Cloud-based Enterprise Mashup Integration Services for B2B Scenarios

Robert G. Siebeck*†
robert.siebeck@sap.com

Till Janner*
till.janner@sap.com

Christoph Schroth*
christoph.schroth@sap.com

Volker Hoyer*‡
volker.hoyer@sap.com

Wolfgang Wörndl†
woerndl@informatik.tu-muenchen.de

Florian Urmetzer*
florian.urmetzer@sap.com

*SAP Research CEC St. Gallen
Blumenbergplatz 9
9000 St. Gallen, Switzerland

†Technische Universität München
Institut für Informatik
Boltzmannstr. 3
85748 Garching, Germany

‡University of St. Gallen
MCM Institute
Blumenbergplatz 9
9000 St. Gallen, Switzerland

ABSTRACT

We observe a huge demand for situational and ad-hoc applications desired by the mass of business end-users that cannot be fully implemented by IT departments. This is especially the case with regard to solutions that support infrequent, situational, and ad-hoc B2B scenarios. End users are not able to implement such solutions without the help of developers. Enterprise Mashup-/ and Lightweight Composition approaches and tools are promising solutions to unleash the huge potential of integrating the mass of end users into development and to overcome this "long-tail" dilemma. In this work, we summarize different patterns on how to realize B2B collaborations between different Mashup platforms. We also evaluate cloud computing infrastructures available on the Web as they might be a suitable platform for B2B integration via Mashups. As a major result, we provide an overview and first results of an architecture and implementation of a prototype that implements an API for Enterprise Mashup integration services. This prototype is built on the basis of different existing cloud infrastructures and services.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; H.4.5 [**Information storage and retrieval**]: Online information services; H.5.3 [**Information interfaces and presentation**]: Group and organization interfaces

General Terms

Design, Economics

Keywords

Enterprise Mashups, B2B, Integration, Cloud Computing, End-user Development, Patterns

MEM2009 workshop in conjunction with *WWW2009*, April 20-24, 2009, Madrid, Spain.

1. INTRODUCTION

Currently available solutions to support B2B collaborations focus on the automatization of long-term business relationships and still lack to provide their users intuitive ways to modify or to extend them according to their ad-hoc or situational needs. Conventional proceeding in the development of such applications directs to an immense use of time and work due to long development cycles and a lack of required business knowledge. As a result, the long tail of situational, ad-hoc, tactical, or individual solutions for the mass of business end users are often not implemented at all (see figure 1 on the basis of [2] and [10]), or do not fully support the evolved business needs [3]. Especially in the area of applications to support B2B collaborations, current offerings are characterized by a high richness but low reach, like B2B hubs that focus on many features enabling electronic collaboration, but lack availability for especially small organizations or even individuals [17, 19]. The other extreme, solutions with a low reach but high richness such as email or phone, lack standardization and formularization which makes them inappropriate for automated or special enterprises' needs [17, 19] (see also figure 2).

New development approaches are needed to overcome theses hurdles and to involve the group of non-technical business users into the development process in order to address this long tail of their requirements, to realize cost-effects and efficiency gains [23], and to overcome the traditional problems between IT department and business units - poor quality of support and low reaction time [8].

Especially Enterprise Mashups, a new generation of Web-based applications, seem to adequately fulfill the individual and heterogeneous requirements of end users [8] and to foster End User Development (EUD). To shorten the traditional development process following the waterfall model, these new breed of applications are developed by non-professional programmers, often in a non-formal, iterative, and collaborative way by assembling existing building blocks [3].

Major result of this work is the presentation of this new development style in combination with recently emerging cloud infrastructures and services. We introduce an architecture of a cloud-based Enterprise Mashup Integration Ser-

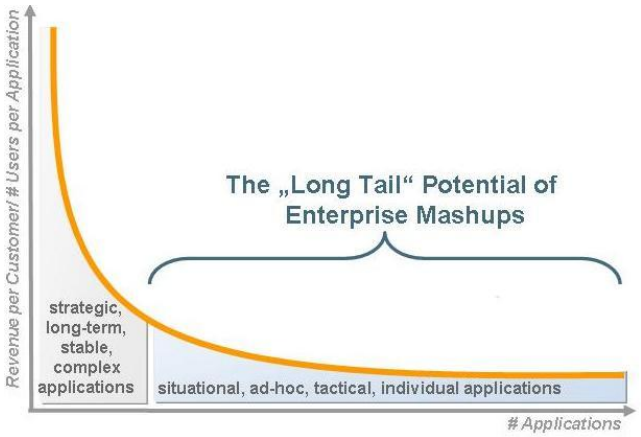

Figure 1: The "Long Tail" Potential of Enterprise Mashups [2, 10]

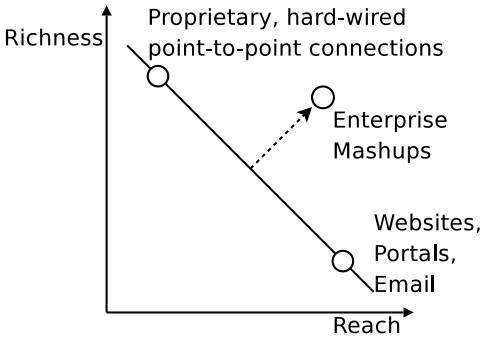

Figure 2: Trade-off between, and improvement of *richness* and *reach* [18]

vices Platform that enables users of Enterprise Mashups to create solutions for their ad-hoc, situational, and individual needs in B2B collaborations.

