A Portal-Based Web Service Development Using a Mashup Approach

Shah J. Miah
Shah.Miah@vu.edu.au
School of Management and Information Systems
Faculty of Business and Law, Victoria University, Footscray Park Campus, Victoria, Australia

Justin C. W. Debuse
jdebuse@usc.edu.au
Faculty of Arts and Business, University of the Sunshine Coast, Queensland, Australia

John Gammack
j.gammack@murdoch.edu.au

Diarmuid Pigott
d.pigott@murdoch.edu.au
School of Information Technology, Murdoch University, Perth, Western Australia, Australia

Abstract

A mashup is an integrated Application Programming Interface (API) that combines data from different data sources or third party web services. In recent years, mashups have been implemented to enhance information management in many web applications. Mashups provide a combined API that is both technologically valid and compatible with other web applications, allowing them to provide customized solutions to business end users through rapid data and service combination. Traditionally portal architectures aggregate content but have limitations which mashup solutions can overcome. We propose an alternative service development approach to the distributed architecture of web portals, using a mashup technique. We show how a generic design based around web service architectures can better meet end user needs, and how customized solutions can readily be built for business problems. Prototype mashup applications, developed through action research, were evaluated favorably for usefulness, speed and ease of use by both technical and business representative users.

Keywords: mashups, web services, end users, web portals

1. INTRODUCTION AND MOTIVATION

The innovation in web service design to meet organizational and end user needs is still emergent. Many studies have identified the rapidly growing requirements of web technology advancement, for example identifying important attributes of web services (Xu, Sharma, &

Hackney, 2005). Accordingly, emerging technologies for web services have recently drawn significant interest in the web community. Digital library services (Pearce, 2006) provide examples where emerging technologies have been used for enhancing end users' active participation. Web applications driven by end users however do not necessarily use traditional requirement-and-build approaches, especially for web based service development (Macías & Castells, 2007). Therefore, developers have started using other technologies for web based user interface design, service composition design, and social or community based features. These are designed to create increasingly interactive applications, where shape evolves through active usage patterns rather than being pre-specified.

Web based applications must meet end users' diverse needs including customizable provisions in various applications, such as portals (Chena & Macrediea, 2010). In particular for web services, where dynamic composition of autonomous processes is involved, traditional architectures are too prescriptive to provide the requisite flexibility. Indeed, such issues in end user provision have been a problem for decades (Backus, 1977; Kell, 2009).

Portals aggregate information around a user community's interests, and increasingly content is sourced, linked and maintained from that very community, linked to the popular and emerging technologies and devices they use in business. Second generation portals (X. D. Wang, Yang, & Allan, 2006) implement a Service-Oriented Architecture consisting of three layers, with web services in the middle layer. The so-called second generation portal is defined as a grid-based portal that provides a unique interface for end-users to access distributed resources. This approach left behind issues of content management and additional programming requirements. Static technologies (HTML based) for web services however suffer from potential weaknesses such as lack of dynamicity, scalability and security, as well as inflexibility. In conventional portals, user required functionality is often restricted by the limitations of web technologies in information retrieval and processing (Lausen, Ding, Stollberg, Fensel, Hernandez, & Han, 2005). Full semantic interoperability as envisaged in the semantic web concept remains a longer term potential. In contrast, recently emerging Web 2.0 technologies such as mashups, wikis, and blogs offer an innovative capability to end users, business entities, and communities. This paper describes the mashup conceptual framework within the relevant background, and its evaluation in a business context, for developing an end-user-enabled web portal.

The rest of the paper is organized as follows: Section 2 presents relevant web service development work. In section 3 we describe the methodology used to develop the mashup solution model. Section 4 presents the problem space identified for this research and its rationalization. Section 5 describes the proposed architecture for the identified business problem. Section 6 presents the evaluation procedure and outcomes, followed by conclusions and discussion in section 7.

2. BACKGROUND

Mashup Applications

Web services provide applications offering 'platform independent' service components available on the Internet. Today many application services are 'service composition' in structure, aggregated from a combination of diverse third-party web services (Srivastava & Koehler, 2006). This type of composition no longer uses fixed HTML, rather seamless, dynamic applications assembled using mashup concepts into an integrated experience. HTML based solutions suffer from issues including dynamicity, scalability and customizability for end users (Datta, Dutta, Thomas, VanderMeer, & Ramamritham, 2002). Thus, it is critical to business acceptance that end users participate in mashing up existing services to leverage and integrate effective new service applications (Miah, Gammack, & Greenfield, 2009).

Mashups are web applications which can enhance value creation by enabling virtual organizations to move from service to user level (Dillon, Wu, & Chang, 2007); they seamlessly merge existing service content or applications from multiple web sources, using web services, XML (Yee, 2006) and relatively simple web Application Programming Interfaces (APIs) (Miah & Gammack, 2008). The main components of the mashups are the data and an API for providing data access to the user. Publication of APIs for mashup data is increasing; for example the Programmable Web group offers 980 data sources. With a trend to cloud computing, mashups are seen as a way to get the most use

out of the non-traditional data sources provided by data vendors. The basis for mashup systems may be graphical or textual. These demonstrate the need to establish a paradigm for steady and consistent growth in mashup utilization in web programming. In this paper we focus on the technological background of mashup through outlining requirements of a combined web API for a specific case application, and propose a generic architecture of a prototype for a web portal.
Mashups can solve a wide range of web service issues; examples include mapping services, rare gaming console availability and location recording, and weather information (Maximilien, Ranabahu, & Gomadam, 2008).

