Fifth International Conference on Construction in the 21st Century (CITC-V)
"Collaboration and Integration in Engineering, Management and Technology"
May 20-22, 2009, Istanbul, Turkey

Two BIM Based Web-Service Patterns: BIM SOAP Façade and RESTful BIM

Umit Isikdag
Independent BIM Consultant, Ankara, Turkey
egetera@superonline.com

Jason Underwood
University of Salford, Manchester, U.K.
j.underwood@salford.ac.uk

Abstract
In the near future, Building Information Modelling will be applied in different areas of the AEC industry. Building Information Models (BIMs) will be used as resources to enable interoperability of software and 'Integrated Project Delivery based on Building Information Modelling' will be realised as a process of managing a project over a single shared information backbone. Thus, facilitating the collaborative use of shared BIMs is becoming important in parallel with the industrial demand in the field. On the other hand, in the software industry service-oriented architectures are becoming more popular in terms of enabling and facilitating, data and system level interoperability, system level integration and collaboration over distributed environments. In this context, this paper presents two web service patterns that will help in facilitating BIM based information sharing over the web and web based collaboration using BIMs. The paper starts with review on, levels of systems integration and web services. The two web service patterns developed are elaborated on later in the paper.

Keywords
BIM, REST, SOAP, Information integration, Web services

1. Introduction

In recent years, Building Information Modelling has become an active research area within the AEC domain in order to tackle the problems related to information integration and interoperability. The key reason behind the advent of BIMs is enabling interoperability (seamless exchange and sharing of information) between various different applications used in the construction industry and throughout the lifecycle of the building. Building Information Modelling is applied in many different areas, i.e. either BIMs used as a resource to enable interoperability or Building Information Modelling has been realised as a process of managing a project through a single shared information backbone. In addition, the development in web technologies has resulted in the emergence of service-oriented architectures that makes it possible for remote applications to inter-operate using standard web interfaces. The service orientation enables loose coupling of applications over the web, i.e. several applications can communicate and interact with each other without the need of knowing the details of their working environment. Each of these applications (or data layers) that take part in such a web-based interaction in a serving form (as a data/component or application service) is known as a web service. Web services and Service Orientation are becoming de-facto standard architectural approaches for data and application interoperability and

integration. Although the trend in the software industry is towards enabling application interoperability over web services, the AEC industry is still not fully benefiting from the service oriented approaches, as the focus of the industry is still very data integration oriented. The research presented in this paper aimed to define patterns for facilitating service-oriented and model based interoperability in the AEC industry. The research started with a review on methods of software systems integration, these methods were reviewed in two stages: information and service level integration approaches. The technology review in the second stage of the research involved the review and evaluation of technologies for establishing service-oriented approaches. Two main architectural styles of service-orientation were also investigated in the second stage of the research and the results of this investigation is summarised in the paper. The following stage of the research was focused on re-visiting the exchange and sharing approaches of BIM, which were also identified as a part of our previous research. The final stage of the research involved the definition of web-service patterns in two architectural styles investigated throughout the research. The first pattern BIM SOAP Façade focuses on enabling interoperability through simplistic and coarse-grained web interface in an object-oriented nature. On the other hand, the second pattern RESTful BIM focuses on enabling interoperability through a fine-grained web interface in a resource-oriented nature. The following section of the paper summarises the background of the research, while the latter sections introduces the patterns defined.

2. Systems Integration and Web Services

Integration can be defined as a strategic approach for binding many information systems together in order to support their ability to exchange information and leverage processes in real time (Linthicum, 2003). The literature review in the area of software systems integration (Linthicum, 2003; Erl, 2004; Ruggiero, 2005; Imhoff, 2005; Hohpe and Woolf, 2003) highlighted two main integration approaches. The first approach *Information Integration* allows data and information to move between source and target systems without involving its application logic. Methods for *Information Integration* can be classified in 4 different categories as Data and Information Exchange methods, Data and Information Sharing methods, Data Transformation methods, and Relate-Search-Retrieve methods. The second approach *Service and Application Integration* allows integration through interaction of system components. Methods for *Service and Application Integration* can also be classified in 4 different categories as Remote Procedure Calls, Messaging, Distributed Objects, Web Services and Portals.

