Guidelines for Efficient and Effective End-User Development of Mashups

Saeed Aghaee and Cesare Pautasso

Faculty of Informatics, University of Lugano (USI), Switzerland
first.last@usi.ch

Abstract. End-User Development (EUD) is an emerging research area aiming at empowering non-technical users to somehow create or design software artifacts. Mashups provide a high potential for EUD activities on the Web. Users on the Web can tap into a vast resource of off-the-shelf components in order to rapidly compose new lightweight software applications called mashups. In this paper, we provide a set of guidelines to design EUD systems for mashups that are widely referred to as *mashup tools*. The guidelines are derived from our experience with the (ongoing) design and evaluation of `NaturalMash`, a novel mashup tool targeted for a wide range of users to create feature-rich mashups.

Keywords: Mashup, End-User Development, Mashup Tool

1 Introduction

Mashups are a popular type of Web applications built out of the composition of heterogeneous components available through the Web [1]. End-User Development (EUD) [2] of mashups aims at exploiting them by a wide range of users on the Web. EUD systems for mashups are referred to as *Mashup tools*. They usually provide users with an intuitive composition language and environment for the code-free development of mashups. Despite the excessive number of mashup tools emerging form both academia and industry [3], only a few of them have been successful. Tools from academia are mostly research prototypes that rarely reach a large user community. Also, many industrial mashup tools, such as Microsoft Popfly and Google Mashup Editor have been discontinued.

The designers of mashup tools face a number of key challenges, including the need for defining high level, intuitive descriptions of computations and integration logic to be combined with suitable abstractions to represent Web widgets, Web services and Web data sources as reusable components. In this paper, based on our experience with the (ongoing) design and usability evaluation of `NaturalMash` [4], we propose a set of guidelines informing the design of next-generation mashup tools.

2 NaturalMash: A Natural Mashup Tool

`NaturalMash` (Figure 1) is a mashup tool targeted to support a wide range of users, including specially non-programmers, to build sophisticated, feature-rich

mashups. The goal behind the design of `NaturalMash` was to enable an optimal learning experience, where not only the challenges imposed by the activities at hand are properly balanced with the users's skills, but also the users gradually learn to master new challenges and skills [5]. Also, we intended to ensure an intuitive form of interaction that users feel comfortable and familiar with.

To these ends, we have followed a formative user-centered approach in order to have access to users' feedback at a very early stage of development and make sure users were kept central in our design so as to avoid as much as possible mismatches between users' expectations versus system behavior.

`NaturalMash` introduce a novel composition technique based on What You See Is What You Get (WYSIWYG), Programming by Demonstration (PbD) [6], and controlled natural language programming [7]. The WYSIWYG interfaces, augmented with PbD (visual field), enables a natural means for the design and manipulation of the presentation layer, and partially the application logic layer, of mashups. Natural language programming compensates the shortcoming of WYSIWYG and PbD to express the application logic of mashups. The choice of structured natural language also fits with the need to ease the tool's learning curve. The mashup descriptions written in natural language are readable and understandable by any English-speaking end user.

One important deficiency of programming in structured natural languages, however, is that it is cumbersome to learn the restrictions of these languages. To overcome this issue, `NaturalMash` introduces an autocomplete menu that provides suggestions as the users types in the text field. Autocompletion also enables incidental learning. While browsing the suggestions provided by the menu, users gradually discover what are the available components that can be mashed up.

`NaturalMash` incorporates a stack bar that represents the components that can be used as the "ingredients" for the mashup. The stack gives an overview of the functionalities offered by the tool, representing them with an icon and a name, thus making them easy to recognize and memorize. From there, users can drag-and-drop ingredients on the visual field or text field. The list of components used by the mashup being created is shown by the component dock.

In the design of `NaturalMash`, we have adapted the paradigm of live programming based on WYSIWYG, in which the life-cycle of edit/compile/run is fully automated by the system. As a result, users will be able to more easily bridge the gulf of evaluation (the degree of difficulty of assessing and understanding the state of the system [8]) to incrementally validate and learn from their small steps one at a time. This in turn leads towards an optimal learning experience.

To non-professional users, data flow is a very challenging concept to grasp when learning how to develop mashups [9]. `NaturalMash` semi-automates the data flow design using a semantic data integration framework.

Up to now, we have successfully completed two design iterations and evaluations. The results of the two usability evaluations were promising and provided us with valuable analytical insights on the ongoing design of `NaturalMash`. In this paper, we used the results to develop a set of design guidelines that will be presented in the next section.

