End-User Programming for Web Mashups
Open Research Challenges

Saeed Aghaee and Cesare Pautasso*

Faculty of Informatics, University of Lugano, Switzerland
first.last@usi.ch
http://www.pautasso.info/

Abstract. Mashup is defined as the practice of lightweight composition, serendipitous reuse, and user-centric development on the Web. In spite of the fact that the development of mashups is rather simple due to the reuse of all the required layers of a Web application (functionality, data, and user interface), it still requires programming experience. This is a significant hurdle for non-programmers (end-users with minimal or no programming experience), who constitute the majority of Web users. To cope with this, an End-User Programming (EUP) tool can be designed to reduce the barriers of mashup development, in a way that even non-programmers will be able to create innovative, feature-rich mashups. In this paper, we give an overview of the existing EUP approaches for mashup development, as well as a list of open research challenges.

1 Introduction

Facilitating software development from reusable components has always been one of the priorities in software engineering [1]. Recently, the proliferation of reusable Web resources, in the form of Web APIs, Web widgets, and Web data sources, has again brought up the notion of reuse within Web engineering with the advent of Web mashups.

The key characteristic of mashups, distinguishing them from other forms of service and software composition, lies in a development approach being carried out in a *lightweight* manner, in which simplicity and usability are more of a priority than quality and completeness [2]. This enables end-user composition activities, in which ordinary Web users are themselves the developers of creative mashups, which can fulfill their personal needs, and can be rapidly adapted as soon as their situational needs change [3]. However, developing mashups still requires significant technical skills. These range from knowing how to reuse components, to at least a basic understanding of programming and familiarity with Web technologies. Yet, such skills by definition are not mastered by non-programmers.

To address the above challenges, one solution is to reuse and adapt existing mashups that, thanks to directories such as ProgrammableWeb [4], can be

* PhD supervisor

easily discovered and shared. However, this should be complemented by leveraging End-User Programming (EUP) [5] to reduce the complexity of mashup development as much as possible, to the extent that even non-programmers can develop and share their desired mashups. In doing so, mashups can reach their full potential to serve as user-centric situational applications on the Web, from which the vast majority of Web users can benefit.

Our research objective is to design, implement, and evaluate a EUP approach for mashups that satisfies the following two requirements. 1) supporting the needs and abilities of non-programmers. 2) enabling creation of any types of mashup that can be developed using manual approaches (i.e. programming and scripting languages). To this end, in the rest of the paper, we will provide a brief survey of existing EUP approaches for mashups, and further discuss a number of open challenges, which are not yet fully solved by the state of the art.

2 Overview of Existing EUP Approaches for Mashups

The research efforts behind the design of EUP approaches for mashups (so-called "mashup tools") have resulted in the growth of this field as an interesting research topic spanning areas including Model-Driven Development (MDD) [6], programming languages [7], software and service composition [8], and Human-Computer Interaction (HCI) [9]. Existing mashup tools can be classified according to the EUP technique [10] they utilize as follows:

– **Spreadsheets.** The advantage of using spreadsheets for creating mashups lies in its ease-of-use, intuitiveness, and expressive power to represent and manage complex data [11]. Mashroom [12] adapts the idea of spreadsheets and adds the nesting tables feature to support complex data formats such as XML and JSON. Husky [13] is also another spreadsheet-based tool aiming at streamlining service composition. However, the main shortcoming of such tools is the lack of support for designing the mashup User Interface (UI).

– **Programming by Demonstration (PbD).** PbD enables users to teach a system to do a task by demonstrating how the task is done [14]. Intel Mash Maker (IMM) [15] utilizes PbD to extract, store, manage, and integrate data from the Websites being browsed by the user. Vegemite [16] is another browser-based tool like IMM which adds scripting capabilities. The use of scripting allows users to augment and operate the extracted data. The focus of these tools are more on data extraction and visualization, and therefore, they do not provide support for service composition and orchestration.

– **Domain-Specific Language (DSL).** DSLs are small languages targeted for solving certain problems in a specific domain. DSLs can also be used as a EUP technique for reducing programming efforts [17]. The Enterprise Mashup Markup Language (EMML) [18] is a DSL based on XML for creating mashups. It supports variety of components as well as the use of scripting languages. Swashup [19] is also another DSL for mashups, based on Ruby-on-Rails. It simplifies invocation, and integration of Web APIs and data sources. Though these DSLs help to reduce programming efforts, they still can not be used by non-programmers due to the difficulty of learning their syntax and vocabulary [14].

