Mashups as an Architecture for Knowledge Management Systems

Stefan Bitzer
University of Göttingen
Institute of Information Systems
sbitzer@uni-goettingen.de

Stefan Ramroth
University of Göttingen
Institute of Information Systems
mail@stefan-ramroth.de

Matthias Schumann
University of Göttingen
Institute of Information Systems
mschumal@uni-goettingen.de

Abstract
The potential and significance of mashups for future corporate applications are currently discussed intensively. So far, first approaches for mashups in some areas of knowledge management have been developed. The potential of mashups as an architecture for knowledge management systems has not been analyzed at all. In order to provide a structured view, we firstly systemize mashups and derive a reference architecture for mashups. In a second step we carry out an analysis of mashups as an architecture for KMS. Subsequently, we examine integration potentials of mashups in a standardised architecture of KMS. We show that the architectures of mashups and KMS share a lot of similarities. Furthermore, we demonstrate that mashups and standardized KMS can complement one another in an integrated architecture and we present open research questions in this area.

1. Introduction

The potential and significance of social software for corporate applications has received a lot of attention recently. Especially mashups as a tool of social software are seen as highly relevant for the development of future enterprise applications [1]. Whereas the significance of mashups for the corporate sector is still debated, there is a consensus regarding the importance of knowledge management for enterprises [2]. For the support of knowledge intensive processes flexibility is a critical factor [3]. Mashups are known to be dynamic, adaptive and user centred [4]. Against this background, an analysis of mashups as an architecture for knowledge management systems (KMS) makes sense. In this regard it should be noted that mashups are a relatively new phenomenon, and especially the architectural models are still in an early stage of development.

2. Mashups

2.1. The term mashup

The term mashup originates from the music business and was adopted for internet technologies in the context of Web 2.0 [6]. Whereas in the music business the term mashup denotes mixing different music tracks, in connection with Web 2.0 the term implies mixing several Web services, data and further content. These components are provided from different producers mostly free of charge over the internet. A simple example for a mashup is iGoogle (http://www.google.com/ig) shown in figure 1, where components from different sources can be aggregated into a personally customized Web site [7].
By using mashups, Web services and data can be integrated into own applications. Besides the integration, mashups offer further potentials. Through bonding of several Web services, previously separate applications and contents can be combined and lead to a new benefit for the user. The integration of Web services is done through standardised Application Programming Interfaces (API) [8].

As already mentioned in the introduction, mashups are considered as Web 2.0 applications and hence fulfil some principles of this concept [9]. For example, mashups use the “Web as a platform”. This means that mashups are Web applications and access Web services by using the internet as their network. Mashups are also typical examples of the Lightweight Programming Model (LPM). By using simple standardised APIs and intuitive programming languages, the management and composition of mashups is kept extremely simple, making the technology accessible for every user. According to the definition given at the beginning of this chapter, mashups integrate and combine Web services, data and contents. This description of the function of mashups is very general and could just as well be achieved by other technologies. Therefore it is suggested to extend the proposed definition. As typical for Web 2.0 applications, users can create mashups themselves. This kind of user participation is called user-driven or user-centred micro-orchestration [10, 11]. In this process, the user is able to obtain the necessary data and services from various sources and combine them according to his ideas. Thus the user can access the requested information ad-hoc and aggregate them. The only requirement is that the data is available in the required granularity [11]. Up to now, these possibilities were hard to realize since existing applications had to be implemented and administrated by IT-professionals. IT-professionals were also needed for each customisation on the basis of the user’s requirements.

Now mashups offer a user-focused view which allows the creation of applications and information by the user and for the user. In most cases he knows better than the IT-Developer, which services he needs and in which configuration. Thus the application created shows the knowledge behind the application. This knowledge can be opened to others, so that they can adopt and potentially extend it. This procedure starts a collective learning process and amplifies the collective intelligence of an organisation [12].

Through this extended definition, mashups fulfil practically all the Web 2.0 principles formulated by Tim O’Reilly [6]. Summarizing this chapter, we define mashups as a Web application, in which the user can autonomously combine Web services, data and further content.

2.2. Classification of mashups

Despite the fact that mashup technologies are still very new, there are already a lot of different kinds of mashups which possess different functionalities and address a distinct target group. For a comparison of mashups and KMS, we first need an overview of the different kinds of mashups, to identify the type of mashup that fits in its main features a KMS. In the following, we therefore develop a classification. Regardless of the differing opinions in the literature on the specifications of mashups [5, 7, 8, 13, 14, and 15], it is possible to deduce a classification. In this paper we distinguish mashups on the basis of the functional range, the target group and the location of the technical implementation (see Figure 2).

