for enabling end-user service composition have been largely ignored ([9], [10]). The current technical aspects of service composition are not of much interest to ordinary users who want to capitalize on the benefits offered by Service Oriented Architecture. What really matters to end users is how these technologies are presented to them and how they can easily use these advanced technologies to perform their desired tasks.

Promising approaches for a lightweight user-driven application design are Mashup platforms (overview provided by [4]), which came up with the growth of the Web 2.0. The graphical composition style constitutes a first step towards user empowerment and increasingly shifts the creation of individual applications to the domain experts. Mashup platforms like Yahoo! Pipes[3] or Open Mashup[4] enable the creation of more sophisticated service-based applications by aggregating web feeds, web pages and web services from different sources. Unlike customizable web portals, users can define relationships between modules by dragging and linking them together within a visual editor. The output of one module can serve as the input of another. However, mashups mainly focus on data aggregation and still lack of concepts to create composite service-based applications by end-users ([3], [9]). Moreover, they require modeling skills and good understanding of computing concepts such as message and data passing which most users do not have.

---

[1] http://www.igoogle.com
[2] http://www.my.yahoo.com
[3] http://pipes.yahoo.com
[4] http://www.open-mashups.org

Namoune et al conducted focus groups to discuss the risks and benefits of end user composition of service-based applications. End users were mainly concerned about the privacy and security of their personal data, as well as the underlying technical complexity they might encounter when composing service-based applications [6].

Despite the continuous progress in service-oriented technologies, service composition by end users (non-programmers) is an area in its early stages. Therefore, identifying the needs and specific requirements of ordinary users is a crucial prerequisite to the design of *"easy to use"* and "*easy to understand*" service composition environments. 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 and service composition to create composite applications. This goal although desirable, opens other interesting challenges to HCI.

3 Experimental Set Up

3.1 Procedure

The study at hand was conducted in the form of a contextual interview/inquiry, wherein 15 non-technical students at the Manchester Business School participated. Each individual interview took approximately one hour. Beyer and Holtzblatt argue that contextual interviews are very useful for identifying user contextual needs and how specific actions are performed in detail [2]. Moreover, they enable researchers to understand users' environments and their work conduct; thus portraying actual user behavior. During the interview we set up a focus in regard to the objectives of the ServFace project[5] and how these objectives will be fulfilled. In this study, the focus is to enable ordinary users to build composite software applications that are tailored to their needs using a light-weight simple composition tool called *ServFace Builder*. To guide the interview several widespread examples (e.g. iGoogle, Google Map Search Service), low fidelity prototypes, and a high fidelity prototype of a future authoring tool were used. The detailed steps participants were asked to complete are the following:

1. Define web services, widgets, and web applications, and explain how these software artifacts work from a user perspective. Following the participants' answers, the interviewer provided the exact definition of each term with examples
2. View a mock-up of the composition tool and make initial comments and impressions
3. Walk through a simple service composition example "student course enrolment" in which the purpose and main aspects of the tool were explained

[5] www.servface.eu

(Figure 1). Through this carefully selected example that suits participants' environment and background we aimed to effectively communicate the idea of "service composition by end users"

4. Indicate their views regarding service composition and evaluate the mock-ups of the tool
5. Go through a service composition scenario and build a composite service using an early online prototype of the tool (Figure 2). Concrete task description: "You are a team assistant for a team of 50 people. You can use MS Office well and like to play around with the tools you use and customize them to fit your needs. Since your colleagues often need to attend conferences it is your responsibility to organize the travels. In the past you spent a lot of time searching and booking suitable flights and hotels. In the future, you want to build an application to allow your colleagues to do their own booking without spending a lot of time on it. *Thus you want to build a special tool that facilitates travel booking*"
6. Indicate their final views regarding the composition tool and the general composition approach

3.2 Materials

3.2.1 Motivating Examples

To simplify the definition of software services, widgets and web applications for our non-technical audience, we used widespread examples that most people are familiar with, in particular: Google Map Search Service, Date and Time Gadget by Google, and Google Suit (email, Calendar, Documents, Web, Reader, etc). These examples were only shown to users after they had provided their definition and examples.

3.2.2 Low-Fidelity Prototypes and Student Scenario

Participants were presented with a set of mock-ups of our service composition tool using Microsoft PowerPoint (figure 1). The mock-ups demonstrated how a student can create a composite application that allows her to register to a particular course of study. The interviewer went through the process of visual service composition and explained the necessary steps. Participants were then invited to make comments or ask questions.

