Developing a Prototype of REST-based Database Application for Shipbuilding Industry: A Case Study

Jeong-cheol Jeon
Technical Planning
HHIC-TMS Co., Ltd. (TMS)
Busan, Korea Republic
cutejjc@gmail.com

Jaehwa Chung
Dept. of Computer Science
Korea National Open University (KNOU)
Seoul, Korea Republic
jaehwachung@knou.ac.kr

***Abstract*—This paper presents a case study of an application prototype for shipbuilding industry using RESTful web service. Majority of database applications for traditional enterprises are based on the two-tier with simple client-server or the three-tier with additional RPC middleware. Because the outdated old configuration is not extensible, an alternative is required to link the database to the Web or mobile platform. This type of modification on the legacy system causes large maintenance cost since the modified architecture can affect other parts of legacy system. Motivated by these issues, this paper proposes a simple and flexible application prototype utilizing open source frameworks and the RESTful web service. The prototype utilizes Groovy and Grails framework which is an open source web application framework on the Java platform. The major benefits of the proposed prototype are simply able to extend to heterogeneous client platforms such as mobile and .NET, and maintain by using high-productive web application framework. As a case study, the proposed prototype is applied to the management task of BOMs (Bill of materials) for shipbuilding.**

Keywords— REST, RESTful, web service, Grails, Groovy, BOM, shipbuilding, manufacturing

I. Introduction

As internet and mobile technologies gain popularity, the business scope of shipbuilding industry is expanding and the business methods are evolving [1]. Under these circumstances, it is difficult for these traditional companies with legacy system to adopt the state-of-the-art IT technologies which are rapidly advancing. In particular, if there are user requirements for mobile or web technology, it is not easy to implement such features on considering the configuration of legacy system. This happens as most existing applications were designed to use inside companies' network only. Thus, there was no consideration for the users of external network.

Shipbuilding industry tends to use the old systems as long as possible. It is not a simple task to fulfill the requirements of integrating web and mobile technology with these existing systems which are developed with the early versions of Delphi and RPC (Remote Procedure Call)-based middleware [2]. A lot of cost is required for the re-engineering and re-development. Particularly, as the conventional RPC-based middleware is developed with dedicated bundled tools and libraries from the software provider, the system is not extensible and flexible and commonly requires massive costs for upgrading the development environment. For that reason, the application developers of the existing system have little choice but to stay on the outdated development environments.

Early versions of Delphi which were released before the proliferation of internet and mobile technologies do not provide libraries with functionality required for integrating world-wide web and mobile technologies (e.g. Toolkits for web services, Unicode support), so they have limitations on extensibility. Moreover, if the system depends on software supplier, as the system is getting old, the maintenance cost tends to be kept increasing. Therefore, to avoid the dependency on the particular products of software vendors and to ensure service continuity, re-engineering (or re-development) of the existing system is required to switch to an enhanced configuration.

To overcome these issues, we reviewed two alternatives which are SOAP-based and RESTful web services [3]. SOAP-based web services are a mature and standardized approach to replace RPC-based middleware [11]. However, SOAP-based approach is complex and restricted on certain environments as it requires special toolkits, According to the review result, we chose RESTful web services to implement the application prototype. Meanwhile the RESTful web services are based on web technology, software with RESTful API can be easily extended to web and mobile platform. Also, REST does not cause such vendor lock-in related problems of the existing middleware because REST is not a proprietary technology of a particular supplier, but an open architecture style available to everyone.

To demonstrate the effectiveness of the chosen approach, we applied REST technology to the BOM (Bill of materials) management of the shipbuilding industry.

The main contributions of this paper are as follows:

- Analyzing the problems of legacy RPC-based systems in shipbuilding industry
- Presenting an overview of RESTful web services in comparison with SOAP-based web services.
- Providing an implementation of a RESTful prototype using modern web application framework for better extensibility.

The rest of this paper is organized as follows. Section 2 covers the concepts and implementation of the adopted technologies for this case study, then in Section 3 discusses the business service to apply, the structure of the implemented prototype, and the results of the implementation. Finally, Section 4 concludes with reviews and implications of the results.