Fig. 1. `NaturalMash` environment: the ingredients stack gives an overview of the tool functionalities (available mashup components). From there, users can drag-and-drop ingredients to start building mashups. Alternatively, they can use the text field to discover desired components. With the help of the autocomplete menu, users type in the text field the imperative description of the mashup in natural language, and immediately see the result in the visual field.

3 End-User Development of Mashups: Guidelines

The design of effective and efficient mashup tools requires to trade-off abstraction against expressiveness and generality against specificity. From a technical perspective, one of the main tasks of mashup tools is to hide the heterogeneity and complexity of Web technologies behind an easy-to-understand abstraction. From a user modeling perspective, the challenge lies in the broad diversity of user skills that need to be targeted and in the large number of domains in which mashups can be applied to.

In the following we enumerate a set of guidelines resulting from our experience with the design and evaluation of `NaturalMash`. We believe these guidelines can help addressing the mentioned tradeoffs towards the design of efficient and effective mashup tools letting non-professional users create sophisticated mashups.

– **Use natural metaphors:** Bringing the user interface closer to the user's way of thinking and working can significantly increase usability. This can be achieved by, for instance, reconciling the semantics of the presentation objects with the semantics of the target application domain (domain-specific metaphors), or using metaphors that are well understood by non-professional users. For example, a recent study on the understandability of service composition languages [10] shows that the visual wiring paradigm (i.e., visual control flow and data flow diagrams) that is widely utilized by existing mashup tools, is not a familiar metaphor for non-professional users.

The `NaturalMash` interface adopts the generic metaphor of visual context and textual content, where the visual field contains the actual mashup output and

the text field consists of the imperative descriptions of the output in natural language. Provided that the imperative natural language descriptions are abstract enough, this metaphor is understandable by almost everyone. This hypothesis was also supported by the results from our user studies.

– **Design at meta-level:** Mashups can be built and used in different domains of applications. These domains range from daily utilities of Web users (i.e., consumer market) to narrowly specialized domains and enterprise environments. It is important to identify the application domain in which users are willing to and have shown a clear need to develop mashups [11]. This is a well known problem in EUD, as the importance of task and domain specificity was already pointed out by Nardi [12] in the context of end-user programming. From the point of view of the tool designer, a closed approach which narrowly targets a single application domain may present some limitations. Application domains usually change over time. This may results in changes of the initial requirements and assumptions based on which the mashup tool was designed. Also, a mashup tool targeting a specific domain may not perfectly fit into, or be easily transformed into, a tool targeting another domain. Therefore we advocate a meta-design approach [13], whereby a generic mashup meta-tool is designed and from it domain-specific mashup tools can be derived by it users.

The meta-design elements in `NaturalMash` include (i) selection of the available ingredients (components), (ii) the look and naming of ingredients, and most importantly (iii) the language style used to describe them. In the latter case, the description of each ingredients used in the text field can be changed by users to tailor the "language" of the tool to their domain.

– **Support different levels of expressiveness:** Being able to only create so-called "toy" mashups is the main criticism against existing mashup tools. An effective mashup tool should provide enough expressive power to allow the creation of sophisticated mashups. On the other hand, the usability of a system may be affected by the degree of expresivenss it offers. To avoid this issue, [14] proposes three levels of user tailoring including customization, integration, and extension. In case of mashups, all these three levels are relevant and thus should be supported by a mashup tool. Customization means modifying an existing mashup through parameterization or user interface manipulation. Integration is the process of creating new mashups and should be allowed at all the levels of data, business logic, and presentation tiers. Extension allows extending the functionality of the mashup tool by developing new components.

In `NaturalMash`, customization is enabled through the visual field. Integration is mainly supported by natural language programming. The plan is to also enable extension for professional users to create and add components the tool library.

– **Avoid complex user interfaces:** Simple user interfaces can largely decrease the learning cost. We emphasis simplicity in terms of elements, content, and language used in the user interface of a mashup tool. Many existing mashup tools have a complex tab-based environment with nested user interface elements. They also commonly use very technical terms (e.g., "regular expressions", "mashup components", etc.) in their user interface.

These are issues that we tried to avoid in the design of NaturalMash. We intended to keep the user interface as simple and easy-to-use as possible. As illustrated in Figure 1, the interface is composed of only four non-nested components (visual field, text field, component dock, stack). We also attempted to use non-technical and easily understandable terms in the interface (e.g., "ingredients" vs. "component library").

– **Build an online community:** Online communities are of importance in boosting the ability of users to learn how to use the tool through creating, sharing, and reusing mashups, knowledge, and experience [12]. Crowdsourcing can also be applied in an online community to persuade professional users to enrich the component library for non-professional users [15].
We plan to investigate the mentioned impacts of online communities in the context of mashup EUD. More importantly, we are interested in in-the-wild testing of our meta-design using an online community.

