Design Principles of Enterprise Mashups

Volker Hoyer[1,2] and Katarina Stanoevska-Slabeva[2]
[1] SAP Research CEC St. Gallen and
[2] University of St. Gallen, mcm=institute, Switzerland
volker.hoyer@sap.com, katarina.staneovska@unisg.ch

Abstract: A new kind of Web-based application, known as Enterprise Mashups, has been gaining momentum in the last years. Enterprise Mashups implicate a shift concerning a collaborative software development and consumption process. End users combine and reuse existing Web-based resources within minutes to new value added applications in order to solve an individual and ad-hoc business problem. Novel design principles are currently about to emerge allowing to cover the long tail of user needs. In this paper, we introduce the terminology used in context of this new paradigm and present an Enterprise Mashup Stack which consists of three layers: Web-based resources, widgets, and Mashups. Based on this model, we elaborate on the design principles of upcoming intermediaries and the mass collaboration form, the lightweight composition style as well as the perpetual beta development model. By means of the EU funded research project FAST, we show their real-world applicability.

1 Introduction and Motivation

The process of development of Web-based business applications in companies follows usually the typical process of software development involving first assessment of user requirements followed by a long process of development and testing. The functionality of the resulting application is actually a compromise of user requirements, as not all user requirements, can be considered. As a result there is usually a long tail - a term first coined and popularized by Chris Anderson [And06] - of many specific user requirements or dynamically changing user requirements that are not met. The business users require Web-based applications that exactly meet their daily needs and which can be developed and adjusted within days and not within months or even years [CBG+07].

At a first glance, innovative approaches of software development, re-use and deployment as for example services-oriented software development provide new opportunities to organize software in a new way. However, as argued in [SC07], established implementations of Service Oriented Architectures (SOAs), i.e. the widespread Web Services Stack [ACKM04], also have shortcomings to cover the dynamic 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 business requirements within days, their limitation on focusing on internal enterprise activities as well as the missing integration of the actual end-user. To meet the organizational flexibility and people needs, a new technical and development paradigm is required which integrates the knowledge workers in the software consumption and development process. This paper presents a concept how Mashups can be used by enterprises to implement a commons-based peer production business application development.

In general, the commons-based peer production represents a new model of economic production. According to Yochai Benkler, who coined the term, „... it refers to production systems that depend on individual action that is self-selected and decentralized, rather than hierarchically assigned" [Ben06]. Thereby, the creative energy and knowledge of large number of people („Wisdom of Crowds") is used to react flexible on continuous dynamic changes of the business environment. In contrast to hierarchical organizations (where centralized decision process are used to decide what has to be done and by whom) or market mechanisms (when tagging different prices to different jobs serves as an attractor to anyone interested in doing the job), people in the peer production have the freedom to work collaborative in decentralized communities on business problems. In this peer production, the users are characterized as knowledge workers who work primarily with information or develop and use knowledge in the democratized workspace [Dav05].

Enterprise Mashups built on Service-Oriented Architectures have the potential to extend the reach of SOA by putting a face on services [JCH$^+$07]. Driven by the consumer market, upcoming tools and forecasts of market research institutes like Gartner, Forrester, or McKinsey show the practical relevance of the paradigm [HF08]. However, an analysis of the underlying principles of Enterprise Mashups is missing so far. In particular, there are no concepts how available Mashups can be stored and managed, how they can be made searchable and which processes are necessary to enable users to flexibly re-use them for their specific needs. This position paper is devoted to fill this gap by discussing the major design principles in context of the Enterprise Mashups paradigm.

The reminder of this paper is structured as follows: Chapter two deals with a clear definition of Enterprise Mashups. In chapter three, we introduce a designed Enterprise Mashup Stack with the relevant vocabulary used in context of Enterprise Mashups and elaborates on the design principles enabling to address the long tail of user needs. By means of the research project FAST, we show their real-world applicability in the course of chapter four. Finally, chapter five closes this article with a brief summary and an outlook on future work.