Web services can be defined as components and resources that can either be invoked over the web or reached by standard web protocols using standard messages. Web Services can provide standard web based interfaces to legacy and current applications in order to enable them to exchange data by standard protocols and expose their functionality over the standard web based interfaces. He (2003) indicated that two constraints exist for implementing the Web Services: i) Interfaces must be based on Internet protocols such as HTTP, FTP, and SMTP and ii) Except for binary data attachment, messages must be in XML. Two definitive characteristics of web services are mentioned as Loose Coupling and Network Transparency (Pulier and Taylor, 2006). As explained by the authors, in a traditional distributed environment computers are *tightly coupled*, i.e. each computer connects with others in the distributed environment through a combination of proprietary interfaces and network protocols. Web services in contrast, are *loosely coupled*, i.e. when a piece of software has been exposed as a web service it is relatively simple to move it to another computer. On the other hand, as web services' consumers and providers send messages to each other using open Internet protocols, web services offer total *network transparency* to those that employ them (Pulier and Taylor, 2006). Network transparency refers to a web service's capacity to be active anywhere on a network or group of networks without having any impact on its ability to function. As each web service has its own Universal Resource Indicator (URI), web services have similar flexibility to websites on the Internet.

Two styles of Web Services exist today: SOAP and REST. SOAP is a web service style based on using SOAP (Simple Object Access Protocol) protocol for exchanging XML formatted messages between the networks by using Hypertext Transfer Protocol (HTTP). On the other hand REST is an architectural style where the web service operates by calling various web-resources.

Today, SOAP has become the lingua franca of the web services. The SOAP protocol has two constraints: i) Except for binary data attachment, messages must be carried by the SOAP protocol and formatted by the rules of it, and ii) The description (exposed interface) of the service must be defined in Web Services Description Language (WSDL, i.e. the XML based language used to describe and locate a SOAP web service).

The terms REST and RESTfull web services have been coined after the PhD dissertation of Roy Fielding (Fielding, 2000). As explained by Pautasso (2008), REST was originally introduced as an architectural style for building large-scale distributed hypermedia systems. According to REST style, a web service can be built upon resources (i.e. anything that is available digitally over the web), their names (identified by uniform indicators, i.e. URIs) representations (i.e. metadata/data on the current state of the resource) and links between the representations. Most of the web services that are currently implemented have transformed from interfaces of the legacy systems. As mentioned in Pulier and Taylor (2006), exposing a web service mostly involves enabling the older software (or the data layer of the legacy system) to receive and respond to web message requests for its functionality.

3. Sharing and Exchange of Building Information Models

Building Information Modelling can be defined as a new way of creating, sharing, exchanging and managing the information throughout the entire building lifecycle (NBIMS, 2007). In this context, a BIM can be defined as a computable representation of all the physical and functional characteristics of a building and its related project/life-cycle information, which is intended to be a repository of information for the building owner/operator to use and maintain throughout the life-cycle of a building (NBIMS, 2006). Today, the current key efforts in the area are the Industry Foundation Classes (IFC) and the CIMSteel Integration Standards (CIS), both of which are defined using STEP description methods, and can be shared and exchanged in the three STEP implementation levels (Isikdag *et al*, 2007). The object model of the BIM (i.e. the logical data structure that defines all entities, attributes and relationships) is represented in the BIM schema. The BIM data is usually created by a CAD application and stored in physical files or databases. The data in the BIM can be shared and a snapshot of the model can be exchanged between applications when needed. Three key methods for sharing and exchanging BIMs which were identified in Isikdag *et al*, (2007) as:

- *Data Exchange* by using physical files
- *Data Sharing* through physical files and Application Programming Interfaces(APIs)
- *Data Sharing* through databases

Data Sharing through databases is usually accomplished by using a central database (also known as the Product Model Server or BIM Server) , which the users can manage the model by using database front-ends or interact with the model by using standard or proprietary database Call Level Interfaces –CLIs- (or APIs).

