An End User Perspective on Mashup Makers

Lars Grammel[1] and Margaret-Anne Storey[1]

Computer Human Interaction and Software Engineering Lab,
University of Victoria, Victoria, BC, Canada,
{lgrammel,mstorey}@uvic.ca,
http://www.thechiselgroup.org

University of Victoria Technical Report DCS-324-IR, September 2008

Abstract. This paper presents a review of six mashup makers from an End User Development (EUD) perspective. The fast-paced development of mashup makers and related research in the last two years has created a wealth of features and approaches. To provide an overview of EUD support in current state-of-the art mashup makers, we explore, summarize and compare their features across six different themes (Levels of Abstraction, Learning Support, Community Support, Searchability, UI Design and Software Engineering Techniques). We found that the mashup makers provide many features to support end users, but there is still much room for further improvement. These results can be used to guide both research and tool design. User studies that reveal the difficulties in using mashup makers and comparisons of different notations are likely to be especially fruitful research opportunities.

1 Introduction

Mashups are applications that reuse and combine data and services available on the web. They are developed in a rapid, ad-hoc manner to automate processes and remix information. This enables users to explore information in new ways and saves valuable time that may be lost in laborious routine tasks.

Mashup development is a promising End User Development (EUD) application area for several reasons. The web as the underlying platform provides access to a growing number of publicly available services. By using such services, the development effort shifts from traditional programming towards finding high-level services [1], glueing these services together [1] and creating User Interfaces (UIs). These activities involve less technical details than programming, are potentially closer to the problem domain and can be supported better by tools because they are more restricted. The shift towards a development model based on composition is in line with a shift towards rapid, opportunistic development of situated applications with short lifespans aimed at small audiences [1]. This style of development provides direct rewards for the users and avoids difficult technological challenges such as scalability. Furthermore, the web as a platform facilitates sharing mashups and community building.

Mashup makers are development environments for mashups. Since 2006, several commercial mashup makers with a focus on end users, web developers and business users such as Microsoft Popfly, Yahoo! Pipes and IBM Mashup Center have been created. At the same time, research on mashups and mashup makers started. Since then, both mashup research and tools have developed rapidly and many features have been explored. Our goal in this paper is to compare how these techniques support end users in developing mashups. As far as we know, this aspect of mashup makers has not been evaluated yet in a similar way before.

The rest of the report is structured as follows: first, an overview of the background and related work is given. Second, the methodology of this review is described. Third, the results are presented, and fourth, implications and connections to other EUD research are discussed. Fifth, we describe the limitations of this work and what we have done to alleviate them. Finally, we conclude the paper with a summary of our contributions and possibilities for future work.

2 Background and Related Work

Constant changes in usage culture and the evolution of the web make mashup definition challenging. The original definition of mashups refers to web-based applications that mix two or more different web-based data sources [2, 3] and reflects the ideas of remixing services and using the web as the underlying platform. The scope of this definition has been broadened by several authors [1, 4] to different devices and environments. Rapid development, situated applications and end user focus can be considered central concepts to mashups as well [5].

For this review, mashups are defined as *end user driven recombination of web-based data and functionality.* This definition is close to the original definition and also emphasizes our end user viewpoint. It is less restrictive than the original definition because it does not require that mashups are web applications. Mashup makers then are *tools that support users in the development of mashups.*

The related research comes from three research areas: Research on mashups, research on web page customizations and research on EUD in general.

Research on mashups has focussed mostly on tools and methods to support mashup development. Marmite [6] combines a visual data-flow language with incremental execution and preview of intermediate results. Karma [7] is a programming-by-example system that integrates data retrieval, modeling, cleaning, and integration. D.mix [3] supports automatic website-to-webservice mapping and combines selecting service samples from websites with remixing them in a wiki. C3W [8] combines programming-by-example methods for extracting actions from web pages with the spreadsheet paradigm. Through other research on mashups, researchers have explored patterns in mashups [4], the background and experiences of mashup developers [2], and the activity of mashup design [1].

Research on web page customization focuses on modifying and personalizing existing web pages instead of creating new ones. Chickenfoot is such a tool for customizing web pages by scripts that run in the browser [9]. It extends the browsers basic scripting language with special commands for web site customization, especially selecting elements from a web page based on visible patterns.

