Issues in Implementing Service Oriented Architectures

J. Taylor[1], A. D. Phippen[1], R. Allen[2]

[1]Network Research Group, University of Plymouth, United Kingdom
[2]Orange PCS, Bristol, United Kingdom
email: andy@jack.see.plymouth.ac.uk

Abstract

The growth of business to business interaction on the Internet can be seen to evolve from EDI, through Internet EDI, to the current state of the art – Service Oriented Architecture. Based upon accepted standards, web services provide a platform for the implementation of such architecture. Systematic evaluation of the implementation of a test service oriented architecture highlights both benefits and problems with the current technology. When considering the emerging second generation standards the lessons learned from this study are still entirely relevant.

Keywords

Service orientation, web services, evaluation, EDI

1. Introduction

This paper considers the implementation of web services into a Service Oriented Architecture and the implications for the software developer. Such approaches are currently generated a great deal of interest as potential solutions to such areas as business-to-business and enterprise application integration. While there is a great deal of promise with these approaches, there is little systematic study of their impact upon the development process. This paper considers the evolution of service oriented approaches before proposing a test scenario whose implementation was evaluated. Assessment of the development process and technological issues are presented, and conclusions are drawn considering the potential for the future of such approaches.

2. Business Integration and Service Based Solutions

The ability to develop viable value-chain partnerships and realise their potential in the shortest possible time is critical to the long-term success of commercial organisations. There have been many technologies used to address this need. Before the emergence of the Internet, the most prominent technology used for business integration solutions was EDI. The Electronic Data Interchange (EDI) standard was developed around the 1960's with the intention to automate routine flow of documents between various business support activities (UNEDIFACT, 2004) . EDI works by providing a collection of standard message formats and element dictionary in a simple way for businesses to exchange documents via electronic messaging. While EDI brought several benefits such as a reduction in manual processing of B2B documents, there were also several shortcomings. Companies wanting to adopt EDI required specific middleware and resources to maintain EDI links with their partners.

The Internet introduced a new integration model called Internet-EDI (Scheier, 2003). Web based EDI was more popular and financially accessible to medium sized and smaller firms. An example of this is Wal-Mart's successful implementation of its own Web-based EDI (0. The US supermarket giant had increased trading with smaller firms who were now able to manage the set up costs associated with Internet-EDI. In spite of this, companies still had difficulties exchanging data between them. There was no standard for document exchange formats and each marketplace had adopted their own translation software and exchange formats.

The multi-vendor XML Web Services initiatives brought the concept of integration over diverse networks into reality (Chatterjee & Webber, 2004). The success of these initiatives lies in the fact that all specifications are built on XML; a metadata language that can be transported and used by any hardware platform or delivery device irrespective of the operating system or programming language. Organisations have already been incorporating first-generation Web Services technologies into their application integration projects. While some of these projects have refined their existing enterprise architectures, other firms had redesigned their systems around a Service Oriented Architecture (0.

Within the Service Oriented Architecture each software interface is seen as a self-contained service that maintains its own state, can be dynamically located and invoked in a service repository and has a platform-independent interface contract that can be used by any application.

These functions include the provision of a service contract and the provision of a service repository for locating services at runtime. The service contract must specify:

- the implementation of the communication channel that can be used to interact with the service;
- the supported data representation format;
- the kind of messages that the service can consume or produce;
- a detailed schema for each message involved in an interaction between the service and service consuming application.

The service repository must provide both the repository and searching functionality required to locate pointers to services that are based on several business criteria such as business process type, industry type, business name and more.

3. Studying a Service Oriented Approach

The potential for service oriented approaches is well documented (for example, Papazoglou & Georgakopoulos, 2003; 0). This study tests that potential through the design and implementation of a real world application requiring a service based approach. The test system was specified by a major UK mobile telecommunications provider which wished to evaluate the feasibility of offering personalised and location based services over mobile devices. Their main interest was based around the ability to integrate new services into their existing portal services at a minimum cost, minimum integration complexity and rapid launch to the market. The aim was to design an open architecture that invites service providers to supply commerce services to its customers via their existing portal. The architecture would

supply all the smart processing in resolving the customer's location and in discovering the right services that match the customer's needs.