4. The Patterns

Traditional approaches for reaching and updating information in the BIMs are mostly focused on exchanging the BIM physical file between applications and using shared (central) databases to manage

the information within BIMs. In addition, the use of Model Views (i.e. subsets of BIM defined according to needs of a specific information exchange scenario) is encouraged for facilitating the BIM-based information management throughout the entire building lifecycle. In fact, recent R&D efforts such as SABLE (Sable Web Site, 2005) have shown that it is possible to use SOAP style web services to interact with BIMs for sharing and exchanging building information. Although the literature in the field illustrates examples of the use of shared (central) databases and SOAP style web services to manage BIMs, there has been no efforts on sharing a distributed BIM over the web. Although this approach might seem difficult technically and might cause problems regarding versioning and ownership, if managed successfully it will provide significant benefits as it will prevent data overloading in databases, enable efficient use of hardware and network resources and will be a step towards the use of Distributed BIMs. The following briefly explains two integration patterns proposed in this study. These patterns are based on SOAP and REST style web services. The first (SOAP style) pattern will make it possible to manage a BIM and Model Views in a federated manner. The second (REST style) pattern will enable the exposition of all objects in a BIM as a set of web representations. This exposition will enable the users to work directly with every object of the BIM without the need for knowing where, how and in what form the BIM is stored.

4.1 BIM SOAP Façade

The need:
Currently, BIM based collaboration usually make use of single shared central database (and rarely 1-2 different Model Views that reside in the same shared central database). In parallel with the need for/trend towards collaborative way of working and the increase of the data volume in BIMs, the use of a single shared database with a single model (and 1-2 Model Views) will not be technically feasible, as it will cause information overload on a single database and the overload of network traffic. In addition, as there will be different BIMs and Model Views in the process, real time information mapping needs will also increase. In order to efficiently manage BIM based collaborative environments, the use of (web) Façades is becoming an inevitable need.

The pattern:
The BIM SOAP Façade pattern can be defined as SOAP exposition of multiple BIM databases based on the principles of Façade pattern. The pattern will enable the management of a set of distributed shared BIM databases over SOAP web services. Before getting deeper into the pattern, it might be good to summarize the principles of the Façade pattern itself. According to Gang of Four (Gamma *et al*, 1995), the Façade pattern provides a unified interface to a set of interfaces in a subsystem. As explained in Shalloway and Trott (2002), when one needs a new (simplified) way to interact with a system (resource) or needs to use a system (resource) in a particular way (such as using a 3-D drawing program in a 2-D way), the Façade pattern can be used to build a new method of interaction. Façades can be used not only to create a simpler interface in terms of functions/operations, but also to reduce the number of objects that a client object must deal with (Figure 1).

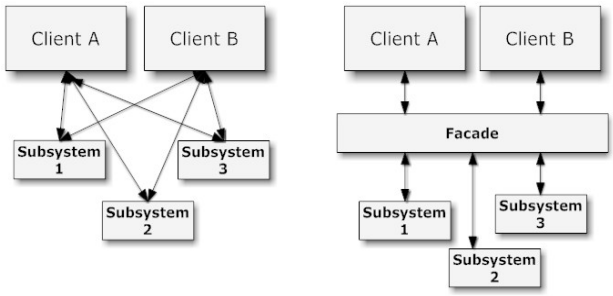

Figure 1: The Implementation of Façade Pattern

The BIM SOAP Façade pattern takes the principles of Façade pattern one step further, i.e. to web services level, and stipulates it for the BIM domain (Figure 2). As the Façade (type) classes are created in object oriented systems, SOAP Façades can also be created depending on the similar architectural principles. The only difference will be that the interface of the SOAP Façades will be defined in WSDL. In this case, the SOAP Façades will be defined as (high level) web services that are using low level web services (here referred to as SOAP interfaces). These (SOAP interfaced) low level web services can then interact with any type of system components (including databases). The next stage of the definition involves the implementation of the SOAP façade idea into the BIM based system architecture.