II. Related Work

This section reviews the related previous studies, then looks at the concepts and the issues of the technologies used in the related architectures.

So far, a number of researches have been conducted to apply REST and SOAP technologies for Service-oriented Architecture (SOA) in the literature. Kumari et al. [5] compare the performance of the Loan Broker application based on REST and SOAP (Simple Object Access Protocol) and conclude that the REST-based web services show better performance than the SOAP-based applications. Feng et al. [6] compare the properties of RPC-style web services with RESTful web services. They conclude that RESTful Web services have better scalability and performance. Liu et al. [8] discuss the issues for reengineering legacy systems with RESTful web services. The comparison of the service-oriented architectures, the advantages of RESTful web services and the reengineering procedures are presented. Rodigues et al. [9] provide the performance comparison of the data exchange formats for RESTful web service on mobile devices. As the result, using JSON (JavaScript Object Notation) instead of XML (EXtensible Markup Language) as data-exchange format reduces considerable bandwidth as well as computing time. Engelke et al. [10] show a case study of REST-based Internet bidding system to replace legacy web services. Ruby on Rails is used for the implementation of the server. C#.Net for Windows and JavaScript for web browsers are used to implement the clients.

Fig. 1 illustrates a typical two-tier software configuration in use. This configuration uses a proprietary protocol dependent on a DBMS (Database Management System)

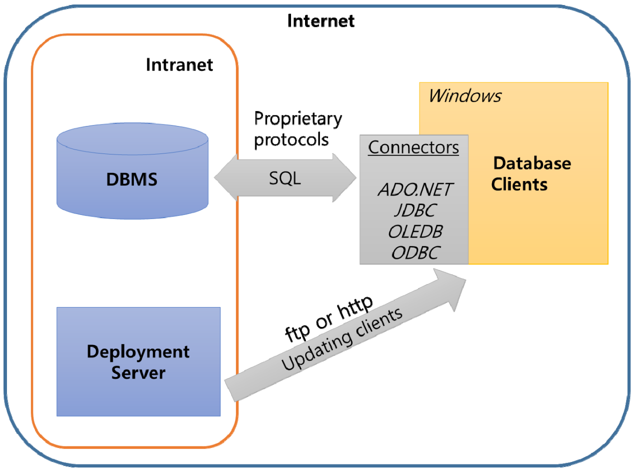


Fig. 1. Legacy application in two-tier configuration

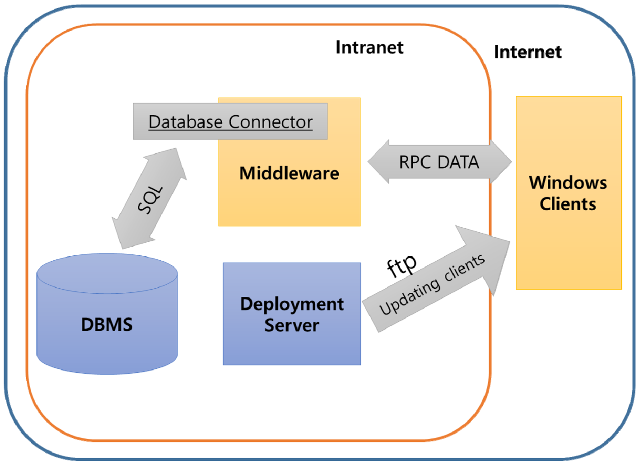


Fig. 2. Legacy application in three-tier architecture

product for exchanging SQL messages with its client applications. Database clients updates are downloaded from the deployment server using ftp or http protocol. Fig. 2 illustrates the existing three-tier software configuration. Middleware maintains connection to the database server (DBMS). Windows clients invoke remotely on the procedures running on the middleware. Middleware in turn processes the requests by performing SQL query on the database server if necessary. Windows clients updates are downloaded from the deployment server using ftp or http protocol.

