buildings

MDPI

Article

An Analysis of BIM Web Service Requirements and Design to Support Energy Efficient Building Lifecycle †

Yufei Jiang [1,*], Xiao Liu [1], Fangxiao Liu [2], Dinghao Wu [1] and Chimay J. Anumba [2]

[1] College of Information Sciences and Technology, The Pennsylvania State University, University Park, PA 16802, USA; xvl5190@ist.psu.edu (X.L.); dinghao@psu.edu (D.W.)

[2] Department of Architectural Engineering, The Pennsylvania State University, University Park, PA 16802, USA; fzl116@psu.edu (F.L.); anumba@engr.psu.edu (C.J.A.)

* Correspondence: yzj107@psu.edu; Tel.: +1-814-880-2305

† This paper is an extended version of our paper published in *Proceedings of the 2012 ASCE International Conference on Computing in Civil Engineering, Special Session on BIM Computer Science Fundamentals*, 2012 [1].

Academic Editor: Tanyel Bulbul

Received: 24 September 2015; Accepted: 25 April 2016; Published: 29 April 2016

Abstract: Energy Efficient Building (EEB) design, construction, and operations require the development and sharing of building information among different individuals, organizations, and computer applications. The Representational State Transfer (RESTful) Building Information Modeling (BIM) web service is a solution to enable an effective exchange of data. This paper presents an investigation into the core RESTful web service requirements needed to effectively support the EEB project lifecycle. The requirements include information exchange requirements, distributed collaboration requirements, internal data storage requirements, and partial model query requirements. We also propose a RESTful web service design model on different abstraction layers to enhance the BIM lifecycle in energy efficient building design. We have implemented a RESTful Application Program Interface (API) prototype on a mock BIMserver to demonstrate our idea. We evaluate our design by conducting a user study based on the Technology Acceptance Model (TAM). The results show that our design can enhance the efficiency of data exchange in EEB design scenarios.

Keywords: BIM; web service; energy efficient building lifecycle

1. Introduction

The topic of facilitating data exchange between different design and simulation tools in the Architecture, Engineering, and Construction (AEC) industry is gaining increasing interest. One notable fact is that AEC has its specific ways to adopt the assistance from computer and information sciences in the whole building design life cycle [2]. Instead of taking advantage of a web-service-based solution, the workstation-based Computer-Aided Design (CAD) approach, which dates back to the 1980s, is still widely used by the AEC industry [2]. In current AEC workflows, simulation processes are not fully integrated into the design and modeling process [3]. In most cases, engineers need to manually build models for simulation, which duplicates work that has already been performed. It was reported that approximately 80% of the effort required to run a simulation is spent on building models [4]. With Building Information Modeling (BIM), attributes and data can be attached to a model, which potentially allows integrated analysis and simulation, especially for Energy Efficient Building (EEB) projects [5,6]. However, most of the analysis and simulation software are developed by experts in different domains, and their data representations are different in nature, which puts a constraint on model reuse and data exchange among tools.

Web service, as a software system to launch functions and exchange data over the web, is a powerful approach to solving the interoperability problem of BIM. There are three kinds of commonly used web service interfaces: remote procedure call (RPC), service-oriented architecture (SOA), and representational state transfer (REST) style architecture [7]. RPC allows two distributed remote heterogeneous systems to call the functions or methods of each other. However, their interactions and function calls are environmentally sensitive, which relate to the implementations in a specific programming language. This design pattern violates the principle of loose coupling in software engineering. To overcome the design defect, SOA was proposed and now SOA has been widely adopted by many web service providers [8]. Instead of relying on a specific implementation, SOA is driven by messages (events) [9]. One of the disadvantages of this architecture is its complexity [10].

The third method is a set of web interfaces based on the REST style architecture, or so called RESTful interface. Implementing web services based on the RESTful interface is most suitable for BIM. The strength of the RESTful interface is its simplicity. In this architecture, the representation of all resources would be abstracted as a Uniform Resource Identifier (URI). The resources here do not just refer to the files or other tangible objects; it also could be a composition of several files, a table, a query, a result set of queries, and any concepts. The web interface is essential to the usability of a web service. As previously mentioned, simulations for different scenes and purposes are highly domain-specific. RESTful web interfaces can effectively isolate and hide internal implementation and domain knowledge from end users. Thus, the domain experts can fully focus on their areas. They can invoke other services and retrieve data from other parties as utilities, without knowing their details. Remote data models could be manipulated by invoking RESTful web interfaces. Clients are loosely coupled with the web service provider, which makes the procedures like data model merging, partial model query, and events notification flexible.

BIM, which is an emerging methodology to facilitate information exchange among the whole AEC community, is not just designed for EEB projects. However, in this paper, we focus in particular on the EEB lifecycle. An EEB project consists of tasks ranging from building constructions to energy analysis. There are many existing simulation tools provided by different parties that assist experts in various domains to perform analysis, such as EnergyPlus and DOE-2 [11]. However, to effectively use them, different data have to be taken as inputs. Figure 1 shows some examples of simulation tools and their required data models [12]. To get the required data, we need to trace back to some source software. Assume an expert wants to model the energy consumptions in EnergyPlus, she will need the data in EnergyPlus Input Data Files (IDF). To extract these data, she is supposed to find the IFC (Industry Foundation Classes) file which is generated by the ArchiCAD. However, the process depends on a number of systems: the EnergyPlus software, an adapter for data format transform, and the source software ArchiCAD. It is quite challenging, without an automated workflow, to figure out the whole process and then acquire the permission of these systems to get the data originated and transformed. To make this easier, if we adopt the RESTful design, the IDF file can be generated and stored when the IFC file is created with the ArchiCAD. It will be provided as a resource online directly for the expert to retrieve whenever he make an HTTP request. The same idea can be applied to the use cases with other simulation tools.

Previous research has presented some broad overview on possible applications of web service in the AEC community, the building lifecycle, and BIM. However, the study that focuses on RESTful web service, which is adopted by more and more leading web service providers, is still inadequate. Moreover, as an emerging area in the AEC area, the EEB lifecycle support gradually involves more participants and digitalized data, which requires a new collaboration pattern than ever before. Accordingly, a study on the requirements and the design of RESTful BIM web service to support EEB lifecycle is necessary. Our goal is to use the proposed requirements and designs to investigate existing platforms and the technology adoption status within the EEB lifecycle supporting area. In addition, we also implement an EEB BIM data RESTful Application Program Interface (API) prototype and conduct a comparative study on some existing platforms and systems. We take advantage of the Technology Acceptance Model (TAM) as our methodology to perform this comparison and evaluation. More specifically, in this paper, we make the following contributions:

- We identify the requirements of RESTful BIM web services to support the EEB lifecycle.
- We propose a set of design models of RESTful BIM web services on different abstraction levels.
- We implement a set of RESTful API on a mock BIMserver as a prototype.
- We compare and discuss our design with existing technologies and implementations by conducting a user study under the guidance of TAM.

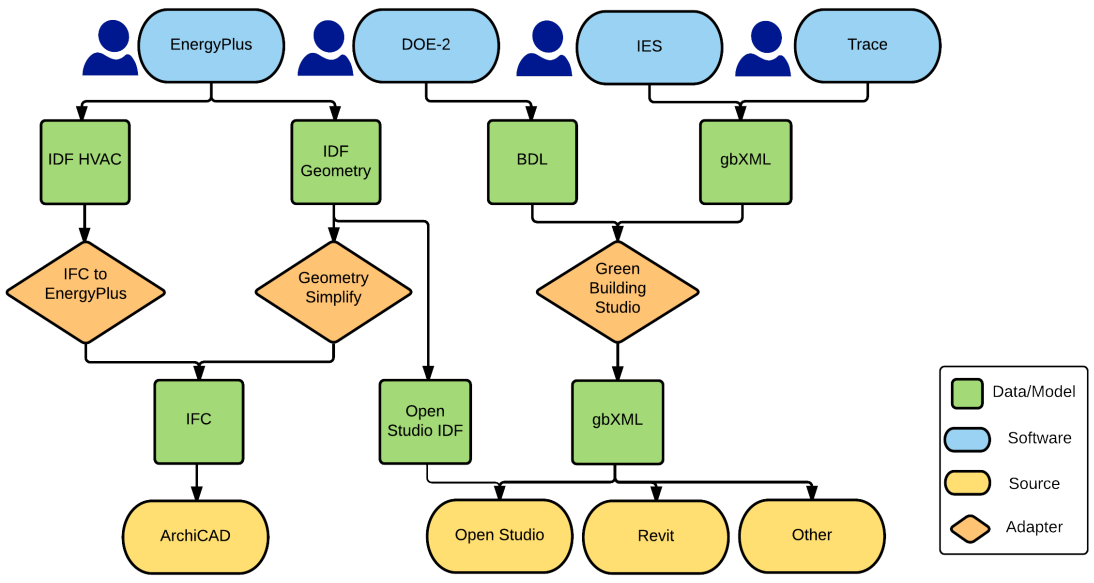


Figure 1. Data exchange in simulation tools of Energy Efficient Building (EEB) projects.

