SENG 403
Topic 7

for

Service-Oriented Software

Prepared by: Muhammad Sarwar
Derek Partible
William Kwok
Tieulong Trac
Philip Athyal
John Flores

Due on: April 11, 2013

Table of Contents

Introduction

Service-oriented software is a software system which involves the architecture and distributed networks of interoperating services [14]. The interoperability of these services provides developers with flexibility and allows them to easily integrate one service with another via well-defined interfaces. During the time of Web 1.0 to Web 2.0 transition, a bulk of service-oriented software systems emerged and brought a substantial change in the way services operated on the web. Developers were provided with a more comprehensive and organized structure to develop on, which as a result made services become more engaging with users.

Service-oriented software introduced the idea of giving opportunity to users to be the providers of the content and provided services where users could generate the content and make decisions. It also introduced mash-ups, which is a term that refers to web applications consisting of two or more independent services working together to form a larger system. The idea of "loosely coupled" services also stemmed from service-oriented software and development. Service-oriented software has made a great impact on how people interact and has largely moved web technologies forward. Because of its significance in the current age of the Web, it is important to understand the principle of service-oriented development, its benefits and implications, and what is involved in the overall process of creating and publishing such software.

Service-Oriented Development

Service Oriented Architecture is rapidly emerging as the main integration and architectural approach in a complex and heterogeneous computing environments. SOA provides a set of guidelines and methodologies for designing and developing software products in the form of a composition of several reusable services. These protocols allow developers to encapsulate information tools as services that clients can access without knowledge of the internal workings [1]. Examples of such products include Mashups, SAAS (Software as a service), cloud computing, etc.

Other than development of software, SOA requires that organizations evaluate their business models, plan design techniques, service oriented analysis, deployment and maintenance plans, and evaluate customer-product relations [2]. The development of software products adopted from these guidelines is referred to as Service Oriented Development.

These development approaches are used to solve software issues such as software inflexibility that would otherwise emerge when maintaining and expanding the software.[3]

Web 2.0

"Web 2.0" is a collection of revolutionary and exciting web technologies that enables [1] the use of lightweight and intuitive Web-based services, and user-generated content using the technology of Web publishing tools. These services and technologies generally involve some amount of social networking and social media. Web 2.0 also incorporates and encompasses around the concepts and technologies of interactivity and tagging [2].

Table 1: Web 1.0 vs Web 2.0

Web 1.0	Web 2.0
Personal website built using Frontpage	myspace
Favourites folder	Netvibes
Encarta	wikipedia
streetmap or mapquest	google earth
Ofoto or Image Station	flickr
Home video	youtube
mp3.com	iTunes
Microsoft Office	Google Docs & Spreadsheets
Computer desktop	Goowy

Table 1 as shown above illustrates the evolution of the Web from Web 1.0 to Web 2.0. This table displays certain technologies available during the time of Web 1.0 and are compared directly to Web 2.0 technologies. Although the entries of the table provide very specific examples, it is clear that many technologies of Web 1.0 are much more personal and static for users than Web 2.0 technologies, which have a large emphasis on user-generated content which may be public or private, and interactivity of the application services for users.

Loosely Coupled Services

Loose coupling is an approach to combining the components in a system in such a way that the components have little to no knowledge of one another. Using the idea of loose coupling, there are service-oriented systems that use the principle of service loose coupling design principle, ensuring that the services inside the system are not tightly contracted. This allows the services to freely evolve without affecting implementation of other services or hampering the system. [4]

A loosely coupled system can be easily decomposed into individual elements for easy maintainability and implementation changes, simplification of testing and error tracking, and in scenarios of severe failures the troubleshooting procedures are also very simplified. [5]

Service oriented architecture defines principles of systems with interoperable services, where the principle of loose coupling plays a major part. Independent design and implementation of services enforces reusability of components since the functionalities are combined together, and also the costs are kept under control when requirements or specifications change.

Mashups

Mashups are a rapidly growing implementation of applications, most commonly found on the web. A mashup is the combination of multiple third-party services: taking advantage of their content, data, and functionality. This integration method is quite similar to music mashups, which takes vocals and instrumental tracks from various songs. [6] Raw data from one source may not have any significance on its own; however through mashups we are able to combine and repackage data to be meaningful.The overall goal of a mashup is to create a combination that is more useful than the sum of its individual pieces. [7] In general, this means that mashups provide a better service for the end user.