2.2.1. Classification on the basis of the functional range. The functions of mashups can be subdivided into three categories: presentation level mashups, data level mashups and logic based mashups [13, 14, and 15].

Presentation level mashups concentrate on the presentation of content. Layout and information are provided in miscellaneous ways as Web services. The implementation is modular, so that the Web services are available as separate components making it possible to arrange them side by side. A simple example is Amazon Light, which aggregates the back-end data of Amazon to offer a customized search (http://www.kokogiak.com/amazon/). In enterprises, presentation level mashup resemble customized portals. The user can decide which feature he wants displayed on his welcome page; comparable to iGoogle.
Figure 2. Classification of mashups in form of a morphologic cuboid

Data level mashups focus on the extraction and combination of data from different sources. They gain information from internal and external resources and integrate them into the user's own internet site. Famous examples for data level mashups are websites combining online maps with further data, e.g., HEALTHmap (healthmap.org), which is a global disease alert map that shows where diseases have been reported [16].

Logic based mashups constitute the most complex type. They can contain presentation level elements as well as data level mashups. In this context the services are combined with the corresponding application logic. They are currently used in price comparison websites like “kayak.com” (www.kayak.com), which use Web services to send inquiries to online travel agencies and book flights afterwards.

2.2.2. Classification on the basis of the target group. There are two kinds of user groups for mashups. On one hand mashups are used for private concerns, so-called consumer mashups, on the other hand they are used in organisations, so called enterprise mashups.

So far mashups are mainly used on the private level. Most of the participants use already established mashups. Only a small group of technology-affine users develop mashups of their own. With new graphical user interfaces, like Yahoo Pipes, the creation of consumer mashups opens up more and more for ordinary users [17]. Besides consumers, an increasing number of organisations use mashups. Enterprise or sometimes also called business mashups [18] concentrate on the individualisation of software by letting the users, respectively the employees, create the software themselves [13, 19]. While many IT departments demand a central IT management and oppose discrete and independent software solutions, others claim the importance of self-made and customized solutions [20]. Through the involvement of the users, mashups can help to harmonise the cooperation between the IT department and the users. Enterprise mashups constitute a technology that enables the user to combine applications and data in the form of Web services to their own requirements. Since not only internal but also external services are used, security aspects play a substantial role for the employment of the different services [21]. Consequently, enterprise mashups are characterised by a secure data handling and a user-friendly as well as controlled application. In October 2007, IBM released the IBM Mashup Center. IBM claims that this software meets the requirements of enterprise mashups described above. It consists of two technologies: IBM Mashup Hub and QEDWiki. The IBM Mashup Hub is a mashup server that stores information feeds. The QEDWiki functions as the user interface and enables non-IT users to "mash" information from any data source [22].
2.2.3. Classification on the basis of the location of the technical implementation. While the different user groups have different requirements for mashups, the location of the technical implementation leads to different techniques for the creation of mashups and to particular characteristics.

Client-side mashups integrate and combine the services or contents on the client, usually in a Web browser. Internal as well as external sources are embedded directly into the client. [23].

Server-side mashups are the pendant counter piece to client-side mashups and integrate content and services on the server. The internal server functions as a proxy between the client and respective providers, so that the work is relocated form the Web application client to the server [24]. In this way, the server functions as a buffer and data cache, allowing among other things interception and repair of incorrect data. More importantly, the server-side solution enhances the handling of security requirements. The entire communication can be encrypted via various security measures. Due to these advantages, server-side mashups are highly relevant for the use in enterprises [14, 21, and 24].

In this paper we analyse mashups as an architecture for knowledge management systems, i.e. mashups are examined for a professional use. This consequently restricts the target group to organisations and to the focus on enterprise mashups. The professional environment for the mashups also influences the location of implementation. On the one hand, security and rights management aspects are essential and, as already mentioned above, can be better managed with server-side mashups. On the other hand, server-side mashups reduce the traffic within the network. Because the integration and combination of the services and data occurs on the server, only the result of this combination is transferred over the network to the user [24]. With client-side mashups, every user needs the complete data capacity, since the integration takes place on the client. Both described aspects strengthen the case for server-side mashups. So far, in enterprises mashups are often used for the aggregation of information or data [19, 25], thus correspond to data level mashups. Nevertheless, especially with regard to the analysis of mashups as architecture for KMS, logic-based mashups have also been considered. Summarizing the statements of this chapter, it can be said that for a consequential comparison we have to select server-side, data level or logic based, enterprise mashups.