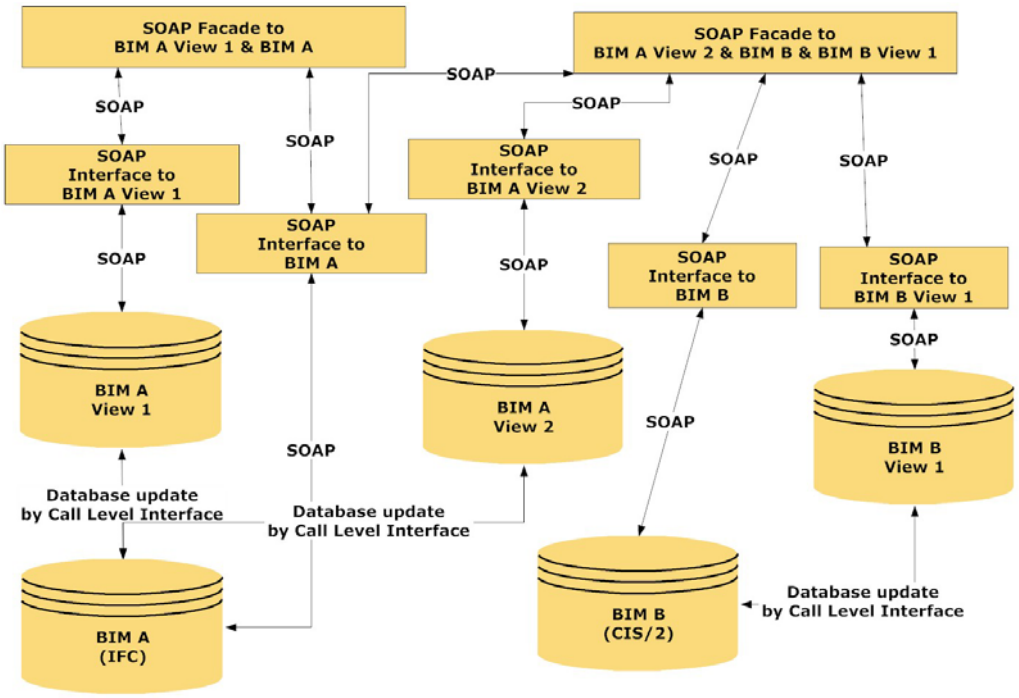

Figure 2: The implementation of BIM SOAP Façade pattern

As seen in Figure 2, in a system designed according to BIM SOAP Façade, the BIMs will reside in shared databases. The Model Views can be generated on-the-fly or stored in the same database with the BIM or even in different databases. Here we will focus on the most loosely-coupled scenario, where Model Views reside in separate databases. In this case, the BIM and Model Views residing in different databases will be synchronised by Call Level Interfaces -CLIs- (i.e. Database APIs), when any change in each of them occurs. In a service oriented architecture all data in the BIM and Model Views will be exposed through low level web services (i.e. SOAP interfaces). Finally, the SOAP Façades will act as the front-ends of the SOAP interfaces, as they will provide simplified and function oriented methods to interact with BIMs and Model Views. Also this will help in reducing the number of SOAP interfaces the client has to deal with when interacting with the models. The SOAP Façades will also enable on the fly model mapping between two different information models and between information model and Model Views (as they can combine the interfaces of several information models and model views). The functions in the SOAP Façades will simply interact with the BIMs and Model Views as they will eliminate the need for information on where the BIM resides, in which database the BIM is stored, and even the users needing to know about the object model of the BIM. For example, an architecture that implemented the principles of BIM SOAP Façade will easily fulfil a SOAP request for mapping the information about columns of a steel building from a CIS/2 model into an IFC Model /Model View without requiring the user or any other information (such as classes to be queried and transferred, location of the models, how they are stored and managed, etc.). The BIM SOAP Façade pattern provides an easy to use, simplified mechanism for management of multiple BIMs over the web. The SOAP Façades in the pattern can be thought of as coarse-grained