The evaluation of the approach was participative in nature based upon a number of broad operational measures:

1. Web service technologies provide the means to incorporate a service oriented approach into an existing infrastructure
2. Using a standards based approach removes issues of platform interoperability
3. Web service technologies are mature enough to develop a service oriented architecture

Evaluating these measures was generally qualitative in nature, based upon developer observation. While generalisation from a single case study is not possible, the findings of the study are presented here as "lessons learned", which will hopefully contribute to further study on approaches to the implementation of service oriented architectures.

3.1. The Test System

The test system (the Universal Commerce Services Management System - UCSMS) needed to provide the means to manage services that could potentially be offered to a customer based upon their profile and location. It also needed to provide an architecture that could easily integrate new services offered by third parties. Any services published with the open registry of commerce services adhere to an agreed set of service interaction standards. These standards consist of agreed data formats, supported services and classification schemes.

The development and implementation of the designed system was carried out using Microsoft tools and technologies. The application was then deployed on a Windows 2003 Enterprise Server. The delivery of content/services was based on two forms of customer profiles. The static profile was built from the customer subscription of services and interests. The dynamic profile will be generated from the discovery of the customer's location – for the purposes of the prototype implementation, location was determined from randomly generated co-ordinates, rather than a live location based system.

Most of the functionality required by this system was be built and deployed within a homogenous environment. However, partner services did not have to be developed for this specific platform. This increases the overall opportunity to engage into new businesses partnerships and increase revenue from new service offerings to mobile consumers. As the desired system had to rely on heterogeneous and platform agnostic service interfaces, service oriented offered the ideal solution. Figure 1 presents a high level representation of the UCSMS system architecture.

The UCSMS system is split into two main logical sub-systems. The first sub-system is the main broker application which is made up of customer account services, subscription management and personalisation management. The second system includes the handling of communication and data exchange interfaces with third-party services and components. These interfaces entail partner commerce services, integration with mobile portal and interfacing with external services such as payment processing.