Such three-tier applications using the legacy RPC-based middleware (e.g. RMI, CORBA, DCOM, DCE) and simple two-tier client / server applications using JDBC (or ODBC, OLEDB, ADO.NET) usually require to open many communication data channels such as TCP ports (varies according to the type of services) on the company network firewall to accept connection requests from the clients of external users. To handle these network related issues, the involvement of the network administrators is usually required. For the legacy applications in these configurations the system operators need to manage this additional connection-related information such as TCP or UDP ports according to the structure of the application, and which creates security issues followed by another problem of becoming vulnerable to failures. Network administrators and application operators should collaborate to solve such troubles, and the lack of understanding of the different sectors may prolong troubleshooting or delay problem resolution.

For user requirements which need columns to be added to a database table on the existing three-tier system, the software maintainers should go through the following procedure.

1. Maintainers add columns to a database table.
2. Change and build the code on the middleware.
3. Import the generated client stub code file to the client development environment.
4. After modifying the client source code, go through building and testing.

5. Finally, upload the pre-built binary file onto the deployment server for updating the windows client executables.

To minimize network firewall issues and improve this inefficient maintenance process of the existing system together with the vendor lock-in avoidance (discussed in section 1), this paper applies the RESTful web services.

SOAP-based web service is a traditional alternative to RPC-based middleware, but it requires toolkits which we failed to find one for the legacy development environment. And SOAP-based web service is known to be complex and heavyweight compared to REST-based web service. Previous studies [5, 6, 7] concludes that REST is better in performance compared with SOAP-based web service. The additional protocol layer and XML schema processing required for SOAP-based web service may affect the performance but REST does not have these overheads. For these reasons we chose RESTful web service for this prototype.

REST (REpresentation State Transfer) is a network-based architectural style first introduced and defined by Roy Fielding (2000) to provide interoperability between computer systems on the Internet. A REST-based or RESTful architecture is defined by a set of constraints. RESTful architecture has resources at the core which are identified by URIs (Uniform Resource Identifiers). There are predefined operations for manipulating the resources. These resources can represent data objects as one of various data formats such as JSON or XML.

REST uses http protocol as a transportation medium to build lightweight, flexible and scalable web services. It is not a protocol but an idea of developing web-like services and it is not tied to a specific platform or technology. Lately, REST gets developers' attentions along with recent dramatic evolution in mobile and web technologies. The key property of REST is the simplicity that comes from the fact that the REST utilizes the http protocol of web technology itself. As mentioned by Feng et al. [5], unlike SOAP-based web service which runs on the “Web”, REST is in the “Web”. Since REST uses http the core technology of WWW, and also uses the same configuration with any ordinary web application, most of state-of-the-art web technologies can be applied including web-cache. In addition, REST's basic four operations including POST, GET, PUT and DELETE http methods can be mapped to CRUD (Create, Read, Update, Delete) operations of relational databases. Therefore, REST can help to ensure consistent development patterns and make system structure easy to maintain. Developers can utilize various supporting tools and documentation available online.

TABLE 1. COMPONENT MAPPING FOR TWO-TIER SYSTEM

Before	**After**
Database	DB + Web Application Server
SQL code	Grails ORM code
OLEDB (ADO.Net)	JDBC
Windows Clients	REST Clients (Web browsers)
Deployment Server	Web Application Server
FTP protocol (for client updates)	HTTP or HTTPS protocol

In this study, we use the Grails [4, 12, 13] framework to implement the REST-based application prototype which replaces the existing application based on the old inextensible architecture. The Grails web application framework is a combination of a variety of open source libraries based on the Spring Framework and Groovy, a dynamic programming language for Java platform. Grails uses latest software design paradigms such as Convention over Configuration (CoC), Don’t Repeat Yourself (DRY), and sensible defaults on Java platform to improve the productivity of the development process. We decide to use Grails framework for this prototype because it has the benefits mentioned above and based on robust Spring framework [14] which is running on Java, as the stability is one of the most important decision factors for the company. Java platform and Spring framework are considered to be stable as they have been adopted and used by large organizations including Korean government [15].