The remainder of this paper is structured as follows: chapter 2 gives an introduction into challenges of B2B integration. Also, the used terminology and techniques as well as currently available solutions on the market are presented: Enterprise Mashup platforms are presented as a means to solve the discussed challenges; Cloud-computing infrastructures are evaluated as a platform to host integration services for Enterprise Mashups. In chapter 3, we discuss the value of Enterprise Mashups for B2B scenarios and present five basic patterns how Enterprise Mashups can be used to support B2B collaborations. As those five patterns still lack to solve solutions for certain B2B problems, we introduce another pattern. Chapter 4 provides an insight into this sixth pattern and proposes a Cloud-based infrastructure to realize a prototype. In chapter 5 we summarize this work and give an outlook on future research on this topic.

2. BACKGROUND AND RELATED WORK

2.1 B2B integration challenges

As mentioned in chapter 1 one of the challenges for enterprises implementing a B2B collaboration is the trade-off between *richness*, that is solutions with a broad functionality, and *reach*, the number of users the solutions is suitable for [18] (see also figure 2):

On the one hand, there are *proprietary, hard-wired point-to-point connected* systems which offer high richness, but have a low reach. Those systems include commercially available *B2B hubs* or *B2B communities*, offering many features enabling electronic collaboration, but lack availability for many participants, especially small organizations, as they usually cannot afford expensive solutions. Also, high formalization makes these systems unsuitable for ad-hoc situations, where flexibility is important.

On the other hand, there are solutions offering a high reach but low richness. Those comprise *classical* communication techniques, like *Websites, Portals or Emails*. They are available for nearly every person or organization, but lack standardization and formalization, and thus making them inappropriate for highly automated processes or enterprises' special needs.

Service oriented architectures (SOA) have been presented as a solution to organization's integration problems for the last years. While SOA has simplified some integration issues, it still has to face some difficulties [11]. Enterprise service buses (ESB) are used to integrate the different services in a SOA driven company. However, most ESBs are not designated for cross-organizational collaboration, and thus raise problems when establishing such a collaboration. Also, SOA may simplify the integration of new services, but this still can only be done by expensive developers. End-users usually are not able to realize integration scenarios. This leads, beneath high costs for integration projects, to inflexibility, because integration projects last longer, although market competition demands a timely response to uprising requirements [13].

Another challenge in B2B integration is the ownership of and responsibility for processes [14]: "in many inter-organizational settings [...] business processes are only sparsely structured and formalized, rather loosely coupled, and/or based on ad-hoc cooperation – and often there is no explicit or implicit agreement of process ownership."

According to Ward-Dutton [22], inter-organizational collaborations tend to involve more and more participants, while the growing number of participants also draws a huge amount of differing requirements. Also, the participants may act according to different roles, controls and priorities, which used to be different: "Historically, the focus for collaboration was participation within teams which were managed according to one set of rules: most often all participants worked within a single department. Now, in supporting supplier and partner co-innovation and customer co-creation, the focus is shifting to collaboration which has to embrace participants who are influenced and restricted by multiple domains of control and multiple differing processes and practices." [22, p. 6]

Minsk et al [15] describe a shift from static B2B approaches to new, dynamic B2B integration, which can adaptively react to unexpected disruptions, allow a rapid customization and can manage increasing complexity by the use of end-to-end business processes.

The problems presented in the last paragraphs can be, at least partially, solved by the use of so called B2B hub soft-

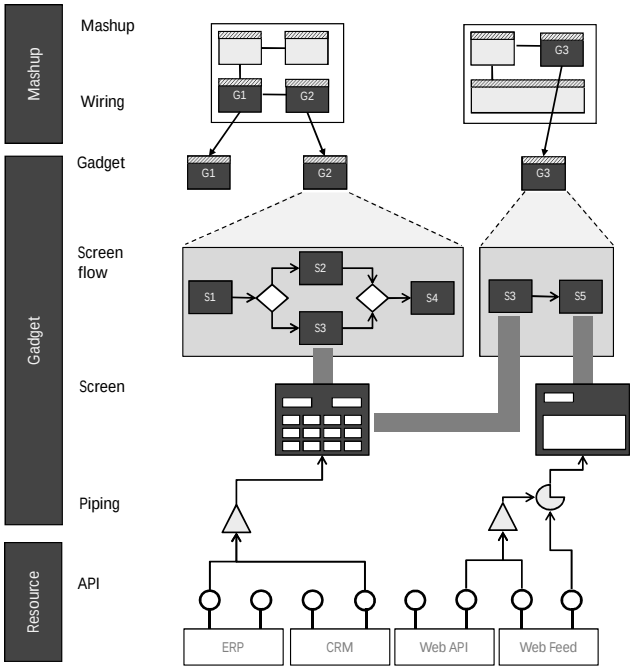

Figure 3: Mashup terminology [10]

ware. Commercially available solutions that support B2B collaborations can be classified according to their functional capabilities [20]: Both *Electronic data interchange translators (EDI)* and *Managed file transfer (MFT)* have a longer history, while B2B gateways only have emerged during the last decade. Additionally, there are *integration as a service (IaaS)* offerings, as well as *business to business process outsourcing (B2BPO)* providers, which out-source the service of integration for the collaborating organizations. However, most of the available solutions aim at supporting medium to larger companies, resulting from their high costs and long implementation times, which make them unaffordable to smaller organizations. Consequently, these offerings are not suitable for short-term collaborations which need to be set up in an adhoc manner.

