An Overview of Current Approaches to Mashup Generation

Thomas Fischer and Fedor Bakalov
University of Jena
07743 Jena, Germany
{fischer.thomas|fedor.bakalov}@uni-jena.de

Andreas Nauerz
IBM Research and Development
71032 Böblingen, Germany
andreas.nauerz@de.ibm.com

Abstract: Mashups allow users to bring together data and services from various web applications in order to create a new integrated tool that serves their needs. In the last few years, a variety of mashups frameworks has been proposed that promise to simplify the mashup creation process so that every user is able to create mashups. In this paper, we give an overview about these approaches and identify their limitations. The main insight is that the average user will not possess the necessary skills to create mashups that meet his needs with these tools. We therefore propose that a tool is needed that allows for the automatic ad-hoc generation of mashups.

1 Introduction

A mashup can be defined as a situational Web application that extracts and combines data and functionality from different sources to support user needs and tasks [Mer06]. In the last few years, a number of frameworks have been proposed to simplify the mashup creation process so that users without programming background are able to create mashups. In this paper, we compare these frameworks with respect to the skill requirements of the user. The analysis is based on a representative overview of mashup frameworks that have been proposed by business and research.

The rest of the paper is organized as follows. In Section 2 we describe the categorization of the framework overview and provide the overview itself. In Section 3 we identify limitations of the existing approaches. Section 4 concludes this paper with an overview about the topics that should be addressed by future research on mashup frameworks.

2 Analysis

In this section, we give an overview about different mashup frameworks and their paradigm to mashup development. Figure 1 depicts the mashup framework landscape in relation to the development paradigm as well as the required skills of the user.

Mashup Framework Categorization Model. The **user skills** can be divided into the categories **developer**, **power user**, and **casual user**. A developer should be familiar with programming, web technologies, different APIs as well as the usage of development tools. A power user has no programming skills, by definition of this paper, but does have de-

tailed functional knowledge about a specific tool or set of tools. Casual users only have the skills to use the functionality of a web browser and are able to navigate through the Web. The skill requirements of users depend on the **development approach** that is used to create the mashup. The development approach of mashups can be based on **manual**, **semi-automatic** or **automatic** creation. Manual creation means that the data sources and functionality have to be integrated by programming or scripting. Semi-automatic creation covers spreadsheet based mashup creation, wire-oriented tools and programming by demonstration. Automatic creation means that the mashup is generated without direct user interaction. This means that resources (e.g. knowledge web services) are selected and invoked automatically. In addition, mashup creation can be called automatic and adaptive, if this process generates mashups tailored to the changing user interests, tasks and experience of a user, which means that it is based on a specific user model [BKN07].
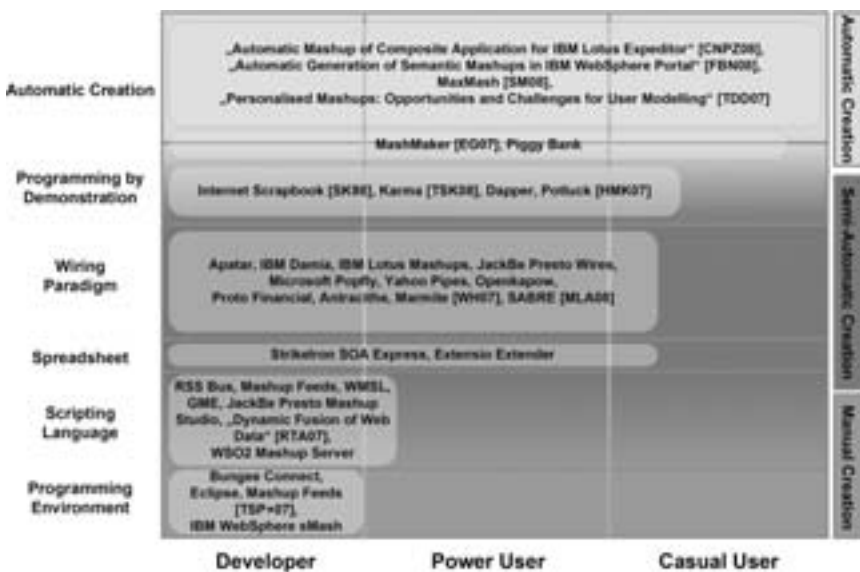

Figure 1: Overview about Mashup Frameworks

Frameworks based on the Programming Paradigm. A number of tools create mashups based on an integrated development environment. *IBM WebSphere sMash*[1] is a development and execution environment for dynamic web applications. It enables easy reuse of web services and enables rapid integration of different web services. *BungeeConnect*[2] is another platform that is offered as an online service enabling users to create comprehensive web applications. Bungee automates the import of publicly available web services as well as traditional databases and data warehouses. This manual development of mashups is supported by a variety of other IDEs and can only be done by experienced developers.

