A mashup architecture for web end-user application designs

2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008)

A Mashup Architecture for Web End-user Application Designs

Shah J Miah [1] and John Gammack [2]
Institute for Integrated and Intelligent Systems,
Griffith University, Nathan Campus, QLD 4111, Australia
E-mail: [1] s.miah@griffith.edu.au, [2] j.gammack@griffith.edu.au

Abstract— **Application design driven by user needs is an increasing trend: such applications may not be anticipatable using traditional requirement and build approaches. Mashup refers to an integrated Application Programming Interface (API) that combines data from different data destination or third party sources for web services. This web service provides a combined API that is technologically valid and compatible with other web applications. In recent years, web mash ups have been tested for solving many issues in existing web applications, such as e-science. In this paper, we analyzed a real service problem in a current virtual organization to show current limitations of using a distributed architecture, and to describe a web service orientated architecture using the mashup concept. We provide a schematic solution for a paradigmatic user application problem, illustrated by a specific map usage need based on geographical information system data. Specifically, our illustrative application is finding the closest hotel within walking range of a meeting venue. Based on the case issues, we outline a generic architecture that offers a dynamic solution for web portals providing services for dynamic user needs.**

Index Terms— **mashup, web applications, hotel finding problem, end users, user mashup.**

I. Introduction

Web services are one kind of web application providing 'platform independent' service components available on the Internet. Application services are assembled from a combination of different but suitable third-party web services that appears as service compositions [9]. This type of composition is no longer being written using fixed HTML code, but rather from seamless, dynamic applications assembled using mashup concepts into an integrated experience. In this new era of IT and communication, Web developers have started mashing up different existing services to leverage and integrate new service applications.

Yee (2006) describes mashup as using "XML and web services to reuse or "remix" digital content and services. A mashup is a web application that seamlessly combines web content from more than one source, provided through relatively simple Web Application Programming Interfaces (APIs). Distinguishing the data (service) level from the metadata (user) level Dillon (2007) describes mashup as both enabling virtual organisations and as the key to value creation. In this paper, we focus on of the user level of mashup in outlining a combined web API for a paradigmatic case, application and propose a generic architecture for one specific type of service application development.

Our running illustration is accommodation finding from combining map and hotel data, using the problem of discovering acceptable hotel accommodation within walking distance of a specific meeting venue (or other target destination, such as tourist locations).(see [13]) This is a familiar problem to business travellers, who typically need the information only for a short time, but is a requirement that lies outside the normal business processes of hotels to service due to its user-centric nature.
While the source information is normally implicit on the web, a mashup allows relevant information to be integrated where existing web service architectures have potential problems.

Such problems include scalability, performance, flexibility, and implementability [3]. For example, a web service is expressed as relevant with a Web Services Descriptions Languages (WSDL) that specifies only the syntax of messages that enter or leave a program [2]. "In which order 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 although there are specifications languages such as WSCI and BPEL4WS. Semantic web application development researchers have addressed this problem by providing grid based solutions. For instance, Dillon (2007) describes GridSpace which incorporates soft aspects of the grid, service-orientation, transient service space, and web architecture. In fact, GridSpace architecture provides infrastructure to enable service discovery and Mashup at various levels. In addition Fox and Pierce (2007) have suggested applications, infrastructures and technologies for e-Science environment. At a broader range (enterprise and distributed environment) these authors claimed that web 2.0 can provide narrow grids for building web services that may provide a robust managed environment. But in this case, one of the limitations for grid based solution is that, its interoperability interfaces for data rather than for infrastructure. In this paper, our focus extends this to outline an architecture that can provide generic and interoperable features. Our architecture uses the Mashup concept in enhancing web service composition issues for the Portal based applications.

Service composition often solves explicit business processes issues in achieving goals related to the business perspective, and focused on business process based steps. But sometimes the requirements of service remain implicit and it requires data handling for customers/end user's service support. For example, the explicit goal of a web based hotel system is to offer service for reservation and after or in-use services, while its one of the implicit service is required by the customer/end user for finding their closest hotel to

where they would like to be. This requires a specific implicit service that will sort the hotel name according to how close/far, costing, and hotel status, so that customer can decide. These may be considered under the rubric of end user designed or directed requirements [1] [7].

