ISSN:0975-9646

A Comprehensive Framework for Reengineering Method for Web services architectures

Venkatsampath Raja Gogineni[1], Dr.Syed Umar[2]

[1]*B.Tech Student, ECM dept., K L University, vaddeswaram, Guntur, AP.*
2Assoc.Professor, ECM dept., K L University, Vaddeswaram, Guntur, AP

Abstract: The web services technology and service-oriented architectures are quickly developing and extensively supported. In particular through the Web services framework have greatly enhanced the flexible and interoperable implementation of service-oriented software architectures. But, it is fairly difficult for existing web applications to expose functionality as services in a service-oriented architecture, because when web applications were built, they served as monolithic systems. In this paper we describe an advanced e-learning system as our case study, we consider central aspects of a re-engineering approach for the Web services platform. While our aim is to provide components of the legacy system also as services in the new platform, re-engineering to suit the new development paradigm is as important as re-engineering to suit the new architectural requirements.

Keywords: Software Re-engineering, SOAP Web services, Web Services Framework, web services.

INTRODUCTION

The World Wide Web is rapidly being adopted as the medium of collaboration among organizations. Web applications are today legacy systems, which constitute valuable assets to the organizations that own them. A web application is an application delivered to users from a web server over networks such as the Internet or an intranet. Web applications are popular due to the ubiquity of the web browser as a client [1].

Performances aiming to develop the Web from a document- to a services-centric environment are bundled in the Web Services Framework WSF [2]. In the WSF philosophy it is to open the Web for application-to-application use. It provides a framework based on description languages to describe services, a publication and discovery facility acting as a marketplace for providers and users of services, and protocols allowing services to be invoked in a distributed environment.

Through the Web services technology becoming mature, it will impact existing Web-based and other distributed systems. Several software systems will be re-engineered for the Web services platform in the future. Re-engineering these systems as service-oriented architectures SOA will become achievable.

We investigate central aspects of a re-engineering method for the Web services platform, Web services framework deployed as a service-oriented architecture poses new architectural constraints on these systems [3]. But we found that the new development style associated with the Web services framework also influences the re-engineering approach. Preparing an existing software component as a service not only for later deployment in a service-oriented architecture, but also as a software entity that can be made available through the WSF description and discovery mechanisms, is a central requirement that a re-engineering method needs to embrace. We will use an e-learning system as our case study here to illustrate our findings [4]. This system is a Web-based, distributed system that is characterised by complex interaction processes.

Case Study an E Learning System:

Nowadays, the Web is the predominant platform for computer-supported teaching and learning. The latest developments have seen more interactive media among the Web resources, allowing interactions between human users and provided services and also interactions between the services themselves. We have been involved in the development and extension of a Web-based teaching and learning environment called IDLE [4] – the Interactive Database Learning Environment.

The IDLE is a Web-based integrated learning and training environment for an undergraduate course programme. This system provides a wide range of educational features from audio-supported lectures and animation-based tutorials to active learning features for genuine database development. Several of the components are audio server and audio player, animation and simulation features, a graphical modelling tool, an execution tool for a database language, a workspace feature for student projects, a content repository, a feedback feature, an evaluation tool, and a Web server and other delivery software. The complexity and diversity of the components requires architectural support in order to develop, extend, and maintain the system. An explicit interface between components has the past supported extension and maintenance of the system [5].

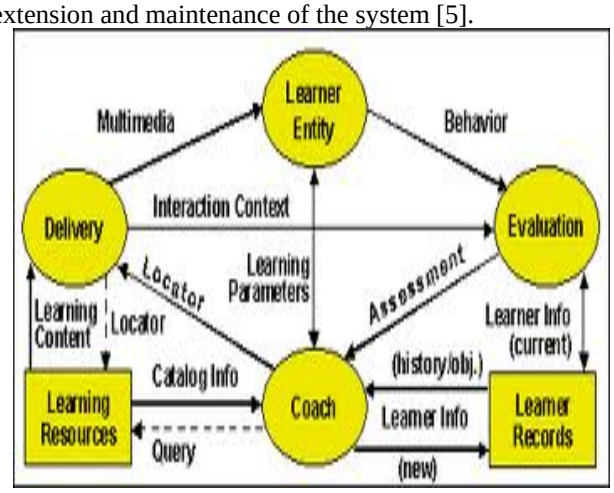

Figure – 1: Learning Technology Standard Reference Architecture (LTSA).

