I.J. Information Technology and Computer Science, **2012, 11, 34-41**
Published Online October 2012 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2012.11.05

Using Web Services Standards for Dealing with Complexities of Multiple Incompatible Applications

Quist-Aphetsi Kester
Faculty of Informatics, Ghana Technology University College
kquist-aphetsi@gtuc.edu.gh

Ghartey Nana Gyankumah
Faculty of Informatics, Ghana Technology University College
nanagharteytech@yahoo.com

Ajibade Ibrahim Kayode
Faculty of Informatics, Ghana Technology University College
kayvaldo@yahoo.co.uk

Abstract— Organizations' dependence on custom enterprise software and web applications from independent software developers and software companies create a lot of problems such as integration, interoperability, security, and system maintenance. This paper seeks to provide a better approach by using Web services standards in dealing with the complexity of multiple incompatible applications that were written in different programming languages on multiple computers and also making 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. The combination of Service Oriented Architecture and Web services will be used to provide a rapid integration solution that will quickly and easily align Information Technology investments and corporate strategies by focusing on shared data and reusable services rather than proprietary integration products.

Index Terms— Integration, Interoperability, Service Oriented Architecture, Web services

I. Introduction

Over the course of many years, businesses find themselves in need of comprehensive enterprise integration solutions and turned to using products developed specifically for that purpose. The adoption of service oriented Architecture and web services provide a rapid solution to solving this problems faced by organizations [1] [2] [3]. In many organizations solicitation of custom enterprise software and web applications from independent software developers and software companies are common. Unfortunately, these custom enterprise application integration (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 paper has the following structure: section II consist of related works, section III gives information on the methodology, section IV use a study of Government of Ghana Ministry Integration project to demonstrate the approach of integration, section V gives information regarding the architecture used, section VI provided a framework underlying the structure supporting the service-abstraction level and section VII concluded the paper.

II. Literature Review

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" [4].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 [4].

Inherently, a service is a software component that contains a collection of related software functionalities reusable for different purposes [5].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.

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 [3]. Few papers e.g., [6],[7], 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[8]. The IT 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 [9]. These same factors should be addressed by the methodology for implementing SOA projects [10]. We now discuss two SOA methodologies that attempt to embody some or all of these factors.

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 [11].

III. Methodology

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 [12].

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) [13]. The services are the fundamental unit of the architecture for sharing business information across department and application boundaries [14].

The paper seeks to achieve two main aims, the first aim is to integrate systems such that they can interoperate and provide services to each other, whereas the second aim is to provide an avenue for addition of new services in the future by interconnecting with other services without changing the entire systems. 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 data stored in them, so it's not practical to discard existing systems. It's more cost-effective to evolve and enhance EIS (Enterprise Information Systems). An example such system is a legacy system. Other examples may be weather services, which can be used to get the weather information. Any application on the network can use the weather service to get the weather information.

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 [14].

IV. The approach

A study of the Government of Ghana Ministry Integration project was used to demonstrate the approach. The project aimed at integrating eight ministries. The existing systems were using a variety of architectural approaches, languages and platforms that were not compatible with each other. 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. The project further aimed at providing new services in the future by interconnecting with other ministries, agencies and government departments without changing the entire system.

A proposed system was to be developed using SOA with web services for the entire process of building and deploying applications for the project. The application was aimed at providing interoperability functionalities through rich Internet clients and XML Web Services for the following ministries:

1. Ministry of Education
2. Ministry of Roads and Highways
3. Ministry of Environment, Science & Technology
4. Ministry of Lands and Natural Resources
5. Ministry of Energy
6. Ministry of Communications
7. Ministry of Transport
8. Ministry of Water Resources, Works & Housing

Composite application solutions were created within each system for adoption of Web service. The Web Services Description Language (WSDL), a standard programming interface was used to provide access to applications, and Simple Object Access Protocol (SOAP), a standard interoperability protocol was used to connect disparate applications within the system.

