I.J. Information Technology and Computer Science, **2013, 07, 1-11**
Published Online June 2013 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2013.07.01

Integration of Independent Applications and EAI Systems using Service Oriented Enterprise Bus and Open System Application Development Standards

Quist-Aphetsi Kester, MIEEE
Lecturer at the Faculty of Informatics, Ghana Technology University College
E-mail: kquist-aphetsi@gtuc.edu.gh

Dr. **Koumadi Koudjo M**
Lecturer at the Graduate School, Ghana Technology University College
E-mail: kkoumadi@gtuc.edu.gh.

Prof. **Nii.Narku Quaynor**
Professor, Department Computer Science and Information Technology, University of Cape Coast

***Abstract*—** Businesses today are dependent on custom enterprise software and web applications from independent software developers and software companies. This creates a lot of problems such as integration, interoperability, security, and system maintenance. Enterprise Application Integration (EAI) and Business-to-Business integration control several key technologies and swift advancement in technology to meet the increasing needs for integration in the enterprise which often results in a lot of challenges due to differences between one proprietary approach and another. This paper seeks to provide an approach of integrating independent applications and EAI systems by using Web services standards and open application development standards in dealing with the challenges faced in integrating applications. This will make it possible for organizations to add a new layer of abstraction that is open, standards-based, and easy to integrate with any new or existing system and also make easy for data discovery as well as building of new concepts from existing data. The combination of Service Oriented Architecture and Web services will be used to provide a rapid integration solution and also publishing services in a manner that new concepts can easily be built from existing services.

***Index Terms*—** Integration, Interoperability, Service Oriented Architecture, Open System Application, Enterprise Application Integration

I. Introduction

Enterprise Application Integration (EAI) is the process of linking different applications together within a single organization or across organization boundaries in order to simplify and automate business processes to the greatest extent possible, while at the same time avoiding having to make changes to the existing applications or data structures. In the words of the Gartner Group, EAI is the unlimited sharing of data and business processes among any connected application or data sources in the enterprise [1].

As corporate dependence on technology has grown more complex and far reaching, the need for a method of integrating disparate applications into a unified set of business processes has emerged as a priority. Users and business managers are demanding that seamless bridges be built to join them. In effect, they are demanding that ways be found to bind these applications into a single, unified enterprise application. The development of Enterprise Application Integration (EAI), which allows many of the stovepipe applications that exist today to share both processes and data, allows us to finally answer this demand. [2]

In today's enterprise infrastructure, system and application integration is a critical concern. Enterprise Application Integration has existed since the early 2000s, but the central problem that it attempts to solve is much older. EAI is an approach, or more accurately, a general category of approaches, to providing interoperability between the multiple disparate systems that make up a typical enterprise infrastructure. Enterprise architectures, by their nature, tend to consist of many systems and applications, which provide the various services the company relies upon to conduct their day to day business. A single organization might use separate systems, either developed in-house or licensed from a third party vendor, to manage their supply chain, customer relationships, employee information, and business logic. [3]

Service Oriented architecture (SOA) has gained popularity in recent years due to its enabling functionality or services to upgrade and extend existing software applications. SOA is an architectural approach to build and deploy software applications that is interoperable by design.SOA has grown as companies endeavor to leverage their existing client base and to integrate their acquired software with their clients' existing ERP system and also it makes software connectivity capabilities very easy. Unlike EAI no middleware is needed as adoptions of standards enable services to interact directly. It also enhances reusability capacity of software, resulting longer life of existing assets. A successful SOA implementation makes it easier to customize and upgrade existing applications thereby reducing total cost of ownership. [4]

EAI products prove to be expensive, consume considerable time and effort and are subject to high project failure rates. Additionally, because these custom applications are proprietary, many of the projects result in additional difficulties. Importantly, modifications to such applications require developing almost the entire system from scratch. Recent experience shows that a better answer is available by using Web services standards. The adoption of service oriented Architecture and web services provide a rapid solution to solving this problems faced by organizations [5] [6] [7].

The paper has the following structure: section II consist of related works, section III gives information on the methodology, section IV discusses the approach used for the integration, V talks about implementation as well as results and concluded the paper.

II. Related Works

Enterprise application Integration enables an enterprise to integrate its existing applications and systems and to add new technologies and applications to the mix. Enterprise organizations must weigh the cost of replacing existing systems with new systems against the cost of merging existing systems with new systems. Discarding existing systems is never an easy choice: companies have invested huge sums of money to install, use, and customize these systems. Not only are their personnel comfortable with using these systems, even if the software is rife with drawbacks, but often the company's way of doing business has evolved to fit with these systems. It is difficult to just walk away from such an investment.

Likewise, bringing in a replacement system has its costs: there's the purchase price of the new system, plus the training and customization costs. The investment in the new system can be as large, if not larger, than the investment in the existing system[8].

Prior to EAI, integrating applications and data within a corporate environment was an expensive and risky proposition. Companies were trying to merge applications that often ran on different hardware platforms and had no protocols for communicating with other software packages outside of their own narrowly defined realm. In a sense, companies had "islands" of business functions and data, and each island existed in its own, separate problem domain as shown in figure1 [8].