The Learning Technology Standard Architecture LTSA has been developed by the IEEE Learning Technology Standards Committee LTSC in order to provide a framework for the development, evaluation and discovery of learning technology systems. It provides a basic architecture consisting of process and storage components and the interactions between them. LTSA identifies process components such as coach, learner entity, evaluation, and delivery, and storage components such as learning resources or learner records as shown in Figure -1. The objective is to enable sharing and reuse of learning technology components [6].

The main objective here is to explore architectural aspects of re-engineering – illustrated in the context of Web-based teaching and learning environment (TLE) design and development. Service-oriented architecture (SOA) shall form the backbone of the re-engineered system. The notion of learning objects is central in the case study context. It comprises content units but also functional components such as delivery or storage [7]. The context of educational technology is a learning object in a self-contained learning resource or function that is both a unit of retrieval and a unit of assembly. In the software engineering context, service notion is more appropriate, referring to an encapsulated, reusable computational entity. Services are similar to components in that their properties are described in form of interfaces separated from the executable entities. While composition is an issue here, we expect services to have component character.

SOAP-Based Web Services

A web service is a technology commonly used to implement SOAs. Service interfaces are defined in the WSDL language, and service users and service providers communicate using the SOAP protocol. Two attributes in a WSDL interface, “style” and “use,” define the SOAP communication between service users and providers. The style attribute has two possible values: “RPC” and “document.” The use attribute refers to data encoding and has two possible values: “encoded” or “literal.” Consequently there are four possible combinations of these two attributes. Two combined options that are common in practice are RPC-encoded and document-literal. They are described next.

RPC-Encoded SOAP

In the RPC style, the SOAP message is equivalent to an XML-based remote method call. The name and type of each argument is part of the WSDL interface definition. The body of the SOAP request necessarily contains an element indicating the operation name and sub-elements corresponding to the operation arguments. The encoded attribute indicates that data is serialized using a standard encoding format. This format is defined by the SOAP specifications and contains rules to encode primitive data types, strings, and arrays. Figure 2 represents an RPC-encoded interaction. The RPC-encoded style was popular in the first years of the Web services technology because of its simple programming model and the similarity between service operations and object methods. However, it is not a good choice, because interoperability problems can arise from deficiencies in the SOAP-encoding specifications [Ewald 2002].

Document-Literal SOAP

The SOAP message body in a document-literal style request can contain arbitrary XML (the business document). The WSDL definition does not have to specify named parameters, and the XML content of the message body does not follow a standard structure as in the RPC style. The literal attribute indicates that no standard encoding format is used—data in the SOAP body is formatted and interpreted using the rules specified in XML schemas created by the service developer. The XML schemas that define the data structure of the request and the response are the key elements in the interface definition. Figure 3 shows a document-literal interaction.