Understanding this issue in relation to the explicitly relevant issues in developing web based service application is required to effectively address the web services composition issues, and the general problem of user directed requirements applies to numerous virtual business operations, not merely our simple hotel example. Therefore we have selected a realistic web service application for case analysis towards a model mashup solution and design for rich internet applications

The virtual business we analyse here (real but anonymised) is a wholesaler's portal based in Europe in web business for more than ten years selling computer goods, hardware, electronics household goods etc. We investigated this business process and found that they used a complicated distributed architecture. We propose a flexible alternative solution using mashup which is suitable and easy to implement within the business.

The rest of the paper is organized as follows. First we briefly describe background information on Mashup technologies in comparison to traditional distributed technologies, for virtual organizations. Section 3 describes the case research as motivation for proposing the mashup solution in this paper. Section 4 describes the related technologies for mashup architecture. Section 5 describes a proposed architecture of the mashup solution. And finally section 6 comprises discussion and conclusions.

II. TECHNOLOGIES FOR USER SERVICE

Processes are rapidly changing in businesses with the potentialities for enhanced virtual operation using emerging technologies. Also changing are customer demands, marketplaces, competition, business partners, and advantages conferred by innovative *use* of new technologies, rather than their mere presence. Such attributes imply businesses problems may be solved in a dynamic way but the current web development technologies does not yet support fully such types of service solution.

While distributed computing offers grid services for flexible solutions to the service needs for the businesses, current technology suffers from service complexity as service needs grow. The focus becomes on designing service processes for evolution not just connecting structural units, and the architectural philosophy changes accordingly.

Table 1 shows the main differences between distributed and service oriented architectures

Distributed architecture	Service Oriented architecture
designed to last	designed to change
Firmly coupled	loosely coupled
integrate silos	compose services
code oriented	metadata oriented
long development cycle	interactive and iterative development
cost centered	business centered
middleware makes it work	architecture makes it work
favors homogeneous technology	favors heterogeneous technology

Table 1: Difference between distributed and service orientated architectures - Sourced from [10]

The trend is less about providing more design middleware and replacing the old systems and more about reusing the old applications in a new way by remixing the system solutions. Service can be composed from businesses' existing services. Therefore, to cope with change, empower users and enable innovative concepts, mashup offers flexible composition of the existing services within an improved user interface environment-improved API suitable for the target users.

There are various existing technologies for assembling mashups [4]; [11]; [12]. Such technologies include XML which is used for data retrieving, web services such as simple object access protocol (SOAP), and web services description language (WSDL) for delivering data to the client. WSDL allows abstract services which are neutral regarding message formats or network protocols used in network communication. SOAP is a protocol that allows exchange information over HTTP. On the other hand, Really Simple Syndication (RSS)/Atom for providing syndication standards, screen scraping for pulling data from other sites, and Asynchronous JavaScript and XML (AJAX) for interacting various technologies for asynchronous loading and displaying data. AJAX is also a cross platform technique that uses a combination of XML and CSS when making up web information.

AJAX plays important roles in the Mashup for enhancing user experience by enabling easy process of sending or receiving data to/from the third party sources. Merrill (2006) described that AJAX comprises technologies such as XHTML and CSS for presentation; Document Object Model (DOM) API for dynamic display and interaction; Asynchronous data exchange for XML data; and scripting (JavaScript) at browser-side. The purpose of these technologies is to develop a web experience for user by sending and receiving data from the sources such as content servers.

For example, Google Maps API currently utilizes AJAX that creates a powerful user experience. For example, a web application behaves as a local application and there is no scrollbars to manipulate that can force for page reload [11].

These technologies provide a mingle interaction between the web services or APIs and content providers.

Both service level mashup and user level mashup functionalities are required in a comprehensive architecture. Various researchers have proposed architectures for mashup applications. For example, Patton (2007) suggested a general architecture with three tiers. These tiers are a) a *resource* tier which provides common sources of data used in the mashup, b) an *application* tier which carries together these resources into a hybrid or has its own functionality for presenting data, and c) a user base which accesses the site contents.