interfaces to BIM objects, and SOAP interfaces can be thought of as finer-grained interfaces to BIM objects. In fact, as the SOAP interfaces will be developed in parallel with the process related requirements of the application domain, they will rarely (or never) allow with one-to-one interaction with every single BIM object.

4.2 RESTful BIM

The need:
In order to enable this one-to-one interaction with BIM objects, we will need the finest-grained interface (similar to a database CLI) in the web level. Developing this interface for every BIM and Model View (especially when the model views are at instance level or dynamic) will be very difficult and require specific programme functions to generate SOAP interfaces on the fly covering every class of the model. One solution to this problem is entirely changing the architectural style, i.e. leaving the SOAP approach, and to focus on RESTful, resource oriented approaches for creating the finest-grained web interfaces to the BIM and Model Views. The second pattern, RESTful BIM, focuses on the exposition of all BIM objects as web resources. Thus, deriving from the principles of the RESTful approach we named this pattern as RESTful BIM.

The pattern:
As depicted in Figure 3, the exposition of BIMs as REST style web services can be accomplished by several methods. The first method is storing BIMs in a relational database and automatically exposing every object of the model as a representation in a REST Web Service. The second method is storing BIM in Model Server and automatically exposing every object of the model as a representation in a REST Web Service. The third method is using REST frameworks to manually define resources to be exposed. The first and second methods can be named as REST Database Exposition. The third method can be named as manual resource creation.