This paper is organized as follows. First, we introduce previous efforts on BIM-based building life cycles and web-service-facilitated BIM in Section 2. Then, we present the details of requirements for RESTful BIM web service to support EEB in Section 3, and propose our designs in Section 4. Section 5 introduces our implementation. We evaluate and discuss our approach in Section 6. We conclude the paper in Section 7.

2. Background and Related Work

2.1. Building Information Modeling

Data and information fragmentation, redundancies and inefficiencies have been challenging the AEC community in recent years, and BIM is being widely adopted in the industry to foster communication and collaboration [13]. According to the National Building Information Modeling Standard [14], Building Information Modeling is defined as "a digital representation of physical and functional characteristics of a facility", and it serves as "a shared knowledge resource for information about a facility forming a reliable basis for decisions during its lifecycle from inception onward". BIM acts as an integrated platform for team members to share and exchange project information through comprehensive object-oriented building models [15,16]. It utilizes CAD that ties all building components together with rich information embedded [17], and can be implemented in the lifecycle of a project representing the plan, design, construct and operate phases [18].

2.2. *BIM Applications*

2.2.1. BIM to Support Building Life Cycle

The traditional AEC industry relies on two-dimensional (2D) printed drawings and CAD files that store 2D geometry information of building elements [19,20]. With the adoption of 3D graphic prototyping technologies, 3D CAD emerged and was quickly introduced to the construction industry [20]. BIM is the latest generation of object-oriented CAD systems (OOCAD) and is capable of assigning both graphic and non-graphic attributes to building elements [19]. In addition to supporting collaborative work processes, BIM has been successfully implemented in other areas in the AEC industry. The use of BIM goes beyond the planning and design stage of the project throughout the building life cycle: it can be utilized as a tool to facilitate sustainable design and analysis [21], automatic safety checking and hazard identification [22], and automated construction process measurement [23]. It bridges the building information across the design teams to construction teams and the operators by allowing each party to retrieve, append and modify information during their period of contribution. Figure 2 shows 25 BIM applications throughout the lifecycle of a building (plan, design, construct and operate) which are developed by the Computer Integrated Construction Research program at the Pennsylvania State University [24].

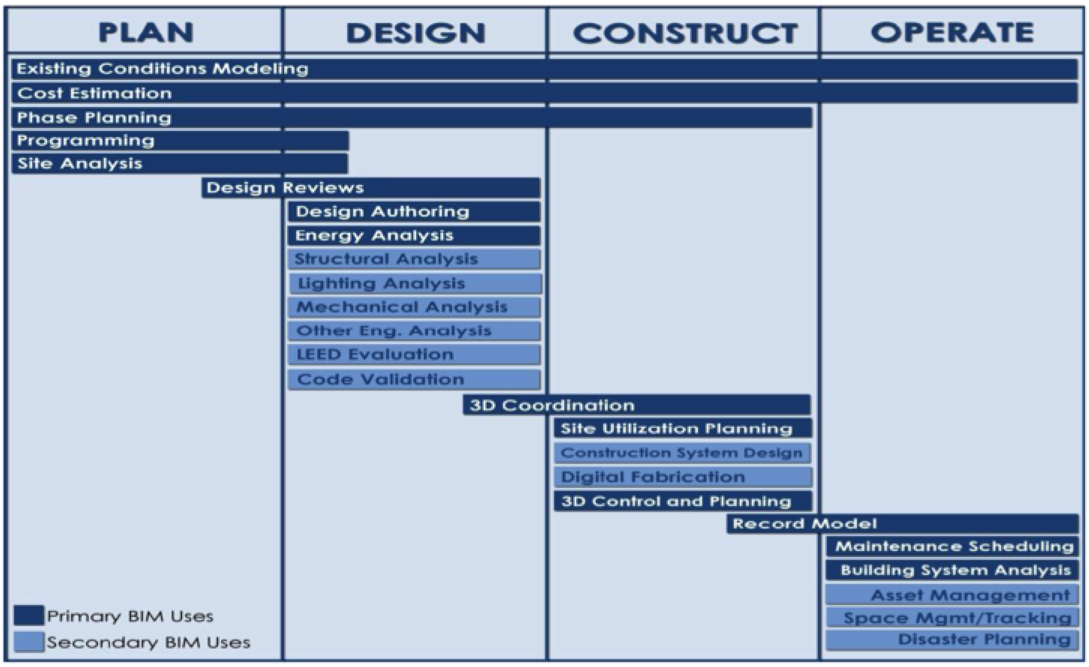


Figure 2. Twenty-five Building Information Modeling (BIM) applications.

2.2.2. BIM to Support Energy Efficient Building Design

Several previous efforts have tried to integrate building information models with energy simulation tools. Bazjanac [25] first disclosed the impact of a National Building Information Model Standard (NBIMS) on building energy performance simulation and analysis. He [26] further proposed a new methodology to semi-automate Building Energy Performance (BEP) simulation. This methodology demonstrates benefits when applied to simulations with EnergyPlus. Young *et al.* [27] surveyed that the lack of software interoperability and functionality is rated as one of the greatest obstacles to improving the business value of BIM. Moon *et al.* [28] conducted case studies to evaluate the interoperability of a BIM based architecture model and performance simulation programs (e.g., EnergyPlus). Jiang *et al.* [1] discussed a set of core BIMserver requirements to effectively support information sharing in energy efficient retrofit projects. These requirements have incorporated the

BIM functionalities into efficient energy building designs and filled in the gap between the original BIM standard and other platforms. In addition, they developed a tool called QueryGenerator to automatically generate BIMserver Java query code [29]. Yu *et al.* [30] investigated a methodology to integrate BIMserver with OpenStudio to efficiently exchange energy efficient building data in a uniformed manner which provides a convenient tool for data transformation and make it easier for information exchange. Liu *et al.* [31] proposed a framework to embed change management (CM) in BIM. A prototype system platform is implemented based on this proposed framework.

BIM has become a critical realm in the AEC industry, and its implementation will benefit owners, contractors, designers and operators [32]. A well-designed BIM system can help save lifecycle cost and cycle time, improve sustainable performance, and reduce human resources [33,34].

3. Requirements

In this section, we discuss the requirements of the RESTful BIM web service to support EEB lifecycle from end user perspectives.

3.1. Information Exchange Requirements

In EEB design and maintenance, there are a substantial number of participants and stakeholders. Accordingly, scalability is the first concern in the information exchange requirements. Simple Object Access Protocol (SOAP) is one of the existing designs for communication; however, the performance of the server drops when the whole system scales up [35]. A typical BIM-assisted project needs collaboration between architects, engineers, contractors, and project managers. It is possible that conflict checks and clash detections will be performed round after round whenever updates happen. The BIM web service should take advantage of the stateless feature of REST to avoid repeated construction and improve the performance. "Stateless" means that a BIM web service does not store its clients' states. The stateless nature of RESTful service improves the scalability of the server and allows application-aware caching intermediaries, which make the client-side access more efficient.

The second requirement of facilitating information exchange in EEB lifecycle is resource accessibility. A key to building a RESTful BIM web service is to identify and abstract appropriate resources that will be exposed to the end users or their clients. A resource is an object with a type, associated data, relationships to other resources, and a set of methods applied to it. BIM has its data model to organize data from different parties. In EEB lifecycle supporting scenarios, the data models do not only contain the geometry information of a building, but also have airflow simulation information, lighting information, Heating, Ventilating, and Air Conditioning (HVAC) information, and geographic information. By adopting the RESTful design pattern, each single resource should map to a Universal Resource Identifier (URI). Resources in the same category should be grouped into collections. Each collection is also a resource, which should also be available by mapping to a URI. This mapping can be done in a recursive bottom-up manner to make all resources accessible. In the current AEC community, there are already some general designs of BIM data models following OO principles. The RESTful interface can be abstracted from these "objects" and the relationship between objects. Additionally, since the exposed interfaces can be accessed by anyone along the EEB lifecycle, authentication should also be considered in the Application Program Interface (API) engineering. The notable point is that authentication implies client's state tracking, which may contradict RESTful stateless principle. A solution is to adopt some open standard authentication mechanisms like OAuth [36], which tracks client's state in a different layer [37]. In practice, OAuth has been reported to work well with REST [37]. In addition to that, all communication should take place over certified Hypertext Transfer Protocol Secure (HTTPS) protocol. Different access control policies should be applied to people as the roles they play vary in the whole lifecycle.

3.2. Distributed Collaboration Requirements

The topic of supporting distributed collaboration has been studied for two decades since the modern enterprise has more and more "dynamic boundaries" and consists of a significant number of "autonomous business units" [38]. Those large engineering and manufacturing enterprises found out that the paper-based collaboration pattern was dramatically bogging down the whole workflow. An important outcome of the research in this area is Product Data Management (PDM), which could be dated back to the mid 90s [38]. The core idea of PDM is to track and manage "meta-knowledge" (e.g., design release management, change management, and impact analysis) through the whole product lifecycle [39]. To support distributed collaboration, a RESTful BIM web service should at least play a role of PDM first to support distribution collaboration.