Mashup technologies are still emerging, and standards are currently volatile. Both Google's mashup editor and Microsoft's Popfly were discontinued in August 2009, despite building substantial user communities. JackBe focuses on enterprise mashups and suggests emerging standards in this area with its attention to security and governance aspects. In business user contexts mashups can be classified by their respective emphasis on information, process or web-page and user interface customization (Grammel & Storey, 2008). Web application mashups generally offer more than simple merging of services and content. These sites typically add value through benefiting users in ways that are different and better than the individual services they leverage (Ort, Brydon, & Basler, 2007). Enterprise mashup (Clarkin & Holmes, 2012) and user generated applications (Slatnick, Parkins, & Dheap, 2008) can be seen as examples of adding value through benefiting users.

Related Work

Under the Service Oriented Architecture (SOA), mashups with various purposes, tools and technological capabilities have been used to build composite applications by combining disparate service components that provide a new service function to end users. For instance, semantic based mashup (Ngu, Carlson, Sheng, & Paik, 2010) has been introduced for service composition of non-web components such as portlets, web applications, native widgets and Java Beans, in a web service with minimal programming effort. DAMIA (Simmen, Altinel, Padmanabhan, & Singh, 2008) proposed a method for mashing up data from Excel, Notes, Web services and XML documents. Yahoo! Pipes ("Yahoo Pipes," 2009) offers a method of combining data from various sources for users to view. We therefore focus on mashup technologies and how these can be used specifically in replacing a distributed structure for web portal systems.

Mashups represent both a concept and a technology for integrating web applications or services coherently (Auinger, Martin, Nedbal, & Holzinger, 2008). While source content is normally implicit on the web, mashup technology helps process relevant content to be integrated; existing web service architectures have the potential for problems such as scalability, performance, flexibility, and ability to implement (Dillon et al., 2007). For example, a web service is expressed as relevant with a Web Services Description Language (WSDL) that specifies only the syntax of messages that enter or leave a program (Pearce, 2006). The order in which messages have to be exchanged between services must be described separately in one specification. However, the composition of this type of service flow is still manually obtained, though there are specification languages such as WSCI and BPEL4WS available. Semantic web application development researchers have addressed this problem by providing grid based solutions. For instance, GridSpace incorporates soft aspects of the grid, service-orientation, transient service space, and web architecture (Dillon et al., 2007); its architecture provides infrastructure to enable service discovery and mashup at various levels (Dillon et al., 2007; Miah & Gammack, 2008). Others, including Fox and Pierce (2007), have suggested applications, infrastructures and technologies for e-Science environments. In general (enterprise and distributed environments) these authors claim that Web 2.0 can provide narrow grids for building web services that may provide a robust managed environment. Previous research for grid-based solutions has focused on data rather than infrastructure (Miah & Gammack, 2008). Understanding these we have selected a realistic web service application for case analysis in this study.

3. RESEARCH METHOD AND PRACTICE

Action research was adopted to identify actual and situated principle and practice for web service development, learning lessons from the study by Vidgens (2002), in which the action research method is described for obtaining web development attributes, using the Multi-view

methodology (Avison & Wood-Harper, 1990) as a framework. Action research is also described as an approach for social science, in which the people being studied are given control over the purpose and procedures of the research (Babbie, 2004). As such, it was necessary to find the most relevant methodology for conducting the solution development. We thus employed a new mashup methodology introduced for e-learning ecosystems (so called "Jampots") (Dong, Zheng, Xu, Li, Yang, & Qiao, 2009) to enhance sustainability by achieving greater usability and individualization in the application design by the end users. The methodology focuses on six operations in a mashup application including publish, syndicate, assemble, orchestrate, consume and cooperate. Similar to this approach, we operationalized our prototype on the three most relevant operational matrices (Dong et al., 2009) such as publish, syndicate and assemble, since they were applicable to the business case in which end users are the mashup designers. These operational matrices are focused on user sided activities precisely (publish allows user to publish their content; syndicate allows user to combine and filter their content; and finally assemble allows user to assemble into their own mashup components) and support our key objective of encouraging users to build their own applications.

4. PROBLEM SPACE AND RATIONALIZATION

Traditional web portal systems are problematic as they are based on rigid platforms with tight rules for exchange of data and services. Processes are rapidly changing in businesses and the processes are getting more complex, requiring multiple users from different management levels to participate and collaborate to address the complexity (Miah et al., 2009). Businesses need to explore the potentialities for enhanced virtual operation, through the use of emerging technologies such as mashups. This use has been demonstrated in personal learning environments (Hsiao, Li, & Lin, 2008) and course management (Hardy, Pinto, & Wei, 2008). Both represent a better way for managing complex learning resources from multiple sources. In the business realm, customer demands are changing, as well as marketplaces, competition, and business partners. Therefore, advantage is conferred by the innovative use of new technologies rather than their mere presence. Such attributes imply business problems may be solved in a dynamic way; however, the current web development technologies do not yet fully support such types of service solutions (Miah et al., 2009).