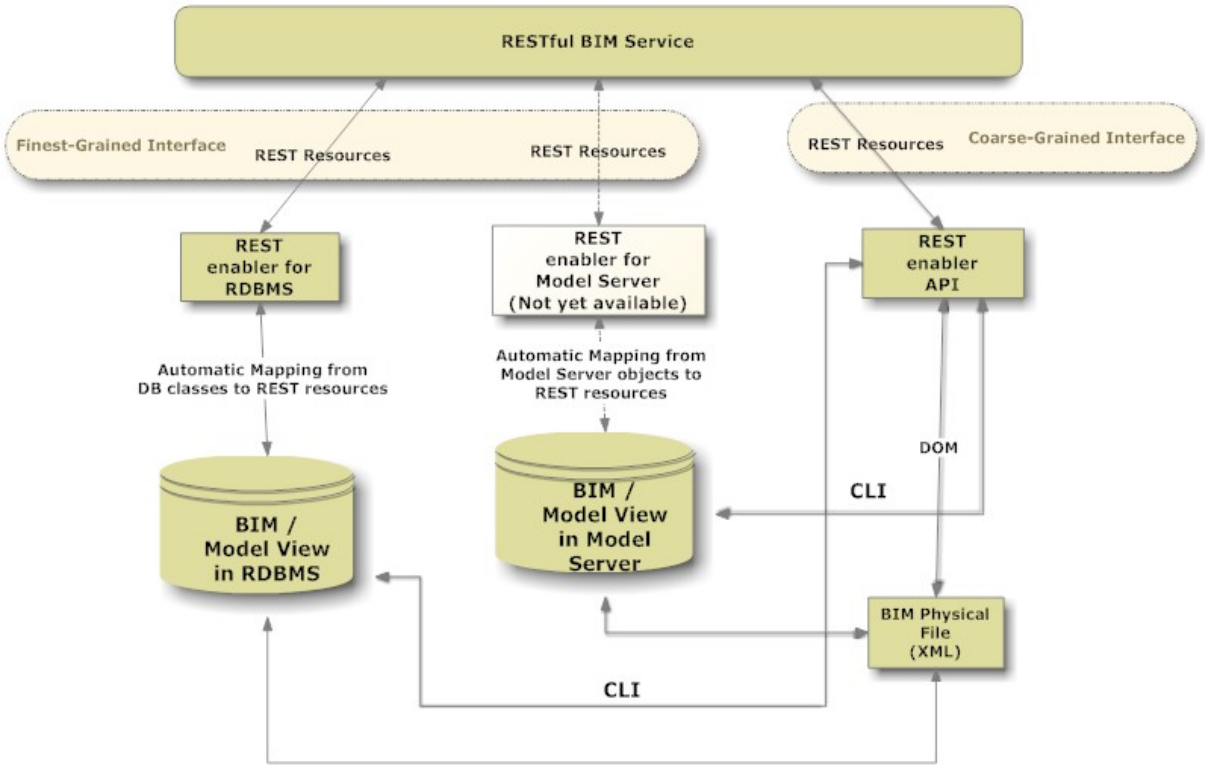

Figure 3: The Implementation of RESTful BIM pattern

Exposing every BIM object as a REST resource will provide the finest-grained interface to the BIM, which will enable the client application to directly interact with the BIM objects. Recent technological developments have resulted with the interfaces that make it possible to automatically map the entities in a relational database to REST resources. Thus, today it is possible to automatically generating REST resources from a BIM that is residing in a relational database. This process is very straight-forward and simple using REST enablers such as sqlREST (sqlREST, 2008).

Once implemented, the sqlREST generates a resource for each entity (table) in the database and each instance (object) in the tables. Every kind of resource has an XLink attribute href with an URL. For example, when an IFC model residing in a relational database is exposed, a highest level set of resources will be provided as:

<resource>
<IfcColumn xlink:href="http://localhost/sqlrest/ IfcColumn/">IfcColumn</IfcColumn>
<IfcBeam xlink:href="http://localhost/sqlrest/ IfcBeam/">IfcBeam</IfcBeam>
…
</resource>

If we move one step further in the resource environment and navigate to the IfcColumn resource, by entering its URI (i.e. http://localhost/sqlrest/IfcColumn), a list of columns in the model will be provided as the list resources, where a new URI is presented for each object in the model. The URIs will be generated based on the Globally Unique Identifier GUID of the objects in the IFC model. In this level, the resource list will be provided as:

<resources>
< resource xlink:href="http://localhost/sqlrest/ IfcColumn/2j6LMcUOL/">2j6LMcUOL </resource>
< resource xlink:href="http://localhost/sqlrest/ IfcColumn/1avLMcAS2/">1avLMcAS2 </resource>
...
</resources>

Although it is very straight-forward to automatically map relational database classes to REST resources, no tools are currently available to accomplish a similar mapping from BIM Product Model Servers to REST resources.

Thus, in order to generate REST resources from the BIM that is residing in a Model Server, a REST enabler API should be used. The resource creation and mapping from BIM classes to REST resources will be manual (i.e. by program code), and thus the level of granularity will be lower (as this mapping will mostly be domain or application oriented.). In contrast to the automatic exposition of BIM in Model Servers, there are tools for creating the resources manually, and the most commonly know tool is RESTlet (RESTlet, 2008).

On the other hand, if the BIM is residing in an XML file, it can either be imported into a RDBMS or a Model Server and the representations (i.e. model data) can be reached from on-the fly generated resources. Another option in this case is generating resources and reaching representations by using DOM and REST API as interfaces.

5. Conclusion

Service and resource oriented architectures will shape the next generation of the World Wide Web. The common view in the software industry denotes that the biggest strength of the Web 2.0 will lie behind the service and resource oriented architectures, which will enable seamless access to data and application services regardless of their location and the environment they operate in. In fact, the AEC industry

(although having their own valid reasons behind it) is still very much focused in establishing data and process level integration. For example, the industry professionals and researchers are still very engaged in developing schema standards for exchanging (geometric and semantic) building information (i.e. IFC, CIS2). Another research trend in the industry is towards enabling efficient building lifecycle management through the use of shared digital building models, which are based on these schema standards. These two trends together currently termed as Building Information Modelling.

