First Monday, Volume 16, Number 8 - 1 August 2011

f i ® s t   m ¤ ñ d @ ¥
PEER-REVIEWED JOURNAL ON THE INTERNET

Reading Tools

Trust issues in W...

Lee, Kaufmann, Buss

Abstract
Review policy
About the author
How to cite item
Indexing metadata
Print version
Notify colleague*
Email the author*

RELATED ITEMS
Author's work
Government policy
Book
Book reviews
Dissertations
Online forums
Quotations
Resources
Media reports
Web search

SEARCH JOURNAL

All

CLOSE

* Requires registration

Trust issues in Web service mash-ups
by Kevin Lee, Nicolas Kaufmann, and Georg Buss

Abstract

With the emergence of Web service mash–ups (Web applications that integrate different data sources), online data integration and aggregation is increasingly becoming the online norm for both commercial and non–commercial users. With such widespread adoption of data integration from discrete sources, the question emerges as to whether the resultant mash–up can be considered as trustworthy. This paper explores the concepts behind Web service mash–ups to determine the factors influencing their trustworthiness. The focus is on examining data quality and data assurance issues for both data providers and mash–up consumers.

Contents

Introduction
Classification of Web service mash–ups
Trust for data quality
Trust for data assurance
Conclusions

Introduction

Web services in general can be defined as a software system that is explicitly designed to support interoperable and platform independent machine–to–machine interaction over a network, utilizing a predefined interface (W3C, 2004). A Web service mash–up (simply called "mash–up" here) follows this general paradigm of a Web service and can be defined as a Web–based application that combines data from Web services. A mash–up uses three main, logically and physically disjoint components: i) the mash–up logic; ii) the client's Web browser; and, (iii) at least two data sources (with at least one external data source). The resulting output is highly integrated data, which is specially optimized for human consumption (Merrill, 2006).

When considering data integration in mash–ups the location where the mash–up logic is executed is important, as this is the point the trust occurs; this differs for different mash–up methods. The most popular type of implementation is on the client side, where the Web browser does both the data integration and presentation. The most used technique for client side data integration is asynchronous JavaScript and XML (AJAX). Alternatively, implementations using Java Servlets or PHP integrate data sources at the server side (Merrill, 2006).

An important characteristic of mash–ups is that they can be created with modern, easy–to–use frameworks, opening their possibilities to a wide audience. With their increasingly ubiquitous usage, an important question is whether mash–ups can be considered as a trusted entity. Trust in this context refers to the reliability, accuracy and confidence of the presentation and usage of the data. Trust has to be looked at from the perspective of both the mash–up consumer and the mash–up provider.

A consumer's trust of mash–ups as an information source strongly depends on data quality. It is difficult to determine what a consumer considers high quality because of the unpredictable usage by consumers. Mash–ups are also highly dynamic so the result can change dramatically with changes in the data sources. Consumer reliance on the mash–up is also unpredictable prior to public exposure. This is further complicated with the usage of multiple often uncredited

data sources. For these reasons it is difficult for consumers to trust the data quality of a mash–up. The original data provider's priority is to be assured of an unabusive usage of their data in order to be willing to make their data accessible. As consumers and data providers are identical in the case of common Web 2.0 community applications, they have to be convinced of both trust perspectives concurrently (see Figure 1 that compares mash–ups to single source data repositories for data assurance and data quality).
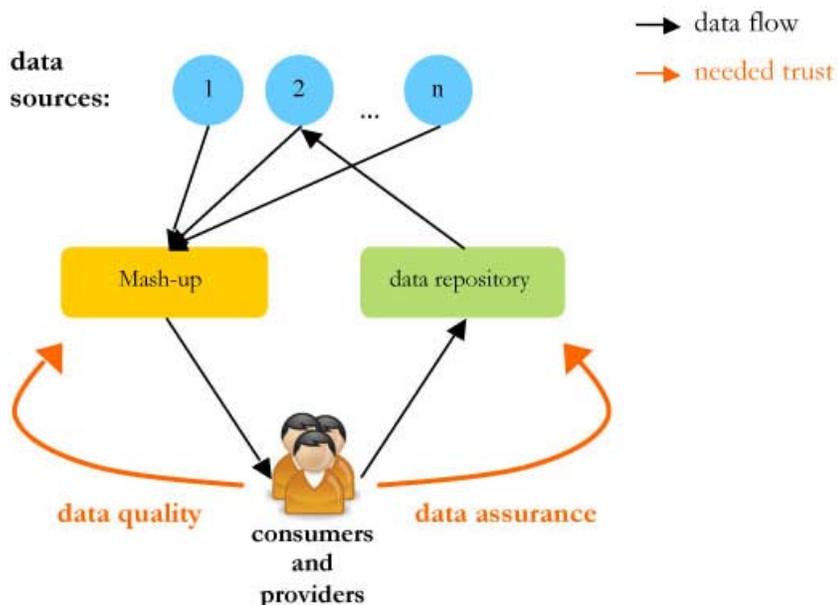

