BUILDING COMPLEX SYSTEMS ON TOP OF WEB 2.0
Integration of Web 2.0 Services using Enterprise Service Bus

Pavel Drášil and Tomáš Pitner
Masaryk University, Faculty of Informatics, Botanická 68a, 602 00 Brno, Czech Republic

Keywords: Web 2.0, Service-Oriented Architectures, Service Integration, Mashups, ESB, BPEL.

Abstract: Service-oriented architectures are a predominant architectural style in current enterprise software systems while Web 2.0 is a predominant paradigm in current web environment. Even if the ideological and technological bases of the two are quite different, many similarities can be found in their view of services as basic building blocks and service integration as a way of creating complex applications. The "Web 2.0 Platform", introduced in this paper, bridges these two worlds by applying enterprise-oriented technologies in Web 2.0 service integration. Its advantages are shown in a case study of a learning environment, based on Web 2.0 services, supporting the learning patterns required by Inclusive Universal Access in learning.

1 INTRODUCTION

The term "Web 2.0" itself still has some notion of controversy in it (Berners-Lee, 2006). But the ideas it represents are by no means the most important driving force of current web development and can bring a renewed vigour to many long-established fields of computer science such as knowledge management (wikis, tagging, folksonomies) or software development methodologies (frequent releases, perpetual beta).

This paper focuses on yet another field, in which the Web 2.0 experience can be useful – software architectures. In section 2 we briefly review the concepts of service and service integration in both traditional and Web 2.0 sense. Section 3, as a result of our research, proposes a new architecture for Web 2.0 services' integration based on enterprise technologies. Consequently, the case study in section 4 utilizes the platform to support learning founded on Inclusive Universal Access. Finally, the conclusion rounds the paper off and identifies some extension points for further research and development.

2 SERVICE-BASED SYSTEMS

Loosely coupled standalone services with defined interfaces can be used as basic building blocks of complex systems. This idea can be traced back to the concept of "modular programming", introduced in 1970s. In the late 1990s, this principle came up again labelled "Service-oriented architecture" or simply SOA (Gartner, 1996). One decade later, the term "Web 2.0" was coined as a general label for the new spontaneously evolved trends apparent in the web (O'Reilly, 2005).

When considering the totally different origin and evolution of SOA and Web 2.0 concepts, it is surprising how close they got at the end - at least from a software architect's perspective. They both promote easily accessible services and "composite applications" built on top of them.

2.1 Service Integration

Both traditional SOAs and Web 2.0 deal with integrating and orchestrating simple services in order to create aggregate services able to support complex requirements of their users. However, the technological bases of the two are quite different which leads to different solutions.

The most traditional way of implementing the SOA principles is employing the "Web services" technology based on HTTP, SOAP, WSDL and UDDI specifications. However, modern SOA implementations accompany this low-level technology by the "Enterprise Service Bus" (ESB) messaging layer and also by the "Web Services Business Process Execution Language" (WS-BPEL,

Drášil P. and Pitner T. (2009).
BUILDING COMPLEX SYSTEMS ON TOP OF WEB 2.0 - Integration of Web 2.0 Services using Enterprise Service Bus.
In *Proceedings of the 4th International Conference on Software and Data Technologies*, pages 179-182
DOI: 10.5220/0002259101790182

or just BPEL) engine for service orchestration.

Interoperability of Web 2.0 services is achieved in a different way. Web 2.0 service integrations, usually called "Mashups", are based on simple protocols (HTTP used in either RPC or REST style), simple data models (JSON, custom XML or RSS/ATOM) and often moved from servers to clients (Drášil et al, 2008). As opposed to traditional SOAs, no formalized or machine-readable notations are used for describing the service API and no service discovery mechanisms are employed. APIs are described textually and service discovery is left up to the community.

The popularity of Web 2.0 services has made an impact on legacy applications too. Many present applications come with a built-in support for various Web 2.0 services. Probably the best known representative is Flock, a Firefox-based web browser with integrated support for 21 Web 2.0 services (at the time of writing).

3 PROPOSED ARCHITECTURE

"Web 2.0 Platform" is our contribution to the field of Web 2.0 services' integration. Its basic idea is that it delivers the functionality, actually provided by remote services with various APIs, to its client applications through a provider-neutral WSDL-defined interface. This way, it bridges the traditional SOA and the upcoming Web 2.0 paradigms (as described earlier in section 2).

Even though the platform was developed primarily for facilitating integrations of Web 2.0 services, it allows us to integrate virtually any software service, whether it has a Web 2.0 flavour or not. It can be used for integrating Web 2.0 services with legacy applications as well.

3.1 Design

"Web 2.0 Platform" consists of a platform core and an arbitrary number of pluggable components called "connectors" (see Figure 1).