More specifically, distributed collaboration requires a well designed mechanism to synchronize contributions made by all teams. For example, it is highly possible that the geometry information is updated after clash detection or energy evaluation. Under this circumstance, the geometry needs revision on the basis of the supporting report. However, malfunctions may exist due to desynchrony in other remote on-going development version branches. It would be problematic for geometry engineers to tell upon which version the clash detection report is based, if more than one commitment of updates in the geometry is submitted into the BIMserver.

To tackle the malfunction caused by desynchrony, the BIM web service should retain a version control system to enable each side to get informed on the change logs and the current version status. By indexing all the changes in the data model, a RESTful BIM web service should support the roll-back feature. With a version control system, a BIM web service can adopt incremental backup which is faster than a full backup. More advanced features, such as making the version control system decentralized, will make a project iterate faster. However, since this advanced feature needs every participant maintain their branches of a full copy of the BIM data, more training is required. Thus, a decentralized version control system is an optional requirement.

Beyond that, a BIM web service must overcome those traditional PDM system shortfalls that might be exposed by emerging challenges in EEB lifecycle supporting area. First, a traditional PDM still uses a file or a document as a basic unit of querying and storing. However, in EEB design and analysis, a class or an abstract entity (e.g., a space, a boundary, or a facade) is a basic unit to work on, which is hard to map to a specific document. A BIM web service must use an additional abstract layer to make files and documents hidden from its clients. Second, a traditional PDM integrates and manages the product data in a static way which is like a warehouse. In the scenario of EEB design, integrated BIM models make some global analysis and optimization possible. A BIM web service host should have the capability of generating new data and knowledge in the process of merging fragmented data from different departments. At last, traditional PDM targets intra-enterprise communication and collaboration, instead of inter-enterprise collaboration. The nature of EEB design is inter-enterprise based. As we have mentioned, different enterprises use heterogeneous systems and data schema. Using the web service with simple and unified interfaces to ease the data exchange between the distributed participants who work on heterogeneous local systems is a must.

3.3. Internal Data Storage Requirements

Internal data handling can be addressed using file-based and model-based approaches. These two approaches have different strengths and shortcomings. A file server stores files statically to provide data persistence and retrieval service. A model-driven server essentially is a web service server. To be more specific, it is a Data-As-a-Service server. In this context, data refers to models. It does have a data persistence layer. However, it will not directly expose the information in the data persistence layer as "data" to its clients. It parses the data in the file first and uses these data to construct its own pre-defined data structure and maintain a comprehensive model as its internal data storage. This process is similar to a marsh up, which takes different web services as sources to render them into a single new service [40].

Comparing web service servers and file servers, we can find out that a file server's structure can be quite clear and simple, which is easy to implement. The underlying database could be a classical relational database. Adopting such a file server will not impact current energy-efficient retrofit workflows. A file server does not necessarily need a unified file format or a format translator. It stores files and facilitates the collaboration of sections of design and simulation teams that share the same file format. The basic unit of a file server is a file, while sometimes users have to query data that is distributed among different files. For example, a simulation tool may need the height and width of all windows. However, a file server may not be able to parse the information it stores semantically. It also cannot support such data export or other advanced functions such as model merging and partial model query [5]. A model-driven server can compensate for the file server's drawbacks. The advantage of this method is that some advanced functionalities (e.g., clash detection and model merging) could be performed on the stored model on the fly [41].

3.4. Partial Model Query Requirements

To query data from a RESTful web service, a request is sent out by an HTTP call. This HTTP call usually carries a payload that contains the data schema of interest in some lightweight formats like JavaScript Object Notation (JSON). The JSON files could be generated by applications like OpenStudio automatically. JSON files are easy to be parsed by a machine. However, it is not an easy task for an end user to compose a long piece of JSON data schema to perform a query, especially a partial model query. Thus, there are two major requirements on the issue of partial model query composing. The first one is to automatically generate queries from Model View Definition (MVD). The other one is to have a domain query language.

3.4.1. Automated Query Generation from MVD

Automated Query Generation should take advantage of some existing documents that describe resource data schema as input. Those documents could be written in natural languages or a domain language that is familiar to domain experts. MVD meets these criteria, which makes it a potential candidate of query generation input. An MVD is defined as a subset of the Industry Foundation Classes (IFC) [42] schema to satisfy one or several specific data exchange requirements. For example, the "Concept Design MVD" has been developed to facilitate selective extraction or importing of relevant building information [26]. Whereas an MVD is independent of a particular IFC release, it is implementation dependent. It is critical to automate the data exchange process by generating queries from MVDs automatically. The automated generation of queries from MVDs will facilitate, simplify, and streamline information exchanges between tools built around BIMservers, and enable flexible queries to BIMserver. This functionality will also enable a BIMserver to selectively export relevant information for one or multiple tools.

3.4.2. Query Language

The other perspective to address the problems in partial model query is to develop a domain-specific high-level declarative query language. Some general query languages have already been available (e.g., Structured Query Language SQL). However, it is difficult to achieve the partial model query of BIM data models directly by using a general query language [43]. Thus, one of the requirements is to move closer to a seamless system integration using the fundamental concepts behind the ontological engineering which helps to define domain-specific concepts. Current integration efforts are usually based on how information are interpreted which is the syntax to describe the information. Under this circumstance, an AEC industry query language should be designed for such a purpose. However, with the increasing complexity of the information, we need to completely and correctly exchange information among heterogeneous systems, and the language itself should be unambiguous. For the areas that have special requirements on query and design, adopting a domain-specific language (DSL) has been proven as a viable solution. An example is a Very high-speed integrated circuit Hardware Description Language (VHDL), a hardware description language [44]. To design an AEC domain specific query language, one should start from a query language that can execute partial queries on an IFC/MVD compliant data model. Dating back to 2003, Predictive Modeling Query Language (PMQL) is an effort that tried to achieve partial model query of IFC files [43]. Generalized Model Subset Definition (GMSD) goes one step further to support better "partial model query on specific model view," and less "request-response cycles" [45]. Recent years have seen other designs like Building Information Model Query Language (BimQL) [46] and Query Language for 4D Building Information Models (QL4BIM) [47].

In summary, a domain specific query language is needed and should have at least some of the following characteristics:

- Compatible with MVDs: Queries should be able to be (semi-)automatically composed based on the existing MVDs,
- Compatible with the RESTful interface,
- Easy for normal users to construct queries, and
- Hidden from the users behind GUI and support transaction queries with ACID (atomicity, consistency, isolation, and durability) [48], and able to reuse routine queries.

4. Design

In this section, we present three design patterns of RESTful BIM web service for developers from three different levels. More specifically, from the low level to the high level, they are library component as a service, building information model as a service, and application as a service.

4.1. Library Component as a Service

A BIM service system consists of many different components. In this design pattern, we create each component inside a BIM service as a standalone RESTful web service. Such components could be authentication service, security plugin, or query engine. The benefits brought by such a design style are multiple-fold. First, RESTful style architecture decouples each component of a system. Each component interacts with each other via a set of RESTful web interfaces, which provide better encapsulation than library calls [49]. Additionally, the dependency between components is minimized. Therefore, the whole system is easy to decompose, integrate, refactor and migrate. When necessary, it is easy to change web interfaces into library calls. However, on the reverse side, a system built based on library calls is hard to be split into several standalone RESTful web services. Third, because each component is designed to be standalone RESTful web service, functional tests could be done earlier. Thus, it is feasible to launch part of the service ahead of the completion of other components, which meets the requirements of agile development and quick delivery strategy. Fourth, because components interact with each other by HTTP calls, they can be deployed on different distributed clusters, which enhances the scalability and portability of the whole service. EEB lifecycles involve considerable participants and stakeholders at different stages, which is a distributed workflow in nature. A RESTful-web-service-enabled distributed system facilitates such a distributed workflow.

Figure 3 illustrates a library component as a service design. In this example, a RESTful BIM web service consists of a web server, a data server, and other two functional components (component 1 and component 2). From the figure, we can see that the RESTful API is not only used for exposing the available service for different clients, it is also the communication channel of each component inside the box of the BIM web service system. For example, the web server receives a request to fetch the temperature data of the first floor of a building in degrees Celsius. After parsing this request, the web server will first send out an HTTP request to hit the data server to retrieve the temperature data. The data server then replies an HTTP response that contains temperature data in degrees Fahrenheit. In this case, the web server sends out another HTTP request to component 2 for calling the Fahrenheit-to-Celsius conversion service. At last, the web service returns the correct data to the client.