Existing technologies support the need to provide more design middleware and the replacement of old systems. Rather than being about reusing the old applications in a new way, it is about remixing the system solutions. This can derive services from businesses' existing services, therefore increasing their ability to cope with change through empowering end-users and enabling innovative concepts. Mashups offer flexible composition of businesses' existing services within an improved user interface environment through improved APIs suitable for end users in their context of use (Miah & Gammack, 2008).

This study uses an existing portal based in Central Europe selling software, hardware and household electronic accessories. This site "http://yannipes.eu" is a virtual business offering comparison pricing on products based on third party companies pricing and product data. This leveraging to a full service platform is through their business relationships with the enlisted companies. The portal incorporates the third party company's data and acts as a wholesaler portal. Consumers can purchase product from the range of brands using the latest product information including price, product description and availability (Miah & Gammack, 2008).

Although this type of aggregator business model is commonly found across the web, it presents a mature model for widespread data sourcing. It has a number of issues for both the customer and company. For example, pricing and specification changes of items need to be reflected quickly in the marketplace, to maintain a comparative environment and ensure customers can make more informed decisions. However, the current system, which uses a distributed database architecture, is less dynamic when the enlisted company changes/updates information. New business models making use of the improved and flexible developments, such as Web 2.0, are needed.

In this context of distributed architecture, a mashup could address the currency of the information by providing a more dynamic and evolving platform that adapts to business requirements. As the enlisted companies update their own database, when they add/modify their

existing product range the portal can be dynamically updated. Through mashing up all the third parties' data into a new API the customer can see their desired products in an environment where comparison is easy and the information is current. Previous research has shown that semantic API can be a key method for enhancing web information retrieval and knowledge discovery (Dotsika, 2010). Motivated by this, we outline requirement of a new API using mashups for end users' web service creation.

5. PROPOSED ARCHITECTURE

Figure 1 illustrates conceptual mashup architecture for a solution to the product aggregator service problem. Each third party acts as a data island, providing its data service by acting as a content server; it will receive the data in the form of a mediated request (via XHTTP, REST or SOAP) and return what in other circumstances might be a legible data report. In our example we shall consider data about the product's prices, description, availability, and payment terms, but the principles applied can be generalized to any situation where such a system is required.

In this, as in any mashup style application, the key point is that there may be little if any commonality between the form or expression of data between the different providers. For example, two providers may include date information, with the first in dd/mm/yy form and the second in dd/mm/yyyy. The required representation for the mashup itself may be a fourth schema and style again; for example, the mashup may require dates in mm/dd/yyyy form. This is where the mashup architecture comes into its own.

The mashup designer must facilitate the user request into forms suitable for each of the content providers, and in turn the responses (or lack thereof) must be merged back into a single data stream. The development of the API will specify the form in turn in which the requests can be made, and the manner of presentation. APIs are only as good as their publication, and a final stage in the design deliberation will involve the creation of the specification. If the process will be treated as intellectual property then it may be prudent to conceal the inner workings of the data factoring and merging. The key drawback is that of any closed-source system: the more eyes on the design, the greater the chances of correction. With mashups this may be far more critical than with ordinary coding or data design scenarios – the mashup will only work if the content servers stay constant in their design, and the process of publication of the internals means that the content service owners as well as the API users can verify that the expectations of the system designers meet the initial requirements of the content servers, and continue to do so over time.

We have outlined a generic mashup architecture for business services based on this conceptual model for web end users. Figure 2 illustrates the architecture for a technology product aggregator service. It consists of four layers in line with the ideas presented by Dillon et al. (2007).

Firstly, the service providers-resource layer is where the content servers are located. This is followed by the technologies mashups transient layer, where the mashup technologies create a combined interaction for data from different resources. The virtual organizations-service mashup layer is next, where the mashup services are prioritized for mashing up data from content servers. Finally, at the user mashup layer data is presented to the web end users according to their requirements.

Figure 3 illustrates a prototype demonstrating how a mashup application can be built to replace a retail portal. This prototype contains several feeds that are used to represent different categories of the product. This has been published through the mashup of existing content. In this case, the existing content is the price lists and the banners of the special offers displayed from the source websites. A combined API just re-presents this information using different tags in the new system. In an application such as in iGoogle as pipeline, contents are combined from different web resources and enlisted into the product catalogues by the business owners. End users can use similar content processing for assembling any mashup operation. This type of personalized solution can be prototyped for any similar business. The business can add or modify the secondary content at any time for target customers. In addition, their target customers can achieve greater uptake through the ability to compare between products from different brands, therefore making buying decisions easier and better informed. Our aim here has been mainly to describe the

architecture and its motivation, but now that a prototype has been built, claims can be tested directly for user cases in future.

6. SYSTEM EVALUATION AND REFLECTIONS

Various experimental approaches, primarily qualitative, have been used to test mashup applications. Wang, Yang and Han (2009) conducted experiments to test the efficiency and effectiveness of a mashup development application called Mashroom. End users, for whom spreadsheets were a familiar programming model, were shown examples and experiments measured the time taken to build mashup applications through the Mashroom solution platform. Different types of mashups, including aggregation from web sources, data analysis and aggregation for comparison, were all effectively developed within a few minutes by non-expert end users.