"Connectors" provide functionality to the platform and subsequently to its client applications. Each connector has to provide a formal description of its interface in the form of an abstract WSDL definition. Remaining WSDL sections are added by the platform core when the WSDL is published.

As you can see in Figure 1, we differentiate two kinds of connectors – primary and secondary. Primary connector is just a wrapper for some external service. Even if it can do some necessary pre- or post-processing, it delivers its functionality primarily by calling the external service. In contrast, secondary connectors are higher level components that deliver their functionality by calling other connectors – either primary or secondary – and processing the results using built-in orchestration logic. This division is of course of no importance for client applications and there is no difference between calling the operation provided by a primary and a secondary connector.

The platform core takes care of routing requests and responses between client applications and connectors, publishes WSDL descriptions of available connectors and manages user accounts. This includes storing users' credentials for particular Web 2.0 services.

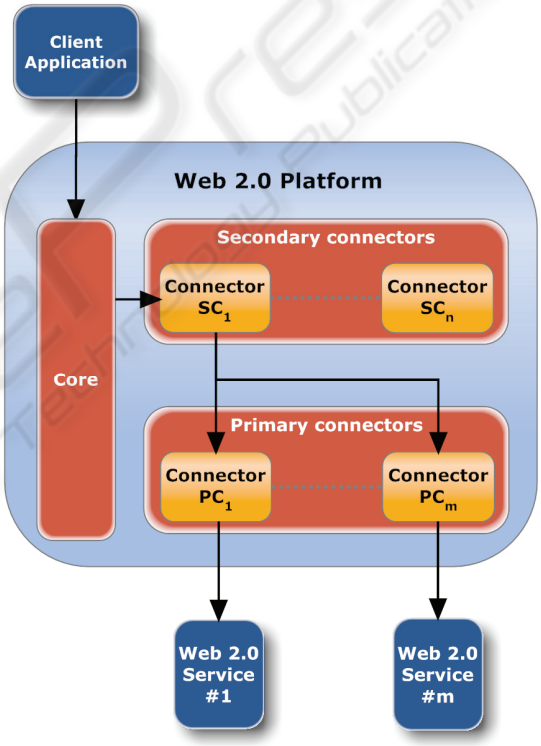

Figure 1: "Web 2.0 Platform" architecture and an example request flow.

3.2 Technologies Involved

The technological bases of the platform are well known, widely supported, enterprise-oriented and time-proved technologies, such as SOAP, WSDL, JBI and BPEL.

The platform was developed as a JBI ("Java Business Integration") application, running in the Apache Servicemix JBI container. JBI provided us with a standardized message format, message

routing facilities and hot-deployability of the connector components. The platform core is made up of several modules ("service units" in JBI language), whose detailed description is behind the scope of this article. As for the connector components, there are no limitations for their internal structure (as long as they are able to provide the WSDL and the declared functionality). In secondary connectors, services can be easily orchestrated using a declarative notation of BPEL. In cases where the descriptive power of BPEL is not sufficient, a full strength of the Java language can be used.

3.3 Selection of the Right Services

There are no technological barriers preventing incorporation of virtually any Web 2.0 service with public API into the "Web 2.0 Platform". However, choosing the right service to deliver the functionality needed may not be an easy task.

As stated in (Drášil et al, 2008), a substantial number of services, that use the fashionable "Web 2.0" label, were created for direct usage by humans only and do not allow any programmatic access. There are also services that offer an application interface, but it does not cover all the functionality provided by the service. Another aspect affecting the practical usability of particular service in the platform is the design of the API. Not all Web 2.0 service APIs, we have met so far, are designed for the communication to be seamless and effective. For example in Digitalbucket, a storage-oriented Web 2.0 service, many request-response interactions may be necessary for converting the name and path of a known file to its system id, required for any further operation with the file.

In addition to functional and technological aspects, one should consider also legal and other non-functional characteristics of the selected Web 2.0 service. From the legal point of view, there are at least two points that one should find out in the selected service's "terms-of-service" document. Firstly, one should ensure that the service allows using its API for the intended purpose and in the intended way. Secondly, one should carefully consider if the rights reserved by the service provider are acceptable with respect to the character of the data to be stored in the service.

One may be hesitant about storing sensitive or critical data in a remote service. Web 2.0 services provided free of charge often give no guarantees regarding users' data security or availability. However, if this is a concern, there are also Web 2.0 services that pledged to keep a high standard of provided service, such as Amazon S3 promising to ensure at least 99.9% service availability under financial penalties. A free choice of the services used is therefore one of the most important features of the "Web 2.0 platform". It allows platform adopters to choose the services that best satisfy their particular requirements or, if no such Web 2.0 service can be found, even to implement the functionality on their own.