Figure 1: Two faces of trust in the Web service mash–up data exchange process.

Mash–ups have three characteristics that make them suitable for many applications: They are flexible to use, comparatively easy to create and do not need specialized infrastructure. This flexibility is increasing as more mostly free data sources are becoming available making it increasingly easy to find adequate data input. The ease of creation and maintenance is provided by specialized frameworks like the Google App Engine (http://code.google.com/appengine), the YUI Library (http://developer.yahoo.com/yui/) from Yahoo or the JavaScript toolkit Dojo (http://www.dojotoolkit.org), which provide the functionality to create mash–up applications even without deep technical knowledge. Due to the fact that mash–ups can be built with logic being executed on the client, the only requirement a mash–up needs for execution are a compatible Web browser and an Internet connection. That makes mash–ups very easy to deploy and to maintain.

Classification of Web service mash–ups

Indexes and directories of mash–ups (ProgrammableWeb, 2009) provide a window on the wide variety of mash–ups. Based on these indexes, the following classifies mash–ups into six categories. Due to the nature of mash–ups, the categories in this classification are not intended to be completely discrete; many mash–ups exist between and across multiple categories.

- **Visualization mash–ups** present data in a human consumable way. Commonly this is through dynamical charts or maps. The most popular contemporary use of this is for overlaying location data on a Google Maps interface such as for live epidemic data. Other common examples are Flickrvision (http://www.flickrvision.com) and WikiMapia (http://www.wikimapia.com).

- **Multimedia mash–ups** are portals that take advantage of sites that host media files such as Flickr or YouTube and metadata sites such as IMDB (http://www.imdb.com). They typically combine the media with the metadata adding value for the consumer. Examples are Neave Television (http://www.neave.com/television) or Vdiddy (http://www.vdiddy.com).

- **Marketplace mash–ups** use distinctive market information like prices or stock status to support or initiate commercial activities. Examples are price comparison tools like PriceGrabber (http://www.pricegrabber.com) and market tracking tool Google Finance (http://www.google.co.uk/finance).

- **Information collections** is a comparatively old type of mash–up that gathers mostly text–based information on specific topics. They therefore preferentially use news feeds or

open encyclopedias like Wikipedia. Examples include Google News (http://news.google.com) and Yahoo News (http://news.yahoo.com).

- **Community & social mash–ups** use data provided by online communities and social sites to enhance commercial services. This usage of free data for commercial services, often called crowd sourcing, considered as a key benefit of Web 2.0 (O'Reilly, 2005). Examples are online travel sites that integrate hotel ratings from communities like Holidaycheck (http://www.holidaycheck.com) or Tripadvisor (http://www.tripadvisor.com).

- **Messaging mash–ups** relate to or provide messaging functionality for nearly instant communication between two or more users. The most popular contemporary ones are those that relate to the instant messaging site Twitter (http://www.twitter.com). Examples include mash–ups that combine Twitter and other data such as SadStatements (http://www.sadstatements.com) and TwitterVision (http://www.twittervision.com).

The possible domains for mash–ups can be separated into non–commercial and commercial applications. Most mash–ups are non–commercial (examples in the classification above), but commercial mash–ups are increasingly emerging. On the non–commercial side, mash–ups usually have a purely informative character. They are often created to visualize and/or aggregate data and to solicit public user input (Merrill, 2006).