There is a large body of literature on general EUD research. Several introductions, overviews and books are available (e.g. [10–14]). The concept of a gentle slope of complexity [15] and the model of different barriers in programming systems [16] are especially relevant for this review of mashup makers. Creating systems with a gentle slope of complexity is the concept of allowing the user to learn new functionality step-by-step, without requiring major learning efforts without immediate results [15]. Ko *et al.* identified six types of learning barriers (design, selection, coordination, use, understanding, and information barriers) by studying how non-programmers learned VisualBasic using Visual Studio [16].

3 Methodology

The objective of this paper is to discover, organize and summarize the features and approaches of current mashup makers from an EUD perspective. Our research methodology is a qualitative, exploratory tool analysis.

Six mashup makers developed by major companies such as Microsoft, IBM and Google are reviewed (see Table 1). We group them by the kind of mashup they can generate. **Information mashups** retrieve data from one or several data sources, process that data, and publish the results either as feeds or in widgets. **Process mashups** automate processes by orchestrating services, forms and other resources in a workflow and often include data entry. **Web page customizations** change websites by removing elements, adding additional widgets and changing the UIs of websites.

Each reviewed mashup maker was used and evaluated for about three to six hours. During the evaluation, mashups were created, the different parts of the system were explored and used, and related documentation and tutorials were read. Features relevant to the themes were added to a feature matrix (see Table 2) as they were discovered during this evaluation.

We have chosen to structure the review by six themes. These themes emerge from our research on EUD (*EUD*) [10–16], mashup (*MASH*) [1, 2, 4, 5] and mashup maker research (*MM*) [3, 6–9, 17, 18], and from our previous experience with mashup makers (*EXP*), as indicated by the abbreviations for each theme below. The themes are used to guide the tool evaluations. Each theme has one or more guiding questions.

– **Levels of Abstraction** (*EUD, MASH, EXP*). What levels of abstraction are supported? Which notations are used on the different levels of abstraction? How are they integrated?
– **Learning Support** (*EUD, MASH, MM*). What documentation is available? What support besides the documentation is provided? How is the concept of a gentle slope of complexity supported? What barriers in working with the mashup maker exist?
– **Community Features** (*EUD, EXP*). How do mashup makers support collaboration between the users?

- **Searchability** (*MASH, MM*). Which features help the users in finding relevant mashups and mashup elements?
- **UI Design** (*EXP*). How are users supported in creating mashup UIs?
- **Software engineering techniques** (*EUD*). What debugging support is provided by the mashup makers? What testing features do they offer? Is version control available?

4 Results

The results are presented according to the evaluation themes. A summary of the features in the mashup makers is given in Table 2. Please note that the focus of this paper lies on summarizing the approaches in mashup makers and not on recommending the best mashup makers.

4.1 Levels of Abstraction

Three different levels of abstraction are used to distinguish how much programming and computer knowledge is required. There is a tradeoff between the required knowledge and the range of solutions that can be developed [15]. Since raising the level of abstraction requires encapsulating concepts, the number of choices available on higher levels of abstraction is usually restricted.

On a **high level of abstraction**, no programming knowledge is required, but the flexibility is usually restricted to reusing and parametrizing mashups which are developed by others. Working with a high level of abstraction is supported by the reviewed mashup makers in several ways: *reuse of complete mashups* created by others, high-level *mashup and widget parametrization*, *automatic reuse of data extractors* for websites, and *programming-by-example* concepts, such as extracting data by example and dragging & dropping feeds on widgets.

On an **intermediate level of abstraction**, knowledge about concepts such as dataflow, data types or UI widgets is required, but the technological details of these concepts are encapsulated in notations. Thus what can be changed or created is limited by these notations. The reviewed mashup makers support mashup development on an intermediate level of abstraction mainly using visual Domain Specific Languages (DSLs). *Visual dataflow languages* help the users in the creation of information mashups. *Visual workflow/process orchestration languages* support the creation of process mashups. The visual languages are used in combination with property dialogs for the parametrization of mashup elements. Data sources are mostly accessed through mashup elements in those notations. *Dialog-based wiring of widgets* provides an alternative to visual dataflow languages that saves screenspace, but depends on visual widgets. On an intermediate level of abstraction, there is typically a distinction between design and runtime mode, although the mashup makers try to reduce this disinction, e.g. using previews.