2.2 Enterprise Mashup Platforms and Tools

Mashups, being a relatively new technology, lack a clear definition in literature. Hoyer and Fischer [7] distinguish between Consumer and Enterprise Mashups, where both are used to combine data elements from different sources and aggregate them. Enterprise Mashups raise additional needs, like security, availability or quality. Also, according to them, "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." [7, p.11]

Literature lacks a clear definition of Mashup-related terms, like resource, widget, API (application programming interface) et cetera. We use the terminology presented by [7] and [10], which consists of the following parts (see also figure 3):

- *Resources* contain content, data or application functionality and represent the core building blocks for Mashups. Resources can be accessed through APIs, which encapsulate the resources and describe the interface through which the they are made available.
- *Widgets* or *gadgets* primarily put a face on the underlying resources by providing a graphical representation for them and *piping* the data received from the resources. Piping can include operators like aggregation, merging or filtering. According to [9], the Mashup Stack can be extended for Enterprise Mashups by *complex gadgets* which consist of several *screens*. Screens are fully functional by themselves, and their pre- and post-conditions drive the transitions among them to tie them together, forming a *screenflow*.
- *Mashups*: "By assembling and composing a collection of widgets [called *wiring* in figure 3] 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." [7]
- Consequently, a *Mashup platform* is a Web based tool that allows the creation of Mashups by piping resources into Gadgets and wiring Gadgets together.

Hoyer et al [9, 10] also point out three further design principles of Enterprise Mashups, *emerging intermediates*, *mass-collaboration* and *lightweight resource composition*. Emerging intermediates name the functionality that provides a registry for the growing amount of resources which are available for use in Mashups. They offer features to collect, classify, describe, rate and monitor the resources to make them available to the end-users. Mass collaboration describes the participation of many end-users in the process of creating Mashups, Gadgets and Resources, by using and sharing them on the one hand, and helping to improve them by giving feedback or developing themselves, on the other hand. Lightweight resource composition stands for an easy (re-)use of once created artifacts, like Resources, Gadgets or Widgets. Through piping and wiring, as depicted in figure 3, users are able to use the same artifact in different scenarios. Also, according to [6] this includes *late binding*, which means that each artifact is not bound to other artifacts when it is developed, but when it is used.

There are several products available both on the Consumer and on the Enterprise Mashup market, table 1 lists some important examples [7].

2.3 Overview on Cloud Computing Infrastructure and Services

Cloud computing can bee seen as a novel way of delivering IT-enabled services[1] to customers. They are used "where massively scalable IT-enabled capabilities are delivered *as a service* to external customers using Internet technologies." [16] As cloud services are delivered to the customer on demand, they provide both scalability and elasticity, where scalability is the ability to deliver the performance needed

[1]Services in this context have to be seen in contrast to components like hard- and software

Vendor	Product	Enterprise Mashup	Consumer Mashup
Dapper	Dapp Factory		■
Google	iGoogle		■
IBM	Mashup Center	■	
JackBe	Presto Edge	■	
Kapow	Mashup Server	■	
Microsoft	Popfly		■
Netvibes	Netvibes		■
Yahoo	Pipes	■	■
SAP	Research Rooftop	■	

Table 1: List of Mashup tools [7]

by customers, whereas elasticity also includes the "ability to support those needs in large or small scale at will. The key issue with elasticity is the ability for a system to scale both in an upward direction (for example, to millions of users) and in a downward direction (for example, to one user)" [16]. Another important aspect of cloud computing is the payment model: customers only pay for the resources they use, that is the computing power they consume [1].

In [4], three different styles of Cloud Computing are characterized:

- Infrastructure-as-a-Service (IaaS[2]) delivers infrastructure services as processing power and storage capacity. It can be seen as an elasticity-extended version of hosting computers in a data center.
- Software-as-a-Service (SaaS) provides software in the internet, without giving the user knowledge or control over the underlying infrastructure.
- Platform-as-a-Service (PaaS) is about delivering portals or platforms to simplify the access and combination of SaaS-offerings.

According to Buyya et al [1], a *cloud* is a parallel and distributed system accessible from anywhere in the world on demand; they primarily describe a IaaS computing cloud. It consists of a collection of connected and virtualized computers. These computers are dynamically provisioned and may appear as one or more unified computing resources to the user of the cloud. These physical computers host many virtualized machines. Through a resource allocator users can access the virtual machines and run their applications on them. Features like accounting, dispatching and monitoring support the operation of the computing cloud for the cloud's provider and the end-users. Service Level Agreements (SLA) ensure the reliability and availability of the cloud computing resources for the end-user.