– **Adopt user-centered design:** We have realized that a formative user-centered design is a promising method to design "natural" mashup tools. This method ensures receiving early feedback from users and applying it on every step of the design process.

4 Related Work

Stemming from both academic and industrial research and development, a number of mashup tools have been designed. Most of existing industrial mashup tools, such as Yahoo! Pipes (http://pipes.yahoo.com/), IBM Mashup Center (http://www.ibm.com/software/info/mashup-center), and JackBe Presto (http://www.jackbe.com/), as well as early academic tools (e.g., Marmite [16]) are designed for expert users with advanced technical knowledge.

Recently, however, the attempt in academia has been to design mashup tools supporting non-programmer users as well. For instance, ServFace builder [17], DashMash [18], and RoofTop [19] provide a full WYSIWYG approach, which, however, does not provide as much expressive power as NaturalMash (thanks to natural language programming). In terms of natural language programming, IFTTT (https://ifttt.com) is a similar system, which, even though it is solely based on natural language, restricts the user's input using a structured visual editor. Also, IFTTT only allows to create mashups based on a single control-flow pattern (if this then that).

5 Conclusion

Mashups provide a vast potential for EUD activities on the Web. In this paper, we proposed a set of guidelines to design next-generation EUD systems for mashups (mashup tools). The guidelines were a result of our experience with the design and evaluation of NaturalMash. The next big step is to bring the tool to the real world and conduct in-the-wild testing.

Acknowledgments This work has been supported by Swiss National Science Foundation with the SOSOA project (SINERGIA grant nr. CRSI22_127386).

References

1. Benslimane, D., Dustdar, S., Sheth, A.: Services mashups: The new generation of web applications. IEEE Internet Computing **12** (2008) 13–15
2. Lieberman, H., Paternò, F., Klann, M., Wulf, V.: End-User Development: An Emerging Paradigm. In: End User Development. Springer Netherlands (2006)
3. Aghaee, S., Nowak, M., Pautasso, C.: Reusable Decision Space for Mashup Tool Design. In: Proc. of EICS. (2012)
4. Aghaee, S., Pautasso, C.: EnglishMash: Usability Design for A Natural Mashup Composition Environment. In: Proc. of ComposableWeb. (2012)
5. Repenning, A., Ioannidou, A. In: What Makes End-User Development Tick? 13 Design Guidelines. Springer Verlag (2006)
6. Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A., eds.: Watch What I Do: Programming by Demonstration. MIT Press (1993)
7. Petrick, S.R.: On Natural Language based Computer Systems. IBM J. Res. Dev. **20** (1976) 314–325
8. Norman, D.A., Draper, S.W.: User Centered System Design; New Perspectives on Human-Computer Interaction. L. Erlbaum Associates Inc. (1986)
9. Mehandjiev, N., Lecue, F., Wajid, U., Namoun, A.: Assisted Service Composition for End Users. In: Proc. of ECOWS 2010. (2010)
10. Namoun, A., Nestler, T., Angeli, A.D.: Service Composition for Non-programmers: Prospects, Problems, and Design Recommendations. In: Proc. of ECOWS. (2010)
11. Casati, F., Daniel, F., Angeli, A.D., Imran, M., Soi, S., Wilkinson, C.R., Marchese, M.: Developing Mashup Tools for End-Users: On the Importance of the Application Domain. IJNGC **3** (2012)
12. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press (1993)
13. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N.: Meta-design: A Manifesto for End-User Development. Commun. ACM **47** (2004) 33–37
14. Mørch, A.: Three Levels of End-user Tailoring: Customization, Integration, and Extension. In: Computers and Design in Context. MIT Press (1997)
15. Nebeling, M., Leone, S., Norrie, M.C.: Crowdsourced web engineering and design. In: Proc. of ICWE. (2012)
16. Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: Proc. of CHI 2007. (2007)
17. Nestler, T., Feldmann, M., Hubsch, G., Preussner, A., Jugel, U.: The ServFace Builder - A WYSIWYG Approach for Building Service-based Applications. In: Proc. of ICWE. (2010)
18. Cappiello, C., Matera, M., Picozzi, M., Sprega, G., Barbagallo, D., Francalanci, C.: DashMash: A Mashup Environment for End User Development. In: Proc. of ICWE. (2011)
19. Hoyer, V., Gilles, F., Janner, T., Stanoevska-Slabeva, K.: SAP Research RoofTop Marketplace: Putting a Face on Service-Oriented Architectures. In: Proc. of SERVICES. (2009)