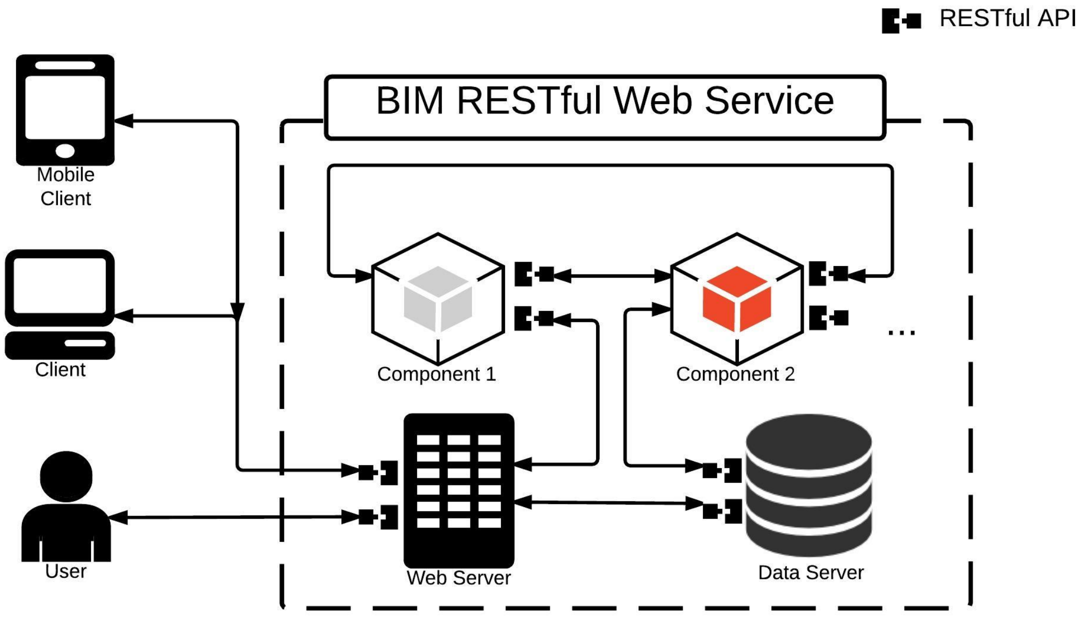


Figure 3. Library component as a service.

4.2. Building Information Model as a Service

In the design pattern of building information model as a service, there is a BIMserver playing the role of central data hub. Each application is built surrounding this BIM data hub. Each application interacts with central BIMserver via RESTful APIs. Figure 4 illustrates the design of the building information model as a service. The BIMserver is not only a data host but also a data pipe among all applications built upon it. Taking advantage of this model, a workflow with N applications merely connects these applications with the central BIMserver, rather than building $N(N-1)$ directed data pipes. Because of the adoption of HTTP-protocol-based RESTful API, the way of connecting each application and BIMserver is unified. All applications can rely on a set of fixed HTTP methods such as POST, GET, DELETE, and PUT to perform state transition on the resources in the BIMserver without any prior knowledge. It also eases exception handling. A set of fixed HTTP status codes tells a client how a server responds to a request. A client without any prior knowledge of the server can still take appropriate operations by parsing returned HTTP status code. For example, in any RESTful web service, error code 404 means resource not found. Based on this, a client could be programmed to have error handlers to each type of error code in advance. Second, each application can subscribe to the specific service exposed by the BIMserver via a RESTful API. Whenever an event is triggered by the operation launched by one application, other applications who subscribe to such type of events will be notified and automatically perform corresponding routines.

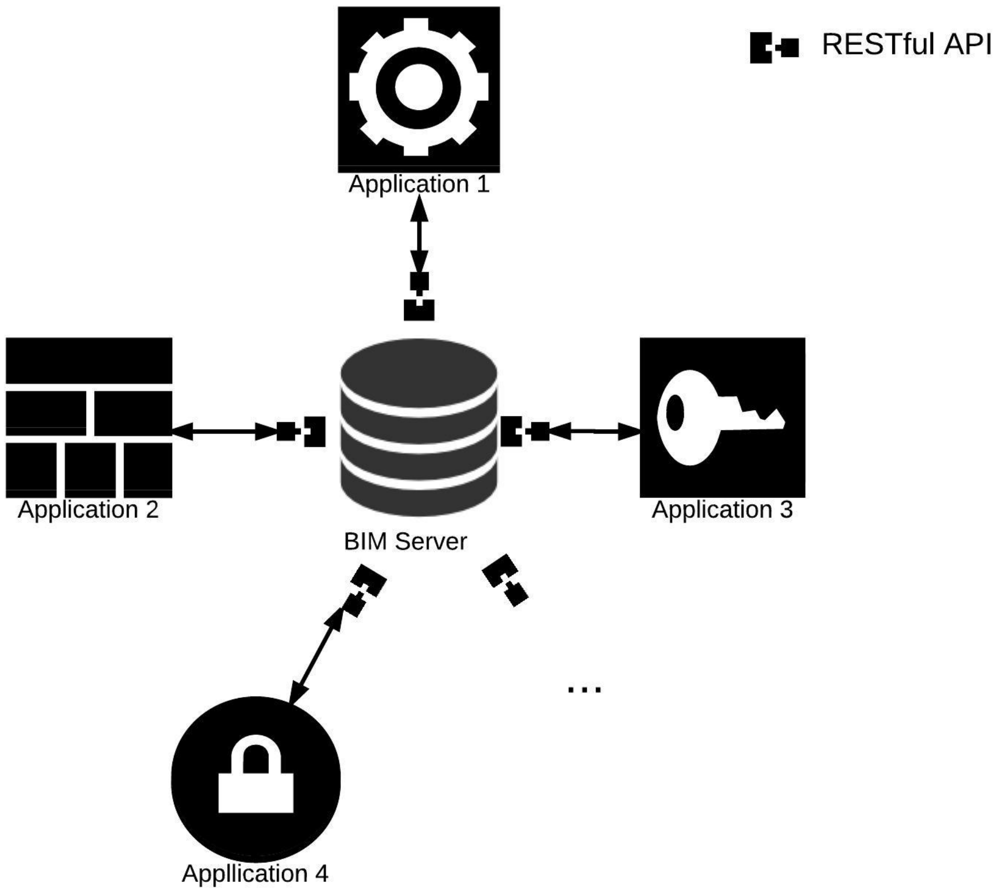


Figure 4. BIM as a service.

4.3. Application as a Service

In the model of application as a service, there is no central data server. Each service has its BIM data storage infrastructure, a local copy of BIM data, the application built upon it, and a set of RESTful APIs. Each service could be a service provider and a service requester at the same time. An information exchange session could be established easily among arbitrary nodes of service in the network. Service initial requests and response payload could be sent over HTTP protocol. BIM data could be queried, pushed, and synchronized between any two services. Such collaboration pattern allows each service has highly heterogeneous internal implementation and data models, which helps architecture designers and analysts take advantage of their existing software assets. Additionally, project data is hard to be compromised by a participant, since there is no central data server and every application works on its data copy. Thus, procedures of authentication and reading/writing permission granting are simplified. Even an untrusted third party can join the collaboration any time. If an EEB project requires a complete project BIM data warehouse that contains all most-up-to-date updates for newcomers to copy from, a node of service could be assigned to be "master" node. Project maintainers can "select" useful updates from all participants. Please note that "master" is just a naming convention. From a technical perspective, all nodes of service in a network are equally important, or, we can say the node called "master" just offers the service of integration. Figure 5 illustrates this design pattern.

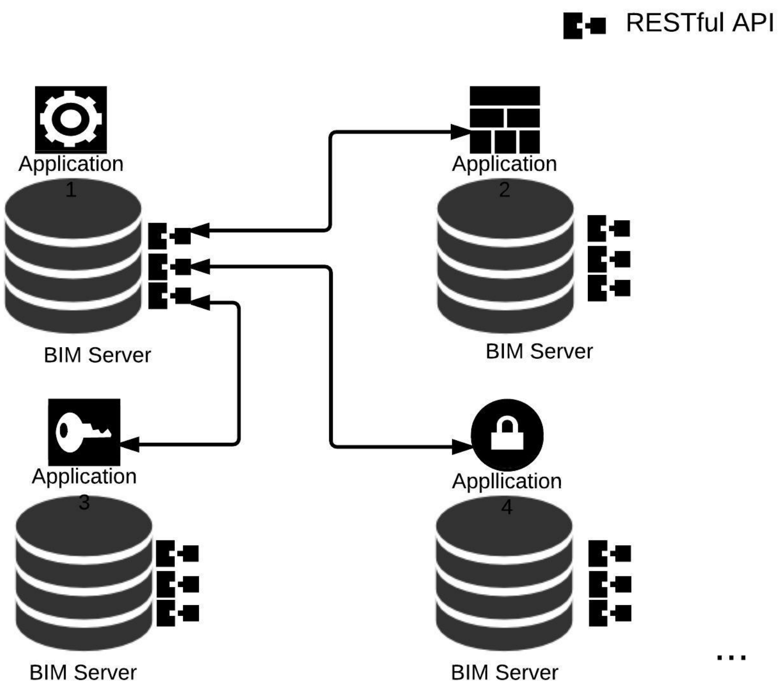


Figure 5. Application as a service.