Enterprise mashups have to be easily applicable by the user. Mashup-building platforms offer a way of achieving this goal. The following chapter describes a mashup stack to be used as reference architecture for such platforms.

2.3. Mashup stack

Even though mashups are not much examined in the literature, approaches for mashup stacks have already been developed [19, 26]. Despite different names and designs, the mashup stacks possess the same core characteristics. On the basis of these stacks, the framework of mashup-building platforms can be shown. Figure 3 depicts the fundamental structure of a mashup stack. It consists of five layers, which build on each other and show at the same time the procedure for the development of a mashup.

The lowest layer “resources” represents the data and the Web services. They are the basis of a mashup and can be obtained from internal or external sources. Because the content is available in different shapes and formats, it has to be prepared for the integration into a mashup. The same applies for legacy applications. Due to the loose coupling in a mashup, legacy applications have to be provided as loose components, e.g. by using wrappers or screen scrapers [27].

The second level “API” constitutes the access layer and provides standardised and suitable application programming interfaces (API). Besides desktop APIs (e.g. Microsoft Office rich APIs) and intranet APIs (like DCOM), APIs for internal and external Web services are important. Two types of APIs for Web services have to be considered both based on the Hyper Text Transfer Protocol (HTTP): resource and activity-oriented architecture [28, 29]. In both architectures HTTP can be regarded as an envelope that packs and sends the information. The Representational State Transfer or, in short, REST architecture as a resource-oriented form uses simple http methods like GET, PUT, DELETE or POST to address Web services as resources.

The activity-oriented architecture uses remote procedure calls (RPC) to address objects in a distributed network. The essential difference to the RESTful architecture is that HTTP is only used for the transmission of the xml-based RPC. For this, SOAP has been established as the associated message structure. Web services using the activity-oriented architecture are called WS-* Web service, resource-oriented are called RESTful Web services [28, 29]. In summary, the second layer describes the functional and non functional characteristics of the resources and enables the loose coupling.

The two described layers are the basis for the creation of mashups. To open mashups to non-technically experienced users, so called widgets are created in the third layer.
to enhance mashups with program logic, similar to traditional software development tools. In this most complex type of mashup, for example iteration, looping or complex data types are used. This way of creating mashups is almost similar to conventional software development, so that mashup-building platforms have to be prepared for this kind of requirements.

The output layer determines all aspects of the publication of the developed applications. For one thing, security policies have to be managed, e.g. who is allowed to access which mashup. Mashups with noncritical contents and results can be provided as a separate Web service. Secondly, the output layer deals with the location where the mashup and the mashup building platform are hosted [19, 26].

3. Mashups as an architecture for knowledge management systems

The following chapter analyses to what extent mashups can be used as architecture for knowledge management systems. The previously presented mashup stack is employed as a basis for the comparison between the architecture of mashups and knowledge management systems (KMS).

3.1. KMS architecture

A KMS is defined as an information and communication technology system, developed to support and enhance explicit and tacit organizational knowledge. A KMS assists users in all tasks of knowledge management, e.g. knowledge creation, storage/retrieval, transfer and application [31]. In order to classify KMS, several architecture models with differences in focus and origin have been proposed. Especially regarding the level of abstraction there are significant differences. A lot of architectures show the functions of KMS on a very abstract level. One example is the generic KMS architecture of Frank [32]. Based on the level of abstraction of knowledge, three consecutive layers are proposed. The examination is largely detached from concrete implementations and functions of the system. The same applies to the three-tiered architecture of Chua [33]. The abstraction level is high and the architecture was developed for consultants as well as technologists. A more specific classification and an example for a comprehensive layer architecture has been done by Lindvall et al. [34]. They classify the different KM tools into seven layers. A number of potential functions are discussed, however a structured design for a KMS is not provided. Moreover, the literature provides further, similar architecture approaches, e.g. Ovum KM tools.