3.2.3 High-Fidelity Prototype

The evaluated ServFace Builder was developed in the frame of the EU-funded research project ServFace. The tool utilizes the advantages of web service annotations [5] enabling a rapid development of simple service-based interactive applications in a graphical manner. The tool applies the approach of service composition at the presentation layer, in which applications are built by composing web services based on their frontends, rather than application logic or data [7]. During the design process, each web service operation is visualized by a generated UI (called service frontend), and can be composed with other web service operations in a graphical manner. Thus,

the user, in his role as a service composer and application designer, creates an application in WYSIWYG (What you see is what you get) style without writing any code. It is worth noting that the tool depicted in figure 2 has been improved and looks different now based on users' feedback and design recommendations gained through this study. However, for the interest of this paper, figure 2 shows the version of the tool used to steer the contextual interviews.

Figure 1. Mockups of the Potential Simple Composition Tool –ServFace Builder-

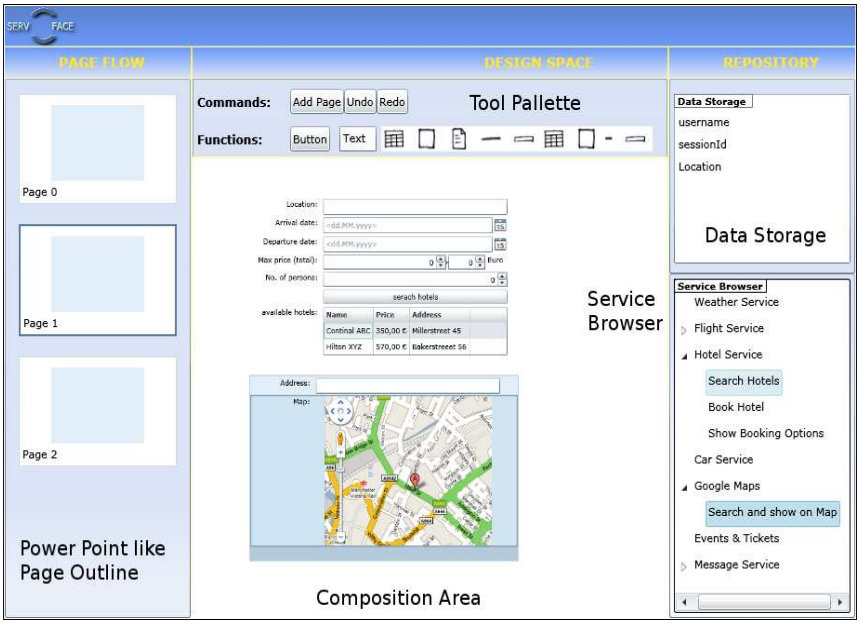

Figure 2. Early Prototype of the Simple Composition Tool –ServFace Builder-

4 Results

4.1 User Perception of Software Services

Users provided diverse definitions of web services. 20% of the users defined services as services that are available on the web, while another 20% of the users defined services as the provision of information or knowledge. The remaining users referred to services as: a tool to build the web and applications, online communities, search engines, and interactive elements. 40% of the 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 46% of the 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).

Once the concept of services was introduced to the participants and the Google Map was presented as an example, users 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.

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. 33% of the 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.

To sum up users were able to provide a very general definition of services abstracted from technical details and describe the functionality of web services, and web applications. Users seem to perceive service-hosting sites as single services instead of a collection of web services as shown by the provided examples. However, users had no knowledge of the term "widgets". Users heard about Web 2.0 and already used its technologies like blogs. Surprisingly many of them had already built web sites by their own.

4.2 User Perception of Service Composition

All users liked "to develop their own software 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

4.3 End User Composition Problems

As previously indicated the main objective of our evaluation is to identify conceptual problems that can be generalized to other mashup editors and service composition environments and propose measurements to resolve them. Motivated by the travel booking scenario, participants pinpointed several drawbacks with the lightweight composition. Table 1 lists and explains both conceptual and usability problems that could face end users while composing services. Other usability problems that are very specific to the ServFace Builder only are not reported below. We include a severity rating for each detected problem using an ordinal scale (low, moderate, high).

Table.1. Conceptual and Usability Problems of Service Composition by End Users

Conceptual Problem	**Severity**	**Usability Problem**	**Severity**
1- Awareness of service composition/connection: despite introducing the concept of "building applications by users" in the walkthrough example, some participants had problems understanding the purpose of the composition tool (i.e. that the tool is designed to build applications). Instead, they thought single services are software systems which operate independently. All users failed to notice that web services can be connected together. After we informed them about the possibility of combining services together and asked whether they want the tool to perform it on their behalf, all users preferred to be involved in the process of combining services because they do not have full confidence in the system and feared it might cause problems.	High	**1- Direct manipulation of services**: selecting and placing services into the main canvas was not intuitive to the participants and caused problems. Upon placing services into the design area, users had difficulty trying to move them around to create an organized visual layout. Moreover, users wanted to adjust the size of services layout but this is currently not supported by the tool.	Moderate