Frameworks based on Scripting Languages. The development of mashups is supported

---

[1] http://www-01.ibm.com/software/webservers/smash
[2] http://www.bungeelabs.com/

by various tools that are based on certain scripting languages such as *Google Mashup Editor*[3] (GME), *Web Mashup Scripting Language* (WMSL) [SHSG07, p.1305], *Dynamic Fusion of Web Data* [RTA07] [TAR07], *WSO2 Mashup Server*[4]. In general, it seems to be too complicated for a non-developer to create such scripts in an appropriate time, because more complex mashups will need a considerable amount of rather complex script code.

Frameworks based on Spreadsheets. Spreadsheet-based tools such as *StrikeIron SOA Express for Excel*[5], *Extensio Excel Extender*[6] focus on the remix of data. Unlike with wire-oriented tools, the data is directly inserted into a spreadsheet. This means that the outputs of a data source are written to cells that have been previously selected by the user. The cell values then serve as inputs of subsequent data source queries. StrikeIron SOA Express and Extensio Excel Extender utilize SOAPful web services to create mashups. In addition, Extensio Excel Extender can provide access to SAP, several databases as well as flat files.

Frameworks based on the Wiring Paradigm. Wire-oriented tools such as *Apatar* [Khi08], *IBM Damia* [SAM+08], *Marmite* [WH07], *SABRE* [MLA08], *JackBe Presto Wires*, *Microsoft Popfly*, *Yahoo Pipes*, *Openkapow*, *Proto Financial*, *Anthracite*, *Lotus Mashups* [7], remix and merge data, functionality, or presentation through a graphical wiring of basic building blocks. This manual connection is sometimes called wiring or piping of different modules, connectors, components or blocks. The available components provide different functionality (e.g., data retrieval, data transformation, data presentation etc.) and have to be connected to achieve the desired coordination of the mashups. The tools often support different data source types such as RESTful and / or SOAPful (e.g. Openkapow, Proto Financial, Anthracite) web services, database, spreadsheets and CSV files.

Frameworks based on Programming by Demonstration. Programming by demonstration enables users to learn a system by the provision of examples. The *Internet Scrapbook* [SK98] system allows users with little programming skills to automate recurring browsing tasks. The user is enabled to indicate the fragments from different web pages that are interesting to him and aggregate them into a personalized mashed-up page. The extraction of the data is based on the HTML structure of the specific web page. *Dapper*[8] is an online service that is able to create an API for any web site. The source web site has to be initially specified, then the user can select graphically from some sample outputs the fields that should be extracted. *Karma* [TSK08] utilizes programming by demonstration to extract lists of data from web pages through simple drag-and-drop of elements of a web page. The system leverages the DOM tree information of the browser and creates a table of the data. The data can be automatically joined with other tables derived from other pages, by a match of attribute name and value pairs. Huynh et al. propose the *Potluck mashup tool* [HMK07]. Potluck takes as input a set of URLs that contain the data that should be

---

[3] http://code.google.com/gme/
[4] http://ws02.org/projects/mashup
[5] www.strikeiron.com/tools/tools_soaexpress.aspx
[6] www.extensio.com
[7] http://www.jackbe.com, http://www.popfly.com/, http://openkapow.com/, http://pipes.yahoo.com/pipes/, http://www.protosw.com/, http://www.metafy.com/products/anthracite, http://www-01.ibm.com/software/lotus/products/mashups/
[8] http://www.dapper.net/

remixed. The datasets from the web pages are shown in a tabular environment. Fields that represent the same semantic attribute can be set as equal through a simple drag and drop of the field names. In this case, the presentation is automatically rearranged. Thus, the mediation between the different semantic heterogeneities is done implicitly by the user, while he is acting on the retrieved data. Furthermore, the system enables the user to clean up and homogenize the data in a faceted browsing environment.

