Reengineering Open Source CMS using Service-Orientation: The Case of Joomla

Tagel Gutema
tagelgutema@gmail.com

Dagmawi Lemma
Department of Computer Science, Addis Ababa University, Ethiopia
dagmawil@yahoo.com

Abstract

To fill the gap of the business requirements on an already existing open source product, reengineering can be performed to make it flexible and used by other systems. This can be achieved by using appropriate technologies and methodologies. Service-Orientation is an emerging technology that is currently being used as facilitation for reusability. Although service-orientation is applied for different types of systems, its application on open source CMS is questionable with respect to simplicity, time, and experience. This paper provides guidelines and prototypes in using service-orientation for this type of specific products.

The paper reengineers an open source CMS (Content Management System) after a thorough understanding of the open source system, in this case Joomla. The paper tries to apply service-orientation on Joomla which in turn enables to expose its component services for use. The methodology that is employed to reengineer the case is a SOSR (Service-Oriented Reengineering). SOSR involves basically on creating the 4+1 view models revealing the existing system to be reengineered in order to create a visual model. Once the existing system is understood and visualized, SOFSE (Service-Oriented Forward Software Engineering) is applied for service-orientation. To realize the service-orientation, the paper uses common technologies that facilitate SOA.

Keywords: Content Management System; Business Process Execution Language; Service-Oriented Architecture; Reverse Software Engineering; Service-Oriented Forward Software Engineering; 4+1 view; Service-Oriented Software Reengineering

1. Introduction

It is crucial to define the term reengineering before introducing the substance of this paper.

Software reengineering is defined in [1] as the study, examination, and modification of the internal system or functionality of an already existing system or product so as to reconstitute it in a new form or design with new features, often to take advantage of newly emerged technologies, but without significant or major change to the native functionally and purpose of the system. Similarly, it is explained in [2] that it is to first maintain the existing functionality, then prepare the existing system for the changes and finally add fresh technology.

Reengineering can be applied on an already existing open source product or software. Open source products could be one of the preeminent opportunities to use them as we find them freely and specifically we can consider open source Content Management Systems (CMS). Open source CMSs can be customized and used for several business needs and requirements. However, a single open source CMS may not satisfy the whole needs of an organization as it has a limited number of functionalities. Hence, looking for another open source module or component might be a solution to fill the gap of the business requirement. It is less likely to make different CMS components homogenous, as each may have been built to meet specific requirement which may diverse from others. Besides, each of these systems has specific objectives and are also implemented in different technologies.

To use open source product modules or components, one way is to apply service-orientation and expose them for reusability and this experimentation will be done in this paper. This can be achieved by using appropriate technologies and methodologies.

2. Statement of the Problem

Organizations always formulate a cost benefit analysis before making any investment on Systems. The investment analysis result may fall on developing a new system or reusing the already existing product that may be used with the old one or buying a new product. Considering the second option, i.e., reusing already existing products, is time saving. But, due to the nature of the heterogeneity of systems, making use of an existing system with less modification may not be an easy task. This could be an obstacle to meet the desired target within a short period of time.

In the old days applications were developed to address a specific business need and to add additional business requirements in the future systems to be developed should be flexible enough so that they can be able to meet frequently changing and new business requirements. The traditional methods of developing and deploying software systems is no longer sufficient due to the difficulties in developing a large number of software subsystems in a short period of time [3]. A new approach is mandatory to provide flexible system that could meet the frequently changing business requirements or needs of the time. In an effort to alleviate challenges in the software process, several models have been proposed and are evolving (such as scrum, extreme programming, etc.). However, addressing the dynamic nature of business requirements may not be solved by focusing on the software process; the type of software architecture applied also matters. Besides, heterogeneous components of different products may not easily work together with new requirements. Therefore, a new approach that makes system components easy and flexible is essential.

Open source CMSs are available in different forms and making use of these product components and modules directly is impossible due to their nature. Therefore, looking for a method or a solution that makes these products more flexible and accessible for reusability is essential.

3. Objectives

The general objective of this paper is to address the problems specified in the problem statement section with an SOA (Service-Oriented Architecture). “An SOA is an architecture that utilizes the core concepts of service providers and service consumers to define a system “[4]. SOA solves problems through the use of reusable components. In SOA, components are developed to solve a specific business problem. These components can be threaded, linked, or integrated into a specific configuration for the current business needs. When new business requirements arise, the system is flexible to be reconfigured [5].