Figure 3. Mashup stack

Depending on the respective context, widgets are defined in different ways in the literature. With respect to mashups, widgets are micro applications for a small individual task in the form of components. For the mashup stack this means that the functions and data are provided as modules [19, 26]. Accordingly, widgets are reusable mashup fragments [30] which can be arranged and composed via drag-and-drop in a graphical development kit [19]. In this way, they convert the interfaces of the second layer into a graphical depiction. For the creation of mashups, widgets are generally optional, since programming can be done manually. Nevertheless, widgets constitute an intermediate layer, which is important for the user-driven micro orchestration.

The fourth layer contains the mashup logic. The separate components (in the form of widgets) are combined into a mashup. In its simplest form, a mashup just consists of a number of widgets arranged on the same screen. Thus the user can create personalised portals that integrate the chosen components. By connecting widgets to each other, more complex mashups can be created. The Web services embedded in the widgets communicate with each other and exchange information. It is also possible
architectural model [33, 35]. It is a comprehensive approach but actual deployment examples are missing. In this paper we resort to the generic KMS architecture developed by Maier. The architecture incorporates different approaches from research and practice [36]. In contrast to other approaches which are oversimplified for our analysis, this architecture is relatively complex. The lower layers are in comparison to three-tiered architectures specified in more detail and the described functions of Lindvall et al. [34] are included as well. In addition to the reasons above, the technological architecture of Maier is advantageous because the layers build upon each other technology [36], similar to the mashup-reference-architecture in chapter 2.3. The architecture consists of six layers (see figure 4): The lowest layer includes internal as well as external data and knowledge sources. Through infrastructure and integration services these resources are transferred into the system. The fourth layer provides knowledge services, e. g. for knowledge search, publication, collaboration or learning. Via the layer personalisation services these knowledge services can be customised individually. The access services in the sixth layer control security and output aspects [36].

3.2. Mashups as architecture for KMS

As already described in chapter 3.1, a KMS is an IT system to develop and support organizational knowledge. Since every person processes knowledge in a different way [37], it is suggestive to adapt a KMS to the requirements of the user as far as possible. The definition as well as the classification of mashups in chapter 2 have shown that mashups are no specific software for a defined problem, but miscellaneously deployable [12]. Since in the best case the user creates the mashup himself, it should fit his requirements. Especially in the knowledge development and linking, mashups are already used in organisations [17, 19]. The next step is consequently to examine mashups as a KMS.

Figure 4 shows a comparison of the layer architectures of knowledge management systems and mashups. The layers are compared and analysed in the following.

**Data and information layer vs. mashup resources.** In both architectures, the required resources are accessed in the lowest layers. In this process, both not only require data and information, but also services and legacy applications [27, 36].

**Infrastructure and integration services vs. API-layer.** The infrastructure services in KMS and the APIs in the mashup architecture share similarities. Both are required for the upper layers to access the resources. In KMS, different services of the internet form the elementary infrastructure. The same applies to the API-Layer in mashups. On the basis of Web-based structures, like the intranet, services and data are integrated via interfaces. However, the difference in this layer between KMS and Mashups is the handling of the integration of legacy systems. While KMS solve the interface compatibility amongst other by the
implementation of middleware [36], in mashups the interfaces have to be adjusted to standardized APIs [38]. Furthermore, the integration services of KMS offer functions that are not provided by the API-layer in the mashup architecture. For one thing, these services administrate, structure and organise knowledge elements. For this purpose metadata systematics, e.g. in the form of ontologies or controlled vocabularies, are used. Furthermore, the integration services contain an overarching user rights management, which includes directories with competences, tasks, organisational rules and authorisations [36]. Both described functions of the integration services are important for a responsible and sustainable dealing with the resource knowledge in enterprises [39].

**Knowledge services vs. widgets.** In this layer, data and services allocated via interfaces are provided in modules. In KMS, these modules are special tools for knowledge management; e.g. tools for knowledge search, knowledge presentation or collaboration. In this regard, the respective mashup layer has a far more general purpose. Besides the aforementioned functions, services and data that are not directly related to knowledge management can be combined and unified. Furthermore, the flexible character of mashups allows for a better customisation of the designated knowledge services to the concrete requirements of the users. Knowledge intensive and individual activities are only insufficiently supported through standard software. As the users can assimilate and adapt the services themselves, mashups are much more flexible than KMS. A simple example is a customized meta-search in mashups. The different available search services can be selected, arranged and customized to the users’ requests [17].