5. Implementation

We implemented a proof-of-concept RESTful API prototype to demonstrate how a RESTful web service can better streamline the data exchange in multizone airflow analysis. We chose multizone airflow analysis for our demonstration because it is an important scenario in EEB lifecycle supporting process. To save users from repeatedly creating models for airflow analysis, DeGraw *et al.* propose a methodology to generate multizone airflow models from energy analysis BIM [50]. Multizone airflow analysis models used by CONTAM, a multizone airflow and contaminant transport analysis software, are simpler than energy analysis BIM used by EnergyPlus [50]. By simplifying and customizing energy analysis models, it is possible to generate airflow analysis models. They describe the detailed object translation process and the technical feasibility in the paper.

5.1. Overall Architecture

The overall workflow of multizone airflow analysis and the role of our prototype is shown in Figure 6. Airflow analysis needs three kinds of data. The first one is the 2D airflow zone which can be generated out of energy models. This part consists of most data of airflow models. This data retrieving process is enclosed by the dotted lines in Figure 6, which is implemented by our prototype. Our prototype was implemented under such a client-server architecture. Our server that publishes the APIs play a role of a BIMserver in the figure. The client plays a role of CONTAM in the figure. The second part is the airtightness of all kinds of materials. This part of data can be fetched from the National Institute of Standards and Technology (NIST) commercial buildings airtightness database [51]. Airtightness will be assigned to every "boundary" of airflow zones according to its attributes. The third part is the weather data, which can be obtained from other sources. The airtightness data and weather data can be explored, accessed and updated by a set of RESTful API in the same way as what we have done on energy models, which fits our application as a service design.

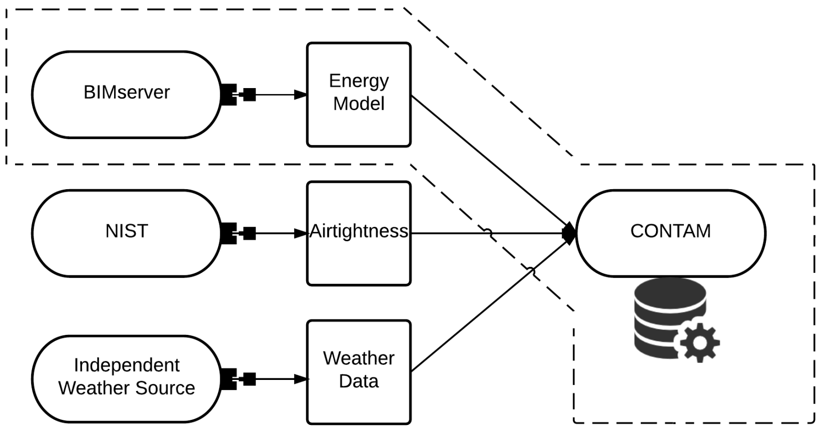


Figure 6. Example to show RESTful application as service.

5.2. Server Side

On the server side, we build a mock BIMserver. To simplify the internal implementation and better focus on web service and API design, we stimulate the on-the-fly energy model generation process in a real BIMserver. We store some pre-defined and previously generated energy models in JSON format in the server. There is no difference from an end user's perspective. The whole set of RESTful API and data schema was defined in the YAML (YAML is a "human friendly data serialization standard for all programming languages [52]". YAML is a recursive acronym for "YAML Ain't Markup Language".) language. Table 1 lists the details of each RESTful API. The server host domain is restfulbimserver.mockable.io. Then a valid URL is *domain name* + *API*. For example, Figure 7 shows a valid curl (curl is an "open source command line tool and library for transferring data with a URL syntax [53]") request that accesses a node named *OS:Space2* in a RESTful approach. In the code listing, "GET" is the HTTP method. A user can use it to hit the endpoint (the URL) of the nodes in an energy analysis model. The parameter, object name "OS%3ASpace2", sets constraints to this query. The returned result of this curl request is shown in Figure 8. Figure 9 shows a bit more advanced request. The expression after the question mark ("?") in the URI is the filter to select the data objects *"paths"* whose elevation is greater than or equal to three among all available paths. Multiple filters can be connected by "&" in a single request. All attributes in a data object can be used as filters in a query. The returned result of this curl request is shown in Figure 10.

Table 1. A summary of RESTful designs and their benefits.

EndPoint	Method	Resource Description
/buildings/{name}	GET	The whole building airflow network model
/nodes	GET, PUT, POST	Enclosed areas of the building are the nodes in the model. by hitting endpoints nodes, users will retrieve all the available nodes in the model.
/nodes/{name}	GET, PUT, POST	This endpoint refers to the nodes specified by their names.
/paths/{name}	GET, PUT, POST	A path is a potential avenue of airflow. Each wall or component may have leakage, then there is a path defined upon it. This end point refers to the paths specified by their names. refers to the paths specified.
/thermalZones	GET, PUT, POST	Thermal zones are not airflow zones. A thermal zone contains all the space which have similar thermal loads for an HVAC system. This endpoint refers to all the thermal zones in the model.
/thermalZones/{id}	GET, PUT, POST	This endpoint refers to the thermal zones specified by their names.
/users/{username}	GET, PUT, POST, DELETE	This endpoint makes users' information accessible.

```
curl -X GET --header "Accept: application/json"
"http://restfulbimserver.mockable.io/nodes/OS%3ASpace2"
```

Figure 7. An HTTP GET request to fetch node information in RESTful (Representational State Transfer) style.

```
{
  "name": "OS:Space2",
  "area": 36,
  "level": "OS:BuildingStory 1",
  "returnAir": [
    {
      "system": "OS:ThermalZone 1"
    }
  ],
  "volume": 108,
  "supplyAir": [
    {
      "system": "OS:ThermalZone 1"
    }
  ],
  "y": 6,
  "x": 0,
  "z": 0
}
```

Figure 8. Node OS: Space2 data in Java Script Object Notation (JSON).

```
curl -X GET --header "Accept: application/json"
"http://restfulbimserver.mockable.io/paths?elevation=>3"
```

Figure 9. An HTTP GET request to retrieve paths whose elevation is greater than or equal to 3.

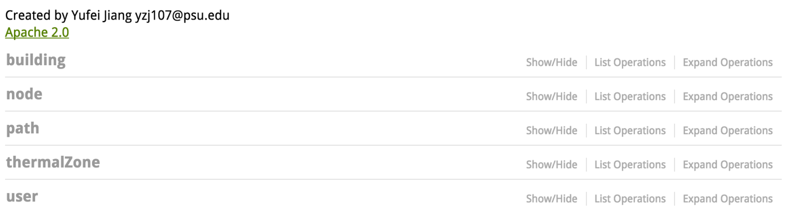
```
{
  "paths": [
    {
      "elevation": 3,
      "name": "OS:Surface 19",
      "area": 72,
      "tilt": 0,
      "level": "OS:BuildingStory 1",
      "azimuth": 90,
      "nodes": [
        "OS:Space 1"
      ],
      "type": "roof"
    },
    {
      "elevation": 3,
      "name": "OS:Surface 12",
      "area": 36,
      "tilt": 0,
      "level": "OS:BuildingStory 1",
      "azimuth": 90,
      "nodes": [
        "OS:Space 3"
      ],
      "type": "roof"
    },
    {
      "elevation": 3,
      "name": "OS:Surface 6",
      "area": 36,
      "tilt": 0,
      "level": "OS:BuildingStory 1",
      "azimuth": 90,
      "nodes": [
        "OS:Space 2"
      ],
      "type": "roof"
    }
  ]
}
```

Figure 10. The path models returned from the server.

5.3. Client Side

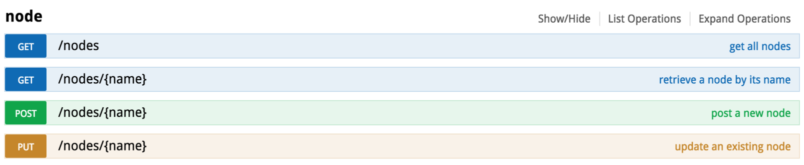
We build a client as a web service requester to play the role of data consumer just like CONTAM, without implementing its airflow analysis logics. The implementation is based on the node.js framework and Swagger framework [54]. It has a web-based user interface. It visualizes all the resources and HTTP methods available on the server side. Our users can interact with the BIMserver through this client which will be used in our evaluation and user study.

Figures 11 and 12 show the screenshots of the client user interfaces. Figure 11a is the original status of the web page when users visit the client. After clicking on a specific resource name such as *node*, the page will display what kind of operations can be performed against this resource, which is shown in Figure 11b. In addition, by clicking a method such as GET, the data schema and more details of this API will be displayed, which is shown in Figure 12. At last, by clicking the *"try it out"* button on a specific API, the client will send a request to the server and display the returned models or the error code. An example is shown in Figure 12. Users with some programming skills can use a shell command or write a script to hit the endpoints in the server side.