Sources for these mashups are generally loosely coupled services. These services are totally unaware of each other and come from separate sources. Sources can include, but are not limited to, services found on Google Maps, Flickr, and even Craigslist. These services alone are hardly related and are not dependent on each other to function. By taking features and information from these applications we can create new services, which uses the data and features in a more useful way. Since the sources are completely independent of each other, the mashup process simply becomes picking and choosing what information and features are needed to create a new and meaningful service.

The process of creating a mashup implies a quick and easy integration, usually pulling information from open source applications and data. Mashups generally involve little to no coding knowledge. Integration focuses more on linking together data and function sources, rather than building systems from scratch. This method relies on services that provide an open application programming interface (API); this makes the process of isolating specific services easier. [8] There are several ways to pull information from a source, this includes methods such as screen-scraping and using URL templates. Screen-scrapings use bots to automate the collection of data, which can be read off a webpage by a human. Using URL templates refers to the use sources that are directly referenced through its address.

The concept of mashups is not new, by any means. The idea of using a composition of reusable services have been around for over decades. The following are some examples for mashups: Wikipediavision is a site that displays where anonymous edits to the site ‘Wikipedia’ originate from. This site is a mashup of ‘Recent Changes feed‘ service provided by Wikipedia and

‘Google Maps’ service provided by Google to create a 2D map view, displaying where the latest edits originate from.

Another map mashup, Trendsmap mashes up the ‘Trending Topics’ service from Twitter and ‘Google Maps’ service from Google. The result is a real-time map of Twitter trends across the globe.

MapsKreig mashes up ‘Apartment and Housing listings’ service from Craigslist and ‘Google Maps’ service from Google to create a real-time map of available places to rent. [15]

SongDNA is another example of a mashup. This mashup comes in the form of a mobile application that can be run on an iphone or an android device. The purpose of the application is to collect and present a huge large amount information about any song. SongDNA provides information including videos, a bio of the artist, its ranking in the charts, tweets, and much more. All this information comes from more than 7 different sources, including MusicBrainz, BBC Music, YouTube, Billboard, and BandsInTown.

Publication

Publication of a web service is not too complicated. The obvious first step is to build your web service. Making sure the service is fully functional with testing will help to increase the quality of the product. Once that is done, the web service should be hosted on an accessible web server so clients can call its functions. Reliability and uptime of the server is essential once users start using the web service.

Now that the web service is made and hosted on an accessible server, it is now time to let users know it is available. There are many ways to do this but one of the techniques we found to be the most common was to register the service at a web service registry. This registry is similar to a phone book or an app store. Developers can register their web service at the registry by providing a web service description language (WSDL) document hosted on the web service’s server. Developers obviously need to also make an account with the registry itself to submit web services to be added. [15]

The WSDL basically describes what features the web service offers, how it communicates, the type of data it takes in and the type of data it outputs. Users need this document to configure their own product to be compatible with the web service. Some registries require the WSDL URL link from users to call a certain web service, so it is a vital part of allowing users to access your web service.

Consumption

Taking advantage of Web 2.0 services with the goal to improve user experience for a website can be quite intimidating. The large amount of tools available is overwhelming, however it is not as bad as it looks. Many Web 2.0 tools are offered for free or for a relatively cheap monthly fee. These tools are web based, so very little (if any) installation is needed.

The first step in using a Web 2.0 service, is deciding exactly what you need. “Identify a Web 2.0 service that will actually accomplish something for you”, suggests Kuchinskas [9]. Once you decide on what kind of tool you want (data storage, interactivity for website, etc.), you can search for a tool that may meet your needs.

The next step is to determine the price you would want to pay for the tool. Most Web 2.0 tools are quite cheap or free (with limited features such as memory) to remain competitive in the market. Narrow down your choices depending on your budget and what each tool offers.

After you have a list of tools that meets your needs and fit within the confines of your budget, now is the time to try out each tool. Personally demoing the tools or implementing them within the environment you intend in using them in is a good way to see if they will work for you. Since Web 2.0 tools don’t require installation on your hard drive, they are easy to remove and add to your projects. [9].

A common way to discover and access web service tools is through a registry. Simply search through a registry of your choice and view the web service description language (WSDL) file provided by the web service. Using the WSDL, configure your own project to be compatible with the WSDL file and call the web service. [15]

Hopefully you now have a tool you are satisfied with in regards to what it offers, pricing,and if it works within the domain you plan to implement it in.