III. DESIGN AND MAPPING OF THE ARCHITECTURE

This section checks how each component of legacy architecture can be mapped to the proposed architecture, and describes the proposed architecture.

Each software component used in the existing legacy architecture can be replaced by the components of the proposed architecture as described below.

■ Replacements of the components in two-tier system

The core components of the legacy two-tier are the database and the Windows clients. APIs for database connection (Ado.Net or OLEDB) are commonly used for exchanging messages between them.

Table 1 shows how these components are replaced in the REST-based architecture. The database server can be used as before but the web application server accepts user clients connection requests instead of the database server. SQL codes are replaced with Grails ORM (Grails’ Object Relational Mapping) codes. OLEDB database connection library for windows clients is replaced with JDBC connection library.

TABLE 2. COMPONENT MAPPING FOR THREE-TIER SYSTEM

Before	**After**
Database	Database
Native DB Client library	JDBC
SQL code	Grails ORM code
Middleware Server	Web Application Server
C language (middleware code)	Groovy or Java Code
RPC protocol (DCE)	RESTful web service (http)
Binary Message	JSON Message
Windows Clients	REST Clients (Web browsers)
Deployment Server	Web Application Server
FTP protocol (for client updates)	HTTP or HTTPS protocol

Windows clients are replaced with REST clients using JavaScript on web browsers. The functionality of deployment server is replaced with web application server. FTP protocol used for updating client applications is replaced with http or https protocol for web-based REST client application.

■ Replacements of the components in three-tier system

The core components of the legacy three-tier are the database, middleware server and the client application. APIs for database connection such as Ado.Net or OLEDB is used for exchanging SQL between the database and the middleware, and RPC is used for exchanging parameters of procedures between the middleware and the client.

Table 2 shows how these components are replaced in the REST-based architecture. The database server can be used as before. Native database connection library used for middleware platform is replaced with JDBC on Java platform. SQL codes are replaced with Grails ORM (Grails' Object Relational Mapping) codes. The middleware server is replaced with Web application server such as Apache Tomcat. C language used for middleware is replaced with Groovy language on Grails framework. RPC protocol is replaced with RESTful web services on http. Messages in binary format are replaced with messages in JSON format. Windows clients are replaced with REST clients using JavaScript on web browsers. The functionality of deployment server is replaced with web application server. FTP protocol used for updating client applications is replaced with http or https protocol for web-based REST client application.

Finally, the new application architecture can be configured as in Fig. 3. This configuration has the structure where messages exchange between the application server and the client applications via RESTful web services. The database connection method between the database and the REST server is performed by JDBC driver. The REST server and its client are connected using the http or https protocol. The REST server uses the RESTful web services to exchange messages with the REST clients. JSON is used as the serialization data format for the message exchange because it is better in performance [9].

IV. Implementation

This section discusses the business tasks to be applied to, the architecture of the implemented prototype and the results of the implementation.