On a **low level of abstraction**, programming knowledge is required, but the greatest flexibility is achieved. *Textual DSL editors* for languages such as HTML,

Table 1. Reviewed Mashup Makers

Mashup Makers for Information Mashups	
Microsoft Popfly (MP)	http://www.popfly.com
Yahoo! Pipes (YP)	http://pipes.yahoo.com
IBM Mashup Center (IMC)	http://www.ibm.com/software/info/mashup-center/
Google Mashup Editor (GME)	http://editor.googlemashups.com/editor
Mashup Makers for Process Mashups	
Serena Mashup Composer (SMC)	http://www.serena.com/Mashups/
Mashup Makers for Web Site Customization	
Intel MashMaker (IMM)	http://mashmaker.intel.com/web/

Table 2. Features of Reviewed Mashup Makers (*Duplicate features in italics*)

Feature	**MP**	**YP**	**IMC**	**GME**	**SMC**	**IMM**
Reuse of Complete Mashups	√	√	√	√	√	√
Mashup and Widget Parametrization	√		√			
Automatic Reuse of Data Extractors						√
Programming-By-Example			√			√
Visual Dataflow Languages	√	√	√			
Vis. Workflow/Process Orchestration Languages					√	
Dialog-based Wiring of Widgets			√			
Textual DSL Editors	√			√	√	
Textual DSLs in Dialog Fields	√	√	√		√	√
Extension APIs	√		√		√	√
Integration of Different Abstraction Levels	√		√		√	√
Tutorials, Help, API Documentation	√	√	√	√	√	√
Discussion Forums	√	√	√	√	√	
Using Shared Artifacts as Examples	√	√	√	√	√	√
Context-Specific Suggestions	√					
Sharing Mashups	√	√	√	√	√	√
Sharing Mashup Elements	√	√	√		√	√
Tagging	√	√	√			
Rating	√	√	√			√
Discussion Forums	√	√	√	√	√	
Artifact-Centered Discussion	√		√			
Social Networks	√					
Text-Based Search	√	√	√			
Browsing Mashups by Structural Properties		√				
Simple Categorization of Mashup Elements	√	√	√		√	
Context-Specific Suggestions	√					
Automatical UI Generation		√			√	
Selecting & customizing UI	√					
Visual UI composition			√		√	√
Textual UI composition				√		
Debugging Output	√	√	√			
Version Control			√		√	

JavaScript and custom scripting languages are used to represent mashups or elements of mashups on this level of abstraction. *Textual DSLs in dialog fields* such as regular expressions enable the flexible configuration of mashup element properties. Providing *extension APIs* allows developers to use integrated development environments such as Microsoft Visual Studio for programming reusable mashup elements.

Due to the tradeoff between the level of abstraction and flexibility in designing mashups, the **integration of different abstraction levels** provides several benefits. Having several abstraction levels allows the user to choose the right level of abstraction for his goal and his skill. It also positively affects the ease of learning the functionality of a system by offering a gentle slope of complexity, and it enables reuse of elements from lower levels of abstraction in higher levels of abstraction.

4.2 Learning Support

Supporting the user in learning to master an EUD system, especially by providing a gentle slope of complexity [15], is a crucial aspect of EUD [14]. Besides the different levels of abstraction and their integration, several other mechanisms are provided by mashup makers to aid users in their learning. **Basic learning features** such as *screencasts and tutorials*, *API documentation* and a *help system* that explains the UI are provided by all reviewed mashup makers. **Community based learning support** is provided in *discussion forums* and through *using shared artifacts as examples.* **Context-specific suggestions** offer the user hints on which mashup elements might be useful based on the current structure of the mashups.

4.3 Community Features

Supporting user communities is essential to the success of EUD tools [10]. Community members provide elements that can be reused by other members, create examples, organize artifacts, and help each other.

Facilities for **sharing** mashups and mashup elements enable the reuse of those artifacts within the community. The *shared mashups* can be run and copied to their workspaces for modification by other users. *Sharing of mashup elements*, e.g. building blocks in MP or feeds in IMC, for reuse on higher or the same abstraction levels provides further areas of reuse.

Collaborative categorization of user-generated mashups and mashup elements helps community members in finding the mashups and mashup elements they search. *Tagging* is a categorization technique in which users, typically the authors, annotate items with keywords called tags to support finding. *Rating* is used in different flavors such as favorites, star-rating, clone-counting and user recommendations to help the user in judging the quality and usefulness of mashups and mashup elements.

Providing **discussion features** such as *general dicussion forums* and *artifact-centered discussion*, e.g. comments on a mashup element, enables the users to exchange ideas, help each other and build a sense of community. Artifact-centered discussions are also a valueable feedback mechanism for the authors and help users in judging the quality and usefulness of an artifact.

Social network systems such as Facebook have gained popularity over the last couple of years. In mashup makers, *social networks* help users to find mashups created by friends, and likely increase the commitment of users to a mashup maker by creating a stronger sense of community.