The specific objectives of this paper are to:

- understand the basic principles of SOA and apply it an open source CMS pertaining to the principles of reengineering.
- enable service-orientation on an open source CMS, in this case Joomla
- introduce a guideline on how to enable service orientation on Joomla.

4. Methodology

Open source CMSs are being solutions for multiple business needs. They can be customized and used for several requirements.

SOA requires at least service reusability and autonomy to satisfy its fundamental principles. Most old systems are not implemented in the context of service orientation and hence reengineering them to comply with the SOA principles could be essential [6]. In this paper, we used the SOSR methodology and extended it for a service broker that composes a set of services as a composite service(s). The SOSR methodology has two phases - RSE and SOFSE. The former one is used to reverse Joomla so that we can visualize with the 4+1 view and RACI chart based on UML diagram and the later one is to model Joomla into the modern target system in the context of service-orientation [6].

5. Literature Review

This section describes the SOA realization in different perspectives from scientific points of view. The technologies described in this section are in detail defined in Section 5.1.

SOA is a new technology choice to integrate systems to improve business processes in an organization with the right tools, skills, and methodology to deliver reusable software applications [7]. In the 21st century, only few organizations are adopting SOA to meet their business needs. SOA is still in its infant stage and many researches are being made towards SOA. These days the word SOA is replacing the traditional middleware as the software technology shifting to service orientation. Dependent, inflexible, and incompatible systems are not suitable for modern IT business as the architectures are expected to be agile and have to respond quickly for the ever changing business needs [5].

A framework is proposed in [8] to enable SOA on CMS systems. This shows that the application of SOA may lead to use a framework to apply SOA. This paper applies SOA on open source CMS (not to create a framework).

Applying service-orientation on a CMS requires understanding the principles of SOA. SOA uses the basic principles of web services to impact service orientation. As web services are standardized across multiple vendors, they are the main implementation techniques for SOA. SOA is a concept that can be influenced by different frameworks and vendors as the web services influence SOA in shaping several of its principles including abstraction, service loose coupling, and service composability [9].

The core compositions of web service technologies - XML, SOAP, and WSDL - build the basic Web services architecture and through further Web service protocols that enable a qualified performance web service. Web services provide reliable, flexible, loosely coupled, and extensible middleware [10].

Analysis on service oriented architecture among different SOA solution vendors like Microsoft, IBM, Oracle, and Red Hat is carried out in [9] highlighting how and to what extent SOA can be achieved with open source solutions and what kind of competence is required for it. In addition to this, a comparison of different SOA solution vendors (both proprietary and open source) is made. The case was based on Vattenfall, which is one of the first five largest energy suppliers within Europe [9]. Having operational systems in different countries brings the challenge of integrating all these distributed systems and this integration is a vital requirement for Vattenfall which uses a proprietary product to integrate across different platforms but the research indicates that the company requires a better infrastructure to be extensible and cost effective. The result of the analysis would help the company to decide on SOA in most cost effective way [9]. To apply this, it requires a thorough understanding of SOA principles.

SOA governance is an essential factor to determine success of SOA [11]. The research result in [11] stated that strong governance and control were key factors to succeed with SOA.

It is shown in [12] that SOA is prominent to strategically align IT and business especially when implemented through the use of Web services. For example, while using HTTP, it is possible to employ widely adopted industry standards such as XML, SOAP, and WSDL which help to run an application on different platforms [12].

In [13], it is described that web service composition is an inevitable aspect to solve complex problems by combining available services and in [10] similar technologies that are being used in this reengineering project with SOA infrastructure are applied. In this paper, basic services are composed and selected as per the request and decision made by the orchestrator BPEL. In [14], it is tried that since it is difficult to compose web services manually; there has to be a mechanism to dynamically generate compositions.

Based on the technologies available and understanding of SOA principles and governance, there are possibilities to successfully apply service-orientation on a CMS system. Therefore, we would focus on reengineering and service-orientation on an open source CMS using the SOA principles.

6. The Proposed Solution

Before discussing the proposed solution, it is important to understand Joomla and how it works.

6.1 Joomla

Joomla is one of the open source CMSs which is released under version 2 of the GPL license.

According to [15], Joomla is one of the most popular CMSs which is proved by a number of awards and too many massive committees. Besides, it has large number of freely and commercially available extensions that help them to do more functionalities more than just simple content management.