2 Enterprise Mashups

In literature, the exact definition of Enterprise Mashups is open to debate [Jhi06] [DMY$^+$07] [MRG08]. In this work, we refer to a definition based on an analysis from different perspective (technical, business, application, consulting, software vendor, and community) combining the academic as well as the industrial world [HF08]: „An Enterprise Mashups is a Web-based resource that combines existing resources, be it content, data or application functionality, from more than one resource in enterprise environments by empowering the actual end-users to create and adapt individual information centric and situational applications" [HF08]. According to this definition, Enterprise Mashups focus on the UI integration [DMY$^+$07] by extending concepts of Service-Oriented Architectures (SOA)

[ACKM04] with the Web 2.0 philosophy [JCH$^+$07]. In contrast to SOA, Enterprise Mashups integrate the business users (i.e., sales manager) in the software development and consumption process implicating a shift concerning the responsibilities of the IT department. Users from the business units characterized by no sophistical IT skills take over more and more capabilities to create their individual operational environment. The role of the IT department is changing toward a service intermediary [HSS08] [CBG$^+$07].

Criteria	**Service-Oriented Architecture**	**Enterprise Mashups**
Time-to-value	Many weeks, months, or even years	Minutes, hours or days
Developer Profile	IT department	Business units (limited programming skills); small teams or individuals
Integration Layer	Application Integration	Focus on UI Integration
Development Phases	Well defined, following agreed-to schedule (although with frequent schedule overruns)	No defined phases or schedules; focus on a good-enough solution to address an immediate need
Functional Requirements	Defined by limited number of users, IT needs to freeze requirements to move to development, requirement creep often caused by changing business needs	As requirements change, Enterprise Mashups usually changes to accommodate business changes; Enterprise Mashups encourages unintended uses
Nonfunctional Requirements	Resources allocated to address concerns for performance, availability, and security; robust solutions	Little or no focus on scalability, maintainability, availability, etc.
Testing	By IT with some user involvement	By users through actual uses

Tabelle 1: Service-Oriented Architecture versus Enterprise Mashups

To understand the changing development environment, table 1 summarizes the findings of a desk research [HSS08] and experiences taken from first implementations of domain specific Mashups with the SAP Research Rooftop Mashup prototype. The comparison of the traditional development approach and the Enterprise Mashups paradigm indicates the changing environment and need for new design principles.

3 Design Principles

In context of the Enterprise Mashup paradigm, novel design principles are currently about to emerge. This section is devoted to present these design principles. They are analyzed from four perspectives: architecture (Enterprise Mashup Stack), community, programming style, and development process [HSSJS08]. Besides a literature review, we integrate our experiences taken during the development activities of the Mashup prototype SAP Research Rooftop [Hoy08].

3.1 Enterprise Mashup Stack

From an architectural perspective, we can describe the relevant components and terms in context of Enterprise Mashups with the assistance of a layer concept. Figure 1 depicts the resulting Enterprise Mashup Stack which consists of the layer resource, widget and Mashup

Resources. The lowest layer contains the actual Web-based 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) allowing equal accessibility to those resources from browsers, mobile devices, or server applications. The resources itself are sourced via a well-defined public interface, the so called Application Programming Interface (API). The APIs encapsulate the actual implementation from the specification and allow the loosely coupling of Web-based resources - a major quality of SOA. According to the REST architecture style, the four CRUD-Operations (Create, Read, Update and Delete) are represented by the HTTP verbs Put, Get, Post and Delete. The Atom Publishing Protocol (APP)[1] represents a first application-level protocol for publishing and editing Web resources by following the REST architecture style. The protocol is based on HTTP transfer of Atom-formatted representations which is documented in the Atom Syndication XML Format. A new imitative driven by Google, called GData[2], uses the extension mechanism of APP and provides also queries and authentication functionalities. It allows sending full-text search queries to the underlying Web-based resource. The returned syndication XML format (Atom or RSS) is based on the Opensearch.org response elements, a specification driven by Amazon. Besides these new lightweight standards, existing application functionalities described with WSDL represent also an important and relevant API for Enterprise Mashups.