Created by Yufei Jiang yzj107@psu.edu
Apache 2.0

building	Show/Hide | List Operations | Expand Operations
node	Show/Hide | List Operations | Expand Operations
path	Show/Hide | List Operations | Expand Operations
thermalZone	Show/Hide | List Operations | Expand Operations
user	Show/Hide | List Operations | Expand Operations

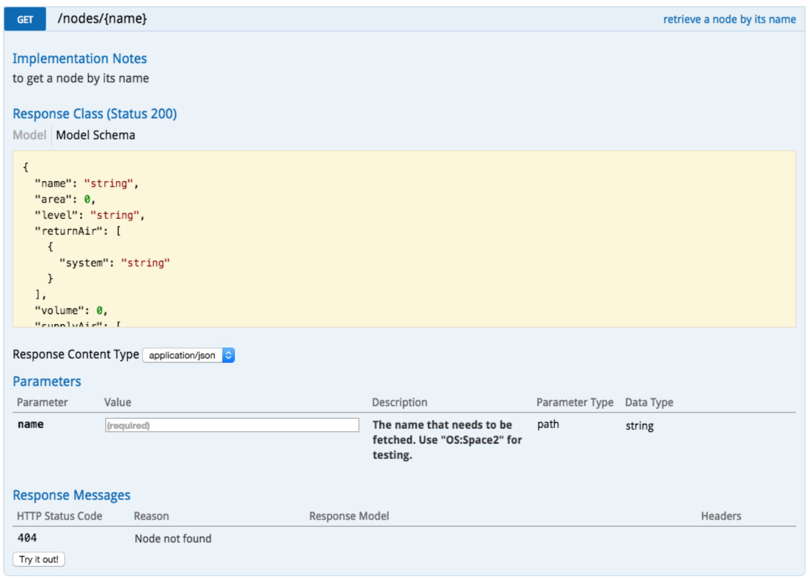
(**a**)


node — Show/Hide | List Operations | Expand Operations

GET	/nodes	get all nodes
GET	/nodes/{name}	retrieve a node by its name
POST	/nodes/{name}	post a new node
PUT	/nodes/{name}	update an existing node

(**b**)

Figure 11. Client user interface screenshots I.


GET /nodes/{name} retrieve a node by its name

Implementation Notes
to get a node by its name

Response Class (Status 200)
Model | Model Schema

```
{
  "name": "string",
  "area": 0,
  "level": "string",
  "returnAir": [
    {
      "system": "string"
    }
  ],
  "volume": 0,
  "supplyAir": [
```

Response Content Type application/json

Parameters

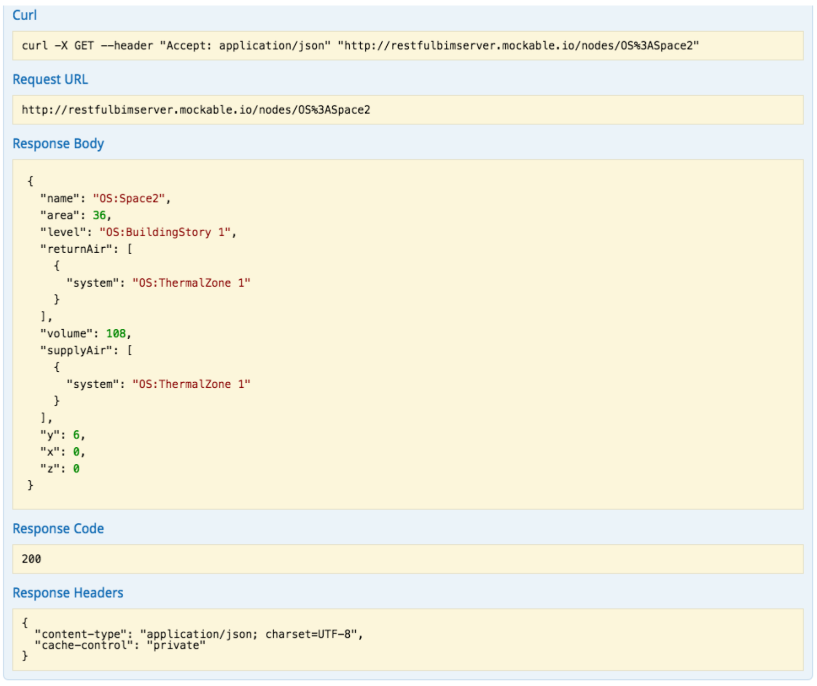
Parameter	Value	Description	Parameter Type	Data Type
name	(required)	The name that needs to be fetched. Use "OS:Space2" for testing.	path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
404	Node not found		

Try it out!

Figure 12. *Cont.*



```
curl -X GET --header "Accept: application/json" "http://restfulbimserver.mockable.io/nodes/OS%3ASpace2"
```

```
http://restfulbimserver.mockable.io/nodes/OS%3ASpace2
```

```
{
  "name": "OS:Space2",
  "area": 36,
  "level": "OS:BuildingStory 1",
  "returnAir": [
    {
      "system": "OS:ThermalZone 1"
    }
  ],
  "volume": 108,
  "supplyAir": [
    {
      "system": "OS:ThermalZone 1"
    }
  ],
  "y": 6,
  "x": 0,
  "z": 0
}
```

```
200
```

```
{
  "content-type": "application/json; charset=UTF-8",
  "cache-control": "private"
}
```

Figure 12. Client user interface screenshots II.

6. Evaluation

6.1. Theoretical Framework and Study Measurements

6.1.1. Theoretical Basis

We use the Technology Acceptance Model (TAM) as our theoretical framework [55], which is one of the most widely used models to check the users' acceptance to a new information technology. It has been successfully applied to examine how users adopt an information system in many different areas [56–60]. The validity of the model itself also has been examined quantitatively in a rigorous approach [61].

The model of TAM and its elements are shown in Figure 13. We can see that there are multiple variables in this model, ranging from users' first impressions of a new information technology to the phase of actual usage. Those variables affect each other and form a directed graph. The two most important variables of TAM are Perceived Usefulness (PU) and Perceived Ease Of Use (PEOU). They are two major drivers to affect users' actual usage of the new technology. According to Ajzen and Fishbein, perceived usefulness is defined as "the degree to which an individual believes that using a particular system would enhance his or her job performance" [62], and perceived ease of use is defined as "the degree to which an individual believes that using a particular system would be free of physical and mental effort" [62]. Different design features and different users' demographic backgrounds may cause different degrees of perceived usefulness and ease of use. Sometimes we categorize those factors as external variables. From Figure 13, we can see that the perceived ease of use is a very important variable since it affects perceived usefulness. If a user feels a new technology is easy to use, then he or she is more likely to believe this new technology is useful. Perceived usefulness and perceived ease of use will together influence users' overall attitude towards the new information technology. Users' intention to use is decided by their perceived usefulness and attitudes. Eventually, users' intention to use will lead to their actual usage and acceptance of new technology.

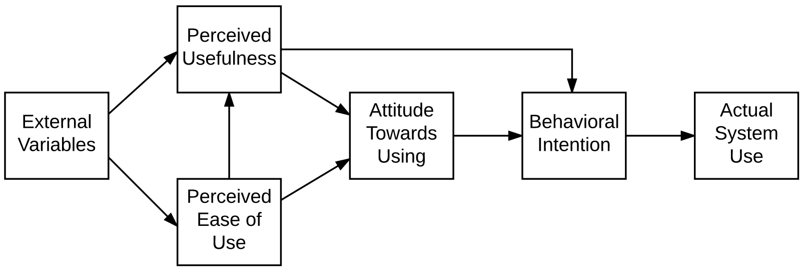


Figure 13. Technology Acceptance Model (TAM) overview.

6.1.2. Measurements and Study Design

Participants. This user study was conducted in the Consortium for Building Energy Innovation (CBEI) project. We invited the researchers and engineers who are involved in this project via emails or in person to participate in our user study. These selected experts share some common characteristics. They all work on the some areas related to EEB design and have experience on the BIMserver, which is an existing open source platform developed by BIMserver.org that facilitates information exchange in building projects. Some of them have experiences with multiple BIM data management systems. However, they also have different expertise and roles. Therefore, we use two tags, "analyst" and "developer", to denote our participants. Analysts can compose analysis logics, operate some domain software, build models, and interpret analysis results. Developers can implement new applications to fulfill those analysis logics, build plugins based on existing platforms (e.g., BIMserver and OpenStudio), and parse data. In CBEI, they usually work together on a research topic. Two tags are not exclusive: if a participant does both jobs in their daily work, she or he can have both tags. There are 10 participants in total. Among them, three are analysts, three are developers, and four are both analysts and developers.

Tasks. Our study contains two parts: user experiments and interviews. First, we ask users to perform a mini project on our prototype. Second, we interview these experts in a semi-structured manner.

In the mini project, we first give each user a short tutorial of our API prototype and how to use our web-based client to fetch data. Then, we show them a target data schema that CONTAM accepts and ask users to try their best to compose a JSON format file that meets that data schema. During this mini project, users are allowed to access any resources and help they need. They also can directly ask questions to us. They must fetch data through our APIs, but not necessarily through our web-based client. Some users may want to directly use curl commands or write their own scripts to hit our APIs.