III. Research Case

The portal "http://yannipes.eu" is a virtual business that sells software, hardware and household electronic accessories. This website offers comparative price for product in the list which is based in the companies prices in the central European area, and leveraging to a full service platform through business relationships with the enlisted companies for whom they independently provide the price and product data. Yannipes incorporates their data and acts as a wholesaler portal. Customers can buy product from the range of brands based on the latest product information (e.g. prices, product description, availability, and payment terms). Using this portal customer can decide about any products based on its competitive brands, specifications, availability, and most competitive price.

This type of aggregator business is found commonly across the web and presents a mature model for widespread data sourcing. However, the system uses distributed database architecture which is less dynamic for the business when a third party changes/updates their information, and new business models in line with improved and flexible (e.g. web 2.0) developments are needed. Figure 1 outlines the current data structure of the Yannipes site.

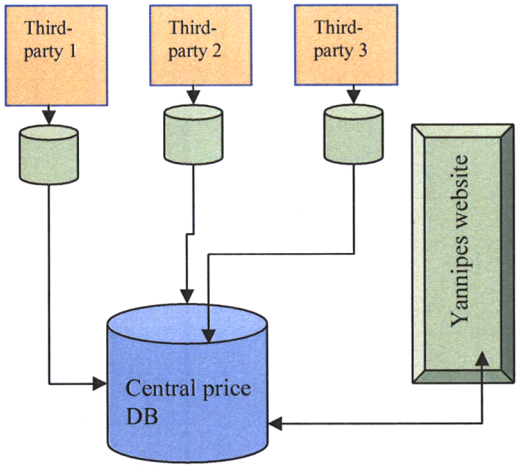

Fig 1: Yannipes general architecture where distributed database technologies are implemented

In this context of distributed architecture, mashup could address the issues by providing a more dynamic and evolutionarily adaptive platform for the business. Third parties can update their own database while they add/modify their own product. A combination of all third parties data resources can provide an opportunity for mashing up all into a new API where the customer can see their desired products in a comparable environment.

In this paper, we therefore discuss a new architecture of a solution that can address this issue. In particular our proposed approach replaces this distributed concept by utilizing mashup concepts to provide a combined API for a business portal where customers can get a more dynamic service system, related to their specific needs

The illustrative application presents a mashup architecture for the paradigmatic "hotel finding problem" where end users can obtain dynamic and user-centrically directed information to make decisions about candidate accommodation purchases.

This solution architecture is intended to be generic, interoperable and applicable in any similar problem domain. In a hotel finding within a selected destination, information such as a hotel's distance from the destination, prices, and room descriptions can be prioritized with mashing up of the hotel website (existing third parties API's). It will ensure end user targetted decision making about which hotel will be closest from a given location while enabling information of price and facilities to be included.

IV. Proposed Architecture

Figure 2 illustrates a conceptual mashup architecture for a solution to the hotel finding problem.

2008 Second IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2008)

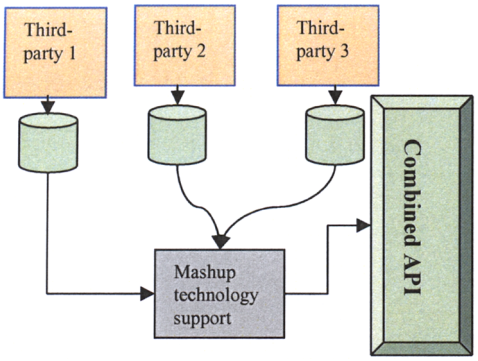

Fig 2: Conceptual model of hotel finding portal.

In this diagram each third party acts as a data island providing its data service by acting as a content server, that is to say 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 hotel's availability, locations, tariffs and status, but the principles applied can be generalised to any situation where the needs for such a system arise.

In this as in any Mashup style application, the critical thing to note is that their may be little if any commonality between the form or expression of data between the different providers (and indeed the required representation for the Mashup itself may be a fourth schema and style again. This is where the Mashup architect comes into their own.

The designer of the mashup must facilitate the user request into forms that are 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.