2- Definition of execution flow of services and application: users were confused about specifying the execution order of the services they added to the design space. In other words, they had troubles defining which service should the application start with and which one should come next, and so forth.	High	**2- System support**: users were also not sure if they were doing the right actions and complained that the system did not inform them about the consequences of their activities, emphasizing that proactive help from the system is important.	High
3- Understanding of technical terms: users were intimated by some technical jargon used in the tool and their meaning such as service operation and parameters, and asked for explanation. Inability to understand parts and concepts used within a design tool may result in users giving up on the tool.	High		
4- Security: there was a security concern from users in relation to using web services that require supplying sensitive information such as: bank details. Users were worried that services retrieved by the tool could disclose their personal information to third party service providers or could be compromised by experienced intruders and hackers.	High		
5- Distinction between design and runtime: users had difficulty understanding the difference between design time and run time. Some users started to input data into entry fields of the services during the development phase and expected the application to process results instantly. Clearly they had misconceptions about the two phases.	Moderate		

5 Discussion and Recommendations for End User Composition

Although users were some times confused about the purpose of the tool, they showed a high likeability towards "composing applications that are tailorable to their needs". This agrees with the current trends that end users are becoming proactive about developing the web [8]. Assembling various services within a single application was favored by the participants because it saves time and effort, is convenient, and offers multiple functionalities, agreeing with [6].

The striking result of this evaluation revealed that the tool was not self-reflective of its composition aspect as users did not attempt to create links between services in the task scenario. This may be attributed to the innovative idea of combining different software components together which end users are unfamiliar with. In contract to customizable web portals (e.g. iGoogle) and social networking sites (e.g. Facebook), where users usually search for and add external services to their pages without having to define relationships between services, this design tool and the alike require users to wire atomic services together. Therefore, it is anticipated that users were unaware of this new composition feature.

An challenge imposed by this tool is how end users with minimum service composition experience can specify the steps they are required to undertake in order to build an application. In particular, it is not an obvious task for users to recognize the services that contribute towards the accomplishment of a particular user goal.

Surprisingly adopting common practices in future design tools does not necessarily improve user experience and their usability. For example, although some users were able to link the page flow section on the right hand side of our service composition tool to the Microsoft Power Point slides section, they showed poor ability to use it appropriately. It is important for designers to come up with and compare several design solutions before committing to a particular design solution.

Other aspects of the tool that created confusion were related to the unfamiliar service computing terms, such as: service operation, parameters. This can be attributed to user unfamiliarity and poor knowledge of technical terms.

In light of the results of user understanding of software services and service composition and identified service composition problems the following tentative design recommendations are suggested:

- Service understanding and representation in service composition environments: users showed a poor understanding of the technical details of web services; thus, we encourage service designers to represent services via user interfaces to naïve users since visual representations can communicate and express the purpose and details of these services more effectively. Whilst abstract representations of services (i.e. black box representations) are difficult to interpret and understand by non-programmers, snippets of code are designed for serious programmers.
- Service composition strategy: there are two fundamental issues to service composition (1) connecting services and (2) identifying the order by which services should be executed. To realize service composition by end users we propose to use a semi-automatic approach (system-guided composition) that seamlessly creates links between services while giving users the power to modify those links as they see most appropriate. Whenever a new service is selected from a list of available services and added to the design space (e.g. service 3, Figure 3), the system should check for service compatibility issues and create the desired connection (Serv 3 and Serv 19, Figure 3). The system may also highlight the possible links from a new added service to other existing services. For the second issue we recommend to use a task modeling view by which users can indicate the goal of their application and the tasks/actions they are to required to perform in order to accomplish their goal. Once the task analysis tree has been created users can associate services to particular actions. The overall aim is to specify the services that contribute towards the accomplishment of a user goal without

worrying about service connections. Furthermore, it is would be very useful if a library containing several task analysis templates of possible assemblies are made available. Users could then reuse and extend these templates according to their specific needs.