On the commercial side, mash–ups can be used externally exposing the results to the public, or internally only. Publicly exposed mash–ups might be used to acquire new data by building a community and thus encouraging participation, to gain market power by binding users or to offer new services/products in order to benefit from a "long tail" (Anderson, 2007). Internally, mash–ups can be used to aggregate information and support the decision–making process in order to increase transparency and efficiency. This special type of mash–up is also known as a situational application mash–up.

Following the definition of Adams and Gerke (2008), a situational application is a tactical, temporary application that is mainly built by the intended user groups themselves to suit their individual needs. According to the characteristics outlined by Chebrakov, *et al.* (2007), they are related to a current specific situation, mainly built by non–professional programmers, often use pre–existing software, have comparatively short living cycles and are mainly intended for informational purposes.

Trust issues in mash–ups can be broadly split into two categories: Data quality and data assurance. Data quality refers to issues that affect the final quality of the mash–up delivered to the consumer. Data assurance refers to the mechanisms that ensure with confidence that the data is accessed appropriately and safely. The following two sections discuss the issues of trust in these broad areas.

■ ——————————————————————

Trust for data quality

In the this section, the influence of data quality on trust in mash–ups will be analyzed. First the potential data sources for mash–ups are classified, then the impact of data quality issues are discussed, before analyzing the available control circuits for ensuring trust in data quality.

Classification of mash–up data sources

In order to determine issues related to data quality, data sources of mash–ups have to be identified and analyzed. The following categorize the most used data sources for mash–up creation:

- **APIs** (Application Programming Interfaces) are bidirectional interfaces that allow standardized access to available data. It is specified by the provider and commonly uses concepts like REST ("Representational State Transfer") or protocols like XML–RPC and SOAP ("Simple Object Access Protocol"). Logic in the mash–upp application uses the API of a site. An example of a publicly exposed API is the GoogleMaps API (http://code.google.com/apis/maps). After the request is processed on the provider side, data responses are usually transmitted utilizing data formats like XML (Extensible Markup Language), JSON (JavaScript Object Notation) or SOAP–Response.

- **Feeds** in contrast to APIs provide a unidirectional (non–interactive) data flow. Their main field of application is provider–controlled content syndication (Merrill, 2006). Used data formats are RSS ("Really Simple Syndication") and ATOM ("Atom Syndication Format").

- **Screen scraping** is a way of collecting data mainly intended for human consumption. The visual output of a Web site is automatically processed in order to retrieve the data it contains. This is typically done by a type of software called "crawler" that pretends to be a human user. Though this method may be inevitable due to the lack of alternative data sources it is often used without the permission of the data provider. The scraping software has to be specifically designed for each data source and is very likely to fail due to a missing data exchange contract. A simple change in the layout of a Web page may cause complete failure (Merrill, 2006).

- **Instant APIs** can be seen as an evolution of screen scraping, though the underlying techniques for data retrieval are exactly the same. Using instant APIs, mash–up creators "outsource" the screen scraping to the API or toolkit provider, who uses his own

techniques for data extraction and makes the resulting data accessible using his predefined interface. Examples are Dapper (http://www.dapper.net/open) and OpenKapow (http://www.openkapow.com).

- **Internal data** is used mainly for creating commercial mash–ups and combines internal and external data. A possible combination is the usage of mash–ups as user–friendly and customizable interfaces on top of a SOA (Watt, 2007). A commercially available product using this scheme is the IBM Mash–up Center (http://www.ibm.com/software/info/mash-up-center), which allows creation and administration of situational applications.

Impact of data quality issues

The following section examines the impact of data quality on trust in mash–ups in both non–commercial and commercial usage.

Mash–ups with a non–commercial background are mainly intended for combination and visualization of existing publically available information; implying that the value for the users is mainly in the way the information is aggregated and displayed. Hence, trust in a mash–up is especially driven by the level of quality the presented information holds.

