Reengineering Legacy Applications for SOA Middleware Integration

Moyo Takonewa, Thinyane Mamello and Kogeda Paul
Department of Computer Science
University of Fort Hare, P. Bag X1314, Alice 5700
Tel: +27 40 6022464, Fax: +27 40 6022464
emawil: {tmoyo, mthinyane, pkogeda}@ufh.ac.za

Abstract-With Service Oriented Architecture becoming contemporary phenomenon in the software industry, it is imperative for such architectures to be extended to Information and Communication Technologies for Development contexts. For Marginalised Rural Areas, such solutions need to be drawn out with due regard to legacy applications and should be adapted for smooth transition.. For numerous legacy applications, integration with current architectures would require an adaptor to interface with SOA middleware. Thus this work looks into adapting multiple web applications to a Web Services implemented SOA middleware.

Index Terms—SOA, ICT4D, MRA, SOAP, WSDL,

I. INTRODUCTION

Candela et al noted that new architectural approaches are emerging or being consolidated and these approaches provide simplified implementations [1]. Additionally, they offer a number of new possibilities for implementing novel user functionality. In this respect, to offer an enhanced quality service to Information Communication Technologies for Development (ICT4D) interventions, we need to embrace these current architectures. An equivalent present architecture, Service Oriented Architecture (SOA) and is widely implemented through Web Services. Wertlen has also proposed a design for such an SOA-based ICT4D services middleware and highlighted how such distributed solutions can enhance the overall quality of ICT4D interventions[2].

Dwesa, a Marginalised Rural Area (MRA) in Eastern Cape, has had some ICT4D activities in the past. These were championed by the two universities of Fort Hare and Rhodes. The institutions run the project under the banner of Siyakhula Living Lab (SLL). As part of the evolution of the SLL services platform and to offer improved user functionality to the community we need to embrace the current architectures.

II. RELEVANT APPROACHES

Though Web Services have been widely used as the implementation of SOA software design paradigm, little has been done to integrate legacy applications with Web Services middlewares.

Upakare at the Open Source Software Resource Centre had attempted to modify Parikshak, a learning management system (LMS) in a bid to make its functionality exposed through Web Services [3]. In their approach, the emphasis was on modifying the legacy application, Parikshak, and a C++ Simple Object Access Protocol (SOAP) implementation, called gSOAP, which was used to expose the functionality through the web service.

Erlikh in his article highlighted that typical legacy systems are composed of many independent programs that work together in a hard-wired cluster of business process flows [4]. He proposed the regrouping of common functions of legacy application that can later be exposed to the middleware using relevant SOAP implementation.

From the above two experiences, it is obviously inevitable to modify legacy applications. The attempts, each of them focused on a single legacy system, which is logical to modify and/or re-program some functionalities. The case differs when we are faced with multiple legacy applications, as is the case with currently deployed SLL eServices in Dwesa. In this case a pluggable-adaptor may suffice in which case basic functionalities are abstracted to the adaptor. Modifying each application would require more time and further, if applications could be grouped with each group containing common functionalities, then an adaptor would be ideal.

III. ADAPTOR FIT-IN

A. Adaptor Model

Obviously, for enabling requests proxying between legacy applications and the middleware, the adaptor has to use more than one mechanism. To send and receive requests and responses from the middleware SOAP would be required. Figure 1 below depicts the model of the adaptor.

Labels 1 to 5 on Figure 1 are a sequence of steps required to send and receive requests between the middleware and the legacy application through the adaptor. For each adaptor, its interfaces are defined and Web Services Description Language (WSDL) is used. Of particular note is the communication between the adaptor and legacy applications, in this case depending on the location of the adaptor, direct method calls or a suitable protocol may be used. Assuming that Model View Controller (MVC) was used in the development of the current eService, method calls will be possible and the adaptor can be used in different eServices with minimal modifications.