Huang, Huang, Li and Yany (2008) described a procedure for evaluating a different mashup application called SituMash. They used the technology acceptance model (Davis, 1989) to measure how users accept the mashup solutions with regard to situation dependent variables, particularly the ease of use (the degree to which users found using the system an effort), and usefulness (the degree to which the use of the system helps increase users' performance in their activity). For mashups both the ease of development of useful applications by non-programming end users and the business or other functional value of the application are relevant. Technical issues, such as the effectiveness and timeliness of composition, and user situation issues, such the new value obtained by service composition, are both relevant. Therefore, in evaluating our framework of mashup application development for business portals, we suggest that a combination of these two approaches offers a comprehensive way to evaluate the system with end users.

Accordingly we conducted a user study using evaluation guidelines given by Huang et al. (2008) and Wang et al. (2009). We measured composition time to build a new mashup service, the system's ease-of-use, usefulness and its situational features (in response to the situation changing). This evaluation experiment involved a convenience sample of 40 third year students in two groups: A, characterized by users with system development knowledge and programming skills (bachelor of IT students), and B, representing typical application end-users in an industry environment (bachelor of business and arts students). These are considered representative of the types of knowledge worker who will design and use mashups in their careers.

At the beginning, mashup application building in iGoogle was demonstrated to both groups. Group A were asked to design/compose a new portal application for an online bookshop using iGoogle mashups. The first task was to record the total composition time for a complete application built by them. End users were then requested to swap their completed applications for evaluation by others. Results are shown in table 2a. This experiment showed that end users can effectively build relevant services using a mashup approach.

In the second task, after building the mashup applications, both groups were asked to adjust their applications with local time setting and geographical features such as indicating the location of their business office. This was intended to identify the situational reasoning and customization features of the application. A questionnaire was used to record participants' responses on the system's features of reasoning, ease of use and usefulness of the developed applications.

Hypotheses were formed and tested for the questionnaire data, using a similar approach to Debuse, Lawley, & Shibl (2007). For each Likert scale statement in tables 2a and 2b, with available responses ranging from strong disagreement (1) to strong agreement (5), a null hypothesis was formed: respondents neither agree nor disagree with the statement. The null hypotheses for group A were:

H1. Respondents neither agree nor disagree with the statement "Mashups can help build applications with situation reasoning features".

H2. Respondents neither agree nor disagree with the statement "Situational features can be composed easily within mashups".

…

H10. Respondents neither agree nor disagree with the statement "I could see that the mashup system would be practically useful".

Null hypotheses were similarly formed for group B, using the Likert scale statements from table 2b. The 95% confidence interval for each mean statement response was used to test the

hypothesis; a lower bound in excess of 3, indicating agreement, or an upper bound of 3 or less, indicating disagreement, would result in the null hypothesis being rejected. The 95% confidence interval statistic means that, if the study was run an infinite number of times (with a different sample from the same population being used each time), then the confidence interval would contain the mean value 95% of the time (Boslaugh & Watters, 2008). The 95% confidence interval is thus described sometimes as giving a plausible range of mean values (Boslaugh & Watters, 2008).

Cronbach's alpha measures internal reliability for questions such as those used in our study, although the literature does not offer much guidance in terms of a minimal value (Kent, 2001). Cronbach's alpha was 0.809 for the Technical group. The statistic only examined the items in table 2a in the paper that used scales from 1-5; it excluded service composition time and "Do you think this type of application can help increase user performance...", along with familiarity with mashup systems and level of knowledge. Cronbach's alpha was also calculated for the business group using all items in table 2b within the paper (thus it excluded familiarity with mashup systems, level of knowledge and level of use); it had a value of 0.876.

Table 1 shows the detailed design of the evaluation experiments; tables 2a and b show the outcomes of the experiments for the two groups. Three group B students were absent during the evaluation session due to urgent personal commitments, although they gave comments on the developed mashup applications later in words.

Further responses from group B were recorded using a separate questionnaire developed from (Grammel & Storey, 2008); the authors argue that mashups are a promising area for end user development (EUD). They evaluate several common mashup makers that provide EUD support features and summarize their approaches from an end user perspective, grouped by the kind of mashup produced, and evaluated using six themes drawn from research and experience. These themes also provided guiding questions for group B, with two questions per theme (Grammel & Storey, 2008), each rated on a 5 point Likert scale. Table 2b contains the results for these items.

The questions given to the two groups were different, and thus could not be compared, with the exception of "Familiarity with Mashup Systems". Further, the "Level of Knowledge" question for the business group was similar to the "Level of knowledge use of Mashup" question in the technical group, and thus was also compared between the groups. For both questions, the assumptions for normality had not been met and hence non-parametric testing was performed instead of a T-test. Specifically, the Mann-Whitney U test was used. This indicated that the groups were not significantly different on either the familiarity, $Z = -1.426$, $p > 0.05$, or level of knowledge, $Z = -0.525$, $p > 0.05$.

As the tables show, evaluation was very positive for all users. The null hypothesis was rejected for all statements within both groups, due to respondents' agreement with the statement. For ease of use and usefulness (traditional criteria for technology use and acceptance), scores around 4/5 were typical, and this was further backed by comments approving the speed and customizability both from technical and business user perspectives, e.g.