The presentation logic of the applications within the ministries were separated from the business logic and the business logic layer was service-enabled making it easier to connect various types of Graphical User Interfaces (GUIs) and mobile devices. The business logic layer was service-enabled and a tightly coupled presentation logic layer was written for each. Importantly, instead of running the presentation logic tier as a tightly coupled interface on the same server in each ministry, the presentation logic was hosted on separate devices making communication with applications in the other ministries easy using a service bus.

The applications within the ministries easily exchanged data by using a Web service defined at the business logic layer because Web services represent a common standard across all types of software. XML was used to independently define the data types and structures [15] [16].

The service-oriented development methodology was used for the entire development process from requirements, analysis to implementation, verification and validation. It provided an effective means to apply and evaluate the service oriented architecture. Through the service-oriented development approach, there was a clean separation between the services provided and the architecture which comprised of components and their relationships.

Unlike other development approaches that limit their scope to a subset of phases such as analysis and design, the adoption of the Service-Oriented Development (SOD) supported the full Service-Oriented Architecture (SOA) lifecycle, including planning, analysis and design, implementation, testing, deployment, and governance activities. Distribution of system components made the system architecture much more modular and decentralized, which contributes to fault tolerance, component reuse, system robustness, maintainability and further positive system properties. However, such distribution of components or systems across buses and networks causes high interaction complexity.

The high degree of distribution made development of these systems very challenging. The number of states and conditions for functional behavior as well as for error conditions increase exponentially with the number of distributed nodes. The right approach adopted to handle this complexity was modeling of the individual functionalities of systems and services independently from each other in an interaction modeling and architecture definition language.

4.1 The Architecture

The Architectural model of the integration of the eight ministries in illustrated in Figure 1 below.

Data is housed in each of the ministry's database and is accessed through the data access layer. Business logics are custom rules and algorithms that handle the exchange of information between a database and user interface. Services are created in the business logic layer and their descriptions are stored and retrieved in the service repository. If a new application wishes to use an existing service, it can query the service repository to obtain the service description and easily generate SOAP messages to interact with it.

The service bus implements, integrates, manages, shares and distribute the services. Through the service bus end users and heterogeneous devices, can interconnect, interoperate and provide services to each other irrespective of the end users' location, device platforms and programming languages. As the business logic layer is service-enabled, the presentation logic of a particular system was separated from the business logic thereby making it easier to interconnect various types of Graphical User Interfaces and devices. Importantly, instead of tightly coupling presentation logics of applications on the same servers which they rely on, the presentation logic was hosted on a separate device in a particular department or ministry and communication with such applications was performed using the service bus.

However, from the architectural diagram, it was observed that there exists a tightly coupled legacy system in a ministry. It is of extreme importance to integrate some of these legacy systems in the service-oriented architecture not because the government of Ghana is not ready to put in place a new IT infrastructure to replace such existing systems or the costs of redesigning or replacing such systems are prohibitive because they are large and complex, but rather, because of the historical roles some of these legacy systems will play within the ministries. Such legacy systems will provide users' needs, even though SOA with web services provide newer technology and more efficient methods of performing tasks within the ministries.

Lastly, as it was expedient for services to reflect and correlate with business processes, the business process engine exists in the architecture to describe business processes, automate and modify them, enforce business rules and drive an automatic flow of execution across the multiple services.