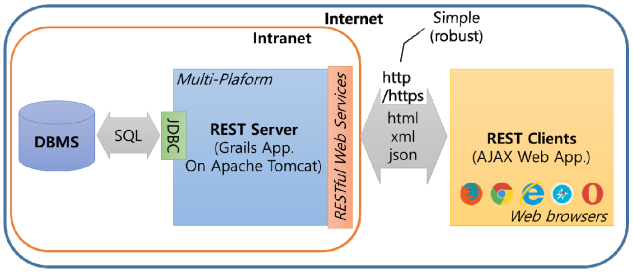


Fig. 3. Prototype architecture with RESTful web services

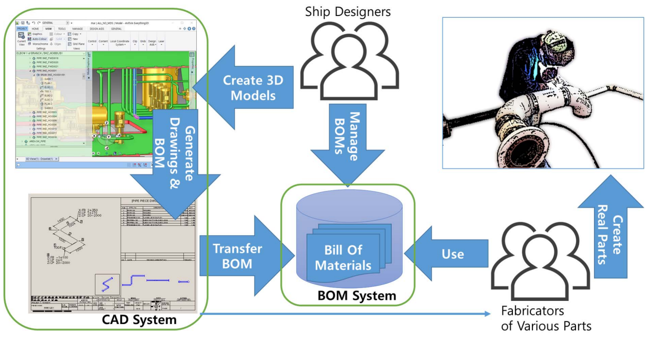
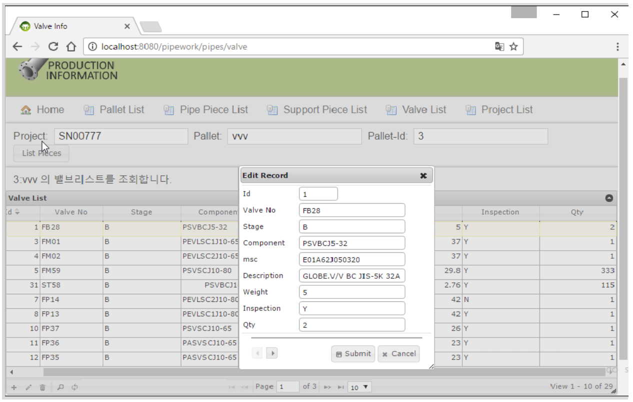


Fig. 4. Business concept of application

The existing system's service we selected for this prototype research is the management and inquiry of bill of materials (BOM) for outfitting parts used for shipbuilding. The reason why we chose this service is that the BOM information is vast and has a lot of internal and external users involved in a shipbuilding project, therefore frequent and continual updates are required to handle user requests.

The concept of business scenarios supported by the system service is shown in Fig. 4. The ship designers create 3D (dimensional) models of a ship using the CAD (Computer Aided Design) system. Next, from the completed CAD models, the bill of materials (BOM) of the parts constituting the ship are extracted to files in text format and transferred to and registered on the database of the BOM system. After registration, additional attributes (such as manufacturer-id, place-of-delivery) needed for purchasing, manufacturing are appended to the pre-registered BOM information of the ship. Thus, the BOM information registered and managed in the database is provided to a large number of involved suppliers (sub-contractors) and used as information necessary for manufacturing (or fabricating) of the parts. Suppliers may update and share the status information of the manufacturing process using a relevant application module of the BOM system.

Fig. 5 and Fig. 6 show the UIs (User Interfaces) of the application prototype which provide a set of functions to manipulate (register, query, modify, delete) the data records in



Project: SN00777 Pallet: vvv Pallet-Id: 3

3:vvv 의 밸브리스트를 조회합니다.

Valve List

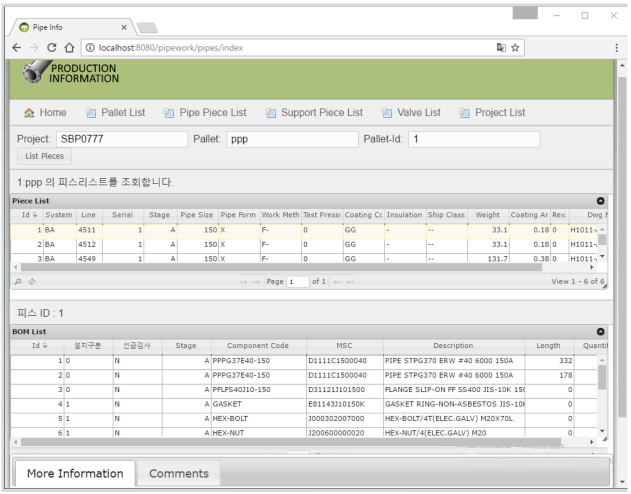
Id	Valve No	Stage	Componen		Inspection	Qty
1	FB28	B	PSVBCJ5-32	5	Y	2
3	FM01	B	PEVLSC1J10-65	37	Y	1
4	FM02	B	PEVLSC2J10-65	37	Y	1
5	FM59	B	PSVSCJ10-80	29.8	Y	333
31	ST58	B	PSVBCJ1	2.76	Y	115
7	FP14	B	PEVLSC2J10-80	42	N	1
8	FP13	B	PEVLSC1J10-80	42	Y	1
10	FP37	B	PSVSCJ10-65	26	Y	1
11	FP36	B	PASVSCJ10-65	23	Y	1
12	FP35	B	PASVSCJ10-65	23	Y	1