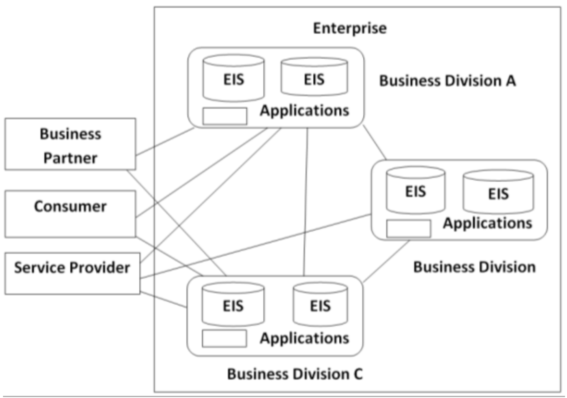

Fig. 1: A Typical Enterprise Domain

Prior to the development of EAI-type approaches, the problems of integration were largely handled using point-to-point integration. In a point-to-point integration model, a unique connector component is implemented for each pair of applications or systems that must communicate. This connector handles all data transformation, integration, and any other messaging related services that must take place between only the specific pair of components it is designed to integrate. When used with small infrastructures, where only two or three systems must be integrated, this model can work quite well, providing a lightweight integration solution tailor-made to the needs of the infrastructure. However, as additional components are added to an infrastructure, the number of point-to-point connections required to create comprehensive integration architecture begins to increase exponentially. A three-component infrastructure requires only three point-to-point connections to be considered fully integrated. By comparison, the addition of just two more components increases this number to 10 connectors. This is already approaching an unmanageable level of complexity, and once an infrastructure includes 8 or 9 component systems, and the number of connections jumps into the 30s, point-to-point integration is no longer a viable option. Each of these connectors must be separately developed and maintained across system version changes, scalability changes, and more (or, in some cases, even purchased at high cost from a vendor), and the unsuitability of point-to-point integration for complex enterprise scenarios becomes painfully clear.[3]

These EAI-type approaches face a lot of problems over time: The data exchange among different systems is inevitable during the process of Enterprise Application Integration. Considerable quantities of data would probably be migrated from one system to another after a fixed period of time. Under other circumstances, data representing business status in one system are required to keep synchronized with that on other systems simultaneously [9].

As in AllBooks, in the vast majority of cases the computer systems supporting the internal functioning of the enterprise have been developed independently to manage the tasks of individual business units. Existing systems manage their own data, following their own data models and governed by their own business rules. This leads to the technical problems experienced by AllBooks International, in which applications are not able to communicate with each other with the reliability and efficiency essential to safely implement the complex business processes of a modern enterprise. This resulted in technological difficulties as a result of ad-hoc integration solutions, such as remote calls, direct reading and writing to application space, or bridges and file transfers used for communication. In addition, applications cannot easily exchange information as the formats and representations of enterprise data are different, and therefore numerous complex individual interpretations are required [10].

Figure 2 and Figure 3 depict the actual situation at a major Australian bank in the mid 90s and 2004 respectively. The boxes represent applications, with arrows or lines showing their point-to-point interconnections. A comparison of the two figures shows that not only the problem has not been solved over the period, but that it may have actually become even worse. The dilemma for the bank analysts is how to properly manage the development maintenance and evolution of these systems. It is difficult to determine what is to be shared between applications. But the use of point-to-point interfaces resulted in a myriad of interface formats and technologies that have to be created and maintained; making it very difficult to accommodate the new applications that will be inevitably required. [10].

With the introduction of the Web, enterprise application integration has taken on a larger significance beyond that of merging application systems. Enterprise servers now handle and maintain huge amounts of data and business logic. The Web enables easy information and service access and it has become a principal means of communication. An enterprise must be able to make its business data accessible to others, from internal employees to external partners, suppliers, and buyers. Figure 4 illustrates this Web direction to which enterprises are currently moving. The success of the Java programming language and the J2EE platform are play a key role in a Web-driven application integration, in large part because they make it easier to develop and implement Web-based applications. [8]

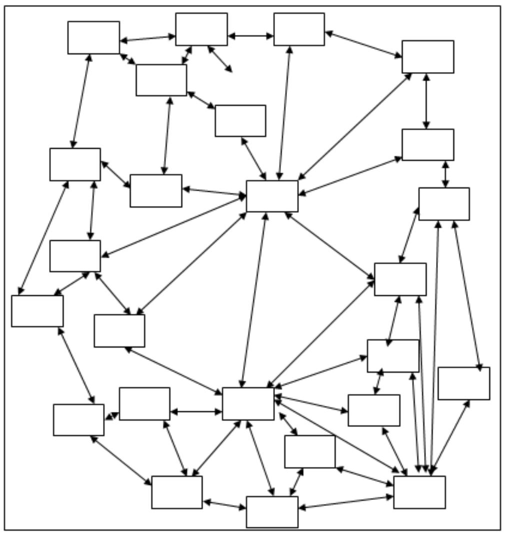
Fig. 2: A Typical Enterprise Domain

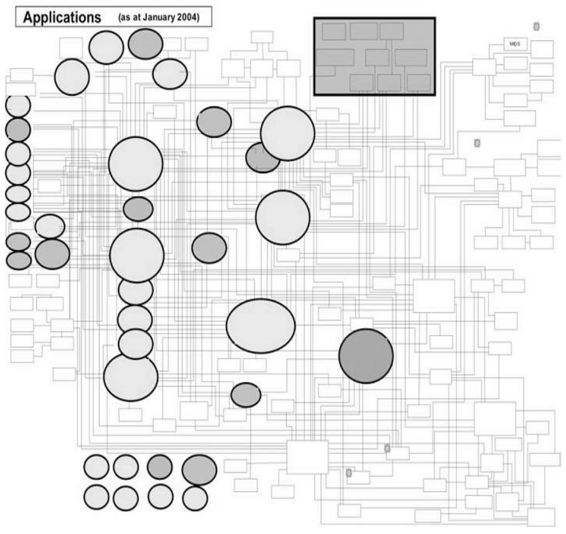

Fig. 3: The situation in a major Australian bank-II

As businesses establish Web presence, Web-driven EAI becomes more essential. Enterprises need to integrate their existing applications and enterprise systems to drive their business-to-consumer and business-to-business interactions, plus their other Web services. Success in e-business is driven by an enterprise's ability to integrate existing applications and extend the reach of these applications to Web-based access. [8]

Primarily, services are implemented as Web Services (WS) which are defined by the W3C as “software systems designed to support interoperable machine-to-machine interaction over a network” [11].It has an interface described in a machine-processable format. Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards [11].

