# Enterprise Mashups: Design Principles towards the Long Tail of User Needs

Volker Hoyer[1 2], Katarina Stanoesvka-Slabeva[2], Till Janner[1 2], and Christoph Schroth[1 2]
[1] *SAP Research CEC St. Gallen, Blumenbergplatz 9, 9000 St. Gallen, Switzerland,*
[2] *University of St .Gallen, Institute for Media and Communications Management (MCM), Blumenbergplatz 9, 9000 St. Gallen, Switzerland*
*{volker.hoyer, katarina.stanoevska, till.janner, christoph.schroth}@unisg.ch*

## Abstract

*A new type of Web-based applications, known as Enterprise Mashups, has been gaining momentum in the last years. Novel design principles are currently about to emerge allowing to cover the long tail of user needs and to provide individual and heterogeneous enterprise applications in a shorter time. In this paper, we introduce the main components of this new paradigm, and discuss the design principles of the architecture (Enterprise Mashup Stack), upcoming intermediaries and mass collaboration, lightweight composition as well as perpetual beta development model.*

## 1. Motivation

At present companies are operating in a dynamic environment and are frequently restructuring internal processes and organizational structures to meet external challenges and opportunities. The needs of employees for information technology support are changing frequently as well. As a result, a phenomenon called the long tail can be observed with respect to the requirements of employees for services and functionalities. End-users are increasingly requiring individualized applications that exactly meet their daily needs and which can easily and quickly be adjusted to changing needs. Established implementations of Service Oriented Architectures, i.e. the widespread Web Services Stack, have several shortcomings to cover this long tail of user needs due to the high technical complexity of the relevant standards (SOAP, WSDL, UDDI, BPEL, etc.), their inflexibility to react on changing requirements quickly, and the missing involvement of the actual end-users.

## 2. Enterprise Mashup Stack

The significant components of Enterprise Mashups are classified and structured in a Mashup stack:

**Resources.** The lowest layer contains the actual Web resources, be it content, data or application functionality. They represent the core building blocks of Enterprise Mashups and are the differentiator of the resource-centric paradigm. According to the lightweight Representational State Transfer (REST) architecture style, each Web-based resource can be addressed by a Universal Resource Identifier (URI).

**Application Programming Interfaces (APIs).** The resources itself are sourced via a well-defined public interface, the so called APIs. The APIs encapsulate the actual implementation from the specification and allow the loosely coupling of existing Web-based resources.

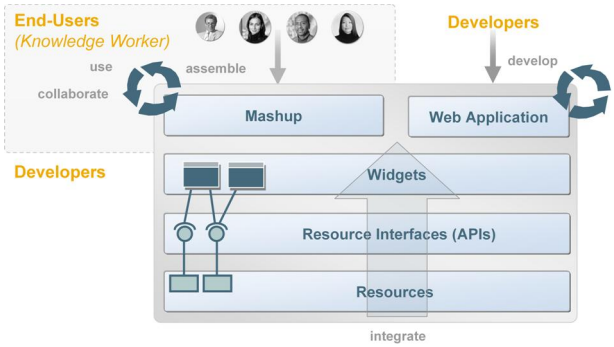


**Figure 1: Enterprise Mashup Stack [2]**

**Widgets.** Widgets represent application domain functions or information specific functions. They are based on resources, are sourced via public APIs, and are responsible for providing graphical, simple and efficient user interaction mechanisms that abstract from the technical description (functional and non-functional) of the Web-based resources.

**Mashup.** By assembling and composing a collection of widgets stored in a catalogue or repository, end-users are able to define the behavior of the actual application according to their individual needs. By aggregation and linking content of different resources in a visual and intuitive way, the end-users are empowered to create their own workspace which fits best to solve their heterogeneous business problems. No skills in programming concepts are required.