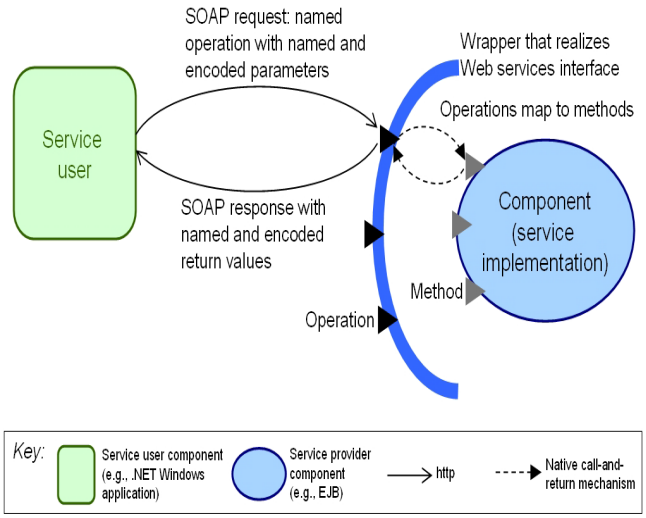

Figure -2. RPC-Encoded Interaction

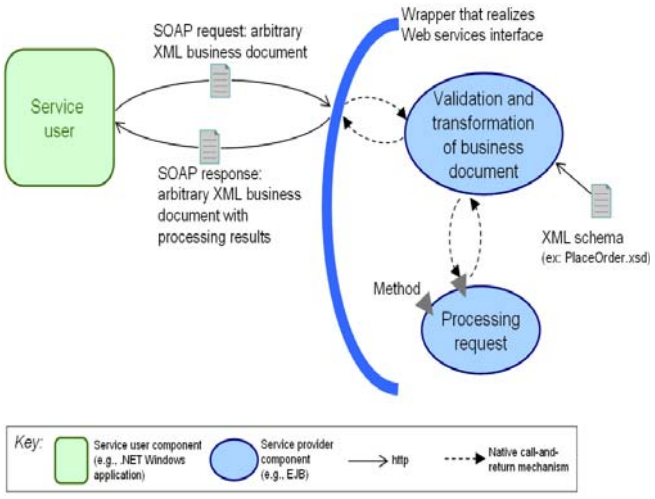

Figure-3. Document-Literal Interaction

The document-literal approach is recommended by the WS-I organization. In an architecture evaluation, the architect should be aware of the differences between these styles. Some Web services toolkits still use RPC-encoded as the default style; therefore, it is important that developers know how to specify the desired style when creating services [8].