A good measure of the level of trust users have in a non–commercial mash–up is its popularity — especially in form of its user frequency. Due to the fact that other dependencies between the mash–up and its users — like binding contracts or artificially created market barriers (*e.g.*, switching costs) — are not relevant in non–commercial mash–ups, popularity might be an adequate measure. Users of non–commercial mash–ups could be aware of the fact that the lack of financial return generally limits the mash–up provider's ability of using revised or generated data. Users therefore possibly anticipate a certain degree of false information, which would not harm their trust in a particular mash–up.

The theory of assimilation and contrast from Sherif and Hovland (1961) can be adapted to the mash–up scenario. This theory is mainly used in marketing science for explaining the impact of cognitive dissonance on customer satisfaction. It states that a small divergence between the individual's expectation and perceived output is generally accepted without leading to negative (or positive) cognitions or reactions. However, if a certain threshold of divergence is exceeded, the experienced contrast is suddenly increased. Transferred to the data quality of mash–ups, this implies that users could be willing to accept incorrect data to a certain extend. How high the threshold is and what implication it actually evokes is specific to the mash–up, the source data and the consumer.

Commercial usage of mash–ups is similar to that of non–commercial usage, except that the tolerance of users towards poor data quality or presentation is lower. Due to the presence of contracts, a lack of data quality may also evoke financial risks like contractual penalties or even liability claims. Because the offered data can be seen as the main asset an external commercial mash–up has, trust in its quality is extremely vital for success.

For internal usage mash–ups are preferentially used as situational applications, which are intended to improve decision–making processes and enhance effectiveness (Adams and Gerke, 2008). Integration of external data sources is explicitly supported and even considered as a main feature of the popular IBM's InfoSphere MashupHub. As soon as external data is integrated in a company's information system, information quality strongly determines the effectiveness of the whole system. Relying on erroneous data could even lead to negative productivity or allow external attacks by delivering aimed misinformation. In the case of commercial usage, external data sources have to be chosen carefully and its accuracy verified before usage.

Even internally generated data can be of questionable quality. The great benefit of situational applications — instant and individualized access to data — depends on the available data sources. Usage of any available internal data for user–generated mash–ups evokes higher requirements on the quality of internal data. Every employee who is responsible for keeping any distributed data updated has to be aware that it might be accessed and processed at any time and for a range of (possibly unforeseen) uses.

Recent practical experiences of IBM strongly correspond to these data quality concerns. During the implementation of situational applications in their own intellectual property division, the condition of internally collected data was found to be poor. To avoid misinterpretation and misrepresentation, data had to be corrected manually before system deployment. After the situational application was launched, employees from other divisions requested details about data sources and aggregation methods because they were unable to judge data quality and therefore had concerns using it. Thus, adequate descriptions were added to the application (Pandaya, 2008).

Control circuits for ensuring data quality

Given a certain amount of available information, there will always be a probability of using data sources of varying quality. This especially applies to information that is provided free of charge, because mash–up providers will always try to keep costs low. According to Akerlof's "lemon" market model (ProgrammableWeb, 2009), a market with varying quality and asymmetry of information tends to fail. Information asymmetry is very likely to exist for mash–up data sources as well, because the value of information can only be evaluated after consumption (Varian, 1998). Thus, the information provider is better able to judge its value. Assuming a positive correlation between information quality and its production costs combined with non–existing quality signs, the information provider is incentivized to maximize their revenue by accepting low data quality; high–quality information could be pushed out of the

market (Akerlof, 1970). The importance of data quality for successful mash–ups indicates that control circuits have to be identified to allow mash–up creators and users to judge this quality. The following gives an overview of possible approaches for ensuring data quality in mash–ups.

Market control can be divided into two categories, those who try to measure the demand for a specific mash–up and those who judge the quality of a mash–up. Measurement of demand is a classical approach to measure a quality of a good. Its underlying assumption is that the amount of a good's consumption is positively correlated to its quality. This effect can be explained by word of mouth and similar spreading effects. This approach has to be adapted for information goods, because of their potentially unlimited reusability and small reproduction costs (Varian, 1998; 1997). A measure of mash–up consumption is the frequency of information memorization because this might imply positive evaluation of its quality. An example of this approach is the online bookmarking storage platform delicious (http://www.delicious.com) that ranks Web resources by the frequency of being bookmarked by users of the service. Another possibility is to judge the quality of information goods by allowing the consumers of information goods to review them after consumption. This can be done by publishing individual or aggregated judgments of good consumption. A big advantage of the market control approach is its simplicity — only little infrastructure is needed to implement this type of control circuit. On the other hand, the achieved output quality of information is limited, because it is always left unchanged and fine–grained rating or correction is impossible.