Widgets. Based on the resources and sourced via public APIs, widgets provide application domain functions or information specific functions. They are responsible for providing graphics, simple and efficient user interaction mechanisms which put a face to the resources and abstract from the technical description of the Web-based resources. By configuring and personalizing, the underlying Web-based resources can be used according to the individual requirements. Therewith, they tend to be designed with a focus on consumption and customization to ensure they are extremely flexible and reusable. However, even if the respected W3C published a draft Widgets specification[3], it lacks on a widespread widget model. Software vendors (like Microsoft, IBM, or Google) define their own widget model, NetVibes has the compelling Universal Widget Architecture (UWA), and OpenAjax

[1]http://ietfreport.isoc.org/idref/draft-ietf-atompub-protocol/
[2]http://code.google.com/apis/gdata/
[3]http://www.w3.org/TR/widgets/

has no component model per se but vital strategies for making Web components together in the same Mashup.

Mashup. By assembling and composing a collection of widgets stored in a catalogue, users are able to define the behavior of the actual application according to their individual needs. By aggregation and linking content of different widgets in a visual and intuitive manner , users are empowered to create their own operational environment which fits best to solve their heterogeneous business problems.

3.2 Emerging Intermediaries

As mentioned in the introduction, Enterprise Mashups have the potential to overcome the shortcomings of the traditional Web Service Stack regarding the lack of comprehensive, trustworthy and widely accepted service registries. Novel forms of intermediaries are currently about to emerge which offer resource registry functionality and extend the role of the traditional UDDI- or ebXML-based implementations. The typical resource discovery and utilization process is therefore different from the Web Service Stack [SC07]: After publishing resources by the provider of a mashable component, 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 consumer to select a resource. The growing number of mashable components [4] implicates an adequate description of the Mashup components for retrieval which are characterized by using the language of business users. According to the Web 2.0 Philosophy, the actual knowledge worker takes over this description process by tagging collaboratively them. It creates a folksonomy, essential a bottom-up, organic taxonomy that organizes the resource on the Web. Further user rating functionalities based on popularity and relevance are used to collect, distribute and aggregate feedback about the Mashup component' behavior by an integrated reputation system. By monitoring performance and by gathering user-based evaluations, the emerging intermediaries lower the risk for users (consumer) to select a bad component for their Enterprise Mashup. Web sites such as Programmableweb.com, Strikeiron.com, Seekda.com represent this new kind of emerging intermediaries which improve the navigation and transparency for users and help them to find the right Mashup component.

3.3 Mass Collaboration

Besides the emerging intermediaries, mass collaboration is a further design principle of Enterprise Mashups. Don Tabscott introduced the term ”prosumption communities”to describe the blurring between producers and consumers [TW06]. Customers participate in the creation of products in an active and ongoing way. Through co-innovation and co-production, the users in the business units do more than customization and personalization;

[4]978 Mashup APIs (http://programmableweb.com), 27.693 online Web Services (http://seekda)

they self-organize their individual operational environment and are empowered to submit new ideas. Without any hierarchy limitations, a knowledge worker creates „quick and dirty“ an Enterprise Mashup, shares it and the experiences with the community and consume 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 further its improvement in a peer production. The described democratized development and consumption process implicates being open. Not only can people share their remixes with three or four best friends, they are able to share them with thousands, and perhaps millions, on the Web.

3.4 Lightweight Composition

The central driver of Enterprise Mashups to address the long tail of user needs is the lightweight resource composition style to build individual enterprise applications by reusing build blocks in different contexts. As depicted in figure 1, the composition takes place both on the widget layer (piping) and on the Mashup layer (wiring) according to the Enterprise Mashup Stack as present before. 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 feeds directly as 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. Intuitive visual environments for the piping composition represent Yahoo Pipes or IBM Diama. The piping composition itself addresses users versed in classical development or data manipulation language. On the widget layer, the user is able to wire existing widgets together with behavior and data relationship by interconnecting visually their input and output parameters. For example, a form widget can be placed on a page, allowing a user to enter data. This data entered can be connected to the input of a widget that provides a Web-based resource invocation, and the output of the resource response can be connected to a widget that renders a visual display. The IBM