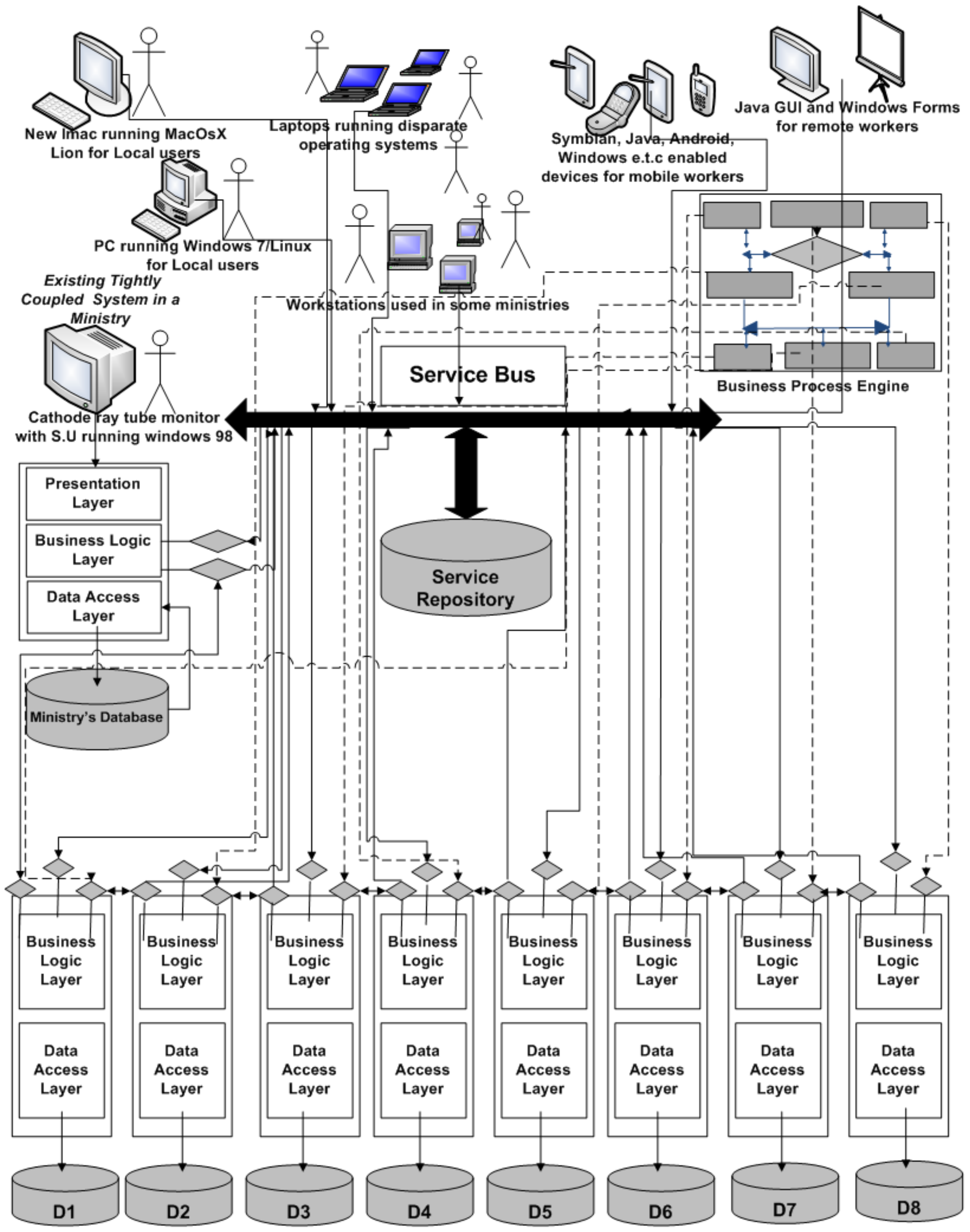