After a user finishes the mini project, we will conduct a one-to-one interview with that user. We first ask them questions regarding their subjective thoughts on our prototype. These questions are designed in the guidance of TAM. Then, we let them compare our design with two counterparts in the BIMserver. More specifically, BIMserver has two ways to query data. The first approach is to write a piece of query code in Java. Figure 14 shows a code snippet from the BIMserver.org advanced query demo. It queries the data of doors whose height is over 2 feet from the first floor. The other way is also based on web interfaces. They integrate BIM Service interface exchange (BIMSie) [63] into their implementation, which accepts SOAP and JSON as web API call payload. However, BIMserver itself does not offer RESTful style web services. The BIMSie interfaces abstract library calls rather than resources. In addition, all requests are sent via HTTP GET method, no matter if it is adding a project or it is deleting a user. Table 2 shows a list of BIMsie APIs that are related to user operation.

```java
@Override
public void query(IfcModelInterface model, PrintWriter out) {
      out.println("Running␣doors␣example");
      List<IfcBuildingStorey> stories = model.getAll(IfcBuildingStorey.class);
      Map<Double, IfcBuildingStorey> orderedStories = new TreeMap<Double, IfcBuildingStorey>();
      for (IfcBuildingStorey storey : stories) {
        orderedStories.put(storey.getElevation(), storey);
      }
      if (orderedStories.size() > 1) {
        IfcBuildingStorey firstFloor = stories.get(1);
        for (IfcRelContainedInSpatialStructure rel : firstFloor.getContainsElements()) {
          for (IfcProduct product : rel.getRelatedElements()) {
            if (product instanceof IfcDoor) {
              ifcDoor ifcDoor = (IfcDoor)product;
              if (ifcDoor.getOverallHeight() > 2) {
                out.println(ifcDoor.getName() + "␣" +
                        ifcDoor.getOverallHeight());
}}}}}}
```

Figure 14. An advanced query sample from BIMserver.org (version 1.1 beta, 2012).

Table 2. User related methods in the BIMSie design.

addUser	setAllowUsersToCreateTopLevelProjects
addUserToExtendedDataSchema	undeleteUser
addUserToProject	removeUserFromProject
changeUserType	userHasCheckinRights
loginUserToken	userHasCheckinRights
getAllAuthorizedUsersOfProject	newUser
getAllCheckoutsByUser	userHasRights
getAllNonAuthorizedProjectsOfUser	deleteUser
getAllRevisionsByUser	getLoggedInUser
getUserByUoid	isAllowUsersToCreateTopLevelProjects
getUserByUserName	isHideUserListForNonAdmin
getUserRelatedLogs	setHideUserListForNonAdmin
getUserSettings	getUsersProjects
getAllUsers	getAllNonAuthorizedUsersOfProject
registerNewUserHandler	unregisterNewUserHandler

Every interview is recorded and transcribed to texts. The transcripts were analyzed to explore users' thoughts related to concepts in the TAM framework. More specifically, we coded a selected range of key phrases. (e.g., "useful", "can be used", "simple", and "easy to use"). We merge those phrases and map them into higher-level concepts in the TAM framework.

6.2. Findings

We analyzed the participants' reactions to the RESTful BIM web service design after experiencing our prototype. Overall, we found out that a majority of participants (eight out of 10) hold positive points of view on our design based on the comparison with their previous experience on BIMserver and other existing BIM data storage systems. They also described many ideas about how this design can be implemented and applied to other sub-domains of EEB analysis and simulation. We have categorized their ideas into these TAM concepts: perceived ease of use, perceived usefulness, altitude towards using, and behavior intention. We also investigated external variables by combining users' own thoughts and our analysis.

6.2.1. Perceived Ease of Use

Most of the participants (eight of 10) believe our design is easy to use. Only one participant gives a negative comment, and another participant feels neutral. Some users believe the most important advantage of our design is its simplicity:

"It is really easy to find the operation I want. There are many duplicated API in BIMsie." (Interviewee 2, analyst and developer).

"I didn't need to know details of those internal implementation when I was using an API."(Interviewee 3, analyst and developer)

"It is easier to infer all APIs you need, all the designs stick to one design paradigm. Its readability and learnability is better." (Interviewee 9, developer)

Some users expressed their perceived ease of use from different perspectives:

"RESTful API is not an ad hoc *design, there are so many existing open source software, libraries, and packages libraries to help you handle RESTful things. I might even not need to write my own code." (Interviewee 10, developer)*

"All solutions for me need additional learning. However, this design is more general, which means you can find more resources, tutorials, and examples to learn. Even if I need an IT guy to help me, I don't need to spend time to teach him too much civil engineering knowledge since RESTful APIs hide those details" (Interviewee 5, analyst).

However, a user also gives a negative comment:

"I feel confused when I was seeing your design, BIMserver interface and those query code. I don't understand them and don't know how to use them. I have been doing EEB structure analysis for several years, I haven't encountered a case that I need web services or RESTful web services. All I rely on is GUI (graphical user interface), like BIMserver's GUI or Revit GUI. I don't know if it (the prototype) is easy to others, it is hard to me" (Interviewee 6, analyst)

6.2.2. Perceived Usefulness

In general, eight of 10 users indicated that they felt our design is useful. The other two users held negative opinions on perceived usefulness.

Among the people who said RESTful style design is useful, many of them expressed that its usefulness is brought by its ease of use:

I used BIMserver quite a lot, I am very familiar with BIMserver. I have many big project experiences on BIMserver. I definitely know the difficulties when there are so many APIs which are just subtly different and so many data objects. If BIMserver or other BIM platforms integrate your APIs, I prefer to use yours. (Interviewee 2. BIM expert and developer).

I think its usefulness is two-fold. First, you build it based on some general concepts, not IFC. Second, compared with writing advanced query code (in Java), this method allows you to write script in any language and just embed a web request. This feature is useful to me. (Interviewee 8. analyst and developer).

"It is like we achieve the same goal with less effort, so it is useful and I prefer it (RESTful web service)." (Interviewee 3. analyst and developer).

A user also uses the examples from other domains to support her opinion:

"I believe taking advantage of RESTful web service is a trend. If you see all leading Internet companies like Facebook or Twitter and check their API documents, you will find that all of their web services are based on REST. This design style is also suitable for EEB design." (Interviewee 1, analyst and developer).

For those who felt neutral or negative, they also gave their reasons:

"My requirements and workflow are quite fixed. My daily challenges mainly are analysis logic issues, not related to data exchange. I am familiar with the tools that I am using. I am not sure if I can get benefits from a new tool" (Interviewee 7, analyst)

6.2.3. Attitude Towards Using

When the users were using our prototypes, they gave various comments and raised some questions. Generally, eight of 10 users had positive attitudes towards using them. Two users had neutral attitudes and no one had a negative attitude.

Some users said they realized some flaws of existing tools when they were using our prototypes:

> *"All BIMsie interfaces use HTTP GET method, I felt GET is not supposed to be used to update server side data." (Interviewee 2, analyst and developer).*

> *"I like how your design organizes everything and hides the internal implementation details. Directly submitting some Java code on a web page may cause some security risks."(Interviewee 9, developer)*

A user indicated that during his usage of our prototype, he can quickly identify where the problem is according to the HTTP standard status codes, which is inspiring:

> *"Error codes are clear, you even don't need to read error messages. I can easily parse them and have routines to respond"(Interviewee 5, analyst)*

A user who gave a negative comment regarding perceived usefulness held a neutral attitude here and said the using experience itself is acceptable:

> *The experience of using your prototype is OK. I don't see many problems there. (Interviewee 7, analyst).*

6.2.4. Behavior Intention

When being asked if they would like to use RESTful BIM web service in their future projects if all functionalities have been fully implemented, seven users said yes, one user said not sure, and two users said no. The behavior intention to use of some users is based on their perceived usefulness and attitude towards using them. They gave similar reasons here to support their decisions:

> *"Yes, I think I will use it. And I don't need to read IFC documents every time." (Interviewee 8, analyst and developer)*

> *"If BIMserver integrated RESTful web interfaces, then I would definitely use it." (Interviewee 3, analyst and developer).*

A user gave a reason from a more comprehensive perspective:

> *"I think it is developer-friendly and user-friendly at the same time. I will use it to request a service, but also use it to launch a service. I know there are many great applications in many labs. Few people know them and many efforts are wasted. They have no incentives to maintain it. If there is a simpler way to deliver your application over web, then more and more researchers can publish their works and they can get more feedback and encouragement. The whole community will benefit from this." (Interviewee 2, analyst and developer)*

One user who had no strong intention to use said he was hesitated because he needs to consider more factors to make a decision:

> *"I got your idea, and it is great. However, a tool that was built based on a great idea might not be a great tool. I must see if its document is well-written. I must see if it is mature and stable or it is still fast changing. Considering if I should adopt a tool is different than talking about an idea" (Interviewee 7, analyst).*

The user who had no intention to use it because she believes she does not need it in her daily work:

> *I think it depends on what kind of projects you are doing. I never need a partial model query and web services. (interviewee 6, analyst).*

6.3. Discussion

In this subsection, we first analyze the external variables that affect users' preference towards our proposal. Then, we discuss our reflections on this study.