Community control in comparison to market control uses the community itself to influence and contribute to the information creation process, which can therefore be seen as the actual information provider. A popular example for this approach is the principle of Wikipedia (http://www.wikipedia.org); each article can be discussed, corrected and complemented by each member of the community. The advantage of community control can be seen in the potentially high–quality level of information output. This relates to an effect known as the "wisdom of the crowds", that was introduced by Surowiecki (2004) and describes the common phenomena of a group being more "intelligent" than one of its members. In the case of Wikipedia, a positive correlation between the number of participating authors and resulting article quality has already been scientifically proven (Kittur and Kraut, 2008). Disadvantages can be seen in the amount of infrastructure needed to enable collaborative work and in a possible dependence of the quality enhancement on the community size. In addition, individual contribution to information that is published on behalf of a whole community provides the possibility of aimed manipulation. An example is the recent discussion about the reliability of Wikipedia articles based on the existing possibility to contribute anonymously. A possible countermeasure might be a limitation of contribution rights.

Reassurance control as a method of avoiding low–quality data can be deducted from a rule of journalism, that every story has to be affirmed by a second, unrelated source before publication. In parallel, particular information might only be considered as validated, if it can be obtained through a certain number of ideally unrelated sources. A strong advantage of this method is its suitability for automatic procession. Integrated in a corporate information system, it could ensure information before using it in crucial situations. Major disadvantages are the potential unavailability of alternative data sources for reassurance and the complex evaluation process if information is not standardized. Furthermore, potentially significant information is very likely to be classified as unsecured due to the lack of comparisons.

Expert control is the most radical approach of data quality assurance to evaluate every piece of information individually before usage. This normally implies manual processing. An example for this approach is the holiday advisory community holidaycheck (http://www.holidaycheck.ch). Each posted review is read and evaluated by a community manager before being approved for online publication. The main advantage of expert control is data reliability. Given a certain degree of knowledge, experts can reliably guarantee a high data quality level. However, this usually leads to high information costs and a comparatively slow speed of information collection.

Trust for data assurance

In the following section, the influence of data assurance on trust in mash–ups is analyzed. Data providers are identified, drivers of the information flow are discussed and control circuits for data assurance analyzed.

Data providers

In order to analyze the importance of data assurance, the actual origins of data used for mash–ups — the data providers — have to be identified. As with data quality, data providers can also be segmented in those with a commercial and non–commercial background. Motivation for commercial data publishing can be distinguished between financially and strategically advantageous. The first option actually means selling data for a monetary reward, *e.g.*, as a new product or service. In this case trust in data assurance can be assumed to be irrelevant for the seller — otherwise he would not have intentionally decided to sell the data beforehand. A strategic motivation for selling data might be the creation of market power by causing lock–in effects or market entry barriers for contenders. In this case, abusive use of the provided data might harm the objectives pursued by the data publication. However, as the

company providing the data normally is in the position of ensuring control over the data source, countermeasures can be taken. This means that trust should usually not be crucial for data publishing either. On the non–commercial side, information is mostly gained through communities and services of Web 2.0. Many Web–based business models are based on user–generated content, which can be seen as the "price" a user pays for service usage. Their reward is the utility service usage creates. In this case, the user is giving his data away to a third party and therefore looses control. If so, trust in this third party is very likely to be a vital factor for their degree of participation.

Drivers of information flow

As community gathered information is a major data source for mash–ups, the drivers of this information flow become a matter of particular concern (Merrill, 2006). Users of Web 2.0 services actively generate and seed content. This phenomenon can be explained by a combination of a technology–push and market–pull effect. The technology–push means that availability of new technologies always encourages people to try it. Market–pull refers to the existing self–motivation people have to participate caused by sociological factors like the need of social interaction (Schroder, 2002; Karla, 2007).