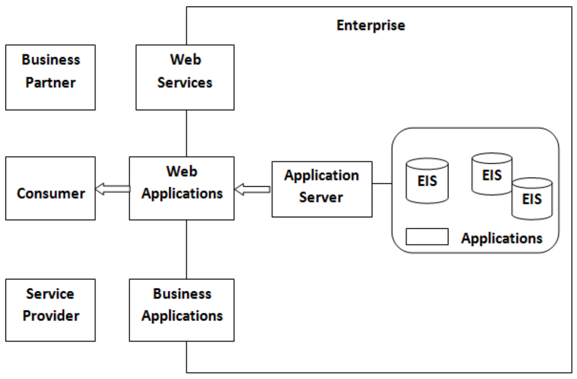

Fig. 4: Web-driven Application Integration

Inherently, a service is a software component that contains a collection of related software functionalities reusable for different purposes. It delivers such operations as data storage, data processing, mathematical and scientific computations, and networking. It is governed by a producer-consumer model in which a service is delivered by a service provider known as the producer which owns the facilities for hosting, running, and maintaining the service, and the client known as the consumer which connects and uses service functionalities. [12].

Much of the research regarding SOA tackles more granular technical issues of development and implementation of Web services, which may be a result of the aforesaid misconceptions [7]. Few papers e.g., [13],[14], deal with the much larger problem of defining what SOA means to the organization and how this definition should then provide the guidance for the development of components to meet business information needs[15]. The Information Technology adoption literature targeting a methodology for development states that there are five categories of factors influencing the decision to adopt SOA (i.e., environmental, organizational, individual, technology, and task characteristics [16]. These same factors should be addressed by the methodology for implementing SOA projects [17]. Two SOA methodologies that attempt to embody some or all of these factors are discussed below.

Teti (2006), an industry analyst, provides a methodology, which entails creating a vision, construction, and execution. He suggests that this model is applicable to many projects, but specifically addresses SOA. The vision creation is driven by a number of inter- and intra-organizational issues that define tasks important to the individuals and the firm (i.e., the constituency); the construction addresses the technology required to accomplish the tasks; and execution seeks to ensure that SOA will facilitate information exchange in the environment.

Bell (2008) provides a SOA methodology that takes a more technical approach. It professes that all software can be considered as services that are designed based on the informational tasks of the organization, configured for transmission in the working environments, constructed with available technologies, and deployed for use by individuals. The methodology represents a conceptual structure that brings together distributed services based on the functionality [18].

A study of the Government of Ghana Ministry Integration project was carried out by Kester et al (2012) where SOA with web services was used to demonstrate the integration of eight ministries. The existing systems within the ministries were using a variety of architectural models, languages and platforms that were not compatible with each other. On the other hand, the Government was not ready to put in place a new IT infrastructure to replace the existing one. But the aim of the IT project was to connect all the ministries such that they can interoperate and provide services to each other and also create a new platform where new services in the future can be added without changing the entire system. The application was developed from the proposed system architecture which aimed at providing interoperability functionalities through rich Internet clients and XML Web Services for all the ministries. [19]

III. Methodology

Providing services through the Web is rapidly becoming the emerging trend. Enterprises are recognizing that it is important for them to provide more of their services, such as customer support and product catalogs, through the Web. Enterprises have come to see that having such services available both in a traditional manner and over the Web enhances their business. The technology scenario is evolving at a breathtaking pace, and EAI is now increasingly being driven by Web-driven requirements and technologies. Web-driven application integration, by making data and services more easily and widely accessible enhances efficiency in business processes. For example, an online stock trading application offered by the financial services industry must be able to handle transactions whose numbers can increase rapidly. It is best, too, if the enterprise can retain the flexibility to develop and add in new applications and extend its existing applications. Up till now, applications were classified as either front-office or back-office applications. Front-office applications are considered to face the customer or end user. Front-office applications include applications for customer relationship management and marketing automation. Back-office applications provide the information infrastructure for running the back-end business processes of an enterprise. Applications provided by an enterprise resource planning (ERP) system are good examples of back-office applications. Traditional EAI focused on integrating the front- and back-office applications. However, traditional EAI is becoming Web-driven EAI. Rather than being targeted to the front end or the back end, most EAI applications are now integrated for the front and back ends and Web enabled. Just as it is imperative for an enterprise information system (EIS) to move to a Web-based architecture, enterprise applications need to be deployed on widely adopted standard application platforms. [8]

SOA with web services changes the way businesses are undertaken and it is a technique of design that guides all aspects of creating and using business services throughout their lifecycle Thus, from their conception to their retirement [1]. And also in defining and conditioning the IT infrastructure that allows different applications to exchange data and participate in business processes regardless of the operating systems or programming languages underlying those applications [20]. SOA is essentially a collection of services which communicate with each other. With SOA, business services are consistently represented and partitioned cleanly. The Web service also supports interoperable machine-to-machine interaction over a network through an interface described by the Web Services Description Language (WSDL) [21]. The services are the fundamental unit of the architecture for sharing business information across department and application boundaries [22].

This paper seeks to provide an approach of integrating independent applications by using Web services standards and open application development standards in dealing with the challenges faced in integrating applications. This will make it possible for organizations to add a new layer of abstraction that is open, standards-based, and easy to integrate with any new or existing system and also make easy for data discovery as well as building of new concepts from existing data. The combination of Service Oriented Architecture and Web services will be used to provide a rapid integration solution such that the systems can interoperate and provide services to each other, as well as provide an avenue for addition of new services in the future by interconnecting with other services without changing the entire systems and also publishing services in a manner that new concepts can easily be built from existing services.