4.4 Searchability

Searching and finding mashups and mashup elements enable users to employ artifacts created by others. *Text-based search* on title, description and assigned tags is the most common mechanism for finding mashups and mashup elements. The results often include the artifact rating and can be sorted by it. *Browsing mashups by structural properties* such as used mashup elements helps users in finding examples for their own mashup. In visual data-flow editors, a *one layer deep categorization of mashup elements* supports users in finding the right elements. *Context-specific suggestions* can provide the needed elements to the user without requiring him to search. Using discussion features, other users can also point out relevant building blocks and example mashups based on questions.

4.5 UI Design

The mashup UIs determine the usability of the mashups. It is therefore important that end users can easily create appropriate UIs for their mashups. The mashup makers provide several different mechanisms for UI design. *Automatically generating the UI* based on the output data of the mashup, e.g. map viewers or image lists, is a convenient way for rapid UI generation, but lacks customizability. *Selecting and customizing* (MP) a single UI component provides more, although still limited, flexibility. The most flexible approach is *UI composition*, either *visual* or *textual*. It enables users to customize UIs to a great extent, but is more complicated than the other approaches.

4.6 Software Engineering Techniques

Given the concerns regarding security and correctness of applications developed by non-programmers [19], providing software engineering techniques for end-users is considered important in the EUD field [12]. **Debugging** is only supported through *consoles* that print debugging output. **Version control** is rather simple and does not contain advanced functions like branching. **Testing** and **change request management** are not supported by any reviewed mashup maker.

5 Discussion

The reviewed mashup makers have many features that improve usability and learnability. Their support for online communities is especially good. However, there are several potential areas of improvement and future research.

Although the mashup makers provide many notations for different tasks and different complexity levels, as well as a variety of learning support features, high learning barriers still remain, especially between different levels of abstraction and different notations. These barriers can be lowered by including more specialized, intermediate notations [15], as for example in MP and IMC.

Further research is required to compare different notations for the same tasks, e.g. visual dataflow languages vs. dialog-based wiring. Evaluation frameworks such as the Cognitive Dimensions of Notations [20] can be used to analyze the strengths and weaknesses of such notations, and formal user studies can compare how users perform on the same tasks using different notations.

Evaluating mashup makers according to the model of six types of barriers in end user programming systems by Ko *et al.* [16] can be used to identify the areas that need most improvement, as well as typical problems users encounter in developing mashups. This form of evaluation can be done via user studies.

An interesting feature that could suppport learning and finding is context-specific suggestions. It is not clear to what extent such suggestions are used, how they fit the needs of the users, nor what their limitations are.

Programming-by-example [11] is used successfully for website data extraction [3, 8], data transformations and filtering [7], as well as for drag & drop gestures in the UI. Using it for other tasks in mashup makers and comparing it to other notations are important research challenges. For example, programming-by-example approaches could aid finding of mashups and mashup elements, which is likely to be a challenge in the future [7].

Regarding UI development, an integration of different design mechanisms, such as using automatic generation to provide a starting point and then being able to modify it using a visual form editor, would combine the strengths of the different approaches. Another important UI improvement is providing a means for advanced information visualization, which could help to leverage the users perceptual system and further increase the usability of mashups.

The overall support for software engineering techniques such as testing and debugging in mashup makers is quite limited. Given the concerns regarding security and correctness of applications developed by non-programmers [19] and the growing popularity of mashups, these shortcomings are dangerous, especially in the context of enterprise mashups. Non-programmer oriented debugging and testing facilities such as debuggers, test frameworks and assertions are researched in the area of end user software engineering (e.g. [21, 12]). Applying those concepts to mashup makers could alleviate the dangers of incorrect and unsafe mashups.

6 Limitations

There are several limitations to the generalizability of this study. Other mashup makers besides those reviewed here exist, and they may contain additional EUD features. We have chosen the reviewed mashup makers in order to get a broad, representative feature set. The mashup makers from major companies are likely to be advanced, as an initial exploration showed, and we made sure to include mashup makers for all three different mashup types (information mashups, process mashups, and web page customization).

Second, there was no standard way in using the mashup makers during the evaluation. A stricter evaluation approach like trying to create the same mashup on all mashup makers was not feasible because of the maturity of the tools and the differences in their functionality and in the kinds of mashups they generate.

Third, the tool review was conducted in July and August 2008. Because the reviewed tools are constantly evolving, the reported feature set here might not reflect their current features.