Frameworks based on Automatic Creation of Mashups. The automatic generation of mashup applications has only recently started to gain interest in the research community. Carlson et al. [CNPZ08] propose a mashup framework for automatic composite mashup applications based on Lotus Expeditor. The framework focusses on mashups composed of non-web service components (e.g., widgets). In our work [FBN08], we have proposed an adaptive and automatic approach for mashup creation. In this mashup framework, SOAP-ful or RESTful knowledge web services are composed automatically by semantic web service composition based on the interests, tasks, and experience of a user. The retrieved data is merged and presented as a mashup. *MaxMash* is a framework that is directed towards automatic generation. It promises to compose and select features of networked applications and generates automatically the mashup application [SM08]. Instead of reusing the existing APIs, it relies on the Extensible Messaging and Presence Protocol (XMPP), which is used for instant messaging applications, and uses the underlying network traffic to create the mashup application automatically.

A number of research initiatives are directed towards adaptive mashups. *MashMaker* is a browser plugin that extends the normal process of browsing with mashup functionality. MashMaker does not require the specification of the mashup in advance by the users, as it is required by the workflow like tools [EG07]. The software guesses a mashup that the user would find useful, based on the current browser content. *PiggyBank*[9] is able to process RDF data, which is embedded in web pages. Manual development of web page specific screen scrapers (e.g., for Flickr pages) allows the extraction of non-semantic data and their transformation to RDF. An advantage of PiggyPank is that the user has not to concern about retrieving and remixing of data, which is done automatically through the merging of RDF triples.

3   Limitations

In this section we outline important limitations of the present mashup development approaches. Manual development (programming or scripting) of mashups requires expertise in the fields of web technologies and programming to create complex mashups (see also [TSK08, p.140], [WH07]. The difficulty of programming or scripting (even for experienced developers) is to transfer the problem domain representation into an abstract, detailed, and technical language of the programming [SBD03, p.287] or scripting environment. Therefore, it is difficult for casual users as well as power users to create personalized mashup applications in an appropriate time. However, a relatively small set of developers

---

[9]http://simile.mit.edu/piggy-bank/

is not able to create mashups that serve the changing and very different needs of a huge mass of end-users.

Semi-automatic tools cover the creation of mashups by end-users. However, they still require detailed background knowledge. In all semi-automatic tools (programming by demonstration, wiring paradigm or spreadsheet paradigm) the end-user has to select manually the data sources that should be remixed in the mashup application. Manual data source selection does not only require the location (e.g. URL) of the data sources, but also knowledge about the structure and semantics of the information that could be retrieved, because otherwise the human agent could not evaluate if a specific data source is appropriate for the specific problem. In fact, it is likely that the end-user also does not know many of the available sources and therefore is limited to a set of known ones, which may not satisfy his requirements such as trust, reliability, security etc. in a specific situation, which can influence the decision quality negatively.

In addition, wire-oriented tools specify mashups by a flow of different components that retrieve, transform, aggregate or present data from disparate sources. Users are overstrained to select appropriate building blocks of wire-oriented frameworks, because the end-users have to translate the problem and domain specific, high level representation of a problem into an abstract, detailed, and technical flow of components, blocks etc., which is similarly stated by Tuchinda et al. [TSK08, p.140] and Carlson et al. [CNPZ08]. In open systems like Microsoft Popfly the count of blocks increases steadily, which makes it increasingly difficult for a user to select building blocks from the cloud for complex mashups.

Another important limitation of manual or semi-automatic created mashups is the lack of adaptivity. This means that if the data sources and functionalities of the provider change their structure or behaviour the mashup application has to be reengineered by the end-users. Due to the above investigation, we can state that manual and semi-automatic creation of mashups has some fundamental limitations. This goes in accordance with Wong et al. [WH07], who argue that most tools still require too much background knowledge on web technologies and programming and focus mainly on lightweight user interfaces. Therefore, we propose fully automatic and adaptive generation of mashups that considers the user interests, tasks and experience. In an automatic generation of mashups the data sources (e.g., knowledge web services) are selected automatically and the retrieved data is merged without user interaction. Furthermore, the ad-hoc creation of mashups, which involves an automatic selection of the data source close to usage, avoids time consuming reengineering and automatically considers changed interests, tasks and experiences. Such an automatic generation requires the extensive utilization of machine processable and reasonable semantics, but promises to overcome the general limitations stated above.