Taking into account that the technology–push might wane over time, market–pull might evolve to the dominant driver of online community data flow. As the existing self–motivation to participate depends largely on soft social factors, a lack of trust in data assurance could be fatal. Due to the high rate of social interaction in modern online communities, rapid dispersion of negative experiences — *e.g.*, concerning data abuse — is inevitable. If basic needs like data privacy and data security are denied, the valuable information flow could run dry. A possible solution might be adequate control circuits that give the actual data provider's control of access to their information and identity.

Control circuits for data assurance

The following section discusses the possible control circuits for ensuring data assurance. The focus is on user generated content and personal user data as these are the most common data sources for mash–ups. Data extraction by techniques like screen scraping is hardly preventable, so the focus is laid on active data publication through APIs.

Classical APIs for personal user data, Web communities and portals all request certain information like name, birth date and e–mail address. Once this data is submitted, it is stored and provider's access can usually not be limited. In some cases, even deletion may not be achieved, even with a direct personal request. User submitted content — such as images, videos or text — is freely accessible by using the provided API in the majority of cases. If the user has a choice, it is usually limited to allow or disallow sharing of particular content for all kinds of access.

In the case of users intending to willingly allow a third party to access stored content in an external Web application, the user is either forced to reveal his user credentials to the third party or to use token–based access granting, *e.g.* by using cookies. For using the latter technique, the user is redirected to a Web application that stores data for log in and access consent. Then, a third party application secures a token with which it accesses stored content.

OpenID (http://www.openid.net) is a protocol that helps prevent identity theft and assists the user in controlling personal information by providing a single sign on system ("SSO"). Users of Web applications supporting OpenID can exclusively use a single URL as an identifier for login and registration purposes. If they are already logged in at their OpenID service provider, identity verification and even exchange of personal data is automatically handled. As by decentralized methods, a token is used to verify a user's identity. Until the user trusts a certain Web application, each identity verification and data exchange process has to be granted manually. Provided that the Web application supports the newer "OpenIP Attribute Exchange Protocol", it can access all information a particular user has previously stored at his OpenID provider and local user databases may even become redundant. As a user can freely decide which OpenID provider to use and every URL a user has control of can be used as an identifier ("delegation"), dependency on a certain provider can be avoided. That makes OpenID a seminal solution for enhancing data assurance, growing in acceptance with sites such as Facebook or Flickr.

A major disadvantage of the OpenID concept is the user's dependence on a single entity. If an OpenID service fails, its users are unable to authenticate themselves. Though that risk could be avoided by using delegation and setting up a redundant account at another OpenID service provider, this is laborious and will mostly not be done as a precautionary measure. Additionally, if the user is not logged in at his OpenID Service provider already, redirection to its login page is still done by a third party application. However, due to the variety of OpenID service providers, keeping fake login pages ready for all of them should not be worth it.

OAuth (http://www.oauth.net) in contrast to OpenID, focuses on control of data exchange between Web applications. It is designed to cover user–initiated as well as not user–initiated data requests. The original application of OAuth is the so–called "three–legged" scenario, where a service provider's Web application holding data, a third–party Web application trying to access this data and the end user are involved. In order to get access to the stored data, the third–party application needs a request and access token. Both have to be accredited by the end user. After the third–party application required a request token, the user is asked by the service provider whether he want to allow the third–party application to send data requests. The distinctiveness of the OAuth protocol is, that this token is still completely independent of granting actual data access. Therefore, the user has to authorize an access token, which can be limited in time and extent of data access (OAuth, 2009). This gives the

user extensive control over third–party data usage.

The second possible scenario is called "two–legged". It covers machine–to–machine communication without direct involvement of the end user. In order to avoid extensive data buffering on the side of the third–party application, this possibility is necessary as well. The 1.0 draft of the OAuth Consumer Request (OAuth, 2009) therefore gives the user the opportunity to grant a third–party Web application permanent access. Once the access token is certificated, the third–party application is able to reuse it until the user cancels it. This allows data access without further end user interaction.