## 3. Emerging Intermediaries

Novel forms of intermediaries are currently about to emerge which offer resource registry functionality and extend the role of the traditional UDDI-based implementations. After publishing resources by the

provider or the actual resource owner, advanced intermediaries monitor continuously the parameters (such as the resource availability and response latency) and provide performance metrics and other evaluation results which may be used by potential customers to select a resource [3]. The growing number of available resources requires an adequate description of resources for retrieval. According to the Web 2.0 philosophy, the actual end-user takes over this description process by tagging collaboratively resources. The user creates a folksonomy, essential a bottom-up, organic taxonomy that organizes the available resources. Further user-rating functionalities based on popularity and relevance are used to collect, distribute and aggregate feedback about the resource's behavior by an integrated reputation system.

## 4. Mass Collaboration

Mass Collaboration is the third major design principle. End-users participate in the creation of Web application in an active and ongoing way. Through co-innovation and co-production, the end-users do more than customization and personalization; they self-organize their individual workspace and can submit new ideas. Without any hierarchy limitations (i.e., only the IT department can create an enterprise application), an end-user creates *"quick and dirty"* an Enterprise Mashup, shares it and the experiences with the community and consumes it immediately. The willingness of users to offer feedback to the application creator, who may be unaware of problems or alternative uses, directly contributes to the adoption of the application and can fosters its improvement in a peer production. The described democratized resource development and consumption process implies an open culture. Not only that can people share their remixes with their best friends, but they are able to share them with thousands or even millions on the Web.

## 5. Lightweight Resource Composition

The central driver of Enterprise Mashups to address the long tail of user needs is the lightweight resource composition style by reusing building blocks in different contexts. As depicted in figure 2, the composition takes place both on the resource layer (piping) and on the widget (wiring) layer according to the Enterprise Mashup Stack. In reference to the UNIX shell pipeline concept, the *piping composition* integrates a number of heterogeneous Web-based resources defining composed processing data chains/graphs concatenating successive resources. The output of each process is direct input to the next one. Aggregation, transformation, filter and sort functions adapt, mix and manipulate the content, data and application functionality of the Web-based resources.

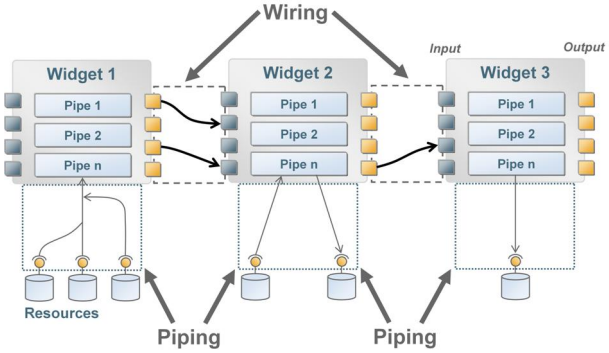


**Figure 2: Piping versus Wiring**

## 6. Perpetual beta

The development of Enterprise Mashups can not be supported by the classical software construction paradigm (requirements, design, implementations and test). To successfully develop Enterprise Mashups in short periods of time, it is in fact necessary to support a perpetual beta development cycle. The users build within hours or even minutes *"quick and dirty"* their individual enterprise applications, often with components that are not under their control. But even if users who are accustomed to the social aspects of Web 2.0 are more tolerant of bugs and poor performance, Enterprise Mashups require a stable mashup environment. Otherwise, this might continue to be a barrier to the early adoption of Enterprise Mashups in business contexts. The development process itself is characterized by an agile development model focusing on the actual working software and not on a comprehensive specification or documentation. In addition, the operative phase is an explicit part of the development model. A disciplined build and development process of new Web-based resources is necessary to allow the continuous improvement and adaptation of Enterprise Mashups.

## 7. References

[1] C. Anderson, *The Long Tail: Why the future of business is selling less of more*, Hyperion Books, 2006.

[2] S. Watt, "Mashups – The evolution of the SOA, Part 2: Situational applications and the Mashup Ecosystem", Online available at http://www.ibm.com/developerworks/webservices/library/ws-soa-mashups2/, 2007.

[3] C. Schroth, O. Christ, "Brave New Web: Emerging Design Principles and Technologies as Enablers of a Global SOA", *Proceedings of the IEEE International Conference on Service Computing (SCC 2007)*, 2007.