7 Conclusion

We have presented a review of six mashup makers from an EUD perspective. Six different themes that emerged from our literature review (Levels of Abstraction, Learning Support, Community Support, Searchability, UI Design and Software Engineering Techniques) were used to guide our evaluation. The mashup makers provide many features to support end users, but there is still much room for further improvement. The results can be used to guide both research and tool design. User studies that reveal the difficulties in using mashup makers and comparisons of different notations are likely to be especially fruitful research opportunities.

8 Acknowledgements

This research was supported by the IBM Center of Advanced Studies (CAS) Toronto. We wish to thank Maleh Hernandez, Tricia d'Entremont, Sean Falconer and Gargi Bougie for their helpful editing suggestions.

References

1. Hartmann, B., Doorley, S., Klemmer, S.: Hacking, Mashing, Gluing: A Study of Opportunistic Design and Development. Technical report, Stanford HCI Group (2006)
2. Zang, N., Rosson, M.B., Nasser, V.: Mashups: who? what? why? In: CHI '08: CHI '08 extended abstracts on Human factors in computing systems, New York, NY, USA, ACM (2008) 3171–3176

3. Hartmann, B., Wu, L., Collins, K., Klemmer, S.: Programming by a Sample: Rapidly Prototyping Web Applications with d. mix. In Proceeding of the 20th Symp. on User Interface Software and Technology (UIST07). Newport, RI, USA (2007)
4. Wong, J., Hong, J.: What do we "mashup" when we make mashups? In: WEUSE '08: Proceedings of the 4th international workshop on End-user software engineering, New York, NY, USA, ACM (2008) 35–39
5. Jhingran, A.: Enterprise information mashups: integrating information, simply. In: VLDB '06: Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment (2006) 3–4
6. Wong, J., Hong, J.I.: Making mashups with marmite: towards end-user programming for the web. In: CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM (2007) 1435–1444
7. Tuchinda, R., Szekely, P., Knoblock, C.: Building Mashups by Example. Proceedings of IUI (2008)
8. Fujima, J., Lunzer, A., Hornbaek, K., Tanaka, Y.: Clip, connect, clone: combining application elements to build custom interfaces for information access. In: UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology, New York, NY, USA, ACM (2004) 175–184
9. Bolin, M., Webber, M., Rha, P., Wilson, T., Miller, R.C.: Automation and customization of rendered web pages. In: UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology, New York, NY, USA, ACM (2005) 163–172
10. Nardi, B.A.: A small matter of programming: perspectives on end user computing. MIT Press (1993)
11. Lieberman, H., ed.: Your wish is my command: programming by example. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2001)
12. Burnett, M., Cook, C., Rothermel, G.: End-user software engineering. Commun. ACM **47**(9) (2004) 53–58
13. Myers, B.A., Ko, A.J., Burnett, M.M.: Invited research overview: end-user programming. In: CHI '06: CHI '06 extended abstracts on Human factors in computing systems, New York, NY, USA, ACM (2006) 75–80
14. Wulf, V., Klann, M., Lieberman, H., Paternó, F., eds.: End User Development. Volume 9 of Human-Computer Interaction Series. Springer Netherlands (2006)
15. MacLean, A., Carter, K., Lövstrand, L., Moran, T.: User-tailorable systems: pressing the issues with buttons. Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people (1990) 175–182
16. Ko, A.J., Myers, B.A., Aung, H.H.: Six learning barriers in end-user programming systems. In: VLHCC '04: Proceedings of the 2004 IEEE Symposium on Visual Languages - Human Centric Computing, Washington, DC, USA, IEEE Computer Society (2004) 199–206
17. Ennals, R., Garofalakis, M.: MashMaker: mashups for the masses. Proceedings of the 2007 ACM SIGMOD international conference on Management of data (2007) 1116–1118
18. Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, Y.H., Simmen, D., Singh, A.: Damia: a data mashup fabric for intranet applications. In: VLDB '07: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment (2007) 1370–1373
19. Harrison, W.: From the Editor: The Dangers of End-User Programming. Software, IEEE **21**(4) (2004) 5–7

20. Green, T., Petre, M.: Usability Analysis of Visual Programming Environments: A Cognitive Dimensions Framework. Journal of Visual Languages and Computing **7**(2) (1996) 131–174
21. Ko, A.J., Myers, B.A.: Designing the whyline: a debugging interface for asking questions about program behavior. In: CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems, New York, NY, USA, ACM (2004) 151–158