3.4 Evaluation

The proposed architecture has all the advantages and disadvantages inherent for all service-oriented architectures. However, there are some notable characteristics with respect to Web 2.0 service integration:

Advantages:

- Any Web 2.0 service with public API can be incorporated, provided that its terms-of-service allow such a usage.
- Legacy systems can be reused.
- Specific or critical services can be implemented locally.
- As long as the primary connector's interface is designed carefully, the service used to deliver the functionality can be replaced if needed.
- As long as the required functional base is covered by existing primary connectors, new mashups can be added in a declarative manner using BPEL-based secondary connectors.
- It takes advantage of a high standard of enterprise technologies and tools.

Challenges:

- Reliability of the "Web 2.0 platform" depends on the reliability of the selected services and on the reliability of the Internet connection.
- Users have to create accounts in all Web 2.0 services in use and provide their credentials to the platform. However, a single user account if often used in all services run by the same company and there is also a growing number of services ready for single sign-on systems such as OpenID.
- Graphical user interface, a key aspect of many Web 2.0 services, is not utilised and client applications have to provide their own. However, there are also Web 2.0 services, such as Amazon S3, that do not have user
- interface at all.
- The impression of being a member of a community, another key feature of many Web 2.0 services, is lost. However, the

community-related functionality may still be available.

4 CASE STUDY

The proposed Web 2.0 Platform architecture allows us to implement an integrated environment for "Inclusive Universal Access in Technology-enhanced Learning".

Derntl and Motschnig (2007) introduced "Inclusive Universal Access" (IA) as an extension of "Universal Access" coined by Stephanidis and Savidis (2001) enhanced with non-technological, human aspects that can contribute to facilitating social and personal growth of students in learning and knowledge sharing settings. IA aims to actively involve learners in all aspects of learning and assessment; to primarily address them on all levels of learning including intellect, skills, and personality; and to employ universally accessible tools to support the educational activities.

Web 2.0 represents an ideal environment giving all learners the freedom to select their favourite service based on their preferences, experiences, and the services they use in the "real life" outside of the education process. In (Pitner, Derntl, Hampel & Motschnig, 2007) we identified the most significant learning patterns for IA in Technology-enhanced Learning and proposed a collection of Web 2.0 services supporting the respective patterns.

Figure 2: "Web 2.0 Platform" equipped with connectors to services employed to accomplish selected IA learning patterns.

As a proof-of-concept for our "Web 2.0 Platform", we select some of the patterns covered by the services and outline their integration into the Platform now, thus creating a complete personalized learning environment solution (see Figure 2). The chosen patterns include – see (Pitner et al., 2007) for a detailed explanation:

- Considering goals and expectations, Learning contracts;
- Project-based learning;
- Sharing and presentation of contributions, Peer teaching;
- Collecting feedback and opinions

5 CONCLUSIONS

The experiments with Web 2.0 Platform as an Inclusive Universal Access learning environment demonstrated the viability of our approach. However, to discover its full potential, it is necessary to extend the platform capabilities by incorporating more Web 2.0 services via new primary connectors, as well as to provide richer functionality of the secondary connectors in order to allow development of more powerful "mashups" covering – in our case – even more learning patterns. Another challenge is to cope with the issues listed in section 3.4.

REFERENCES

Berners-Lee, T., 2006. developerWorks Interviews: Tim Berners-Lee, http://www.ibm.com/developerworks/podcast/dwi/cm-int082206txt.html, retrieved 04/01/09.

Derntl, M., Motschnig-Pitrik, R., 2007. Inclusive Universal Access in Engineering Education, In *37th ASEE/IEEE Frontiers in Education Conference*. IEEE.

Drášil, P., Pitner, T., Hampel, T., Steinbring, M., 2008. Get ready for mashability!, In *ICEIS'08, 10th International Conference on Enterprise Information Systems*. INSTICC.

Gartner Group, 1996. Service Oriented Architectures, Part 1 and 2, SSA Research Notes SPA-401-068 and SPA-401-069, Gartner Press.

O'Reilly, T., 2005. What is Web 2.0?, http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html, retrieved 04/01/09.

Pitner, T., Derntl, M., Hampel, T., Motschnig, R., 2007. Web 2.0 as a Platform for Inclusive Universal Access in Cooperative Learning and Knowledge Sharing. *Journal of Universal Computer Science, Special Issue '7th International Conference on Knowledge Management'*.

Stephanidis, C., Savidis, A., 2001. Universal Access in the Information Society: Methods, Tools, and Interaction Technologies, *Universal Access in the Information Society Journal*, 1, 1.