4 Conclusion

In this paper, we have outlined the different types of development approaches that are utilized by existing mashup frameworks. Especially semi-automatic tools that often provide a graphical user interface have gained wide popularity in the last few years and promise to allow end-users to create mashups. However, as explained in the previous section, the

existing approaches have several limitations. They lack a mechanism to select data sources (e.g., information web services) efficiently. Furthermore, the composition of the different components has to be done manually, which becomes more and more time consuming, if the count of available components increases and the mashups become more complex. As a consequence, we have started working on a tool for automatic generation of user sensitive mashups within IBM WebSphere Portal [FBN08]. The framework automatically selects and combines information web services to create mashups. In our work, we also have described a user model that stores knowledge about user interests and expertise, which are used by the framework in order to generate mashups tailored to the needs of individual users.

References

[BKN07]   P. Brusilovsky, A. Kobsa, and W. Nejdl, editors. *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*. Springer, 2007.

[CNPZ08]   M. P. Carlson, A. H.H. Ngu, R. Podorozhny, and L. Zeng. Automatic Mash Up of Composite Applications. In *Int. Conf on Service-Oriented Computing (ICSOC-08)*, 2008.

[EG07]   R. Ennals and D. Gay. User-Friendly Functional Programming for Web Mashups. In R. Hinze and N. Ramsey, editors, *ICFP*, pages 223–234. ACM, 2007.

[FBN08]   T. Fischer, F. Bakalov, and A. Nauerz. Towards an Automatic Service Composition for Generation of User-Sensitive Mashups. In *LWA 2008 - Workshop-Woche: Lernen, Wissen & Adaptivität*, Würzburg, Germany, October 2008.

[HMK07]   D. F. Huynh, R. C. Miller, and D. R. Karger. Potluck: Data Mash-Up Tool for Casual Users. In *ISWC/ASWC*, volume 4825 of *Lecture Notes in Computer Science*, pages 239–252. Springer, 2007.

[Khi08]   A. Khizhnyak. Apatar Connector Guide. Website, 2008. Available online at http://www.apatarforge.org/wiki/display/GUI/Apatar+Connector+Guides; visited on August 8th 2008.

[Mer06]   D. Merrill. Mashups: The new breed of Web app. Website, 08 2006. Available online at http://www.ibm.com/developerworks/library/x-mashups.html; visited on March 28th 2008.

[MLA08]   Z. Maraikar, A. Lazovik, and F. Arbab. Building Mashups for The Enterprise with SABRE. In *Int. Conf on Service-Oriented Computing (ICSOC-08)*, 2008.

[RTA07]   E. Rahm, A. Thor, and D. Aumueller. Dynamic Fusion of Web Data. In *XSym*, pages 14–16, 2007.

[SAM+08]   D. E. Simmen, M. Altinel, V. Markl, S.Padmanabhan, and A. Singh. Damia: data mashups for intranet applications. In *SIGMOD Conference*, pages 1171–1182. ACM, 2008.

[SBD03]   E. Schwarzkopf, M. Bauer, and D. Dengler. Towards intuitive interaction for end-user programming. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 287–289, New York, NY, USA, 2003. ACM.

[SHSG07]   M. Sabbouh, J. Higginson, S. Semy, and D. Gagne. Web Mashup Scripting Language. In *WWW '07: Proceedings of the 16th Intl. Conf. on World Wide Web*, pages 1305–1306, New York, NY, USA, 2007. ACM.

[SK98]   A. Sugiura and Y. Koseki. Internet Scrapbook: Automating Web Browsing Tasks by Demonstration. In *ACM Symp. on User Interface Software and Technology*, pages 9–18, 1998.

[SM08]   M. Shevertalov and S. Mancoridis. A Case Study on the Automatic Composition of Network Application Mashups. In *ASE*, pages 359–362. IEEE, 2008.

[TAR07]   A. Thor, D. Aumueller, and E. Rahm. Data Integration Support for Mashups Sixth Int. Workshop on Information Integration on the Web, IIWeb, Vancouver, Canada, 2007.

[TSK08]   R. Tuchinda, P. A. Szekely, and C. A. Knoblock. Building Mashups by example. In J. M. Bradshaw, H. Lieberman, and S. Staab, editors, *Intelligent User Interfaces*, pages 139–148. ACM, 2008.

[WH07]   J. Wong and J. I. Hong. Making Mashups with Marmite: Towards End-User Programming for the Web. In Mary Beth Rosson and D. J. G., editors, *CHI*, pages 1435–1444. ACM, 2007.