Table 2 presents a comparison of major cloud infrastructures (Amazon EC2, Google AppEngine, Sun Grid Engine and Salesforce's Force.com) and their characteristics. The *style* classifies the infrastructures according to [4], as introduced above. *Platform* describes the technical infrastructure available to the end-user. *Additional APIs* describe which cloud-based offers, that is SaaS products, the providers offer additionaly to the infrastructure services. The *target application* describes for which type of applications the infrastructure is specifically suitable.

[2]Not to be confused with Integration-as-a-Service, which is also often abbreviated with IaaS.

3. ENTERPRISE MASHUP INTEGRATION PATTERNS

3.1 Enterprise Mashups and their potential for B2B scenarios

As depicted in the last section, Enterprise Mashups are primarily focused on the use in companies to enable the construction of ad-hoc applications by the end-user. However, we believe that they also have advantages in B2B scenarios, especially because they can be used to aggregate data from different sources. Of course, resources do not have to originate from the same organization as where the Mashup application is developed.

Mashups can resolve many of the disadvantages of *B2B hubs*, like e.g. a low reach due to hard-wired connections, by enabling end-user development and lightweight connections of systems. Still, they can help adding richness to existing lightweight solutions such as Websites or Portals by adding a certain level of formalization and standardization and thus enabling automatization, which is an important enterprises' need. This places Enterprise Mashups in the upper right corner of the richness and reach model as shown in figure 2.

"Mashups enable the ease of mixing and transforming various sources of information internally and from business partners. [...] Complexity in B2B operations is often linked with heterogeneous systems and platforms. The tedious integration process and requirements of various support and maintenance for the software is a major hindrance to today's dynamic B2B integration, especially for the Small and Medium Enterprises." [15, p. 319]

3.2 Basic Mashup Integration Patterns

There are several ways to establish a cross-organizational collaboration using Enterprise Mashups [12]:

Pattern 1: *Sharing of the Mashup platform*: a company can give access to other companies to use their Mashup platform to run applications which are deployed there. This pattern is simple to implement, as only access has to be granted, but it lacks automatization and thus is only of limited benefit. Of course, this pattern could as well be realized with a classical Website or a Portal application.

Pattern 2: *Provide a Gadget*: by providing a Gadget, a company can give business partners the possibility to use that Gadget within their own Mashup platform. They thus can connect the Gadget with their own Gadgets and wire them up to build a new application.

Pattern 3: *Provide a Screen*: similarly to the previous pattern, a company can share a Screen. By providing a Screen, they give more flexibility to their partner integrating that Screen in their Mashup application. On the other hand this adds complexity and thus makes the whole process harder to handle.

Pattern 4: *Provide an API/Resource*: this pattern is also more flexible than the previous one, as the partner accessing the API can decide for themselves how the resource is presented to the end-user. However, in this case a developer could be required to build a good interface representing the API.

Pattern 5: *Connect resources*: companies could connect their back-end resources, e.g. ERP systems, to exchange data. By providing a visual representation of the ERP systems through a Mashup, this setting can be expanded to a

Provider	Amazon	Google	Sun	Salesforce
Product	Elastic Compute Cloud (EC2)	AppEngine	Grid Engine	Force.com
Website	http://aws.amazon.com/ec2	http://code.google.com/appengine	http://www.sun.com/software/sge	http://www.salesforce.com/platform
Style	IaaS	PaaS, IaaS	IaaS	PaaS, SaaS
Platform	Virtual Machines (Amazon Machine Images) running Linux or Windows	Python Web application containers	Virtual Machines running Solaris	Apex (programming language of Force.com) Sandbox
Additional APIs	Amazon Web Services, e.g. Simple Storage Service (S3), Simple Queue Service (SQS)	Google APIs, e.g. Google Accounts (user management), Datastore API, Email services	-	Salesforce API, Connectors
Target applications	any	Web applications	any	Enterprise web applications

Table 2: Comparison of some representative Cloud platforms (based on [1] and [20])

Mashup use-case.

If one applies the model of richness and reach as introduced in chapter 1, the patterns also tend to suffer from this trade-off. Pattern 1 has a high reach due to the fact that it has nearly no technical requirements for most participants in the collaborations. The richness would be quite low, on the other hand, because integration with their own systems is difficult or not possible for the parties accessing the Mashup pattern which was shared by their business partner. Pattern 5 on the contrary has a very low reach, as it depends on hard-wired connections to be established. Still, it allows a high richness, as the both back-end systems in the collaboration can be configured to handle any data exchange. Patterns 2 to 4 are in between the two extremes of patterns 1 and 5 regarding the richness and reach. However, the move to the upper right corner of figure 2 to reach both high richness and reach can still be improved.

3.3 Mashup Integration Services for B2B

To address the trade-off between richness and reach for the patterns presented in the last section, we proposed another pattern using *Mashup Integration Services for B2B* in [21]:

As visualized in figure 4, two companies, A and B, have applications deployed in their own Mashup platforms. They connect their own back-end systems to their Mashup applications by accessing them through APIs. Beneath their back-end systems, they both connect to the Mashup Integration Services, also via an API. These services handle the data exchange between both companies and support the collaboration by offering certain features, which are introduced the next chapter.