Conclusions

Establishment

The main benefit of mash–ups can be seen in the integration of multiple data sources. By using modern and interactive ways of data aggregation and displaying, they provide the opportunity to ease human comprehension of data strongly. This classifies mash–ups as a useful tool for many applications. Trust in mash–ups is a vital factor for their success. It is required within two dimensions: Data quality and data assurance (see Figure 2). Their individual importance varies concerning the addressed user group.

Data quality tends to be more important to commercial mash–up users. Due to the fact that information provided by mash–ups is used to improve transparency and effectiveness of business–related decision–making processes, data quality is crucial. If a certain level of data quality cannot be guaranteed, mash–ups will not be able to create a noticeable value surplus or even cause harm to the user. Non–commercial users are very likely to assume and therefore accept a lack of data quality to some extent, which usually does not harm their user experience.

Data assurance — especially data privacy and security — is tending to be a larger issue for non–commercial mash–up users. Because mash–ups are usually utilizing data from online community services, the willingness of users to provide a constant information flow is vital to the existence of mash–ups. Since the user's reward for information supply is mostly limited to non–monetary factors like social interaction, harming a basic need like privacy is very likely to cause rejection or even active resistance. This would definitely harm the attitude of non–commercial users towards mash–ups, because they can be perceived as a major privacy risk. For commercial users keeping control of data sources spreading internal information is supposed to be much easier. Thus, data assurance is not likely to be a factor of major concern.
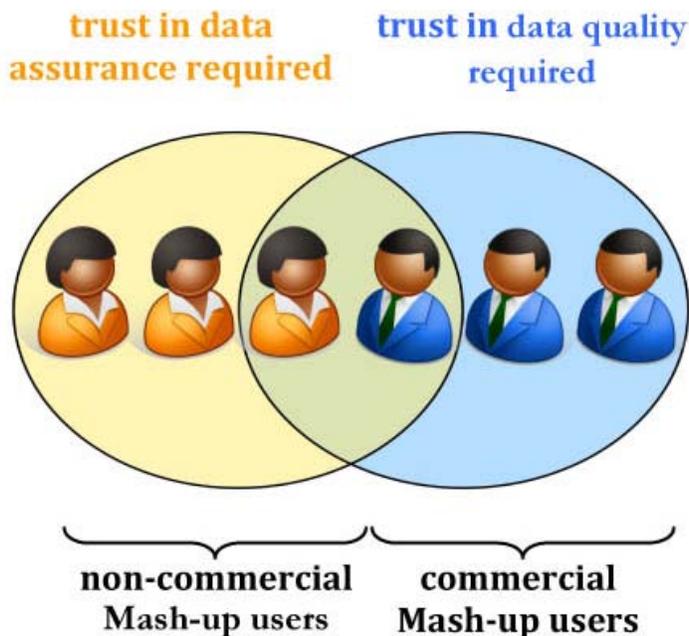

Figure 2: Dominating faces of trust in mash–ups depending on intended usage scenario.

Due to their strong advantages over traditional software, usage of mash–ups as situational applications in commercial domains even offers major potential in the corporate and b2b market. This provides an important chance for mash–ups to outgrow their community roots

and to establish themselves in mature IT systems. However, internal use of mash–ups in a business environment should not be decided overnight. Without individual consideration of the impact integration of external data sources has on the company and its IT system, improvident use of mash–ups might also harm a company's success. Important topics to address are the comparatively high requirements on internal and external data quality as well as the extent of system user rights. "Should users be allowed to integrate any kind of new external data sources?", "Are users allowed to access any kind of internal data?" or "How can the IT administration ensure overall system quality?" are typical questions that need to be addressed beforehand. Above all, deciders should be aware that the higher importance of internal data quality requires additional efforts in most cases. Personnel responsible for data administration need to be enabled to meet these requirements in order to allow full utilization of mash–ups' potentials. FM

About the authors

Kevin Lee is a Lecturer and a member of the WSN Cloud Group in the School of Information Technology at Murdoch University, Perth, Western Australia.
E–mail: kevin [at] kevin-lee [dot] co [dot] uk

Nicolas Kaufmann is an undergraduate student at the University of Mannheim, Germany.