– **Visual Programming.** Programming languages can also be expressed by visual symbols and graphical notations [20]. Visual programming is widely used by existing mashup tools in the form of wiring diagrams, in which users drag-and-drop mashup components (visualized as boxes) and connect them to form a mashup. Examples are Yahoo Pipes (YP) [21], IBM Mashup Center (IMC) [22], ServFace [23], and Presto Cloud [24]. The main problem of these tools, according to a recent study conducted by Namoun et. al., is the fact that the wiring paradigm is difficult to understand by non-programmers [25].

– **Model-based Automation.** This is concerned with automatically creating mashups based on knowledge about the user and the context in which she operates. Due to the fact that there is much more work on the tool side, this technique best serves the needs of non-programmers. The framework proposed by Carlson et. al. automatically creates mashups out of non-web service components [26]. Bakalov et. al., on the other hand, present an automatic mashup generation framework that is also capable of composing Web services (REST and SOAP) [27]. As described in [5], the problem of this technique lies in the high risk of generating irrelevant mashups with respect to the given requirements.

3 Open Research Challenges

– **Simplicity and Expressive Power Tradeoff.** When it comes to create complex mashups, the majority of existing mashup tools are not powerful enough. This can be witnessed by the fact that most of the registered mashups in the ProgrammableWeb are all developed using general-purpose Web scripting languages. If these are called *real* mashups, the majority of current EUP tools are limited to creating *toy* mashups that are not as feature-rich. On the other hand, increasing the expressive power of mashup tools (e.g., DSLs) can potentially result in a decrease in simplicity (gentle learning curve, and ease-of-use). Hence, the major challenge is to cope with this tradeoff.

– **Mashup Components Heterogeneity.** Mashup components are heterogeneous in terms of the technology through which they are made accessible. They can be classified into Web APIs, Web widgets, and Web data sources. Within enterprises, another class of mashup components may encompass legacy services such as databases, Plain Old Java Object (POJO), and Enterprise Java Beans (EJB). The challenge is how to abstract all these heterogeneous components in a way that facilitates their seamless composition.

– **Mashup Composition Techniques.** The development of mashups consists of Process Integration (PI), Data Integration (DI), and UI integration [28]. PI forms the logic of mashup by composing the functionality obtained from Web APIs. UI integration creates the visual front-end, by which users interact with the mashup. This is obtained through integration of various widgets [29]. The underlying data model of the mashup is obtained by the integration of two or more remote data sources [30]. A challenge for mashup tools is to fully support development within all of these three levels.

– **Mashup Evolution.** Mashup evolution can be caused by two reasons. The first is the change in the user requirements, which forces the mashup to be reengineered to meet the new ones [31]. The other is the evolution of the building blocks of the mashup, that in case of Web services is very likely to happen. From the mashup EUP perspective, this has however remained a challenging matter.

– **Online Communities.** Empowering end-user communities is of value in the area of EUP [5]. With the growth of the Web 2.0, online communities and social networks can be used to promote sharing of mashups, technical discussion, and collaborative categorization [32]. Yet, only a few mashup tools, such as YP, offer online communities. Moreover, the potential of these communities to enable mashup development as a collaborative process has still to be fully exploited.

4 Conclusion and Future Work

This paper provides an overview and classification of existing approaches and open research challenges for enabling EUP for mashups. Our future research will be geared towards addressing these challenges by designing, implementing, and evaluating a novel mashup tool. To do so, we will utilize a User-Centered Design (UCD) methodology [33], in which the end-user needs and feedback affect every step of the design process. Getting closer to the mindset of the end-users can help with the design of a more *natural* and *powerful* EUP tool for mashups [34].