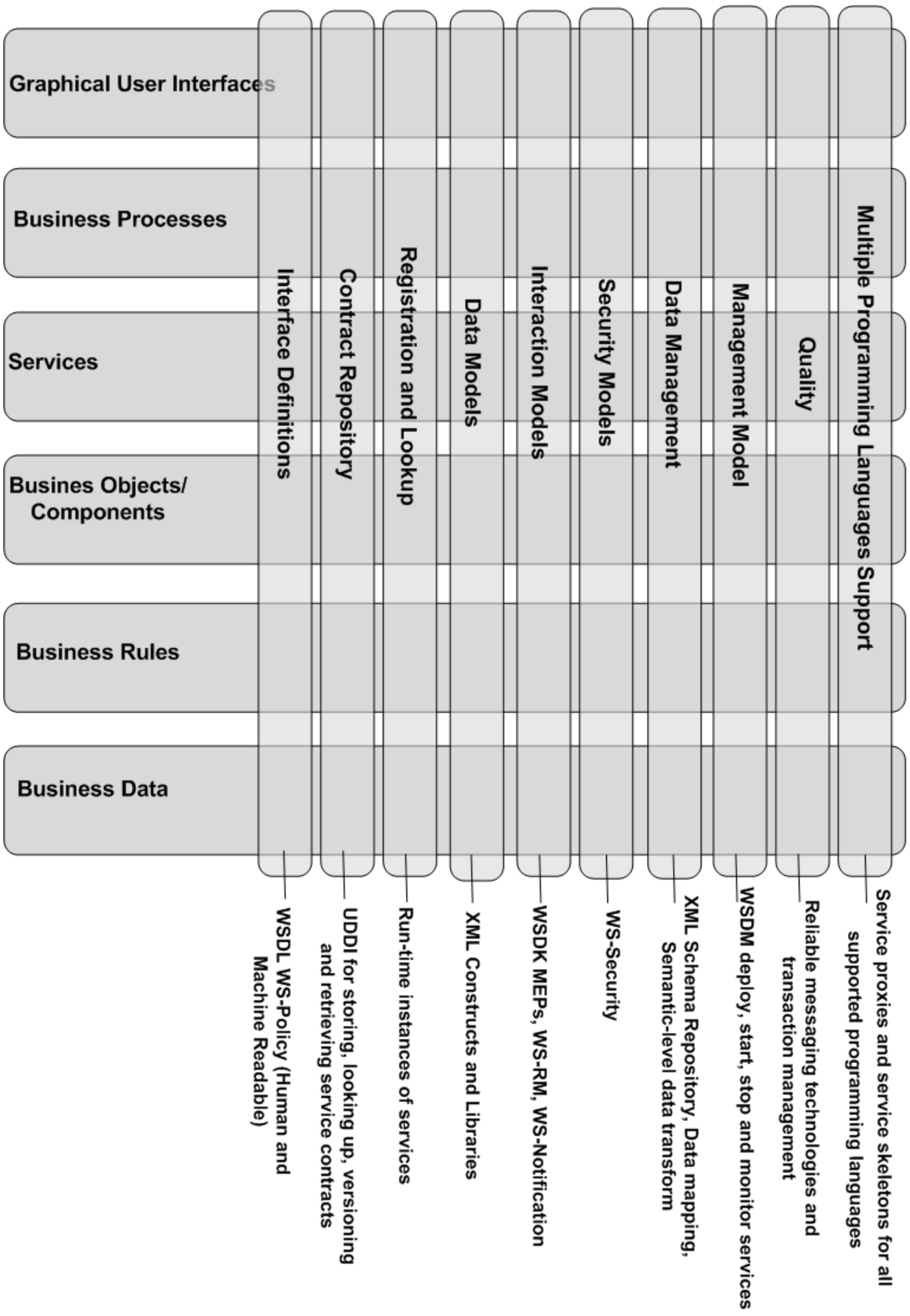