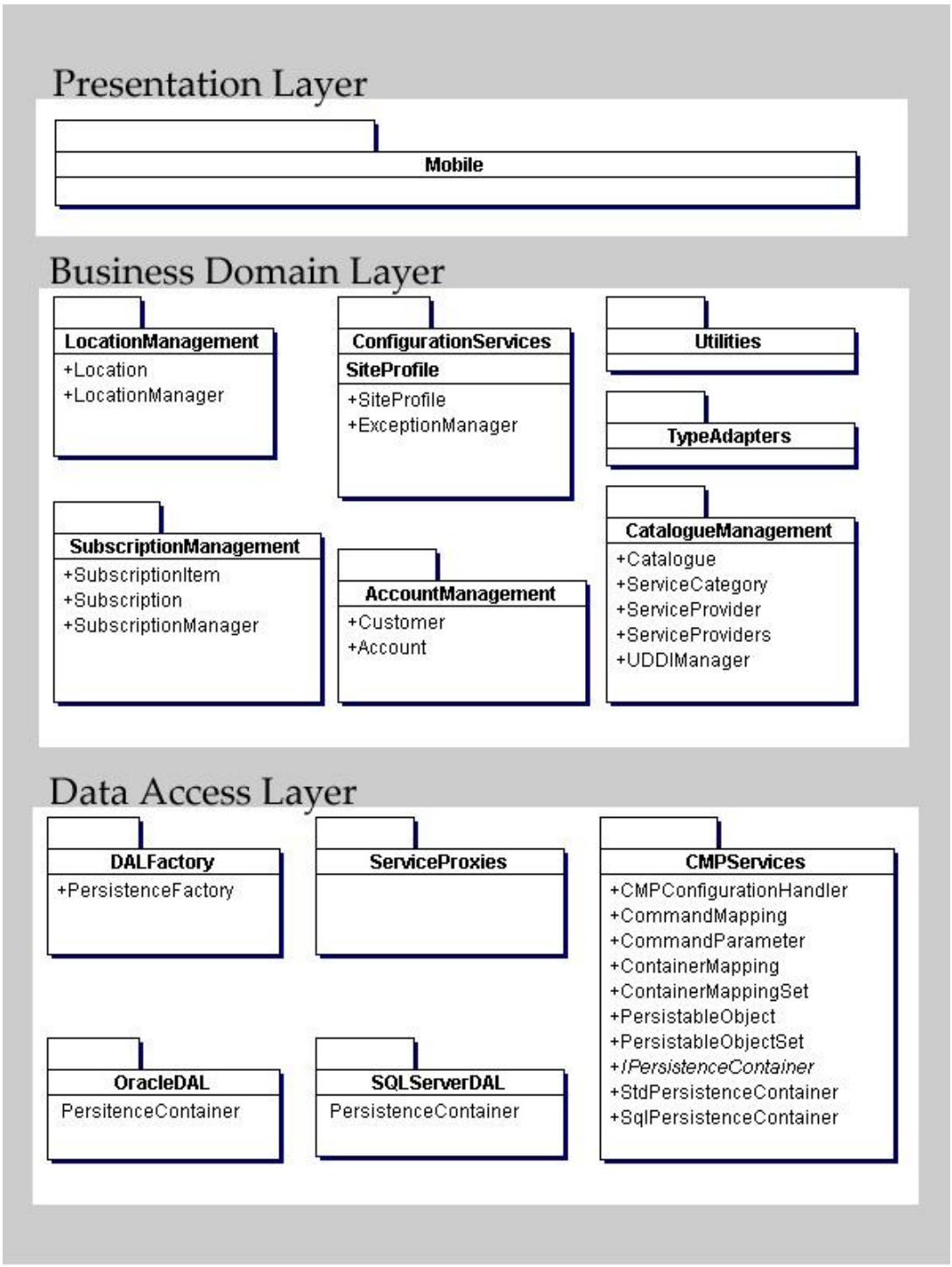

Figure 1 - UCSMS Overall System Architecture

The Service Agent components provide the access to external services and application via service proxies. These components will encapsulate the interface, protocol and coding details required to use each service. Equally, the Service Interface components will expose some of the important functionality that will be required by other internal applications. These service interfaces are implemented using XML Web Services. In order to demonstrate third party integration to the service contract approach, a test scenario was specified – a gift shop service that would provide basic catalogue browsing and product ordering through a defined

interface. This demonstration service was then implemented to simulate three different vendors all providing their business products through the standard contract.

3.2. Implementing Service Discovery - Public vs. Private UDDI Registries

UDDI registries can either be public, currently called UDDI Business Registry (UBR) or private (UDDI Organisation, 2000). The UDDI Business Registry is a free open distributed service that can be used by the general public and provides a simple registration process for publishing business services. Anyone can browse a UBR either programmatically or through a web browser. All public node operators have to comply with all the latest UDDI specifications on how to manage their nodes, UBR node replication procedures, and accessibility to programmers' API to ensure the integrity and availability of the information provided.

The main disadvantage of using a public registry for building enterprise applications is that UBR suffers from a lot of stale data. The UBR allows any public user to register with the registry and publish business entities and services that could possibly be phoney. Private registries do not have this problem. Most private registries will have to either implement an instance of a vendor specific UDDI product or create their own UDDI registry. Using a vendor's UDDI product can bring some additional benefits. For instance, Microsoft's UDDI Server provides a set of language/platform specific APIs to access the UDDI Server private registry. These APIs can be used to access any public or private registry as long as the developer knows the registry's inquiry and publishing access points. For this reason, the test system was implemented with a private registry – this worked well with the portal approach. Users of the system did not locate services themselves, the worked within the portal that did the service location for them based upon their static and dynamic profiles.