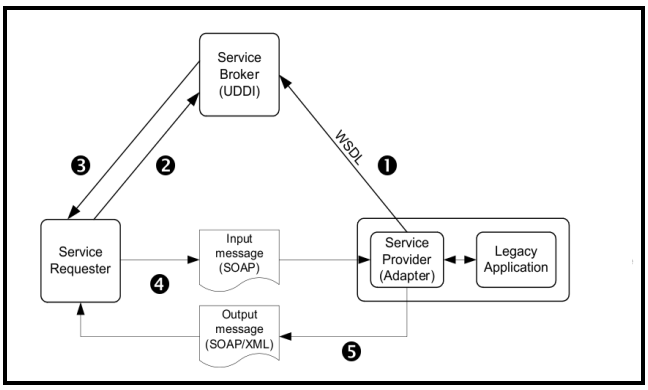

Figure 1: Adaptor Model [5]

B. *Adaptor Structure*

For complex legacy systems (e.g. eCommerce) the adaptor needs to have an enabling structure, the components that allow it to carry out expected functions. Figure 2 below is the proposed structure of the adaptor.

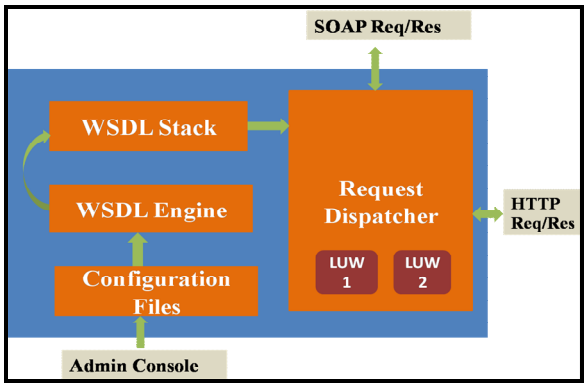

Figure 2: Proposed Adaptor Structure

1) *Request Dispatcher*

The request dispatcher is the core of the adaptor and is responsible for handling the requests from either side and serve responses in compatible format. It may receive SOAP request and then executes direct method calls and then serve SOAP responses.

2) *Logical Units of Work (LUW)*

The request dispatcher itself is composed of smaller modules which may be the regrouped methods of the legacy applications and/or a set of predefined method calls to fully execute a business process as has been discussed by L. Erlikh [4].

3) *WSDL Engine*

This component serves to generate required WSDLs for each SOAP Server within the adaptor. The idea behind is to allow flexibility in modifying the adaptor for different legacy systems. The WSDLs are generated from simple text configuration files and stored in WSDL templates.

4) *WSDL Stack and Configuration files*

The WSDL stack is just a collection of WSDL templates and complex data schemas. Configuration files contain some configurations for particular legacy application. Both are used by the WSDL engine to generate custom WSDLs dynamically.

IV. METHODOLOGY

Since the main focus is to develop a reusable adaptor, modification of the legacy applications will only be done as the last resort. The development will be through programming each component, and the whole adaptor will be prototyped vertically. This is to allow the testing of the whole adaptor's functionality and modify the structure if needed. The adaptor becomes part of the new legacy application, which allows repackaging of the application. The adaptor will be a drop-in folder and its calling will be referenced relative to the legacy application's URL.

V. CONCLUSION

While sustainability of ICT4D projects is an issue, reuse of software components should also be seen as a driver towards both social and financial sustainability. Reuse of software components means low learning overhead (i.e., for MRA dwellers) and each cent spent on developing a software application will find its way back to proper use. Adaptors are therefore a key mechanism for integrating legacy systems into SOA architectures, especially in ICT4D contexts.

VI. REFFERENCES

[1]. L. Candela, D. Castelli, P. Pagano, and M. Simi. *Moving digital library service systems to the grid. Peer-to-Peer, Grid, and Service-Orientation in Digital Library Architectures*, pages 236_259.

[2]. Ronald Wertlen. A Design of a Middleware Solution for Connected Rural Digital Access Nodes Enabling a Multitude of Applications. Master's thesis, University of Fort Hare, January 2010.

[3]. R. Upakare and M. Sasikumar. Converting Legacy Application to Web Services using an Open Source Toolkit: A Case Study. *FOSLET 2006*, page 29.

[4]. L. Erlikh. Integrating legacy systems using web services. *eAI Journal*, 8,2002.

[5]. Laura Koetzle Ted Schadler. *Integrating Legacy Applications as Web Services*. White paper, May 2003. Available online (9 pages).

Moyo Takonewa received his undergraduate degree in 2008 from the University of Fort Hare and is presently studying towards his Master of Science degree at the same institution. His research interests include web services, ICT for development, IM and Presence and mobile development.