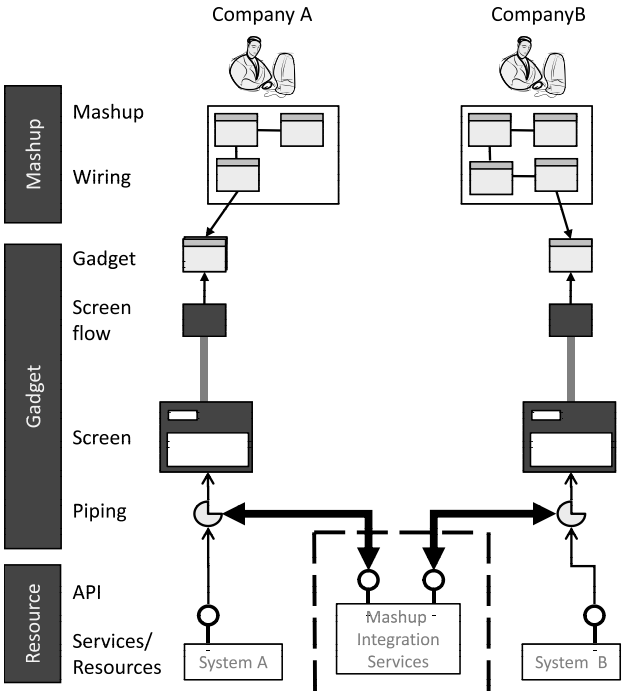

Figure 4: Mashup Integration Services [21]

4. CLOUD-BASED PROTOTYPICAL REALIZATION

The Mashup Integration Services described in the last chapter are being implemented as a prototype in the course of the FAST project [5]. In this chapter, we first describe the layers of the prototype and the requirements for such services, that is what services are needed and what their characteristics are. Consecutively we describe an architecture which describes how these services work together. We furthermore give an outlook on the technical realization of the services using cloud infrastructures and services.

4.1 Prototype Layers

In [21] the requirements of the Mashup Integration Services mentioned in section 3.3 are classified by the three SOA layers according to [19]. The uppermost layer ist the *organizational layer*, which describes the different services that have to be connected to each other. The *language or semantics layer* handles the objects of interaction and their types. The *services and infrastructure layer* is about the technical base to enable the requirements of the other two layers.