However in high hopes of achieving these aims, the existing IT infrastructure will not be replaced by a new one due to extensive investments made in the existing system. This is because they have an enormous amount of useful data stored in them, so it's not practical to discard existing systems.

The approach that will be used will meet the requirements of IT projects by making applications to be more composed of services rather than writing codes from scratch. The approach will save codes and provide a standard way for each of the components of an IT system to interact with shared services, enable components to become building blocks for reuse, shift focus to application assemble rather than design, create new applications out of existing components and integrate applications systems both internally and externally.

SOA with web services will be used for the entire process of building and deploying the applications for the project. This effectively lower the overall costs and improve the ability to rapidly change and evolve IT systems within an organization. The approach will also facilitate the composition of services across disparate pieces of software, whether old or new; departmental, enterprise-wide, or inter-enterprise; mainframe, mid-tier, PC, or mobile device, to streamline IT processes and eliminate barriers to IT environment improvements within and outside organizations [22]

IV. The Approach

The approach used was to get all the independent systems residing within each organization connected so that they can easily exchange data irrespective of the platform upon which they are running on and make the services available for usage by external systems during transactions. A multi-layer architecture approach was used to develop a client-server system to make information processing possible within the organization and also making some services accessible. The architecture of a client server system consisting of Web browser, servlet middleware and database server for processing transactions was developed and services published in the service bus. Some end-user applications were developed using HTML form for user-input and this lowers the data requirement of the client's browser version and it does not impose the requirement of a Java-enabled browser with the latest JDK patch. A middle tier with a Web server running Java servlet was set up and the Java servlet was used to access the database with JDBC driver. A back-end database server was set up and Web services were developed to make it possible for external organizations to request for information from the database through the internet.

Web services with SOA was used to provide an open standard and machine-readable model (WSDL) for creating explicit, implementation-independent descriptions of service interfaces and communication mechanisms that are location-transparent and interoperable. For complexity and unreliability of integrating complex infrastructures using a point to point approach to be taken care of, a middleware will be used to centralize and standardize integration practices across the entire infrastructure. A separate connector to connect each application requiring, components in an EAI-based infrastructure use standardized methods to connect to a service bus (SOA) that is responsible for providing integration, message brokering, and reliability functionalities to the entire network. The systems bundle together adapters for connectivity, data transformation engines convert data to an appropriate format for usage by requesters, modular integration engines handle many different complex routing scenarios simultaneously, and other components present a unified integration solution. A central integration mechanism was used to further consolidate tasks. This create more room for a more flexible architecture, where new components can be added and removed as needed, by changing the configuration of the provider by simply developing modular components making a single service to be easily re-used by multiple applications.

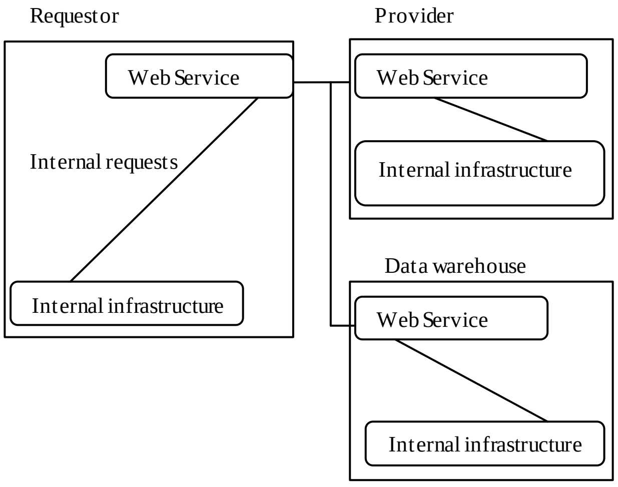

Fig. 5: Web enabled systems for provider and requestor

In the figure above, languages and protocols were standardized to eliminate the need for many different middleware infrastructures and the interactions were based on protocols redesigned and the internal functionality settings were made available as a service. Service-oriented architecture has standardization as a key policy in implementing web services for easy integration of multiple incompatible applications. Web services provide an entry point for accessing local services and with homogeneous components that reduces the difficulties of integration.

Web services were exposed through the interface the functionality performed by the internal systems and this makes the services discoverable and accessible through the Web in a controlled manner(Figure 5) and (Figure 6). Homogeneous components were built to reduced the difficulties of integration and standardized. Service descriptions was made richer and more detailed, covering aspects beyond the service interface. The application integration in environment encompasses three layers: a business process layer, an integration layer, and an application server layer (Figure 6). Each layer, in turn, holds technologies that serve as the application server integration building blocks. The application server layer enable an application integration project to link not only with existing enterprise systems but also with the Web. The application integration platform adds an integration layer on top of application server. This integration layer provides support for application development tools and frameworks. These development tools and integration frameworks are based on the application programming model, and they rely on metadata for generating and providing services. The integration layer also adds support for such functionality as a rules engine, intelligent message routing, and message transformation, all on top of the base functionality provided by the application server. The business process layer serves as the top-most layer for the platform and represents an enterprise's unique way of doing business. This business process layer exposes business process level abstraction by providing support for business process modeling and for the business process engine.