Page 1 of 3 10 View 1 - 10 of 29

Edit Record

Id	1
Valve No	FB28
Stage	B
Component	PSVBCJ5-32
msc	E01A62J050320
Description	GLOBE.V/V BC JIS-5K 32A
Weight	5
Inspection	Y
Qty	2

Fig. 5. UI of application prototype (Editing)



Project: SBP0777 Pallet: ppp Pallet-Id: 1

1:ppp 의 피스리스트를 조회합니다.

Piece List

Id	System	Line	Serial	Stage	Pipe Size	Pipe Form	Work Meth	Test Pressı	Coating Cc	Insulation	Ship Class	Weight	Coating Ar	Rev.	Dwg
1	BA	4511	1	A	150	X	F-	0	GG	-	--	33.1	0.18	0	H1011-
2	BA	4512	1	A	150	X	F-	0	GG	-	--	33.1	0.18	0	H1011-
3	BA	4549	1	A	150	X	F-	0	GG	-	--	131.7	0.38	0	H1011-

Page 1 of 1 View 1 - 6 of 6

피스 ID : 1

BOM List

Id	설치구분	선급검사	Stage	Component Code	MSC	Description	Length	Quanti
1	0	N	A	PPPG37E40-150	D1111C1500040	PIPE STPG370 ERW #40 6000 150A	332	
2	0	N	A	PPPG37E40-150	D1111C1500040	PIPE STPG370 ERW #40 6000 150A	178	
3	0	N	A	PFLFS40J10-150	D31121J101500	FLANGE SLIP-ON FF SS400 JIS-10K 15	0	
4	1	N	A	GASKET	E81143J10150K	GASKET RING-NON-ASBESTOS JIS-10	0	
5	1	N	A	HEX-BOLT	J000302007000	HEX-BOLT/4T(ELEC.GALV) M20X70L	0	
6	1	N	A	HEX-NUT	J200600000020	HEX-NUT/4(ELEC.GALV) M20	0	

Fig. 6. UI of application prototype (Inquiry)

a data grid.

In order to ensure the extensibility for mobile or windows clients in the future, this prototype is designed to use the RESTful web services. However, the involvement of these mobile or windows native clients may increase the cost of maintenance. Therefore the number of different client platforms should be minimized for the maintainability.

Since the proposed prototype is based on a web application, the management of the corporate firewall is simplified (only http/https service ports are needed to open), and the potential deployment issues (updating pre-built client binary) in the legacy system are eliminated. Plus, The REST-based approach provides better accessibility for users outside the company.

In addition, application maintenance inefficiency of the legacy three-tier approach has been improved with the adoption of RESTful web service and the modern web application framework. The procedure for adding columns of a table in a Grails Web application with RESTful web services is as follows. First, developers open to edit the Groovy class code(.groovy) representing the domain object (class) and add the properties corresponding to the database columns to be added, and then modify the JavaScript and HTML in the GSP (Groovy Server Pages) source files for the REST Client, and then perform the test and deploy the application on the server. The application can be modified and tested in a single Grails development environment and can be dynamically verified the code changes during program execution in the development environment.

TABLE 3. DEVELOPMENT TOOLS AND FRAMEWORKS

Type	Description
Development Platform	Java SE 7
Web Application Framework	Grails Framework 2.2.4
Grails plugin for RESTful web service	grails-jaxrs 0.9
Grails plugin for Ajax Data Grid	jQuery Grid (jqGrid) 4.3.2
Integrated Development Environment	Groovy/Grails Tools Suite (3.6.4)
Web Application Server	Apache Tomcat 8.0.36
Database Server	MySQL 5.1
Web Application Server Platform OS	Ubuntu Linux 14.04
Developer Platform OS	Windows 7 Professional
Database Platform OS	Ubuntu Linux 14.04

Table 3 lists the development tools and frameworks used to implement the application software prototype. We used Grails Framework for the implementation of the web application with RESTful web services. We used JSR-311 (JAX-RS) Grails plug-in for implementation of RESTful API and Spring Security (CORE, LDAP, UI) Grails plugin to implement functions related to user authentication. We also used the jQuery UI plugin for the Ajax UIs and jQuery Grid (jqGrid) plugin for the data grid used in the Grails web client.