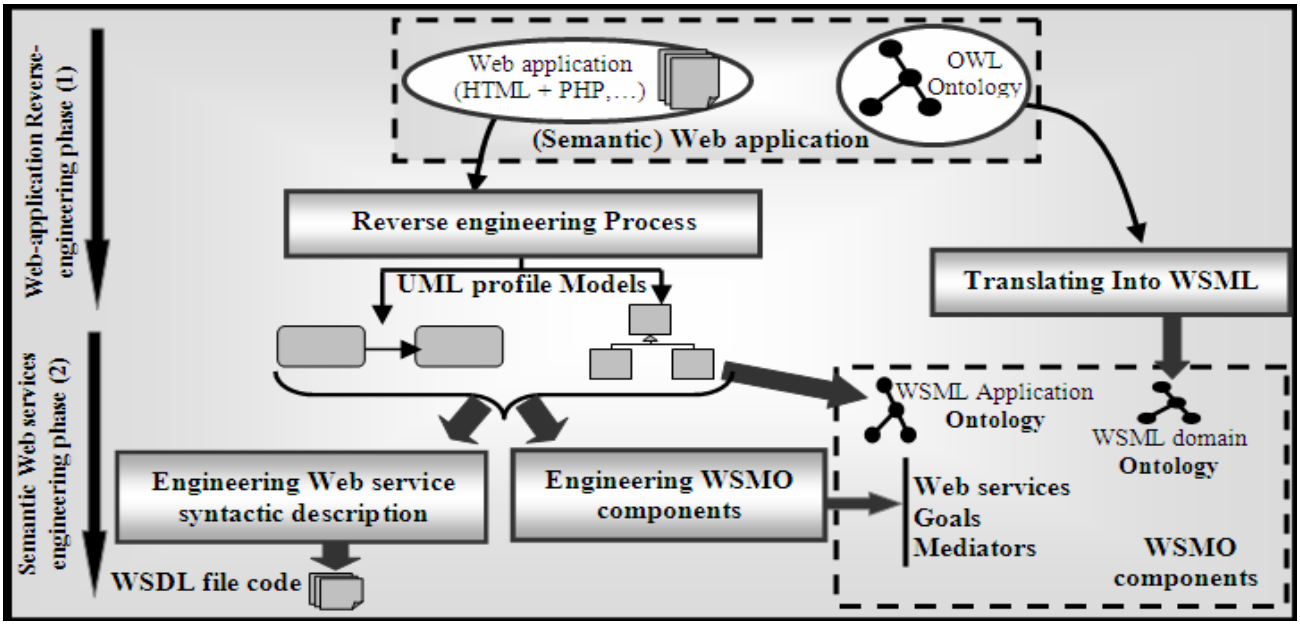

Figure 4: Proposed framework
(proposed framework consists of two major phases: web application reverse engineering and SWS forward engineering)

Web application reverse engineering phase:
Web application reverse engineering phase extracts a model based abstract view of the considered web application, starting from the available implementation. The generated models are expressed according to the proposed UML profile, and consider both static pages (pure HTML with no server-side scripting) and dynamic pages (including executable code at the server side). Reverse engineering aims at recovering design models (Class and Activity diagrams) from web applications code. The resulting models cover static and dynamic aspects of the application.
In our framework both class diagrams and activity diagrams are stored by means of an XML serialisation. The chosen variant for this task is XMI (XML Metadata Interchange). The main purpose of XMI is to enable easy interchange of metadata between modelling tools [9].

CONCLUSIONS AND FUTURE WORK

Reengineering is the process of analyzing a subject system to identify the system components and their interrelationships, and to create representations of the system in an improved or a new form. The work we have described in this paper focuses clearly on the latter goal. The basic idea underlying our work is to reverse-engineering the existing data driven web application to create representations at a higher level of abstraction. We have addressed units of software that have both service and component character. They can be annotated with metadata and discovered through searches based on this metadata. These units are also units of composition in the development of architectures for Web services-based systems. The notion of learning objects has captured this type of software unit in the case study context. Our experience with the prototype service discovery and reengineering system to date is by no means mature. We have collected some initial evidence on how such a tool might be deployed and our case studies suggest that the approach is indeed promising. In the case study we discussed in this paper, we were able to indeed discover interesting

REFERENCES:

[1] Michiaki T. and Kenichi T., "Decomposition and Abstraction of Web Applications for Web Service Extraction and Composition," *in Proceedings of the 2006 IEEE International Conference on Web Services (ICWS 2006)*, USA, pp. 859-868, 2006.
[2] World Wide Web Consortium. Web Services Framework. http://www.w3.org/2002/ws, 2003.
[3] World Wide Web Consortium. Web Services Architecture Definition Document. http://www.w3.org/2002/ws/arch, 2003.
[4] C. Pahl, R. Barrett, and C. Kenny. Supporting Active Database Learning and Training through Interactive Multimedia. In Proc. Intl. Conf. on Innovation and Technology in Computer Science Education ITiCSE'04. ACM, 2004.
[5] C. Pahl. Managing evolution and change in web-based teaching and learning environments. Computers and Education, 40(1):99–114, 2003.
[6] IEEE Learning Technology Standards Committee LTSC. IEEE P1484.1/D8. Draft Standard for Learning Technology – Learning Technology Systems Architecture LTSA. IEEE Computer Society, 2001.
[7] C. Szyperski. Component Software: Beyond Object-Oriented Programming – 2nd Ed. Addison-Wesley, 2002.
[8] Ewald, Tim. The Argument against SOAP Encoding. http://msdn2.microsoft. com/enus/library/ms995710.aspx (2002).
[9] Bouchiha, D. and Malki, M. (2012) 'A framework for reengineering web applications towards semantic web services', *Int. J. Web Science*, Vol. 1, No. 3, pp.180–203.