Conclusion

Service-oriented software has had a huge impact in the way we interact and use the Internet. As discussed, Web 2.0 is essentially the epitome of what service-oriented development is and the capabilities and limitations of this architectural methodology. The transition from Web 1.0 to Web 2.0 sets a great example of the features which were a direct result of service orientation, which included: interactivity and user-generated content, tagging, social media, etc. The emphasis on having the user as a part of the web (a producer) rather than simply a consumer

displays how due to service-orientation, current software on the web may be used in less linear or static ways.

What are also available now on the web are “mashups” which combine parts of web services (which individually, the parts are meaningless) to make something meaningful and useful. These parts of web services must be “loosely-coupled” in order for use in a mashup because of the properties of loose coupling which follows in tandem the principles of interoperable services in service-oriented software. We have observed in our report several different mashups and their uses: Wikipediavision, Trendsmap, MapsKreig, and SongDNA all display interesting features which are clearly defined by parts of other web services.

Finally, the process of publishing and consuming web services is fortunately simple; it involves managing WSDL files, registries, and other Web 2.0 tools. Service oriented development has made a substantial mark in the Internet’s progression and although some of the principles it follows limits some capabilities of software due to its structure, it is clear many things have improved. Our report has covered what service oriented development has impacted; it has provided the Internet’s ecosystem with richer services that provide content from users and commercial producers alike, a streamlined process of publishing and consuming these web services, and has established a well-received principle of loose coupling and interoperability.

References

[1] Wikipedia, the free encyclopedia.
Service-oriented architecture
Retrieved February 20th, 2013 from
http://en.wikipedia.org/wiki/Service-oriented_architecture
[2] Michael P. Papazoglou and Willem-Jan van den Heuvel (2006).
Service-Oriented Design and Development Methodology
Retrieved February 20th, 2013 from
http://arno.uvt.nl/show.cgi?fid=106517
[3] Nicolas Gold, Andrew Mohan, Claire Knight, Malcolm Munro (2004).
Understanding Service Oriented Software
Retrieved February 22nd, 2013 from
http://www.dur.ac.uk/malcolm.munro/papers/docs/IEEE-Software-2004.pdf
[4] Margaret Rouse (2011).
Definition: Loose Coupling
Retrieved February 20th, 2013 from
http://searchnetworking.techtarget.com/definition/loose-coupling
[5] Michael Poulin (2009).
Business Ecology Initiative & Service-Oriented Solution
Retrieved February 21st, 2013 from
http://www.ebizq.net/blogs/service_oriented/2009/02/togaf_90_is_short_on_soa_part_3.php
[6] Duane Merrill (2009).
Mashups: The New Breed of Web App
Retrieved February 21th, 2013 from
http://www.ibm.com/developerworks/web/library/x-mashups/index.html
[7] David Linthicum (2007).
Enterprise Mashups Meet SOA
Retrieved February 23rd, 2013 from
http://www.infoworld.com/t/applications/enterprise-mashups-meet-soa-337?page=0,2
[8] Wikipedia, the Free Encyclopedia
Mashup (Web Application Hybrid)
Retrieved February 24th, 2013 from
http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
[9] Susan Kuchinakas (2007).
How to Use Web 2.0 Inside Your Company
Retrieved February 21th, 2013 from
http://www.cbsnews.com/8301-505125_162-51066093/how-to-use-web-20-inside-your-company/
[10] Amy Erin Borovoy (2012).

Five-Minute Film Festival: Learn to Use Web 2.0 Tools
Retrieved February 20th, 2013 from
http://www.edutopia.org/blog/film-festival-technology-tool-tutorials
[11] ComputerWeekly.com (2008).
In-depth: How to use Web 2.0 at Work (Next-Generation Enterprise IT)
Retrieved February 23rd, 2013 from
http://www.computerweekly.com/feature/In-depth-How-to-use-Web-20-at-work-Next-generation-enterprise-IT
[12] Valerie Venezia (2007)
You Too Can Use Web 2.0
Retrieved February 22nd, 2013 from
http://www.slideshare.net/valeriev/you-too-can-use-web-20
[13] WebTools4U2Use, AuntyTech (2013)
Finding The Right Tool
Retrieved February 21st, 2013 from
http://webtools4u2use.wikispaces.com/Finding+the+Right+Tool
[14] Ian Foster (2005)
Service-Oriented Science
Retrieved February 20th, 2013 from
http://www.jstor.org.ezproxy.lib.ucalgary.ca/stable/3841930
[15] Membrane SOA Registry (n.d.)
Retrieved February 18th, 2013 from
http://www.membrane-soa.org/soa-registry/