- *each user can have their own custom layout that works for them (technical participant 5)*
- *Easy to use, simple navigation, and fast creation (technical participant 3)*

And

- *we can save time. It is very easy to apply in day to day life (business participant 3)*
- *Easy to use, Can find things easily, change style (business participant 14)*

Learning technical skills, albeit "gently", was seen as valuable for business users, who were generally very positive about designing their own mashups, as the results in table 2b show.

7. DISCUSSION AND CONCLUDING REMARKS

This paper described how mashup techniques can replace traditional portal based web service development. We highlight the importance of utilizing the mashup design paradigm by proposing a generic architecture for emerging application development. This type of architecture can leverage and integrate end user relevant information from existing applications on the web. A technical architecture was outlined and this provides a conceptual model for particular implementations, one of which was

prototyped using iGoogle to demonstrate the concept.

To show the effectiveness of the approach representative groups of technical users and business end users evaluated mashups designed to emulate portal functionality, including customization and situated reasoning features. Results showed that useful and easy to use mashups could be developed within minutes. These were also useful and intuitive to users beyond the original developer, a typical limitation of end-user developed systems. The ease of development was noted with even non-programming business users commenting positively about learning the skills to develop their own mashups.

Although this architecture addresses issues in a traditional web portal system, there are limitations with any mashup based approach. Implementing the concept in some business areas would need consideration of issues such as intellectual property and organizational boundaries, both of which can create implementation issues for these types of services. In addition, sensitive data may require encryption when mashed up with data from other sources, although leading mashup platforms do include enhanced security features (JackBe, 2009). Finally, a third party content provider's unwillingness may interrupt the free flow of information. Evaluators saw a role for technical controls to reduce risk and this will be a useful feature to include in future mashup makers.

The main contribution of this paper is the linking of the emerging technology in the form of mashups to better support the requirements of users for up-to-date and interactive web service development, under the guidance of a classical research method. At the same time, in practice, we have shown how a mashup approach can be applied to overcome limitations of current virtual organizations based only on a distributed architecture and without provision for directed data sourcing at the user layer. Detailed specification for this type of application would be problem specific, but the architecture itself is considered an advance in allowing more dynamic inputs from different data sources, directed by the specific context requirements of end users.

8. REFERENCES

Auinger, A., Martin, M., Nedbal, D., & Holzinger, A. (2008). Mixing Content and Endless Collaboration – MashUps: Towards Future Personal Learning Environments. Retrieved 23 March, 2010, from https://online.tu-graz.ac.at/tug_online/voe_main2.getVollText?pDocumentNr=95304andpCurrPk=42744

Avison, D. E., & Wood-Harper, A. T. (1990). Multiview: an Exploration in Information Systems Development. McGraw-Hill, Maidenhead.

Babbie, E. (2004). The practice of Social Research (10th ed.). Thomson, Wadsworth Publication company, CA.

Backus, J. (1977). Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs. *ACM Turing award lectures*.

Boslaugh, S., & Watters, P. (2008). Statistics in a Nutshell: A Desktop Quick Reference. O'Reilly Media, Inc.

Chena, S. Y., & Macrediea, R. (2010). Web-based interaction: A review of three important human factors. *International Journal of Information Management, 30*(5), 379-387.

Clarkin, L., & Holmes, J. (2012). Enterprise Mashups. *The Architecture Journal*. Retrieved October, 2012, from http://msdn.microsoft.com/en-us/library/bb906060.aspx

Datta, A., Dutta, K., Thomas, H., VanderMeer, D., & Ramamritham, K. (2002). Accelerating dynamic web content generation. *Internet Computing, IEEE, 6*(5), 27-36.

Davis, F. (1989). Perceived Usefulness, Perceived Ease of Use and User Acceptance of Information Technology. *MIS Quarterly, 13*, 319-340.

Debuse, J., Lawley, M., & Shibl, R. (2007). The Implementation of an Automated Assessment Feedback and Quality Assurance System for ICT Courses. *Journal of Information Systems Education, 18*(4), 491-502.

Dillon, T. S., Wu, C., & Chang, E. (2007). *GRIDSpace: Semantic Grid Services on the Web:Evolution towards a SoftGrid.* Paper presented at the 3rd International Conference on Semantics, Knowledge and Grid, Xian, China.

Dong, B., Zheng, Q., Xu, L., Li, H., Yang, J., & Qiao, M. (2009). *Jampots: a Mashup System towards an E-Learning Ecosystem.* Paper presented at the Fifth International Joint Conference on INC, IMS and IDC.

Dotsika, F. (2010). Semantic APIs: Scaling up towards the Semantic Web. *International Journal of Information Management, 30*(4), 335-342.

Fox, G., & Pierce, M. (2007). *Web 2.0 and Grids.* Paper presented at the 3rd International Conference on Semantics, Knowledge and Grid, Xi'an, China.