References

1. Mcllroy, D.: Mass-produced Software Components. In: Software Engineering Concepts and Techniques, NATO Science Committee (1969) 138–155
2. Yu, J., Benatallah, B., Casati, F., Daniel, F.: Understanding Mashup Development. IEEE Internet Computing **12** (2008) 44–52
3. Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More. Hyperion (2006)
4. ProgrammableWeb. (Available at http://www.programmableweb.com/)
5. Nardi, B.A.: A Small Matter of Programming: Perspectives on End User Computing. MIT Press (1993)
6. Bozzon, A., Brambilla, M., Facca, F.M., Carughu, G.T.: A Conceptual Modeling Approach to Business Service Mashup Development. In: Proc. of ICWS 2009. (2009)
7. Ennals, R., Gay, D.: User-Friendly Functional Programming for Web Mashups. In: Proc. of ICFP 2007. (2007)
8. Lopez, J., Bellas, F., Pan, A., Montoto, P.: A Component-Based Approach for Engineering Enterprise Mashups. In: Proc. of ICWE 2009. (2009)
9. Wong, J., Hong, J.: What do we "mashup" when we make mashups? In: In Proc. of WEUSE 2008. (2008) 35–39
10. Myers, B.A., Ko, A.J., Burnett, M.M.: Invited Research Overview: End-User Programming. In: Proc. of CHI 2006. (2006)

11. Hoang, D.D., Paik, H.y., Benatallah, B.: An Analysis of Spreadsheet-Based Services Mashup. In: Proc. of ADC 2010. (2010)
12. Wang, G., Yang, S., Han, Y.: Mashroom: End-User Mashup Programming Using Nested Tables. In: Proc. of WWW 2009. (2009)
13. Husky. (Available at http://www.husky.fer.hr/)
14. Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A., eds.: Watch What I Do: Programming by Demonstration. (1993)
15. Ennals, R., Brewer, E., Garofalakis, M., Shadle, M., Gandhi, P.: Intel Mash Maker: Join the Web. SIGMOD Rec. **36** (2007) 27–33
16. Lin, J., Wong, J., Nichols, J., Cypher, A., Lau, T.A.: End-User Programming of Mashups With Vegemite. In: Proc. of IUI 2009. (2009)
17. Prähofer, H., Hurnaus, D., Mössenböck, H.: Building End-User Programming Systems Based on Domain-Specific Language. (2006)
18. EMML. (Available at http://www.openmashup.org/)
19. Maximilien, E.M., Wilkinson, H., Desai, N., Tai, S.: A Domain-Specific Language for Web APIs and Services Mashups. In: Proc. of ICSOC 2007. (2007)
20. Shu, N.C.: Visual Programming. Wiley (1992)
21. Yahoo Pipes. (Available at http://pipes.yahoo.com/pipes/)
22. IBM Mashup Center. (Available at http://www.ibm.com/software/info/mashup-center)
23. 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. (2010)
24. Presto Cloud. (Available at http://www.jackbe.com/enterprise-mashup/)
25. Namoun, A., Nestler, T., Angeli, A.D.: Service Composition for Non-programmers: Prospects, Problems, and Design Recommendations. In: Proc. of ECOWS 2010. (2010)
26. Carlson, M.P., Ngu, A.H., Podorozhny, R., Zeng, L.: Automatic Mash Up of Composite Applications. In: Prof. of ICSOC 2008. (2008)
27. Bakalov, F., Konig-Ries, B., Nauerz, A., Welsch, M.: Ontology-Based Multidimensional Personalization Modeling for the Automatic Generation of Mashups in Next-Generation Portals. In: Proc. of ONTORACT 2008. (2008)
28. Hanson, J.J.: Mashups: Strategies for the Modern Enterprise. Addison-Wesley Professional (2009)
29. Daniel, F., Yu, J., Benatallah, B., Casati, F., Matera, M., Saint-Paul, R.: Understanding UI Integration: A Survey of Problems, Technologies, and Opportunities. IEEE Internet Computing **11** (2007) 59–66
30. Di Lorenzo, G., Hacid, H., Paik, H.y., Benatallah, B.: Data Integration in Mashups. SIGMOD Rec. **38** (2009) 59–66
31. Dorn, C., Schall, D., Dustdar, S.: Context-aware adaptive service mashups. In: Proc. of APSCC 2009. (2009)
32. Grammel, L., Storey, M.A.: An End User Perspective on Mashup Makers. Technical report, University of Victoria (2008)
33. Vredenburg, K., Mao, J.Y., Smith, P.W., Carey, T.: A Survey of User-Centered Design Practice. In: Proc. of CHI 2002. (2002)
34. Myers, B.A., Pane, J.F., Ko, A.: Natural Programming Languages and Environments. Commun. ACM **47** (2004) 47–52