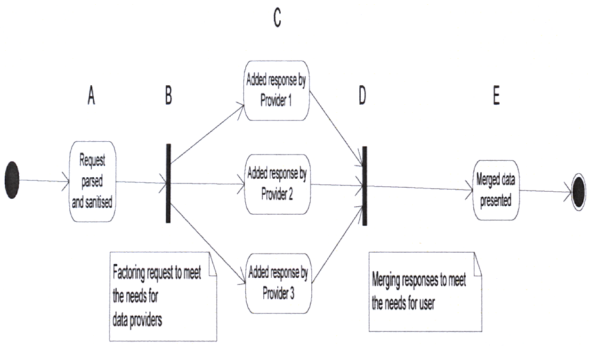

Fig 3: statechart for the mashup data.

In the first state (A in figure 3), a data request has received, analysed for sanity, verified for privileges and rectified where needed and possible, within the bounds of a secure environment

The single request is then fork-transformed (B in figure 3) into the separate requests needed for the content servers. Recapitulating, these can be:

1. transport (via XHTTP, REST or SOAP)
2. port (the service may be on TCP port 80, but may in fact be on any, particularly if it is a secure service
3. permissions (the individual permission/security level could be acquired by the mashup consortium, and factored internally)
4. encoding (while there are mandated encoding requirements for XML, the xml: lang attribute may have to be set, and some form of requisite re-encoding needed)
5. data normalisation (the fields for time/date/service level/price-range or even if the system was extended to bookings, for composite or individuated values for name and address). This may include incorporating standard data for a particular server that is required
6. typification (the individual values for time, date and even area-coding will vary considerably – the most obvious being the international ISO date versus the style employed by the USA)
7. preparation for return (for each request per 1,2 and 3 in this list, there must be a waiting process for a matching response)

The most important thing to recognise at this point is that though the data has been changed to match the requests, they are potential at this point – they have been tailored for the destination content provider, but in other ways they are synonymous. An additional point to note is that at this stage the identity of the requesting agent must either be concealed and replaced by the consortium identity, or else the system as a whole must permit transparent passing of identity (a situation that is not as secure for many reasons).

The individual requests variants are then sent off to the content servers (C in figure 3), where each is value added by the attachment of suitable data

Now a mirror process (D in figure 3, mirroring B in figure 3) to the forking must occur, to permit the merging of data into a single resource. Each of considerations 4,5 and 6 for B must be revisited to make a commonality of data

The final stage involves reinvolvement with the data service requester: depending on the form the API is decided to take this may in itself be an open request (via SOAP, REST or AJAX/XHTTP) or a closed request (delivery of a plain HTML page).

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 is going to be intellectual property then it may be prudent to conceal the inner workings of the data factoring and merging. The chief drawback of this is that of any closed-source system, that 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 de-

sign, 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 that they 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 4 illustrates the architecture for hotel finding portal system. It consists of four layers in line with the ideas presented by Dillon (2007). Which are at least in part orthogonal to the states present in figure 3.

These are the service providers-resource layer where the content servers are located (represented by the various diempotents of state C in figure 3); the technologies mashups transient layer where the mashup technologies create a combined interactions for data from different resources (B and D); The virtual organizations-service mashup layer (A and E) where the mashup services are prioritized for mashing up data from content servers; and finally the User mashup layer where the data is presented to the web end users according to their requirements (the initial and final states of figure 3).

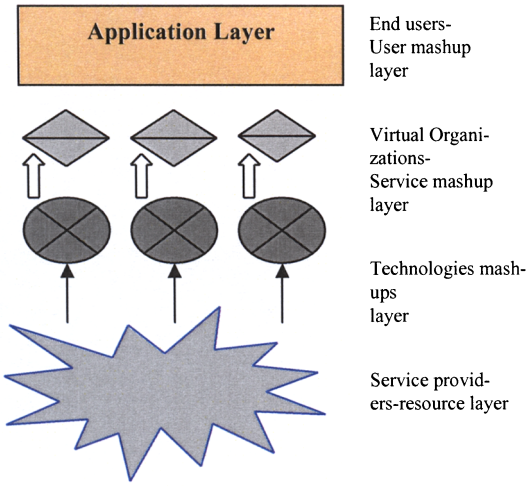

Fig 3: Mashup architecture for a hotel portal system.