There are only two entry points in Joomla: Frontend and Backend request. The Frontend request will only be instantiated through the root index.php and the backend administrator/index.php entry point. By doing so, Joomla minimizes security vulnerabilities as well. Basically, there are four operations - receive request and load libraries, initialize/build application, determine application route (determines selection of appropriate component and execute it), rendering the selected document, and respond appropriately.

It is important to make systems more flexible to service-orientation in this ever-changing environment. This means any software should be designed in such a way to facilitate easy reengineering. Fortunately, most open source CMS systems, including Joomla, are flexible enough to facilitate reengineering. Nevertheless, it is difficult to find a detailed design document made for open source CMS. Therefore, bringing the available source code into visual view requires a thorough study and time. Thanks to the Joomla community coders, most of the classes and methods are documented which, at least, help for code readers. Getting the proper location of component code execution initiation and return point is an essential part of the code where we need to make a focus. Determining the appropriate location where the client should be put, i.e., it could be on the front of the Joomla or in the place where it divides the Joomla into service provider and service consumer that means the service orientation will divide the system into different parts. This does not mean the “site” and the “administrator” as Joomla divides.

Irrespective of any architecture, it is possible to expose each component into the service consumer. The decision where the client code (in this case the SOAP client) should be put requires if the service provider is independent and properly designed to be consumed. Although, how to put the SOAP client and where to put it is decided by the designer. In this paper, it is put where the dispatcher starts to look for the requested components because we want to expose the components for service-orientation.

Joomla prevents direct access to the system. That means it only allows a single point of entry. This is done for the sake of security, especially to prevent a directory traversal attack. But, in this paper the orchestrator is accessing outside the system and Joomla, in its original form, does not allow the service provider. In this paper, it is located in the server where Joomla is located. Hence, this requires breaching the security. In this case, for example, the RACI chart will show the responsible and accountable person in the project. Based on the accountability, the responsible person shall consider the vulnerability to lock it or to put it in the appropriate place or to make appropriate solution for it. The RSE enables to visualize the Joomla system and the SOFRSE helps view the target system with its modern requirements. This solution gives how Joomla components can be exposed to any service consumer. As there are a number of open source CMS products, it is possible to use other Joomla components’ output for any particular consumption if needed. The service consumption may not be only specific to open source CMS, but for any application if needed.

In the solution design part, we will create the SC, SB, and the SP based on the components as they will be considered as services when the system is reengineered. Therefore, reengineering into service-orientation is done on three of the components due to time limitation as the ultimate goal is to show as a prototype.

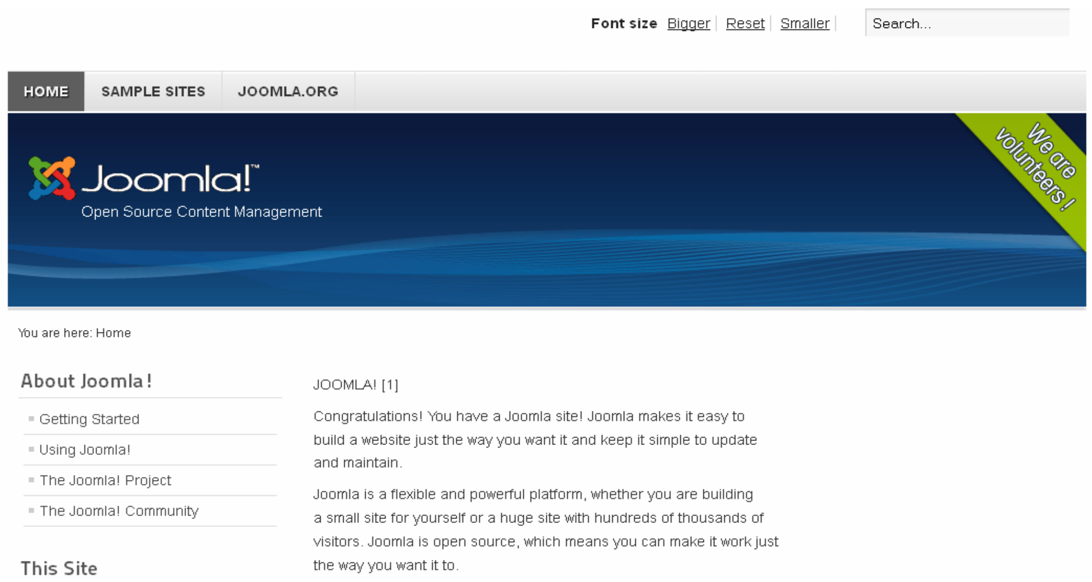
In this paper, due to time limitation, we consider three components: Content is used to display articles, content categories, and content sections, NewsFeeds are any series of news articles, and Contact is list of contacts information. The implementation of these components shows that it is possible to service-orient all the rest of the entire Joomla components whether from the frontend or backend as the way of access and the entry point of components is the same for both parts. The decision is made by the dispatcher based on user request. The access mechanism for the frontend and backend is through index.php and administrator/index.php, respectively. The dispatcher dispatches to the appropriate component based on the input path irrespective of the front and the backend [15]. Therefore, selecting either from the front or the backend does not affect the service-orientation principle.