4. Evaluating a Service Oriented Approach

The aim of this study was to design and implement a partial solution for discovering and delivering location-based services in real-time in order to evaluate the technical issues in implementing a service oriented architecture. The solution has demonstrated that the selected technologies were mature enough to deliver such a system. Nevertheless there were also some additional intricacies that had to address specific issues with some of these technologies. The remainder of this study details both the benefits and problems of carrying out a practical implementation a service oriented approach.

4.1. Benefits of designing Service-Oriented Applications

Web Services technologies must be a strategic part any newly designed enterprise applications. This study has demonstrated that using Web Services does not require entirely new application architecture. In fact, one could say that web-service applications are an extension of component-based applications. Web Services can be easily created in a component-based application by simply adding application proxies to the functionality already offered by the application's components. The benefit of this approach is that if there are any significant risks with the usage of web services, one can still revert back to the component-based architecture without too much impact on the overall system. The same component classes can still be used in a non-service environment. Properly designed service

interfaces takes advantage of existing systems in an extremely cost-effective manner without having to invest in migrating existing applications.

4.2. Interoperability conformance

One of the foremost benefits of service-oriented environments is the intrinsic potential for immediate and future interoperability. Any web-services based system cannot claim to be a service-oriented application unless it measures up to the Basic Profile 1.0 Interoperability standard (0. Although the system was only tested and developed on a Microsoft platform, any data returned by the web services was strictly based on XML Schema standards.

4.3. Limitations of Microsoft UDDI Server

The solution used Microsoft's UDDI Services that is shipped with Windows 2003 Enterprise Server. Apart from some initial deployment issues with installing a UDDI Server on a Windows 2003 Server, the study discovered three main weaknesses in the standard design of UDDI services.

Firstly, the publishing process for registering a service provider and their services was not robust enough. The central issue with UDDI publishing is that service providers have no way of knowing what to information to publish unless there is a documented process on the information required by each process. Furthermore, the GUI-based interface does not force Service Providers to carry out each step in a sequential and logical order. Therefore, the same issues highlighted in section 3.2. about stale data in public business registries could still be encountered in this private open solution. Hence, either a customised user interface or a contract binding publishing guidelines ought to be provided to guarantee a stable UDDI based solution.

The other drawback is related to the latency of discovering service providers and their services due to the lack of support for nested queries. In order to find a service based on location and service category, the user needs to query the registry twice to obtain both business details and service binding details. Application testing recorded a time of, on average, 11 seconds to retrieve a service search on a local machine within the same local area network. Alternatively, one can design a custom component that directly querying the underlying database using SQL rather than the UDDI Inquiry APIs.

4.4. Customisation of service contracts

During the design of service contracts, some service operations required complex type parameters, i.e. a type made out of several simple types. However, when tested in the default Web Service testing page, the service operations that requested a complex type disappeared from the list of operations. To resolve this anomaly, one technique used was to submit the complex type as individual simple types. While this technique worked fine during testing of the web service via the default interface and the service proxy testing, there was still the ambiguity of why a service would not accept complex types as parameters. One method that succeeded resulted in a service operation accepting a complex type was to declare the complex type as a class within the assembly. Once this was carried out, the service could find both the schema and a description of what this type was composed of.

4.5. Completing the Business Model

The bulk of this study was focused on solving the technical design and particulars of a service-oriented system for discovering and delivering location-based services. The remaining business requirements will have to be addressed in a production environment specific to the particular business model. Nevertheless, this study has highlighted key business concerns that must be managed to deliver the full benefits of these technologies.

The system will only be successful if the services provided by the service providers are reliable and robust enough to handle the expected volumes. These non-functional requirements need to be carefully planned and documented as part of the overall deployment and maintenance strategies.

Managing effective customer relationships is essential to every business. Firms can create new market opportunities and enhance business value by understanding how their customers use their services. This can be achieved by tracking customer usage and develop consumer models through data mining from the data collected. Furthermore, adopting flexible service charging options encourages customers to try out other services. Such a scheme would also facilitate the creation of added value services by refining existing products and services offered.