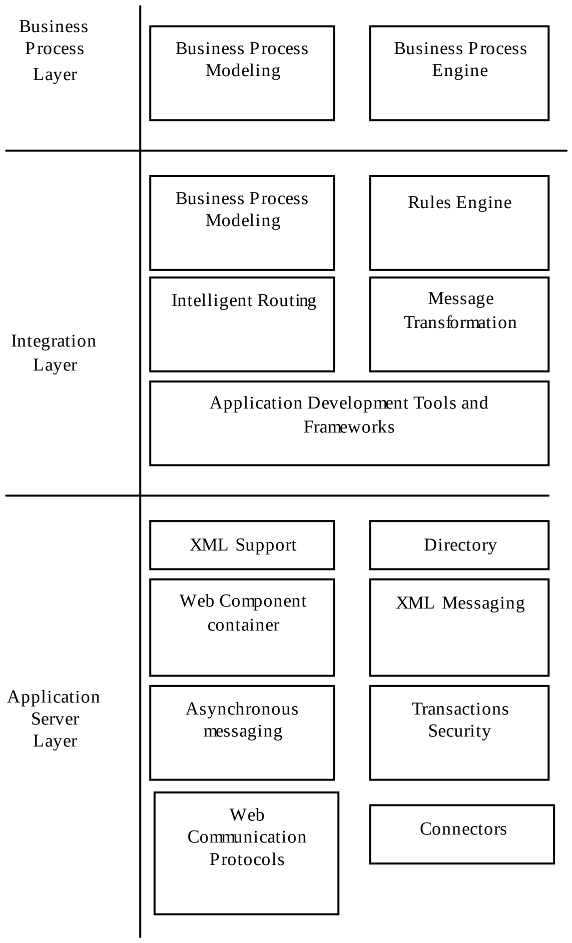

Fig. 6: Application Integration Layers

To integrate applications at the business logic layer, systems were enabled to consume and provide XML-based Web services. The Web Services Description Language (WSDL) contracts were used to describe the interfaces to these systems. And interoperability was further made possible by making the implementation compliant with the Web Services specifications. Figure 8 showed Service Oriented Integrated architecture of Enterprise resources. The Enterprise Resources and Operational Systems consist of existing applications, legacy and COTS systems, CRM and ERP applications, and older OO implementations. Integration Services provide access to the resources and systems of Enterprise Resources and Operational Systems and the components wrap integration services and provide a ‘single point of contact’ for integration services. Business Services represents a logical grouping of component, integration services and operations and also provides high level business functionality throughout the enterprise as well as provides a 'service interface' layer of abstraction to the functionality of components and Integration Services. Business Processes as a series of activities executed in an ordered sequence according to a set of business rules (called a choreography or business process model) and business events.