A detailed presentation of the requirements of the different

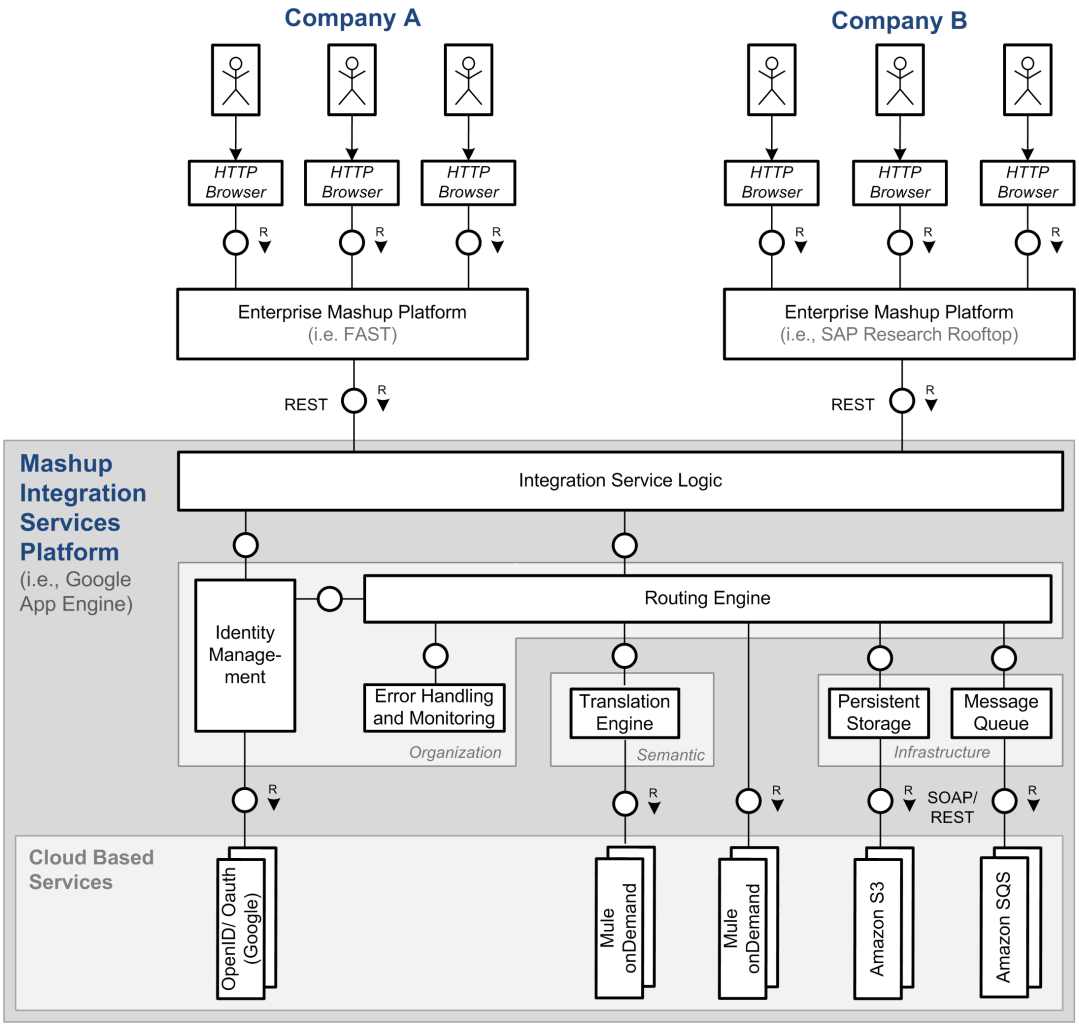

Figure 5: Architecture of the Mashup Integration Services for B2B

identified Mashup Integration Services can be found in [21].

4.2 Prototype Architecture

Figure 5 shows the services described in the last section and their relations to each other. The core services are shown within the box in the middle. The external services shown under the box are attached via APIs to allow the usage of third-party offerings to realize their functionality.

Users access the services through a Mashup platform of their choice. The Mashup platforms are connected via APIs to the Mashup integration services. To use the services, the users have to identify themselves against the user access control service; this service is connected to a user management service, which controls the users and their settings. The user management service is connected via an API to allow the usage of external services, e.g. a corporate user database. All data coming from the users go through a translation engine to unify the data objects and protocols, so that different Mashup platforms can be integrated. The translation engine has an interface which allows connecting other external translation engines to add support for additional protocol and data standards. The translated data is forwarded to the routing engine, which is the core of the Mashup integration services. The routing engine takes care of processing the inputs received from the Mashup platforms and forwarding them to the right recipient; the routing is based on rules, which can be configured through an API. To simplify this, a Gadget could be provided for the end-user. The routing engine is also connected to a message queue via an API. Thus, different message queue engines are attachable. The message queue is responsible for storing and forwarding the messages controlled by the routing engine. Beneath the message queue, a persistent storage, also connected via an API to allow exchangeability, is available to store large data. The error handling and monitoring service allows tracking the message flow to detect errors and to collect statistical data.

Regarding the implementation of Mashup integration services, the question of the used infrastructure arises. Of course, enterprises running Mashup platforms usually have IT infrastructure available. In the B2B context, this would result in many different infrastructures, and a conflict of responsibility for the integration services would arise. Thus, we propose deploying the Mashup integration service as a *cloud-based* service.

Also, there are cloud-based services available which provide the functionality required by the services described in the previous sections. In this way, the Mashup integration service can reuse and leverage existing cloud services to in order to be implemented as efficient as possible. This section shows which services are available and can be used to provide the described Mashup integration services. Of course, primarily the external services (the services shown under the box in figure 5) are suitable to be realized by using cloud-offerings. The core-services themselves need to be developed customly, but they still could be hosted on a cloud environment, e.g. on one of the offerings presented in section 2.3.

4.2.1 User management

There are several services available on the web which provide user management. However, as many enterprises provide their own user management services, we propose to attach the user management service via an API so that it is exchangeable. Of course, a default user management should be provided so that the integration services are not dependant on the avaliability of a corporate user management. OpenID[3] is not exactly a cloud-run service but an identity management protocol supported by many large Mashup platform providers, e.g. Yahoo or Google.

4.2.2 Message queue

The message queue could be realized by using Amazon's Simple Queue Service[4] (SQS). SQS is a web-service which provides a queue for messages and stores them until they can be processed. The Mashup integration services, especially the routing engine, can put messages into the queue and recall them when they are needed.

4.2.3 Persistent storage

Amazon Simple Storage Service[5] (S3) is also a web-service which allows to store files. The routing engine can use this service to store large files.

4.2.4 Translation engine

The translation engine within the core services of the Mashup integration services is primarily focused on translating between different protocols which the Mashup platforms it connects can understand, e.g. REST or SOAP web services. However, if the need of translation of the objects transferred arises, this could be attached to the translation engine. A company requiring such a service could on the one hand develop such a service and connect it to the Mashup integration services. Another possibility for this would be to connect existing translation services, e.g., the services by Mule on Demand[6], which is also a cloud-based offering.

4.3 Interaction between the services

Figure 6 displays the interaction between the different Integration Services. The diagram describes the process of a message being delivered and handled by the Mashup Integration Services Platform. The precondition for this process is that a user already established a route to a recipient.

After having received a message from an Enterprise Mashup tool via an API, the Integration Services first check the access rights of the sender of the message against an external service, e.g. the OpenID-services. An incoming message is processed only if sender of the message is authorized, that is he has the right to deliver the message to the recipient and to use the Mashup integration services; if he's not authorized the processing stops and an error is logged. The error log message is written into a log-file, which could reside on Amazon's Simple Storage Service. If the message has been accepted, it is put in the message queue in Amazon's SQS service. If required, the message is being translated into another format, which can also be done by an external, cloud-based service. After that, the services can begin trying delivering the message to a recipient. Evaluating the recipients of the message is based on the rules stored in the routing engine which have been configured by a user before. Finally, the successful delivery of the message can be logged, or an error if one occured.