Grammel, L., & Storey, M. A. (2008). An End User Perspective on Mashup Makers, University of Victoria Technical Report DCS-324-IR. Retrieved July, 2009, from http://lars.grammel.googlepages.com/paper_mashup_makers.pdf

Hardy, N., Pinto, M., & Wei, H. (2008). *The impact of collaborative technology in IT and computer science education: harnessing the power of web 2.0.* Paper presented at the 9th ACM SIGITE conference on Information technology education.

Hsiao, I. H., Li, Q., & Lin, Y. L. (2008). *Educational social linking in example authoring.* Paper presented at the 19th ACM conference on Hypertext and hypermedia.

Huang, A. F. M., Huang, S. B., Lee, E. Y. F., & Yany, S. J. H. (2008). *Improving End-User programming with Situational Mashups in Web 2.0 Environment.* Paper presented at the IEEE International Symposium on Service-Oriented System Engineering.

JackBe, E. (2009). Safer Enterprise Mashups. Retrieved 23 July, 2009, from http://www.jackbe.com/enterprise-mashup/blog/presto-27-has-been-released

Kell, S. (2009). *The Mythical Matched Modules.* Paper presented at the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications (OOPSLA).

Kent, R. (2001). Data Construction and Data Analysis For Survey Research. Palgrave Macmillan.

Lausen, H., Ding, Y., Stollberg, M., Fensel, D., Hernandez, R. L., & Han, S. K. (2005). Semantic web portals: state-of-the-art survey. *Journal of Knowledge Management, 9*(5), 40-49.

Macías, J. A., & Castells, P. (2007). Providing end-user facilities to simplify ontology-driven web application authoring. *Interacting with Computers, 19*(4), 563-585.

Maximilien, E. M., Ranabahu, A., & Gomadam, K. (2008). An online platform for web APIs and service mashups. *Internet Computing, IEEE, 12*(5), 32-43.

Miah, S. J., & Gammack, J. (2008). *A Mashup Architecture for Web End-user Application Designs.* Paper presented at the Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST), Thailand.

Miah, S. J., Gammack, J., & Greenfield, G. (2009). *Mashup Technologies for Building End-user Enabled Business Portal.* Paper presented at the Third IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST), Istanbul.

Ngu, A. H. H., Carlson, M. P., Sheng, Q. Z., & Paik, H. (2010). Semantic-based Mashup of Composite Applications. *IEEE Transaction on Services Computing, 3*(1), 2-15.

Ort, E., Brydon, S., & Basler, M. (2007). Mashups Styles, Part 1: Server-Side Mashups (Sun Microsystems). Retrieved 23 March, 2009, from http://java.sun.com/developer/technicalArticles/J2EE/mashup_1/

Pearce, J. (2006). User collaboration in websites. Retrieved July 2, 2008, from http://www.nla.gov.au/nla/staffpaper/2006/jpearce.html

Simmen, E., Altinel, M., Padmanabhan, S., & Singh, A. (2008). *Damia: Data Mashups for Intranet Applications.* Paper presented at the ACM SIGMOD International Conference on Management of Data, Vancourver, Canada.

Slatnick, M., Parkins, G., & Dheap, V. (2008). Web 2.0 Meets Telecom. Retrieved October, 2012, from http://www-01.ibm.com/software/industry/frameworks/pdf/Web2_0_MeetsTelecom.pdf

Srivastava, B., & Koehler, J. (2006). Web Service Composition - Current Solutions and Open Problems, IBM India Research Laboratory. Retrieved July, 2009, from http://www.zurich.ibm.com/pdf/ebizz/icaps-ws.pdf

Vidgens, R. (2002). Constructing a web information system development methodology. *Information Systems Journal, 12*(3), 247–261.

Wang, G., Yang, S., & Han, Y. (2009). *Mashroom: End-User Mashup Programming Using Nested Tables.* Paper presented at the International World Wide Web Conference, Madrid, Spain.

Wang, X. D., Yang, X., & Allan, R. (2006). *Top Ten Questions To Design A Successful Grid Portal.* Paper presented at the Second International Conference on Semantics, Knowledge and Grid (SKG 06).

Xu, H., Sharma, S. K., & Hackney, R. (2005). Web services innovation research: Towards a dual-core model. *Journal of Information Management, 25*, 321–334.

Yahoo Pipes (2009). Retrieved July, 2009, from http://pipes.yahoo.com/pipes/

Yee, R. (2006). Mashups, IST-Data Services. Retrieved December, 2009, from http://dret.net/lectures/services-fall06/Mashups.pdf