**Personalisation services vs. mashup logic.** In KMS, individual profiles that fit the requirements of the users can be created via personalisation services. The mashup logic layer resembles the personalisation services. Different modules, potentially in the form of widgets, are combined based on individual users’ requests. As a result, individual mashup portals evolve, just like in KMS. In addition, the mashup logic allows the creation of more complex connections between the services. For instance by connecting the Web services, the to do list could be displayed as a mind map. In this way new applications are developed, which can not be implemented that easy in KMS.

**Access services vs. output.** Both layers provide largely the same functions. They both enable the access and output of the used services, portals or new applications. In mashups, the output layer also regulates the location of the technical implementation, i.e. client or server-side mashups. Due to the mentioned reasons in chapter 2.3, for this comparison mashups are restricted to server-side. In KMS this question does not arise, since KMS are mostly realised in form of client/server architectures [40].

The comparison of the architectures of KMS and mashups shows several similarities. The architectures consist of five respectively six layers that are similar in their respective functions. Furthermore, each layer is the basis for the layers above. However, there are some differences between the architectures. KMS are specifically designed for the support of knowledge management. The knowledge services as the core elements of KMS are existent from the beginning, whereas in mashups these services have to be created as widgets first, possibly by the user himself. More important is the missing of an overarching user rights management in mashups in contrast to KMS. In return, mashups allow the integration of every type of data as well as services and to combine them in higher layers. A maximised customisation to the requirements of the user is enabled through user participation in this process.

### 3.3. Integration of mashups in architectures of KMS

The comparison in chapter 3.2 showed that the architectures of KMS and mashups share many similarities, and that both have their respective advantages. Consequently it is reasonable to combine both architectures and try to integrate mashups as components of KMS. Through this approach, KMS can use mashup technologies to improve the support of individual and intensive knowledge activities. In return mashups are integrated into the infrastructure of KMS and by that into a structured rights system.

Figure 5 shows a potential integration of mashups into the architecture of KMS. The lowest layer “Services, Data and Knowledge Sources” in the combined architecture can be used by both mashups and the original KMS. For mashups this layer represents the core mashup resources. In this process it is important that the services for the mashups persist as independent services [11], so they can be combined autonomously in widgets. This applies for the layer “infrastructure services” as well. The infrastructure layer has to support explicitly the APIs for WS- and Restful Web services. The most important improvement for mashups compared to the former mashup stack, is the addition of integration services. The integration services include a rights management system for all data and services from lower layers [36]. For this reason, the rights management system can be used for mashups and their widgets as well. Furthermore, the services and data can be enhanced by useful metadata in the integration layer. This metadata
can improve and simplify the creation as well as the use of widgets for the user, which is important concerning the user participation within the mashup concept.

The widgets as additional components exist in parallel to the knowledge services of the KMS. Thus, the user can use the specific knowledge services and create own mashup applications with Web services and data. The knowledge services themselves could be provided as widgets. Through this process, every tool can be used flexibly. The combination of knowledge services and the created widgets is done in the personalisation services layer respectively the mashup logic. In case all knowledge services are provided as widgets, this layer only consists of the mashup logic. Otherwise the layer has to be extended with personalisation services of the KMS. In the end, the access and output layer provides user profiles which regulate access to the actual KMS and the created mashups. The shared knowledge platform helps to diffuse the created mashups. Moreover, the user has the possibility to modify mashups and widgets, so that in the course of time an evolution of the initial mashups occurs.

4. Conclusion

The use of Web 2.0 technologies for corporate knowledge management is currently discussed intensely. In this context wikis, blogs and social bookmarking are often mentioned [41]. However, we have shown that mashups also offer interesting possibilities for knowledge management systems.

We first analysed mashups as an architecture for KMS. Because of the found similarities and differences, an integration of mashups into a classical KMS architecture was examined afterwards. Through this integrated solution, one can benefit from the special features of mashups as well as of KMS.

Summarising this paper, it can be said that mashups can be applied as an architecture for KMS. Especially the flexible bonding of new data and services by the user himself increases the usability of such systems and supports knowledge intensive processes. The task for future research will be a closer investigation of mashups in concrete knowledge management scenarios. Two general scenarios are possible. In the first scenario, mashups can be used as a KMS in security uncritical scenarios. In the second scenario for a secure handling of data, an integration of mashup functionalities into a KMS has to be examined. Here, especially the interaction between the integration services, the knowledge services and the widgets has to be investigated. For the future, the increasing implementation of mashups in corporate applications can be expected, so that the importance of mashups for KMS will be further rising.
5. References