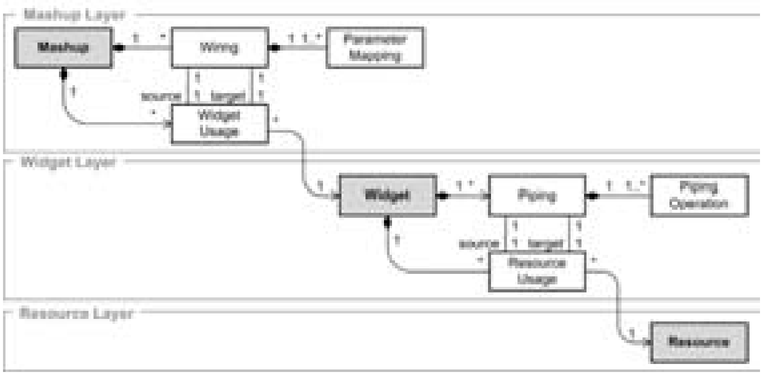

Abbildung 1: Enterprise Mashup Stack: Piping versus Wiring

Mashup Center[5] and the research project EzWeb[6] provide first concepts and platforms for the wiring concept.

3.5 Perpetual Beta Development Model

Regarding from a development model perspective, the classical software construction paradigm (requirements, design, implementations and test) is no longer valid. 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. Unlike traditional SOA-based applications, Enterprise Mashups will never be finalized. A beta version is not only an actual product but also media through which the next beta version is devised with dynamic negotiation with both technological and market innovation [Kak06].

4 FAST Platform

In the frame of the EU research project FAST we are currently developing a Web-based platform to create business relevant widgets built on resources encapsulating existing services from legacy systems [FAS09]. The resulting widgets - in the FAST project we call them gadgets - can be deployed in different Mashup platforms like iGoogle or Facebooks (consumer-oriented), EzWeb (enterprise-oriented) as well as on mobile devices.

According to the Enterprise Mashup Stack, the FAST Gadget architecture is depicted in figure 2. In contrast to existing consumer-oriented widgets characterized by one screen, we propose a *screen flow-oriented* widget model. It can consist of various *screens* (ready-to-compose building block that can be considered fully functional by itself) that are connected with each others by *flow operations* (i.e., gateway). In enterprise environments, widgets are confronted with the presentation of lot of information which can not be handled in only one screen. Enriched with semantic annotations defining pre and post conditions of interconnected screens, FAST assistants the users to create business relevant gadgets. Besides the actual *form* representing the graphical user interface of the screen, a set of *resources*

[5]http://www-01.ibm.com/software/info/mashup-center/

[6]http://ezweb.morfeo-project.eu/

build the content of a screen. In order to handle the disparate interfaces or invocation mechanisms of *services* in the back-end, resource adapters wrap the services to a FAST unified resource structure.