Appendix 1: Figures and Tables

Figure 1. Conceptual Model of Generic Mashup Portal Architecture

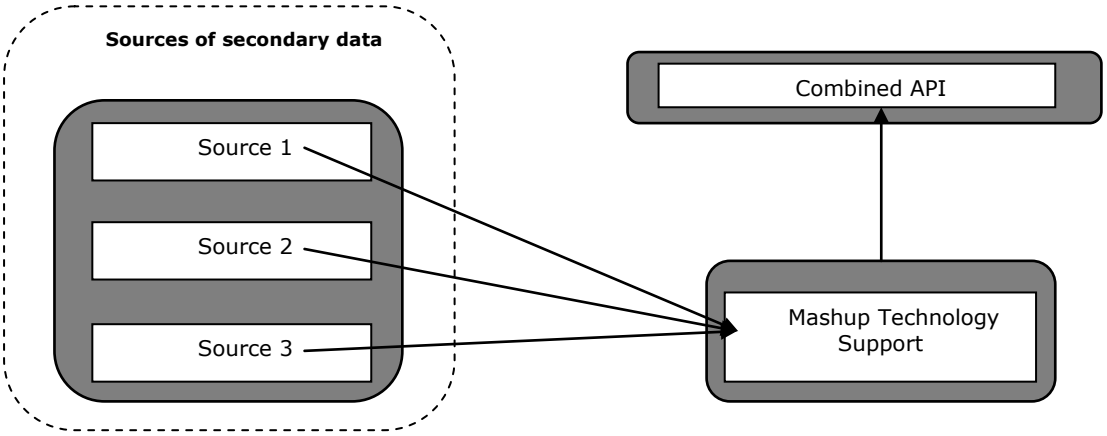
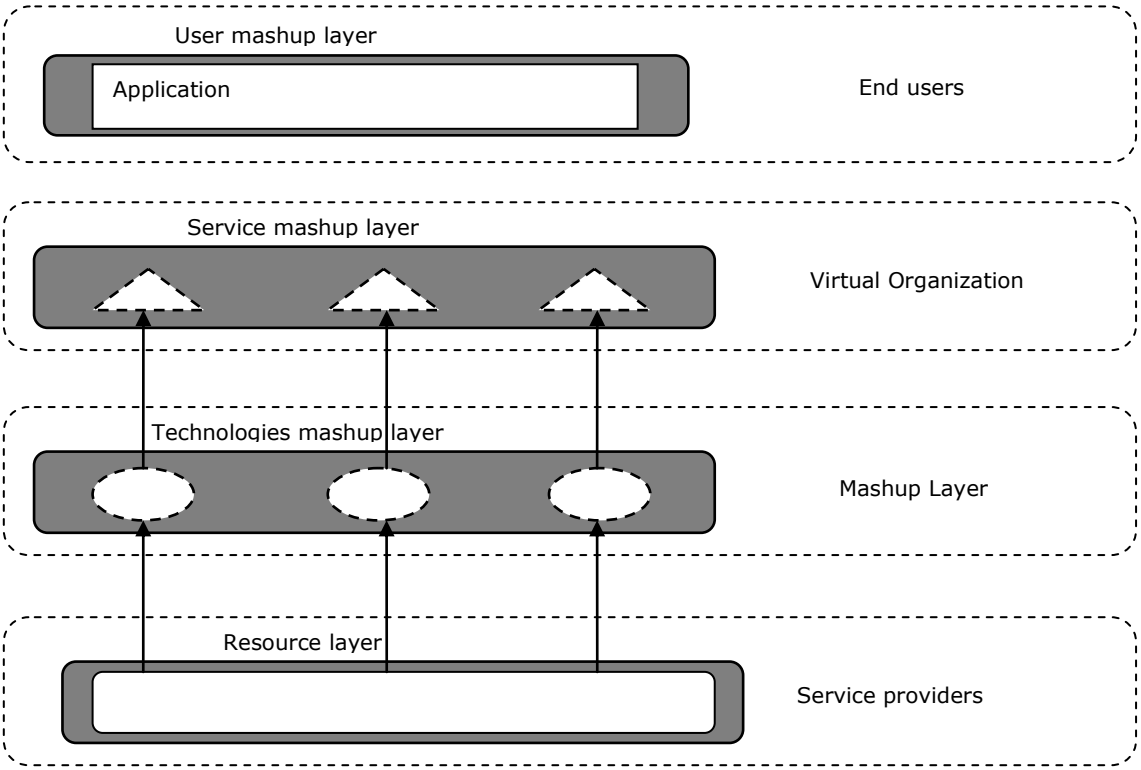


Figure 2. Mashup Architecture for a Technology Product Aggregator Service



ISSN: 2167-1508
v5 n2219

Figure 3. A Prototype of Mashup Portal Applications Developed Using iGoogle Mashups