[3]http://openid.net/
[4]http://aws.amazon.com/sqs/
[5]http://aws.amazon.com/s3/
[6]http://www.mulesource.com/

5. CONCLUSION AND OUTLOOK

In this work, we first gave an insight into challenges of B2B integration, and presented commercial solutions available on the market. We presented Enterprise Mashups as a novel way to solve some of these issues, as they have a certain value in the enterprise world. Then we briefly summarized five patterns, which describe ways how to use Enterprise Mashup tools in the B2B integration context. These patterns all have advantages and disadvantages. Besides that, we identified further issues, which are not addressed by the patterns, but which are important in enterprises' context. These issues include, for example, requirements on security. To address those issues, we proposed a novel way for using Enterprise Mashups in B2B context, which we call Mashup B2B integration services. Finally, we described the different layers and requirements, how such integration services are characterized, and how they could be prototypically implemented by using different cloud services and infrastructures.

An evaluation of the presented integration services has been done in course of the FAST project [5] and can be found in [21]. We refer to [12] for further details on the Mashup Integration Patterns.

In line with the previously mentioned FAST project, the prototype of the described services is being implemented.

6. ACKNOWLEDGEMENTS

This work is supported in part by the European Commission under the first call of its Seventh Framework Program (FAST STREP Project, grant INFSO-ICT-216048).

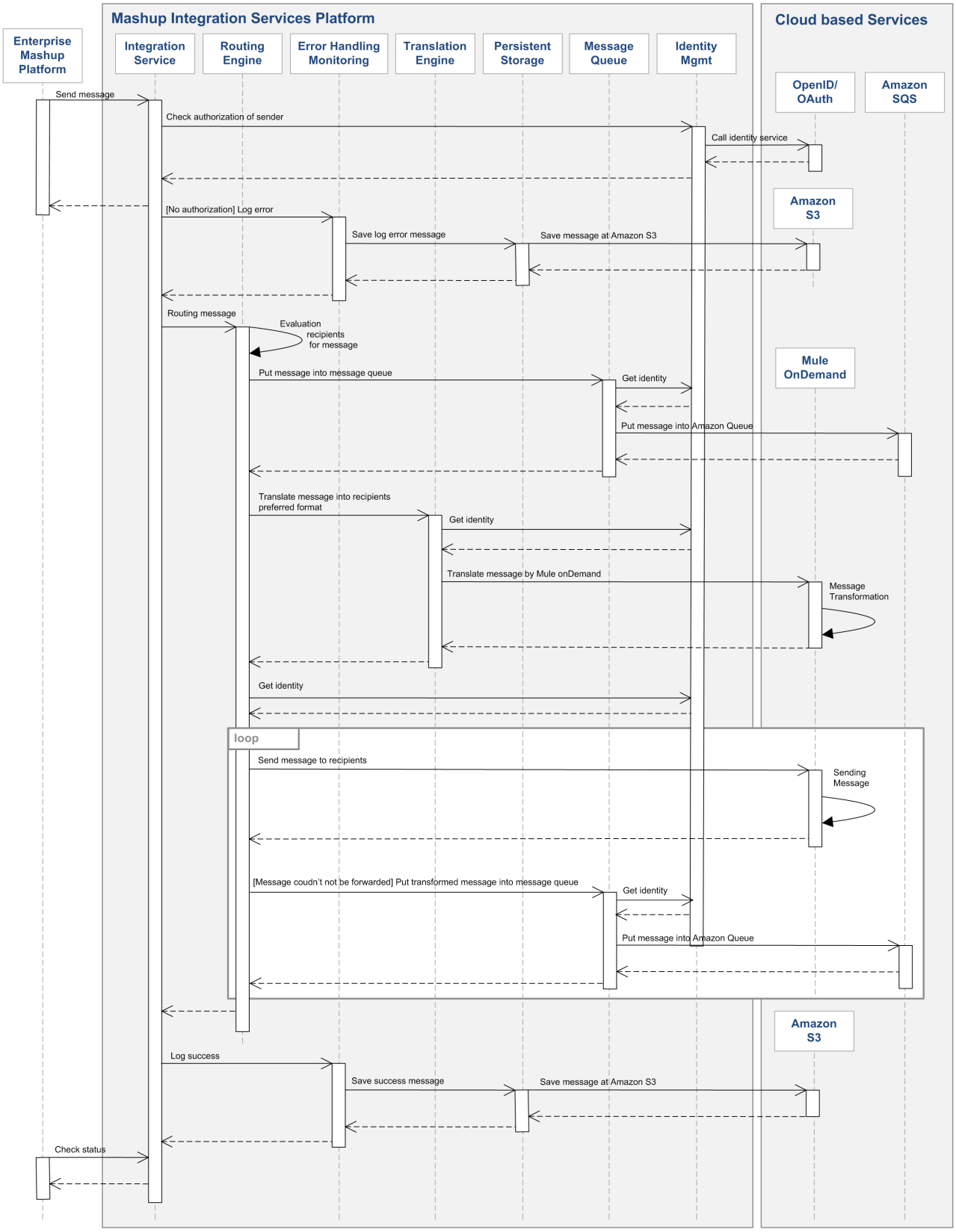

Figure 6: UML Sequence diagram: Interaction between the Mashup Integration Services for B2B

7. REFERENCES

[1] R. Buyya, C. Yeo, S. Venugopal, M. Ltd, and A. Melbourne. Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC-08)*, pages 25–27, Los Alamitos, CA, USA, 2008. IEEE.