In summary, the interoperability trends in the software industry is more focused towards service level integration, while the AEC industry is currently more focused on the data level integration and in enabling process integration through information integration. The study explained in this paper provides an overview of how service oriented architectures can be built in data-intensive knowledge domains such as AEC. In this context, the patterns explained here might contribute in bridging the integration and interoperability focuses of the AEC and software industries. Two patterns that were developed during this research portray that it is technically possible to build up highly customisable BIM based service oriented architectures using both SOAP and REST styles. Although the SOAP style is more mature in terms of current implementations (and current research have shown evidence of SOAP style implementations using BIMs), the REST style (resource-oriented) architectures also provides a lot of opportunities as it can enable finest-grained interactions at the web level.

6. References

Erl, T. (2004). *Service-oriented architecture: a field guide to integrating XML and web services.* New Jersey: Prentice Hall PTR.

Fielding, R. T. (2000). "Architectural styles and the design of network-based software architectures." PhD Thesis. Dept. of Information and Computer Science, University of California, Irvine.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*, Reading, Mass.: Addison-Wesley.

He, H. (2003). "What Is Service-Oriented Architecture", Online at http://webservices.xml.com/pub/a/ws/2003/09/30/soa. html Accessed on July 21, 2004

Hohpe, G., and Woolf, B. (2003). *Enterprise integration patterns: designing, building, and deploying messaging solutions.* Indianapolis: Addison-Wesley.

Imhoff, C. (2005). "Understanding the Three E's of Integration EAI, EII and ETL", *DM Review Magazine*, Online at http://www.dmreview.com/issues/20050401/1023893-1.html Accessed on June 10, 2005.

Isıkdag, U., Aouad, G., Underwood, J., and Wu, S. (2007). "Building Information Models: A review on storage and exchange mechanisms", *Proceedings of CIB W78 2007*, Maribor, Slovenia Editor: Daniel Rebolj, pp.135-144.

Linthicum, D. S. (2003). *Next generation application integration: from simple information to web services*. Indianapolis: Addison-Wesley.

NBIMS. (2006). "National BIM Standard Purpose", US National Institute of Building Sciences Facilities Information Council, BIM Committee, Online at: http://www.nibs.org/BIM/NBIMS_Purpose.pdf Accessed on January 15, 2007.

NBIMS. (2007). "National Building Information Modeling Standard Part-1: Overview, Principles and Methodologies", US National Institute of Building Sciences Facilities Information Council, BIM Committee, Online at: http://www.facilityinformationcouncil.org/bim/publications.php, Accessed on October 12, 2007.

Pautasso,C., Zimmermann, O. and Leymann, F. (2008). "Restful web services vs. "big" web services: Making the right architectural decision", *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pp.805-814.

Pulier,E., and Taylor, H. (2006). Understanding Enterprise SOA, Manning Publications , Greenwich, USA.

RESTlet. (2008) "RESTlet: Lightweight REST Framework" Online at http://www.restlet.org/ Accessed on December 16, 2008

Ruggiero, R. (2005). Integration Theory, Part 1. *DM Review Magazine*, Online at http://www.dmreview.com/dmdirect/20050812/1034584-1.html Accessed on November 5, 2005.

SABLE Web Site. (2005). Online at: http://www.blis-project .org/~sable Accessed on February 12, 2007.

Shalloway, A., and Trottt, J.R. (2002). *Design Patterns Explained A New Perspective on Object-Oriented Design*, Addison-Wesley.

sqlREST. (2008). "sqlREST Enabler for Web Services", Online at http://sqlrest.sourceforge.net/ Accessed on December 10, 2008.