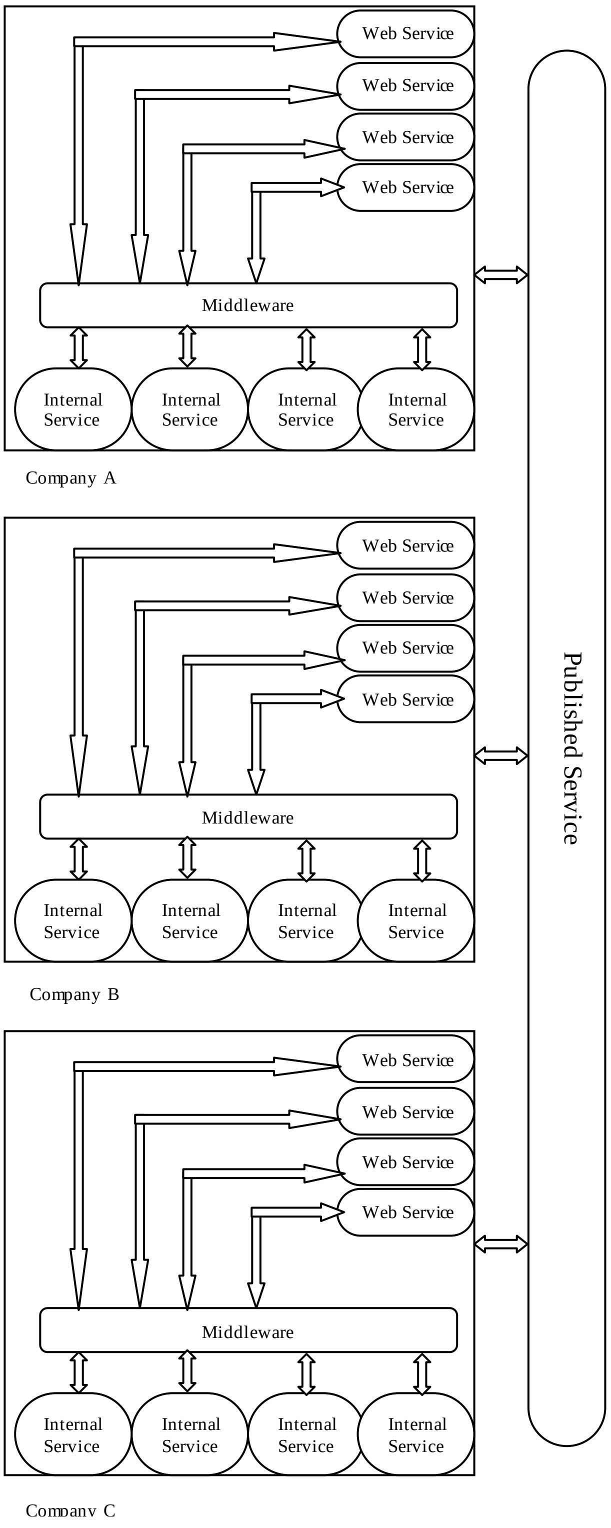

Fig. 7: Web enabled systems for provider and requestor

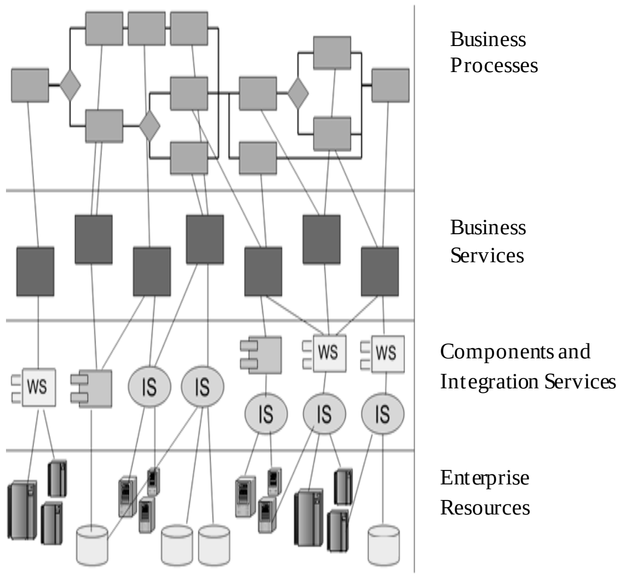

Fig. 8: Service Oriented Integrated Architecture

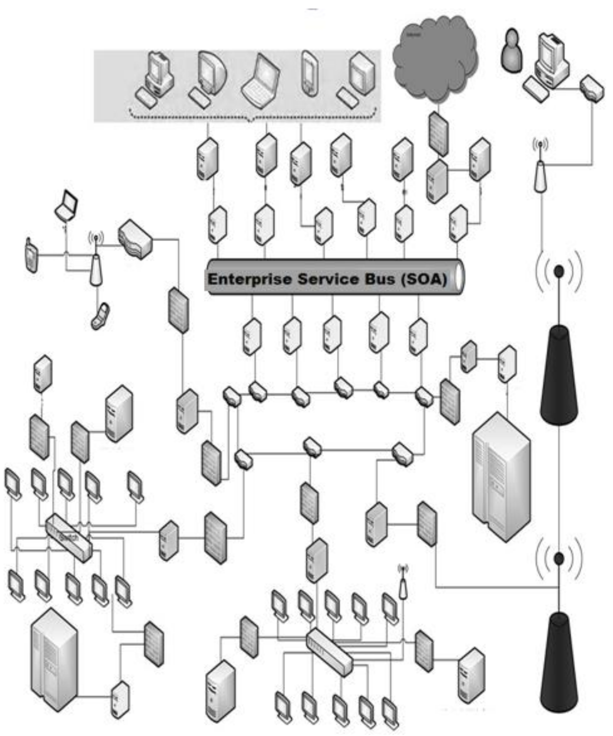

Fig. 9: Integrated Systems via a Service Bus

With integrated enterprise systems, businesses can now focus on core competencies, reduce spending, and reuse existing information in new ways without a major overhaul of their existing infrastructure. This will help them to have flexibility in their service delivery, save cost, and be efficient in service delivery. Figure 9 illustrates the dynamism of platforms interoperability and intercommunication of between integrated systems via a Service Bus.

The metadata of a registry network was stored in an ontology way and published for easy data discovery making it easy for searches to be done to find partners with products based on price range or availability, or to find high quality partners. This made it possible for applications, businesses, partners and suppliers to analyze information about the registries and their characteristics to effectively carry out the process of service discovery.

V. Results

The implementation was done by building a 3-tier database application that uses Java servlets and the Java Database Connection (JDBC). The three tiers consist of Client Tier or user interface, Middle Tier or business logic and Data Storage Tier. The three logical tiers were implemented to correspond to three types hosts namely: the browser or GUI Application to serve the client, Web Server or Application Server and the Database Server (often an RDBMS or Relational Database). Servlets was used for creating HTML user interface pages. The servlets JavaBeans were responsible for business logic and Java classes responsible for data access. These objects used were JDBC to query the database. The HttpServlet class provides methods, such as doGet and doPost, for handling HTTP-specific services.

The web client consists of two parts: Dynamic web pages containing various types of markup language such as HTML and XML, which are generated by web components running in the web tier and web browser, which renders the pages received from the server. The application clients directly access enterprise beans running in the business tier. The clients interact with Java EE servers, enabling the Java EE platform to interoperate with legacy systems, clients, and non-Java languages.

The business tier comprising the business components and entailing the business code, which is the logic that meets the needs of a particular business domain, is handled by enterprise beans running on the business tier. The enterprise bean receives data from client programs, processes it and sends it to the enterprise information system tier for storage or sends it back to the client program.

The enterprise information system tier handles EIS software and includes enterprise infrastructure systems, such as enterprise resource planning (ERP), mainframe transaction processing, database systems, and other legacy information systems. The Java EE application components might accesses the enterprise information systems for database connectivity.

The HTML form for user-input receives input for a form. The form is processed by a servlets which resides in a Tomcat server after it has been filled and submitted by a user.

The middle tier was developed with a Web server running Java servlets which accesses a database and returns an HTML page listing the data. Each field on the form stores information that can be transmitted to the web server and then sent to a Java servlets. Web browsers then communicate with the server by using HTTP. After a user fills the form and submits it, the form data is then sent to a server using two kinds of HTTP requests: get and post.

Java servlets handle both of these requests through methods inherited from the HttpServlet class: doGet(HttpServletRequest, HttpServletResponse) and doPost(HttpServletRequest, HttpServletResponse).

The doGet() and doPost() methods have two arguments: an HttpServletRequest object and an HttpServletResponse object.

The servlet communicates with the user by sending back an HTML document, a graphics file, or other types of information supported by the web browser. It sends this information by calling the methods of the HttpServletResponse class.

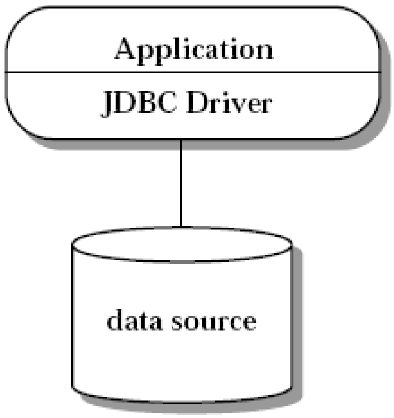

Fig. 10: Architecture of JDBC

A MySQL back-end database server was set up in order for the java servlet to access information in the database using JDBC driver existed. The JDBC is a java API for connecting programs written in java to relational databases. It has a data access interface, assesses many data sources and performs common task like connection pooling, batch updates and transaction management.