As mentioned before, REST applications have the same architecture as the web applications, and this research is not a sort of algorithm study, we want to make it clear that we did not go through the performance analysis of this prototype.

V. CONCLUSIONS

This case study reviewed the structural problems of the legacy applications used in the shipbuilding industry and proposed a REST-based prototype as an alternative.

Through the case study, we learned that the RESTful web services approach is well matched with the database application, and the new approach with the Grails web application framework simplifies the system structure, therefore the application can be rapidly deployed in response to the users' requests compared with legacy approaches. Besides, the web-based approach can also reduce the network related troubles. Above all, the prototype is flexible and extensible, so it is possible to extend to various different client platforms such as Linux, Android, iOS and more.

This case study has some limitations that the implementation includes only the CRUD database operations. We hope that in future there will be more case studies related on various REST-based approaches, such as transaction processing and file attachments.

According to the fact that there has been a lot of discussions on REST such as HATEOAS (Hypermedia As The Engine Of Application State), REST may be the key component providing infrastructure for IoT (Internet of Things) world in near future.

References

[1] Mamaghani, Farrokh. "Impact of Information Technology on the Workforce of the Future: An Analysis", International Journal of Management, Vol. 23, No. 4, pp. 845-850, Dec. 2006.

[2] Krakowiak, Sacha. "An Introduction to Middleware" in Middleware Architecture with Patterns and Frameworks, Feb. 2007.

[3] Fielding, Roy Thomas. Architectural styles and the design of network-based software architectures. Diss. University of California, Irvine, 2000.

[4] Rocher, Graeme, and Jeff Scott Brown. “The Essence of Grails” in The Definitive Guide to Grails, 2nd ed., Apress, 2009.

[5] S. Kumari and S. K. Rath, "Performance comparison of SOAP and REST based Web Services for Enterprise Application Integration," 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, 2015, pp. 1656-1660

[6] Xinyang Feng, Jianjing Shen and Ying Fan, "REST: An alternative to RPC for Web services architecture," 2009 First International Conference on Future Information Networks, Beijing, 2009, pp. 7-10.

[7] B. Upadhyaya, Y. Zou, H. Xiao, J. Ng and A. Lau, "Migration of SOAP-based services to RESTful services," 2011 13th IEEE International Symposium on Web Systems Evolution (WSE), Williamsburg, VI, 2011, pp. 105-114.

[8] Y. Liu, Q. Wang, M. Zhuang and Y. Zhu, "Reengineering Legacy Systems with RESTful Web Service," 2008 32nd Annual IEEE International Computer Software and Applications Conference, Turku, 2008, pp. 785-790.

[9] Rodrigues, Carlos, José Afonso, and Paulo Tomé. "Mobile application webservice performance analysis: Restful services with json and xml." International Conference on ENTERprise Information Systems. Springer Berlin Heidelberg, 2011.

[10] Engelke, Charles, and Craig Fitzgerald. "Replacing legacy web services with RESTful services." Proceedings of the First International Workshop on RESTful Design. ACM, 2010.

[11] Garriga, Martin, et al. "RESTful service composition at a glance: A survey." Journal of Network and Computer Applications 60 (2016): 32-53.

[12] Beckwith, Burt. “Programming Grails.” O'Reilly Media, Inc., 2013.

[13] “The Grails Framewrk.” https://grails.org, September 2015.

[14] Walls, Craig. “Spring In Action Fourth Edition.” Manning Publications Co., 2014.

[15] “eGovFrame Portal.” http://www.egovframe.go.kr, October 2016.