Like the Internet, XML and Web Services technologies are progressively becoming the baseline of competitive necessity. As the emphasis on cost cutting, increasing value and managing risk increases, more firms are looking at Web Services technologies as a solution to these concerns. Moreover, many forward thinking companies have already found new ways to use these technologies to support business processes across organisations. Interestingly, the big software and hardware vendors already have an assortment of products that provide the core services for business process management. Looking back at integration during the days of EDI, the same issues of software, hardware and other resources emerge once again. However, with Web Services technologies, most of these resources already exist within the firm. Companies can reuse and recycle existing resources and infrastructure to incorporate these technologies within their architecture. While the level of confidence, skills and experience increases, firms can start opening the door to new business models that expand on B2B integration and business processing orchestration.

Despite the benefits that Web Services technologies bring to a business, they cannot solely provide a holistic solution to service provisioning and service management. The promotion of collaborative efforts and the emergence of partnerships and alliances is what ultimately deliver the real value.

5. Conclusions

This study was carried out in order to assess the implication for the developer in implementing a service oriented architecture. The underlying web service technologies used to implement such an approach undeniably provide a solid foundation. However, despite the generally positive outcome, Web services technologies are far from mature. For instance, there was no evidence of any cases on how to dynamically bind to a Web Service at runtime throughout the lifetime of this study. Similarly, there was a lack of case studies on orchestration of Web Services. Some of these themes and other operational matters covered in

this evaluation section highlight the importance of further examination and testing before embarking on developing large-scale service-oriented projects.

The focus of this study was on the delivery of single isolated Web Services. The real value is delivered when multiple Web Services are integrated into larger and useful composite services. Equally, business enterprises will discover the real value of a service-oriented system when there are opportunities to use Web Services to improve business workflows and increase operational efficiency.

A recent XML standard has been drafted and targeted to solve these problems. Business Processing Execution Language (WS-BPEL) is the standard used to describe the orchestration of Web Services-based end-to-end business processes (0. This standard was introduced in May 2003 (as BPEL, and renamed WS-BPEL in 2004) and major software vendors have already launched products supporting the WS-BPEL standard. While this standard might turn out to be the cornerstone of the ultimate SOA implementations, the same basic techniques and practices covered in this study will still have to be taken into account.

6. References

Chatterjee, S. & Webber, J. (2004), *Developing Enterprise Web Services*, Prentice Hall, NJ.

Erl, T. (2004), *Service Oriented Architectures*, Prentice Hall, NJ.

Lau, D. (2004), "Build Web sites with BPEL business processes", IBM Corporation, http://www-106.ibm.com/developerworks/edu/ws-dw-ws-buildweb-i.html, date accessed: 19th July, 2004.

Microsoft (2004), "XML Web Services Overview, .Net Framework Developer's Guide", http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconwebservicesoverview.asp, date accessed: 28th January, 2004.

Papazoglou, M. & Georgakopoulos, D. (2003), "Service Oriented Computing", *Communications of the ACM*, 46 (10):25-28, October 2003.

Scheier, R. (2003), "Internet EDI grows up", Computerworld, http://www.computerworld.com/industry/topics/retail/story/0,10801,77636,00.html, date accessed: 25th February 2004.

UDDI Organisation (2000), "UDDI Overview Presentation", http://www.uddi.org/pubs/UDDI_Overview_Presentation.ppt, date accessed: 2nd February 2004.

UNEDIFACT (2004), "EDI Standards Introduction", http://www.unedifact.com/edi/edi_2.htm, date accessed: 18th February 2004.

Web Service Interoperability Organisation (2004), "Basic Profile v.1". http://www.ws-i.org/Profiles/BasicProfile-1.1-2004-08-24.html. Accessed 30th January 2005.

Wilson, T. (2002). "Wal-Mart Cuts the VAN Out of EDI". Network World Fusion. http://www.nwfusion.com/newsletters/asp/2002/01560820.html. Accessed 30th January 2005.