Above is the architecture of the JDBC. It is a two-tier architecture where the top level is visible to application programs and the lower level consists of drivers for individual engines.

The general steps taken in setting up the back-end database server are as follows:

- Importing the packages
- Registering the JDBC drivers
- Opening a Connection to the Database
- Creating a Statement Object
- Executing a Query and Returning a Results Set Object
- Processing the Result Set
- Closing the Result Set and Statement Objects
- Closing the Connection

Below is the "connect class" code snippet implemented for connecting to the MySQL relational database back-end server and a "queryDatabase class" for querying the database.

```
public class Connect {
    public static void main (String[] args) {
        Connection conn = null;
        try {
            String userName = "quist";
            String password = "kester";
            String url = "jdbc:mysql://localhost/backendserver";
            Class.forName ("com.mysql.jdbc.Driver").newInstance ();
            conn = DriverManager.getConnection (url, userName, password);
            System.out.println ("Database connection established");
                } catch (Exception e) {
            System.err.println ("Cannot connect to database server");
        } finally {
            if (conn != null) {
            try {
                conn.close ();
                System.out.println ("Database connection terminated");
            } catch (Exception e) { /* ignore close errors */ } }}}}
```

```
import java.sql.*;
import java.util.*;
public class queryDatabase {
public Vector requestInformation() {
Connection conn = getMySqlConnection();
Vector<String> response = new Vector<String>();
try {
Statement st = conn.createStatement();
ResultSet rec = st.executeQuery(
"SELECT * FROM curricula ORDER BY RAND() LIMIT 1");
if (rec.next()) {
response.addElement("ok");
response.addElement(rec.getString("url"));
response.addElement(rec.getString("title"));
response.addElement(rec.getString("description"));
} else {
response.addElement("database error: no records found");
catch (SQLException sqe) {
response.addElement("database error: " + sqe.getMessage()); } }
private Connection getMySqlConnection() {
 Connection conn = null;
```

The web services were created to enable external systems to request for information from internal systems. Besides SOAP, One of the most popular technologies in creating web services is XML-RPC, a protocol for using Hypertext Transfer Protocol (HTTP) and Extensible Markup

Language (XML) for remote procedure calls. An XML-RPC client is a program that connects to a server, calls a method on a program on that server, and stores the result. Using Apache XML-RPC, the process is comparable to calling any other method in Java because we don't have to create an XML request, parse an XML response, or connect to the server using one of Java's networking classes. In the org.apache.xmlrpc package, the XmlRpcClient class represents a client. An XmlRpcClient object can be created in three ways, each of which requires the URL of the server:

n XmlRpcClient(String)—Create a client connecting to an address specified by the String, which must be a valid web address (such as http://www.quistkester.com) or web address and port number (such as http://www.quistkester.com:2274)
n XmlRpcClient(URL)—Create a client connecting to the specified URL object
n XmlRpcClient(String, int)—Create a client connecting to the specified hostname

To call the remote method on the XML-RPC server, We called the XmlRpcClient object's execute(
String, Vector) object with two arguments:
n The name of the method
n The vector that holds the method's arguments
The name of the method was specified without any parentheses or arguments.
Following is the code snippet we used to instantiate a new XML-RPC client and call the method:

```
XmlRpcClient client = new XmlRpcClient("http://localhost:80");
Vector params = new Vector();
Object result = client.execute("kester.requestInformation", params);
```

The execute() method returns an Object that contains the response. This object was cast to one of the data types sent to a method as arguments: Boolean, byte[], Date,
Double, Integer, String, Hashtable, or Vector.
Like other networking methods in Java, execute() throws a java.net.IOException
exception if an input/output error occurs during the connection between client and server.
There's also an XmlRpcException exception that is thrown if the server reports an XMLRPC
error.

At the end web services were exposed through the interface the functionality performed by the internal systems and this makes the services discoverable and accessible through the Web in a controlled manner. Homogeneous components were built to reduced the difficulties of integration and standardized. The application integration in environment encompasses three layers: a business process layer, an integration layer, and an application server layer. Each layer, in turn, holds technologies that serve as the application server integration building blocks. The application server layer enable an application integration project to link not only with existing enterprise systems but also with the Web. The application integration platform adds an integration layer on top of application server. To integrate applications at the business logic layer, systems were enabled to consume and provide XML-based Web services.

VI. Conclusion

Using SOA with web services did not only reduce the management, maintenance time and cost but it also reduces the amount of deployed code. Moreover, the architecture did not only serve as the blueprint for the system but also the project's team structure, documentation organization, work breakdown structure, scheduling, planning, budgeting, unit testing and integration. The architecture established the communication and coordination mechanisms among components. Services were created and they can be reused in multiple and also new services and applications can be created quickly and easily used with a combination of new and old services. Importantly, services were modeled independently of their execution environment.

References

[1] Gable, and Julie, "Enterprise application integration", Information Management Journal, March/April, 2002.

[2] David S. Linthicum (1999). Enterprise Application Integration p6.