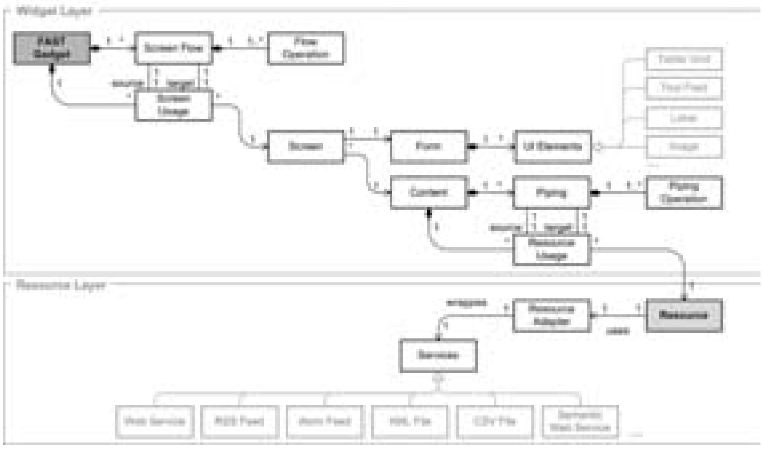

Abbildung 2: FAST Gadget Meta Model

To cover navigation and transparency aspects, the FAST platform combines concepts from the Semantic Web initiatives with the Web 2.0 philosophy. In this sense, a FAST ontology defines the conceptual model for organizing the various components (gadgets, screens, flow operation, resources). In addition, each instance will have a slot to allow annotations with non-predefined tags. Enriched with user profiles and history data, FAST provides relevant mashable components to the user according to his current context. The so-called FAST catalogue is responsible for the intermediary role. For the lightweight composition (in case of FAST it's piping) the so-called FAST Visual Gadget Studio allows the user to choose among preselected domain specific building blocks to create in a drag and drop fashion a widget. To deploy the created widgets, the FAST gadget adapter deals with Mashup platform capabilities and available policies in order to adapt the FAST gadgets to different environments (EzWeb, iGoogle, Facebook, etc.).

In contrast to existing Mashup tools (Microsoft Popfly, Yahoo Pipes, SAP Research Rooftop, Google Mashup Editor, IBM Mashup Center, etc.) [HF08], the FAST platform integrates semantics specified in RDF to organize the growing number of mashable components. By supporting users in reusing the components to value added applications, FAST provides only these components to the users which are relevant in their individual context. Existing Mashup tools are lacking on adequate discovery mechanisms so far. Further, the philosophy of FAST to develop a gadget which can be deployed in multiple Mashup platforms allows the portability and reusability of content on multiple channels (desktop, browser, mobile devices). Especially in enterprise environments, this will reduce costs by consuming the created widgets in multiple environments.

Figure 3 shows the FAST prototype to design screen flow gadgets. By drag and drop, the user is able to specify a process-oriented gadget by connecting the screens with each other. The post and pre conditions in the botton check the semantic interoperability of the screens. For example, the „order screen“ on the left is marked as red, because the fact

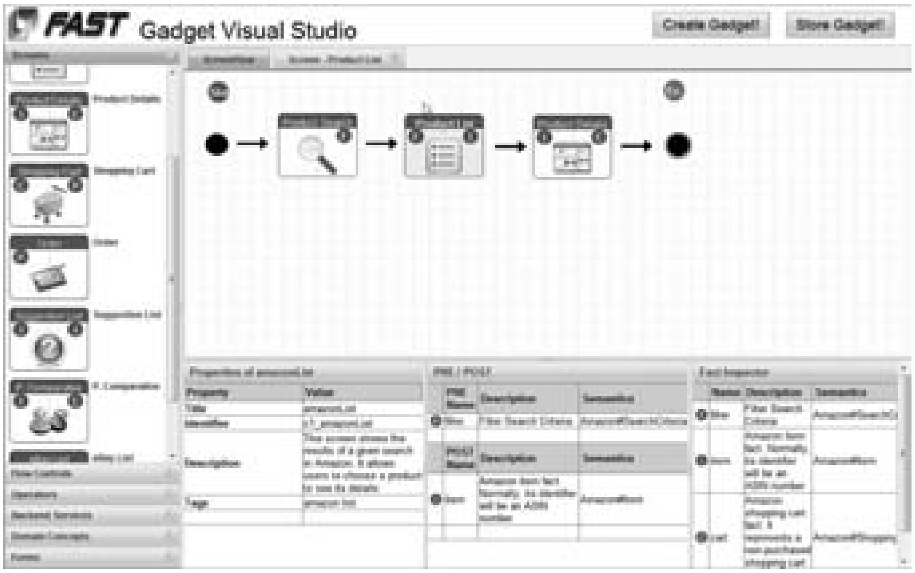

Abbildung 3: FAST Screen Flow

(product number - P) is not available in one of the three selected screens. In case the user moves the „shopping cart screen“ to the canvas, the „order screen“ will marked as green. The „shopping cart screen“ has the post condition P fact allowing the connection with the „order screen“.

5 Conclusion

The aim of the paper is the analysis of emerging design principles of the Enterprise Mashup paradigm allowing to implement a commons-based peer production business application development. In order to achieve this, the main terms related to Enterprise Mashups were defined. Based on experiences from first Mashup prototypes (i.e., SAP Research Rooftop) and a literature analysis, we present novel design principles regarding from the perspectives architecture, community, programming style, and development process. By means of the FAST research project funded by the European Union, we apply the principles in a real-world application.