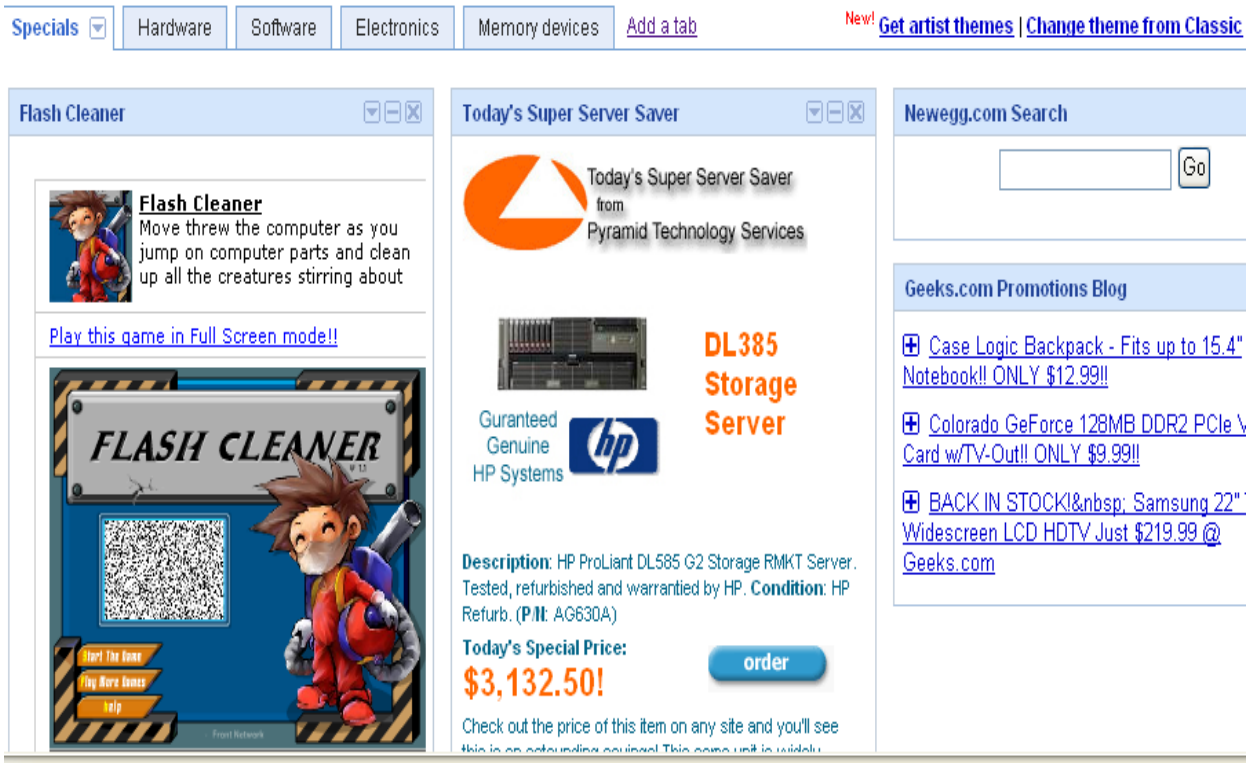


Table 1. Experiment Design for Mashup Evaluation

Students Group	Evaluation assessments	Number of users involved	Assessment criteria
Group A	Test 1	20	Composition time of building a new mashup
	Test 2	20	Situation reasoning features
	Test 3	20	Ease-of-use
	Test 4	20	Usefulness
Group B	Test 5	17	Levels of Abstraction, Learning Support, Community Support, Searchability, UI Design and Software Engineering Techniques

Table 2a. Outcomes of the Evaluation Experiments for the Technical Group (N=20)

Category	Experimental item		95% CI for Mean		
		Mean	Lower Bound	Upper Bound	SD
1. Service Composition	Total time for Service composition (minutes)	22.50	19.93	25.07	5.501
2. System reasoning features	Mashups can help build applications with situation reasoning features	3.85	3.54	4.16	.671
	Situational features can be composed easily within mashups	4.30	3.96	4.64	.733
	Situation reasoning features add value to applications	4.35	4.00	4.70	.745
3. System ease of use	Effort involved to use the mashup in web based book-shops?	4.25	3.88	4.62	.786
	Simplicity of navigation using the mashup application?	4.15	3.84	4.46	.671
	How useful do you think the system is for composing information?	4.05	3.73	4.37	.686
	Mashups can offer a generally useful way of building web based applications	4.10	3.73	4.47	.788
	It was easy to build a new mashup	4.60	4.32	4.88	.598
	This system is relatively simple and straightforward to use	4.70	4.43	4.97	.571
4. System usefulness	I could see that the mashup system would be practically useful	4.35	4.08	4.62	.587
	Do you think this type of application can help increase user performance for any business or online shops (e.g. book, CD sales)?	Yes (all)			

Table 2b. Outcomes of the Evaluation Experiments for the Business User Group (N=17)

Category	Experimental item		95% CI for Mean		
		Mean	Lower Bound	Upper Bound	SD
1. Levels of Abstraction	Application building through mash-ups needs only little programming knowledge	3.71	3.31	4.10	.772
	Through working directly with visual notations it is easy to build specific business solutions	4.12	3.68	4.56	.857
2. Learning Support	Mashup makers should include learning features to help end users make mashups	4.65	4.39	4.90	.493
	A "gentle learning slope" will help business users master mashup design	4.59	4.27	4.91	.618
3. Community Support	Sharing mashups and their building blocks will help business communities succeed	4.29	3.90	4.69	.772
	Tagging and rating user-made mash-ups will strengthen mashup tool sharing and use	4.00	3.52	4.48	.935
4. Searchability	Searching for useful mashups made by others using title and sorting by ratings is good enough for most business users to find what they want.	3.94	3.48	4.40	.899
	Browsing mashups by their elements and context specific suggestions about mashup elements is relevant for business users seeking useful mashups	4.41	4.05	4.78	.712
5. UI Design	End users should be able easily to create their own user interface (i.e choosing themes, colours, text styles etc.)	4.71	4.46	4.95	.470
	Automatic interface generation, although not customisable, is good enough for business end users	4.06	3.50	4.62	1.088
6. Software Engineering Techniques	Business users should learn technical skills to ensure security and correctness of their mashups	4.47	4.02	4.92	.874
	Mashup makers should provide strong technical controls to reduce these risks by business users designing mashups	4.18	3.76	4.59	.809