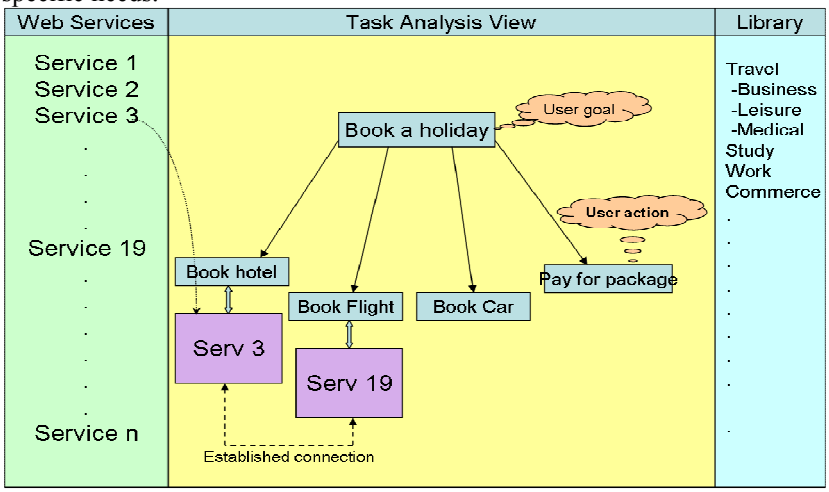

Figure 3. A Potential Service Composition Design Solution

- Service related terms: technical jargon (i.e. service operations, parameter, widgets) is not well understood by ordinary users; therefore, we propose using friendly and self-explanatory titles to elevate technical complexity and enhance users understanding of service composition aspects.
- Manipulation of services: service composition environments should provide a suitable and large design area. In addition, users should be enabled to easily interact with and visually manipulate services (i.e. moving services within the canvas, changing their dimensions, color, deleting services … etc).
- Secure services: in addition to composition-related issues, the system has to deal with security and privacy aspects. In that respect, the retrieved services must be trustworthy and reliable and this should be clearly communicated to the users through symbols (e.g. a secure digital e-sign, verisign identity protection), as well as providing guarantees from service providers to compensate service consumers in case of frauds.
- Continuous user feedback: users were sometimes inquisitive about the current state of their design; hence, we suggest adding proactive assistance (e.g. intelligent software agents) that continuously gives assurances and notifies users about the consequence of their actions, especially in critical situations.

Further to these general design guidelines, we noticed that most of our users designed their application in one page instead of multiple pages. Therefore, we recommend enabling service composition within one design page as it is more convenient and less confusing. In regard to the design tool's name, users suggested using names that reflect their personality and give them a feeling of ownership and control over the tool such as: "myTool, myApplication, youDesign … etc".

6 Conclusion and Future Work

This paper discusses users understanding of services and service composition, and the main issues that end users with no modeling skills or programming knowledge may face during the composition of services using service development environment. Users liked the idea of being able to build their own applications but evidence showed they were not thinking about linking services together to form augmented assemblies. We therefore propose to support service composition environments with intelligent mechanisms that automatically define connections between services (e.g. control and data flow) "system-driven composition" while enabling users to control and customize these connections. In future work, we plan to improve the current service composition tool and address some of the identified and inferred 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 latest version of the ServFace Builder.

Acknowledgments. We thank all students of the University of Manchester who participated in this research study. The work presented herein is supported by the EU-funded project ServFace.

References

1. Alonso, G., Casati, F., Kuno, H., Machiraju, V.: Web Services: Concepts, Architectures, and Applications. Springer Verlag. (2004)
2. Beyer, H. Holtzblatt, K.: Contextual Design: Defining Customer-Centered Systems. San Francisco: Morgan Kaufmann Publishers (1998)
3. Daniel, F., Casati, F., Benatallah, B., Shan, M.C.: Hosted Universal Composition: Models, Languages and Infrastructure in mashArt. In Proceedings of ER'09 (2009)
4. Hoyer, V., Fischer, M.: Market Overview of Enterprise Mashup Tools. In Proceedings of ICSOC (2008)
5. Janeiro, J., Preussner, A., Springer, T., Schill, A., Wauer, M.: Improving the Development of Service Based Applications Through Service Annotations. In Proceedings of WWW/Internet (2009)
6. Namoune, A., Wajid, U., and Mehandjiev, N.: Composition of Interactive Service-based Applications by End Users. In the Proceedigs of UGS2009 - 1st International Workshop on User-generated Services at ICSOC'09, Stockholm (2009)
7. Nestler, T., Dannecker, L., Pursche, A.: User-centric composition of service frontends at the presentation layer. In the Proceedigs of UGS2009 - 1st International Workshop on User-generated Services at ICSOC'09, Stockholm (2009)
8. O'Reilly, T.: What Is Web 2.0?. Retrieved 20 April 2010, from http://oreilly.com/web2/archive/what-is-web-20.html.
9. Ro, A., Xia, L.S.Y., Paik, H.Y., Chon, C.H.: Bill Organiser Portal: A Case Study on End-User Composition. In Proceedings of WISE (2008)
10. Schahram Dustdar and Wolfgang Schreiner.: A Survey on Web Service Composition. In International Journal of Web and Grid Services, Vol. 1, No. 1. (2005)
11. Wong, J., Hong, J. I.: Making Mashups with Marmite: Towards End-User Programming for the Web. In Proceedings of CHI (2007)