However, the design principles serve only as a starting point. Based on the findings presented in this position paper, further work will deal with a development of a reference model for Enterprise Mashup environments by leveraging the St. Gallen Media Reference Model. First results can already be found at [HSS08]. In addition, we are currently developing the FAST platform according to the presented design principles. To measure the economic benefits of the Mashup paradigm in enterprises, we will implement various real-world industry scenarios. A first version of the Open Source FAST platform will be available in March 2009.

Acknowledgments.

This paper has been created closely to research activities during the EU-funded project FAST (INFSO-ICT-216048) [FAS09].

Literatur

[ACKM04] G. Alonso, F. Casati, H. Kuno und Vijay Machiraju. *Web Services Concepts, Architectures and Applications*. Springer, Berlin et al., 2004.

[And06] C. Anderson. *The Long Tail: How endless choice is creating unlimted demand*. Random House Business Books, London, 2006.

[Ben06] Y. Benkler. *The Wealth of Networks. How Social Production Transforms Markets and Freedom*. Yale University Press, New Haven and London, 2006.

[CBG$^+$07] L. Cherbakov, A. Bravery, B.D. Goodman, A. Pandya und J. Bagget. Changing the corporate IT development model: Tapping the power of grassrots computing. *IBM System Journals*, 46(4), 2007.

[Dav05] T.H. Davenport. *Thinking for a Living: How to Get Performance and Results from Knowledge Workers*. Harvard Business School Press, Boston, 2005.

[DMY$^+$07] Florian Daniel, Maristelle Matera, Jin Yu, Boualem Benatallah, Regis Saint-Paul und Fabio Casati. Understanding UI Integration. A Survey of Problems, Technologies, and Opportunities. *IEEE Internet Computing*, 11(3):59–66, 2007.

[FAS09] FAST. EU project FAST (INFSO-ICT-216048). *http://fast.morfeo-project.eu/*, 2009.

[HF08] V. Hoyer und M. Fischer. Market Overview of Enterprise Mashup Tools. *Lecture Notes in Computer Science*, 5364:708–721, 2008.

[Hoy08] V. Hoyer. Zusammengerührt. Ad-hoc-Software aus der Fachabteilung. *ix Magazin für Professionelle Informationstechnik*, (10):98–102, 2008.

[HSS08] V. Hoyer und K. Staneovska-Slabeva. The Changing Role of IT Departments in Enterprise Mashup Environments. In *Proceedings of the 2nd International Workshop on Web APIs and Service Mashups*, 2008.

[HSSJS08] V. Hoyer, K. Stanoevska-Slabeva, T. Janner und C. Schroth. Enterprise Mashups: Design Principles towards the Long Tail of User Needs. In *IEEE International Conference on Services Computing (SCC)*, Jgg. 2, Seiten 601–602, 2008.

[JCH$^+$07] T. Janner, V. Canas, J.J. Hierro, D. Licano, M. Reyers, C. Schroth, J. Soriano und V. Hoyer. Enterprise Mashups: Putting a face on next generation global SOA. In *Tutorial at the 8th International Conference on Web Information Systems Engineering (WISE 2007)*, 2007.

[Jhi06] A. Jhingran. Enterprise Information Mashups: Integrating Information, Simply. In *Proceedings of the 32nd international conference on Very large data bases*, Seiten 3–4. VLDB Endowment, 2006.

[Kak06] M. Kakihara. Strategizing Software Development: Strategic Management of Internet Services Development. In *Proceedings of the International Workshop on Interdisciplinary Software Engineering Research*, Seiten 37–43, 2006.

[MRG08] E.M. Maximilien, A. Ranabahu und K. Gomadam. An Online Platform for Web APIs and Service Mashups. *IEEE Internet Computing*, 12(5):32–43, 2008.

[SC07] C. Schroth und O. Christ. Brave New Web: Emerging Design Principles and Technologies as Enablers of a Global SOA. In *Proceedings of the IEEE International Conference on Service Computing (SCC 2007)*, Seite 8, 2007.

[TW06] D. Tapscott und A.D. Williams. *Wikinomics: How Mass Collaboration Changes Everythink*. Portfolio Hardcover, New York, 2006.