New Imac running MacOsX Lion for Local users
Laptops running disparate operating systems
Symbian, Java, Android, Windows e.t.c enabled devices for mobile workers
Java GUI and Windows Forms for remote workers
PC running Windows 7/Linux for Local users
Existing Tightly Coupled System in a Ministry
Workstations used in some ministries
Service Bus
Business Process Engine
Cathode ray tube monitor with S.U running windows 98
Presentation Layer
Business Logic Layer
Data Access Layer
Service Repository
Ministry's Database
Business Logic Layer
Data Access Layer
D1
D2
D3
D4
D5
D6
D7
D8

Fig. 1: an Architectural model of the integration of the eight ministries

D1 = Ministry of Education Database
D2 = Ministry of Road and Highways Database
D3 = Ministry of Environment, Science and Technology Database
D4 = Ministry of Lands and Natural Resources Database
D5 = Ministry of Energy Database
D6 = Ministry of Communications Database
D7 = Ministry of Transport Database
D8 = Ministry of Water Resources, Works and Housing Database

4.2 The Framework Underlying the Structure Supporting the Service-Abstraction Level

Dealing with collections of services became cumbersome. A concept and framework of service-level abstraction was therefore imperative so that the end users, departments and ministries can deal with collections of services, rather than individual services. Service-level abstractions deal exclusively with services and they define all the important elements of the services.



Graphical User Interfaces
Business Processes
Services
Busines Objects/ Components
Business Rules
Business Data
Interface Definitions
Contract Repository
Registration and Lookup
Data Models
Interaction Models
Security Models
Data Management
Management Model
Quality
Multiple Programming Languages Support
WSDL WS-Policy (Human and Machine Readable)
UDDI for storing, looking up, versioning and retrieving service contracts
Run-time instances of services
XML Constructs and Libraries
WSDK MEPs, WS-RM, WS-Notification
WS-Security
XML Schema Repository, Data mapping, Semantic-level data transform
WSDM deploy, start, stop and monitor services
Reliable messaging technologies and transaction management
Service proxies and service skeletons for all supported programming languages

Fig. 2: the service-level abstraction

The Service-level interface or Service contract definitions**:** Abstracting the accessible interface of services from its technical implementation was of crucial importance. The service-level interface or service contract catered for this. It was a well-defined interface that clearly separated the services' externally accessible interface from their technical implementation using WSDL. Preferably, the interface or service contract was human-readable and machine-readable. Also, a well-defined service contracts was published in the service repository, thereby making it easier for right services to be found.

The Service contract repository: This was the repository for storing, looking up, versioning and retrieving service contracts. Services were searched using taxonomies. The services were categorized appropriately by using the taxonomies in registries such as the UDDI.

The Service registration and lookup: This was the naming service for locating instances of services and run-time resources in a high performance, scalable, and highly available manner. The main difference between the service registration and lookup and the service contract repository was that whereas the service contract repository was used to look up service contracts, the service registration and lookup was used for finding the run-time instances of the services.

Service-level data model: The service- level data model was the data model for all data that was defined in the service-level interfaces or service contracts and exchanged among services. The service-level was defined using XML schema referenced in the WSDL service contracts. The service-level data model for the Service-Oriented Architecture was also defined using existing XML constructs and existing business-specific XML libraries.

Service-level interaction model: This determined which interaction modes were available for interacting with the services. For example, the service level interaction model can be defined by request or reply, request or callback, one-way, publish or subscribe.

Service-level security model: This defined the approach taken to securing services. For example the service-level security models could be how service requesters are authenticated, how and when service requesters are authorized, how SOAP messages are encrypted and how providers login to application and database levels.

Service-level data management: This defined the management of data at the service level. XML Schema repository was used for storing and managing business level data representations. Data-mapping facilities was used for mapping data between different message structures including data filtering, data aggregation, and simple translation function, and Semantic-level data transform facilities define information taxonomies and perform semantic transformations across service domain boundaries.

Service-level management model: This was the approach taken to manage services in their life-cycle. For example the service-level management model provides support for deploying, starting, stopping, and monitoring services.

Service-level qualities of service :This provided support for reliable messaging technologies and various qualities of services including message-ordering, guaranteed delivery, at-most-once delivery, and best-effort delivery. It also provided transaction management capabilities for defining and supporting transaction execution and control including two-phase commit and/or compensating transactions. High availability capabilities included clustering, failover, automatic-restart, load balancing, and hot deployment of services.