In Joomla, the frontend operations are usually page requests, viewing contents, search values, and the input is a page to be requested and similarly the backend is a page request and input values of the form if any [15]. Selecting a single component from the backend requires other associated components like user registration and creation and due to time limitation, we could not select from that part. However, as described above, this does not affect the representation of the selected components. Therefore, randomly selecting any of the components represent the entire components except the content component should be incorporated as it is a must to exist in Joomla to get contents.

7. Prototype

To implement the prototype, SOA implementation technologies (XML, BPEL, SOAP, WSDL, etc.) and other SOA implementing tools are used.

Figure 1 shows a screenshot after Joomla has been service oriented. It shows when Joomla is first instantiated. A result from the content component is also shown.



JOOMLA! [1]

Congratulations! You have a Joomla site! Joomla makes it easy to build a website just the way you want it and keep it simple to update and maintain.

Joomla is a flexible and powerful platform, whether you are building a small site for yourself or a huge site with hundreds of thousands of visitors. Joomla is open source, which means you can make it work just the way you want it to.

Figure 1: Joomla home page after service-orientation

8. Conclusion

It is shown that making a legacy system ready for integration by incorporating service orientation is to mean ready to be used by other applications for reusability. Any client system that requests the server gets the desired output. The SOA technologies enable independent or heterogeneous systems to communicate across a network and through the Internet. This paper tried to apply the concept of service orientation to an open source CMS (Joomla) and this has been done successfully on some components of Joomla.

SOA is usually for different large systems to come to a common purpose for an organization to meet its business needs. However, it is also possible to apply the concept of SOA on an application having too many components by applying interface description and passing message among various services through broker(s) with some modification of both the server and client side code in a way making appropriate for service orientation. The depth of modification may vary form system to system. If the system is very old, it may require too many modifications as it usually becomes tightly coupled and difficult to divide into components or modules. When services are exposed to another application, an organization can benefit from the concept of service orientation since it brings various open source components composed with services orientation technologies and available to the business need by incorporating into other legacy or modern systems within a relatively short period of time instead of developing a new component or system from scratch.

References

[1] "http://www.answers.com/topic/ reengineering#ixzz21KtigFCX

[2] "http://www.hiddenbrains.com/articles/tag/re-engineering-services-business-reengineering-software-reengineering-service-hire-software-developer"

[3] C. L. Liu, N. T. Hua, and A. B. Tucker, "PRACTITIONER' S".

[4] M. B. Juric and M. Krizevnik, "Composite Applications with Oracle SOA Suite 11g".

[5] M. B. Juric, R. Loganathan, P. Sarang, and F. Jennings, "SOA Approach to Integration".

[6] S. Chung, J. Byung, C. An, and S. Davalos, "Service-Oriented Software Reengineering: SoSR," 2007.

[7] J. Melton and W. Group, "Praise for FastSOA".

[8] "The Undersigned Faculty Committee Approves the," 2006.

[9] "Analyzing Service Oriented Architecture (SOA) in Open Source Products". 2010.

[10] Y. Wang, "Web Services for a Software Development Platform," 2010.

[11] J. Franzén, "Shared Experiences from Five Organizations moving towards a," 2008.

[12] J. Lundberg, "Service Oriented Architecture & Web Services", 2008.

[13] E. Krankoc, "Web Service Composition under resource allocation constraints, 2007.

[14] E. K. Kuban, "Abductive Planning Approach for Automated Web Service Composition using only user specified inputs and outputs", 2009.

[15] J. K. Chuck Lanham, Mastering Joomla 1.5, Extension and framework development, second edition, The professional guide to programming Joomla, 2010.