[3] Understanding Enterprise Application Integration - The Benefits of ESB for EAI (2012). Available: http://www.mulesoft.com/enterprise-application-integration-eai-and-esb

[4] Enterprise Application Integration & Service Oriented Architecture (2012). Available: http://www.managementstudyguide.com/enterprise-application-integration-service-oriented-architecture.htm

[5] Prof. Paul A. Strassmann .What is a Service Oriented Architecture - George Mason University, November 19, 2007

[6] Qusay H. Mahmoud, (2005). Service-Oriented Architecture (SOA) and Web Services: The Road to Enterprise Application Integration (EAI) .Available: http://www.oracle.com/technetwork/articles/javase/soa-142870.html

[7] Web Services Architecture Usage Scenarios, W3C Working Group Note, H. He, H. Haas, D. Orchard, 11 February 2004 Available: http://www.w3.org/TR/2004/NOTE-ws-arch-scenarios-20040211/

[8] Rahul Sharma Beth Stearns Tony Ng (2001).J2EE Connector Architecture and Enterprise Application Integration.pp1-3.

[9] Dongjin Yu et al.(2009)" Service Oriented Enterprise Application Integration and its Implementation Based on Open Source Software" Proceedings of the Second Symposium International Computer Science and Computational Technology(ISCSCT '09). p.229

[10] George Fernandez (2006), "A Federated Approach to Enterprise Integration". Thesis pp.10-11.

[11] Web Services Glossary, W3C, 2004, [online]http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211

[12] Builder, C., Bankes, S., and Nordin, R., Command Concepts: A Theory Derived from the Practice of Command and Control, RAND Corporation, 1999.

[13] Ren, M. and K. Lyytinen (2008) "Building Enterprise Architecture Agility and Sustenance with SOA," Communications of the Association for Information Systems 22(1), pp. 75-86.

[14] Papazoglou, M. P., and D. Georgakopoulos, D. (2003) "Service-Oriented Computing," Communications of the ACM, 46(10), pp. 24-49.

[15] Schulte, S., Repp, N., Berbner, R., Steinmetz, and R. Schaarschmidt (2007) "Service-Oriented Architecture Paradigm: Major Trend or Hype for the German Banking Industry?" Proceedings of the 13th Americas Conference on Information Systems (AMCIS).

[16] Teti, F. (2006) "Develop a Service-Oriented Architecture Methodology," Database Advisor Magazine, http://websphereadvisor.com/doc/17991 (current Oct. 15, 2007).

[17] Kwon, T. and R. Zmud (1987) "Unifying the Fragmented Models of Information Systems Implementation," in R.J. Boland and R. Hirschheim (Eds.), Critical Issues in Information Systems Research, New York: Wiley.

[18] Haines, M. (2007) "The Impact of Service-Oriented Application Development of Software Development Methodology," Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS).

[19] Kester, Q. A., Gyankumah, G. N., & Kayode, A. I. (2012). Using Web Services Standards for Dealing with Complexities of Multiple Incompatible Applications. International Journal of Information Technology, 4.

[20] Bell, M. (2008) Service-Oriented Modeling: Service Analysis, Design, and Architecture, New York: Wiley.

[21] XML Information Set, W3C Recommendation, J. Cowan, R. Tobin, 24 October 2001 Available: http://www.w3.org/TR/2001/REC-xml-infoset-20011024/

[22] Ervin Ramollari1 et al. A Survey of Service Oriented Development Methodologies. The 2nd European Young Researchers Workshop on Service Oriented Computing .2007.

Authors' Profiles

Quist-Aphetsi Kester, MIEEE: is a global award winner 2010 (First place Winner with Gold), in Canada Toronto, of the NSBE's Consulting Design Olympiad Awards and has been recognized as a Global Consulting Design Engineer. He is a PhD student in Computer Science. The PhD program is in collaboration between the AWBC/USFC Academics Without Borders/Universitaires Sans Frontieres (formerly AHED-Academics for Higher Education and Development) Canada and the Department of Computer Science and In-formation Technology (DCSIT), University of Cape Coast. He had a Master of Software Engineering degree from the OUM, Malaysia and BSC in Physics from the University of Cape Coast-UCC Ghana.

He has worked in various capacities as a peer reviewer for IEEE ICAST Conference, lecturer and Head of Comput-er science department. He is currently a lecturer and Head of Digital Forensic Laboratory Department at the Ghana Technology University.

Dr. Koumadi, Koudjo M: is a 2009 Wilkes Award Winner BCS. He is the director of the International Institute of Technology and Management (IITM), Togo and a lecturer at Ghana Technology Graduate School. He is a member of IEEE He had his PhD in Telecommunication Engineering at the Advanced Institute of Science and Technology (KAIST), in Daejeon Korea. Masters in Telecommunication Engineering at the Advanced Institute of Science and Technology (KAIST), in Daejeon Korea and B.Sc in Telecommunication Engineering Beijing University of Posts and Telecommunications in Beijing, China. He may be reached at kkou-madi@gtuc.edu.gh.

Prof. Nii Narku Quaynor: is a Jonathan B. Postel Service Award Winner 2007 from the IETF (The Internet Engineering Task Force) and a scientist and engineer who has played an important role in the introduction and development of the Internet throughout Africa. He is a Professor at the University of Cape Coast University, Ghana. He was the director of ICANN for the African Region in 2000.Quaynor graduated in engineering science from Dartmouth College in 1972 and received a Bachelor of Engineering degree from the Thayer School of Engineering there in 1973. He then studied Computer Science, obtaining an M.S. from the State University of New York at StonyBrook in 1974 and a Ph.D. from the same institution in 1977. He was founding chairman of AfriNIC, a member of the United Nations Secretary General Advisory Group on ICT, member of the ITU Telecom Board, Chair and of the OAU Internet Task Force, President of the Internet Society of Ghana, and member of the Worldbank Infodev TAP.

How to cite this paper: Quist-Aphetsi Kester, Koumadi Koudjo M, Nii.Narku Quaynor,"Integration of Independent Applications and EAI Systems using Service Oriented Enterprise Bus and Open System Application Development Standards", International Journal of Information Technology and Computer Science(IJITCS), vol.5, no.7, pp.1-11, 2013. DOI: 10.5815/ijitcs.2013.07.01