Georg Buss is a Ph.D. candidate at the University of Mannheim, Germany.

References

Holt Adams and John Gerken, 2008. "Choosing between mash–ups and traditional Web applications," *IBM developerWorks* (15 July), at http://www.ibm.com/developerworks/lotus/library/mashups-web/, accessed 12 July 2011.

George A. Akerlof, 1970. "The market for 'lemons': Quality uncertainty and the market mechanism," *Quarterly Journal of Economics*, volume 84, number 3, pp. 488–500.

Chris Anderson, 2006. *The long tail: Why the future of business is selling less of more*. New York: Hyperion.

Luba Chebrakov, Andy J.F. Bravery and Aroop Pandya, 2007. "SOA meets situational applications, Part 1: Changing computing in the enterprise," *IBM developerWorks* (23 August), at http://www.ibm.com/developerworks/webservices/library/ws-soa-situational1/, accessed 12 July 2011.

Jürgen Karla, 2007. "Implementierung von regelkreisen in geschäftsmodellen für Web 2.0–publikumsdienste," *HMD Praxis der Wirtschaftsinformatik* volume 255, pp. 17–26.

Aniket Kittur and Robert E. Kraut, 2008. "Harnessing the wisdom of crowds in Wikipedia: Quality through coordination," *CSCW '08: Proceedings of the ACM 2008 Conference on Computer Supported Cooperative Work*, pp. 37–46.

Duane Merrill, 2006. "Mash–ups: The new breed of Web app," *IBM developerWorks* (8 August), at http://www.ibm.com/developerworks/xml/library/x-mashups/index.html, accessed 12 July 2011.

OAuth, 2009a. "OAuth Consumer Request 1.0," Draft 1, at http://oauth.googlecode.com/svn/spec/ext/consumer_request/1.0/drafts/1/spec.html, accessed 17 November 2009.

OAuth, 2009b. "OAuth Core 1.0," Revision A, at http://oauth.net/core/1.0a, accessed 17 November 2009.

Tim O'Reilly, 2005. "What is Web 2.0? Design patterns and business models for the next generation of software" (30 September), at http://oreilly.com/web2/archive/what-is-web-20.html, accessed 1 November 2009.

Aroop D. Pandaya, Luba Cherbakov and Andy J. F. Bravery, 2008. "SOA meets situational applications, Part 3: Examples and lessons learned," *IBM developerWorks* (3 July), at http://www.ibm.com/developerworks/webservices/library/ws-soa-situational3/index.html, accessed 12 July 2011.

ProgrammableWeb, 2009. "Mashup Dashboard," at http://www.programmableweb.com/mashups, accessed 9 November 2009.

Hans H. Schroder, 2002. "Paradigms for the management of innovation: A critical analysis," In: Heinz Strebel (editor). *Innovation und umwelt*. Graz: dbv–Verlag, pp. 23–76.

Muzafer Sherif and Carl Iver Hovland, 1961. *Social judgment: Assimilation and contrast effects in communication and attitude change*. New Haven, Conn.: Yale University Press.

James Surowiecki, 2004. *The wisdom of crowds*. New York: Doubleday.

Hal R. Varian, 1998. "Markets for information goods," University of California, Berkeley, at

http://people.ischool.berkeley.edu/~hal/Papers/japan/index.html, accessed 12 July 2011.

Hal R. Varian, 1997. "Versioning information goods," University of California, Berkeley, at http://people.ischool.berkeley.edu/~hal/people/hal/papers.html, accessed 12 July 2011.

Stephen Watt, 2007. "Mash–ups — The evolution of the SOA, Part 2: Situational applications and the mash–up ecosystem," *IBM developerWorks* (8 November), at http://www.ibm.com/developerworks/webservices/library/ws-soa-mashups2/, accessed 12 July 2011.

World Wide Web Consortium (W3C), 2004. "Web services architecture," W3C Web Services Architecture Working Group (11 February), at http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/, accessed 26 October 2009.

Editorial history