[2] N. Carrier, T. Deutsch, C. Gruber, M. Heid, and L. L. Jarrett. The business case for enterprise mashups. White paper, 2008.

[3] L. Cherbakov, A. Bravery, B. D. Goodman, A. Pandya, and J. Baggett. Changing the corporate it development model: Tapping the power of grassroots computing. *IBM Systems Journal of Computing and Information Technology*, 46(4):743, 2007.

[4] T. Eymann. Cloud computing, 2008.

[5] Fast and advanced storyboard tools (FAST) Project. http://fast.morfeo-project.eu. Accessed on 02/09/2009.

[6] R. Guo, B. B. Zhu, M. Feng, A. Pan, and B. Zhou. Compoweb: A component-oriented web architecture. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 545–554, New York, NY, USA, 2008. ACM.

[7] V. Hoyer and M. Fischer. Market overview of enterprise mashup tools. In *Proceedings of the 6th International Conference on Service Oriented Computing (ICSOC 2008)*, pages 708–721, Sydney, Australia, 2008. Springer.

[8] V. Hoyer and K. Stanoevska-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.

[9] V. Hoyer and K. Stanoevska-Slabeva. Design principles of enterprise mashups. In *Proceedings of the 5th Conference of Professional Knowledge Management*, Solothurn, Switzerland, 2009. Gesellschaft für Informatik.

[10] V. Hoyer, K. Stanoevska-Slabeva, T. Janner, and C. Schroth. Enterprise mashups: Design principles towards the long tail of user needs. In *Paper presented at the 2008 IEEE International Conference on Services Computing (SCC 2008)*, Honolulu, Hawaii, 2008.

[11] T. Janner, M. A. Canas Vaz, J. Hierro, D. Licano, M. Reyers, C. Schroth, J. Soriano, and V. Hoyer. Enterprise mashups: Putting a face on next generation global soa. In *Presented at The 8th International Conference on Web Information Systems Engineering (WISE 2007)*, Nancy, France, 2007.

[12] T. Janner, R. Siebeck, C. Schroth, and V. Hoyer. Patterns for enterprise mashups in b2b collaborations to foster lightweight composition and end-user development. Submitted to IEEE 7th International Conference on Web services (ICWS 2009), 2009.

[13] L. Kutvonen. Challenges for ODP-based infrastructure for managing dynamic B2B networks. In A. Vallecillo, P. Linington, and B. Wood, editors, *Workshop on ODP for Enterprise Computing (WODPEC 2004)*, pages 57–64, 2004.

[14] M. Larsen and R. Klischewski. Process ownership challenges in it-enabled transformation of interorganizational business processes. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*, pages 1–11. IEEE, 2004.

[15] G. C. Minsk, L. S. Poh, H. Wei, and T. P. Siew. Web 2.0 concepts and technologies for dynamic b2b integration. In *IEEE Conference on Emerging Technologies and Factory Automation, 2007*, pages 315–321, Patras, Greece, 2007.

[16] D. C. Plummer, T. J. Bittman, T. Austin, D. W. Cearley, and D. M. Smith. Cloud computing: Defining and describing an emerging phenomenon. Technical report, Gartner, 2008.

[17] C. Schroth. Global industrialisation of information-intensive services: a reference architecture for electronic business media. *Int. J. Product Lifecycle Management (IJPLM)*, 3:191–210, 2008.

[18] C. Schroth. "Richness" und "Reach" organisationsübergreifender, Informations-intensiver Dienste. In M. Bichler, T. Hess, H. Krcmar, U. Lechner, F. Matthes, A. Picot, B. Speitkamp, and P. Wolf, editors, *Tagungsband der Multikonferenz Wirtschaftsinformatik 2008 (MKWI 2008)*, pages 1369– 1380, Munich, Germany, 2008. GITO-Verlag.

[19] C. Schroth, B. F. Schmid, and W. Müller. Designing modular architectures for cross-organizational electronic interaction. In *Proceedings of the United Information Systems Conference (UNISCON 2009)*, Sydney, Australia, 2009. Springer.

[20] C. Schroth, R. G. Siebeck, and B. F. Schmid. Characteristics of cross-organizational electronic interaction and supporting IT service infrastructures. Technical report, Alexandria, St. Gallen, Switzerland, 2009.

[21] R. G. Siebeck, T. Janner, C. Schroth, V. Hoyer, W. Wörndl, and F. Urmetzer. Using mashup integration services in b2b scenarios. Submitted to the 11th IEEE Conference on Commerce and Enterprise Computing, 2009.

[22] N. Ward-Dutton. SOA as a foundation for open, flexible business collaboration. Technical report, Macehiter Ward-Dutton, 2006.

[23] V. Wulf and M. Jarke. The economics of end-user development. *Communications of the ACM*, 47(9):41–42, 2004.