V. Conclusion and Discussion

This paper described how mashup technique can be used to solve specific service issues for end users. In relation with this issue, we proposed a generic architecture using mashup concepts. We also described the relevant technologies that can be used for mashup in different service layers. This type of architecture can leverage and integrate the end user relevant information from the existing web applications in the web. Intellectual property, organisational boundaries as well as the third parties agreements can come across implementation issues for this type of services. Also the sensitive data is to require for confidentially handling for example required encryption need to be used when this data mashes up with data from other sources. The third party content providers unwillingness can interrupt the free flow of information. For some instances, web aggregation and regulations

Other issues with user centric mashups are signalled and these also were planned for in the architecture but require further research. End user computing in general is known for its potentially feral nature [8] and the generally lower quality, security, efficiency and integrity of user designed applications cause problems (familiar from "amateur" spreadsheet developments) that present challenges to enterprise adoption of mashup architectures. Parallels with conventional architectural strategies relevant to future enterprise architecture designs are noted in [6] and suggest areas in which enterprise mashup technology and associated policy is required.

In this paper we have sketched how a mashup approach can be applied to overcome limitations of current virtual organisations based only on a distributed architecture and without provision for directed data sourcing at the user layer. Detailed specification for this type of applications would be problem specific, but the architecture itself is considered an advance in allowing more dynamic inputs from different data sources, directed by specific context requirements of the user.

VI. Acknowledgement

Authors would like to thank Diarmuid Pigott for helpful suggestions and technical advice during the preparation of this paper.

VII. References

[1] Chan SL, "End-user directed requirements: a case in medication ordering In Advanced topics in end user computing" ed MA Mahmood IGI Publishing Hershey, PA, USA, 2002

[2] Christensen & others, "The web services description language WSDL" Available at http://www-4.ibm.com/software/splutions/webservices/resources.html

[3] Dillon, T.S., Wu, C. and Chang, E., "GRIDSpace: Semantic Grid Services on the Web:Evolution towards a SoftGrid", In *proceedings of the 3rd International Conference on Semantics, Knowledge and Grid,* Oct.29-31, 2007. Xi'an, China.

[4] Fontana, J., IBM, Google team on portal, web mash-ups, Network World, 2007, Available at http://www.networkworld.com/news/2007/022807-ibm-google-portal.html

[5] Fox, G.and Pierce, M. (2007), Web 2.0 and Grids, *In proceedings of* the 3rd International Conference on Semantics, Knowledge and Grid, Oct.29-31, 07. Xi'an, China.

[6] Gammack JG, "Designing for End user systems construction: a process approach" in Sutton DJ (Ed), *Proc 8th Australasian Conference on Information Systems,* Adelaide, 1997, pp104-114 http://business.city.unisa.edu.au/acis97/papers/fp058.pdf

[7] Gammack, JG, "Constructive design environments: implementing end-user systems development", *Journal of End User Computing, 1998,* vol.11 n.1, pp.15-23,

[8] Kerr, D.V., Houghton, L., and Burgess, K., "Power Relationships That Lead To The Development Of Feral Systems", *Australasian Journal of Information Systems*, 2007, Vol 14, No 2, pp141-152

[9] Srivastava, B. and Koehler, J., "Web Service Composition - Current Solutions and Open Problems", IBM India Research Laboratory. Block 1, IIT, New Delhi 110016, India. 2006, Available at http://www.zurich.ibm.com/pdf/ebizz/icaps-ws.pdf

[10] Schmelzer, R., "Emerging Trends in SOA: Rich, Smart, Mashed and Governed", ZapThink, 2006, Available at http://colab.cim3.net/file/work/SOACoP/2006_10_3031/Presentations/RSchmelzer10292006.ppt

[11] Merrill, D., "Mashups: the new breed of web application", IBM developers work, 2006, Available at http://www.ibm.com/developerworks/xml/library/x-mashups.html

[12] Patton, T, "Mashups put a new face on the web, 2007, Available at http://articles.techrepublic.com.com/5100-3513-6156271.html

[13] Wong J and Hong J. I, "Making Mashups with Marmite: Towards End-User Programming for the Web CHI", 2007, In proceedings Programming By & With End-Users, Available at http://delivery.acm.org/10.1145/1250000/1240842/p1435-wong.pdf?key1=1240842&key2=6298157911&coll=GUIDE&dl=GUIDE&CFID=9464130&CFTOKEN=91409841

[14] Yee, R., Mashups, IST-Data Services, 2006, Available at http://dret.net/lectures/services-fall06/Mashups.pdf