Support for multiple programming languages: This provided bindings for multiple programming languages. To fully support a wide range of execution platforms and applications, the service platform needed to support multiple programming languages, including generating service proxies and service skeletons for all supported programming languages.

The above description of the service-level abstractions framework was applied to a single operation and defined as part of a service, or a single service made up of multiple operations, or a collection of services drawn from a specific line of ministry of a group of ministries. For example, the service-level data model for a single service defines the data types used by all operations that make up that service. In contrast, the service-level data model for a set of line of business services defined the common language that those services used to exchange data.

V. Conclusion

Using SOA with web services did not only reduce the amount of deployed code, but it also reduced the management, maintenance. 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.

The approach of service-oriented development enabled the project to reap many benefits. Services were created and they can be reused in multiple applications within the various ministries. Also, new services and applications can be created quickly and easily used with a combination of new and old services. Importantly, services were modelled independently of their execution environment and messages were sent to any of the services. Furthermore, there was division of responsibility as individual employees or group within the ministries were using and exchanging services geared towards their lines of duties. For instance, human resource workers in each of the eight ministries would concentrate on human resource issues and they interoperated and exchanged information with each other through a human resource services developed by the service providers using the SOD approach.

References

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

[2] 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

[3] 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/

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

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

[6] 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.

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

[8] 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).

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

[10] 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.

[11] 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).

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

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

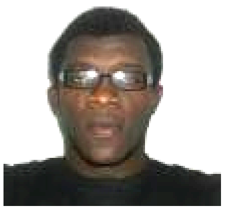
[14] Ervin Ramollari1 *et al*. A Survey of Service Oriented Development Methodologies. *The 2nd European Young Researchers Workshop on Service Oriented Computing .2007*.

[15] Thomas Erl. *SOA: Principles of Service Design*, l, Prentice Hall, 2007, ISBN -13-234482-3.

[16] *Web Services Choreography Description Language: Primer*, W3C Working Draft, 19th June 2006. Available: [http://www.w3.org/TR/2006/WD-ws-cdl-10-primer-20060619/].

Quist-Aphetsi Kester 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 Academics for Higher Education and Development (AHED) Canada and the Department of Computer Science and Information Technology (DCSIT), University of Cape Coast. He had a Master of Software Engineering degree from the OUM, Malaysia.

He has worked in various capacities as a peer reviewer for IEEE ICAST Conference, lecturer and Head of Computer science department. He is currently a lecturer and Head of Digital Forensic Laboratory Department at the Ghana Technology University College-GTUC as well as lecturer on the Coventry University (UK) program at the postgraduate school at GTUC. Research areas are in Software architecture, information security, internet technologies, Signal Speech and Image Processing.



Ghartey Nana Gyankumah is a Teaching Assistant at Ghana Technology University College-GTUC and pursuing an MSC degree in Software Development at Coventry University UK at the Ghana Technology University College-GTUC. His research areas are in Software architecture and methodologies, ecommerce, e-learning, telemedicine, and Internet technologies.

Ajibade Ibrahim Kayodei is an MSC Student in Software Development at Coventry University UK. He had a Masters Degree in Computer Systems from the University Of Ibadan, Oyo State Nigeria and B.Eng, Electrical /Computer Engineering from the Federal University Technology, Minna, Niger State. His research areas are in Software architecture, software methodologies and Internet technologies.

How to cite this paper: Quist-Aphetsi Kester, Ghartey Nana Gyankumah, Ajibade Ibrahim Kayode,"Using Web Services Standards for Dealing with Complexities of Multiple Incompatible Applications", International Journal of Information Technology and Computer Science(IJITCS), vol.4, no.11, pp.34-41, 2012. DOI: 10.5815/ijitcs.2012.11.05