6.3.1. External Variables

Table 3 shows users' feedback and their expertise. We group the rows by user tags. From the table, we can find that two analysts did not show strong interests in this idea while the other eight generally support it. By analyzing this preference pattern and interview scripts, we identify two major external variables.

The first variable is the role an expert played in a project. Different roles cause those users to consider the same issue from different perspectives. The analysts' focus is more on analysis logics. Their work is to refine their analysis algorithm to better abstract knowledge from existing data. They usually interact with service provider's APIs indirectly. For those developers and analysts who also do development jobs, publishing their applications or requesting remote data are challenges they are facing in their daily work. They understand our motivation and get the point of our idea more quickly. This is like Google Maps releasing a set of better web services and APIs. Those mobile app developers will start to use it to refine their applications while the end users might just feel indifferent to this news. However, those end users still get better user experience in the end. They get benefits in an indirect way. This result helps us identify our contribution as a developer-oriented solution rather than an end-user-oriented solution. Instead of additionally lowering the bar of using our design to make it accessible to more people, making it more developer friendly might be the right direction to go.

The second external variable is the nature of a project. The number of data objects needed by a project is various. Some projects need a few kinds of discrete data objects which do not relate to each other. In this case, even if the data size is huge, the query itself is simple. In this scenario, some interviewees pointed out that every approach works similarly in these kinds of problems. Users cannot gain much benefit by importing a new workflow and design. However, some projects' data models are very complex including both data schema and the relationship between those data schema. In this case, a resource-modeling-centric RESTful web service helps both service developers and application builders tackle the problem by following a top-down approach. If an EEB design sub-domain happens to have these features, then the users who work in these areas recognize the benefits of our method.

Table 3. Interview feedback.

User Number	Tags	PEOU	PU	ATU	BIU
1	Analyst & Programmer	✓	✓	✓	✓
2	Analyst & Programmer	✓	✓	✓	✓
3	Analyst & Programmer	✓	✓	✓	✓
8	Analyst & Programmer	✓	✓	✓	–
4	Programmer	✓	✓	✓	✓
9	Programmer	✓	✓	✓	✓
10	Programmer	✓	✓	✓	✓
5	Analyst	✓	✓	✓	✓
6	Analyst	✗	✗	–	✗
7	Analyst	–	✗	–	✗

6.3.2. Reflections on the Study

There is a long debate on when to apply web services and how to choose a proper design style from those co-existing design paradigms. It should be clear that there is no universal best design, but relatively more appropriate design to a specific case. Regarding the case we present in this paper, BIM-enabled EEB design has some unique challenges, such as interdisciplinary, multiple participants and stakeholders, fast evolving community, and deep involvement of researchers, which make its loose coupling collaboration pattern different from some traditional domains. Those challenges are exactly what RESTful web services are good at handling. By communicating with the researchers, we found most of them agree with the points presented in our paper. Even the BIMserver official blog described the current challenge and future direction of the community: "the commercial use and increasing complexity of applications that use BIMserver as their base, demand query capabilities beyond the current plugins. We've always waited for the industry (or academic world) to develop a rich BIM Query language, but time is running out. That is why we decided to remove the Query-plugin type and build one new query language...and we plan to gradually migrate the current API to a new one" [64]. Our research is orthogonal to the BIMserver and other BIM data platforms. They can integrate our approach into their implementations.

An interface is a contract between service providers and users. The web service interfaces that look like method calls in a programming language such as *"addUserToProject"* are actually modeling on logics. It works better when all parties share a similar background or work closely. However, if participants belong to different parties and have heterogeneous knowledge backgrounds, the best contract between them should be the data schema itself, or more precisely "resources", which is more stable and more understandable. That is the reason we use RESTful web service which models interfaces based on resources.

6.3.3. Limitations and Future Work

This study presents some preliminary results collected from 10 interviewees. This research can be regarded as a pilot study on this direction. Being different to some information systems that can be operated by normal users, due to the fact that our proposed requirements, designs, and implemented prototypes are targeting some expert users who work on BIM-enabled EEB design, it is difficult to recruit a large number of qualified interviewees in a short time. In addition, our current prototype RESTful APIs just cover the essential resources for multi-zone airflow analysis. There are many other areas in EEB design to be implemented.

In the future, to perform larger and deeper user studies, we need to build plugins or integrate our API to some existing BIM data platforms and cover more functionalities. Thus, we can expand our user studies from two directions. First, we can collaborate with some groups and work with them on a real-world project by using our applications. By attending their daily work, we can use the ethnography methodology to observe and record how our idea affects their work efficiency. Second, we can invite more researchers to try our application. Then, we can use questionnaires to collect some quantitative data to perform an analysis.

7. Conclusions

In this paper, we investigate the advantages of using BIM RESTful web services to support the EEB lifecycle. We identify the requirements to support the EEB lifecycle from four different perspectives: information exchange requirements, distributed collaboration requirements, internal data storage requirements, and partial model query requirements. Three RESTful style designs at different abstraction levels are proposed and detailed with rationales. We have implemented a set of RESTful APIs and conducted a pilot user study. In the user study, by comparing with the current BIMserver partial model query methods, users generally perceive more usefulness, ease of use, and behavioral intention to use in performing an EEB design task. We conclude that the RESTful BIM web service is a feasible method to improve the efficacy and efficiency of the EEB lifecycle support and enhance the BIM data accessibility.

Acknowledgments: This research is supported in part by the Consortium for Building Energy Innovation (CBEI), sponsored by the Department of Energy under the grant DE-EE0004261.

Author Contributions: Yufei Jiang and Dinghao Wu conceived the concept and design; Yufei Jiang and Xiao Liu performed the experiments; Yufei Jiang and Xiao Liu analyzed the data; Yufei Jiang, Xiao Liu, Fangxiao Liu, Dinghao Wu, and Chimay J. Anumba wrote the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jiang, Y.; Ming, J.; Wu, D.; Yen, J.; Mitra, P.; Messner, J.I.; Leicht, R. BIM server requirements to support the energy efficient building lifecycle. In Proceedings of the 2012 ASCE International Conference on Computing in Civil Engineering, Clearwater Beach, FL, USA, 17–20 June 2012.
2. Watson, A. BIM-a driver for change. In Proceedings of the International Conference on Computing in Civil and Building Engineering, Nottingham University Press, Nottingham, UK, 30 June–2 July 2010.
3. Hirsch, A.; Pless, S.; Guglielmetti, R.; Torcellini, P.A.; Okada, D.; Antia, P. *The Role of Modeling When Designing for Absolute Energy Use Intensity Requirements in a Design-Build Framework*; NREL/CP-5500-49067; National Renewable Energy Laboratory: Golden, CO, USA, 2011.
4. Bazjanac, V. Acquisition of building geometry in the simulation of energy performance. In Proceedings of the 2001 Building Simulation Conference, Rio de Janeiro, Brazil, 13–15 August 2001.
5. Beetz, J.; van Berlo, L.; de Laat, R.; van den Helm, P. BIMserver.org—An open source IFC model server. In Proceedings of the CIP W78 Conference, Cairo, Egypt, 16–18 November 2010.
6. Beetz, J.; Berlo, V.L.; Laat, D.R.R.; Bonsma, P. Advances in the development and application of an open source model server for building information. In Proceedings of the CIP W78 Conference, Sophia Antipolis, France, 26–28 October 2011.
7. Fielding, R.T. Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. Thesis, University of California, Irvine, CA, USA, 2000.
8. Legner, C.; Heutschi, R. SOA adoption in practice—Findings from early SOA implementations. In Proceedings of the 2007 European Conference on Information Systems (ECIS 2007), St. Gallen, Switzerland, 7–9 June 2007.
9. Michelson, B.M. Event-driven architecture overview. *Patricia Seybold Group* **2006**, *2*, doi:10.1571/bda2-2-06cc.
10. Exforsys. SOA Disadvantage, 2007. Available online: http://www.exforsys.com/tutorials/soa/soa-disadvantages.html (accessed on 1 August 2015).
11. Associates, J.J.H. Welcome to DOE-2.com. Available online: http://www.doe2.com/ (accessed on 28 April 2016).
12. O'Donnell, J. SimModel: A domain data model for whole building energy simulation. In Proceedings of the SimBuild 2011, Sydney, Australia, 14–16 November 2011.
13. Campbell, D.A. Building information modeling: The Web3D application for AEC. In Proceedings of the Twelfth International Conference on 3D Web Technology, ACM, Umbria, Italy, 15–18 April 2007.
14. BuildingSMARTalliance. United States National Building Information Modeling Standard, 2007. Available online: http://www.wbdg.org/pdfs/NBIMSv1_p1.pdf (accessed on 3 August 2015).
15. Eastman, C.; Eastman, C.M.; Teicholz, P.; Sacks, R. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*; John Wiley & Sons: Hoboken, NJ, USA, 2011.
16. Liu, F.; Jallow, A.; Anumba, C.; Wu, D. Building knowledge modeling: Integrating knowledge in BIM. In Proceedings of the 30th International Conference on Applications of IT in the AEC Industry, Beijing, China, 9–12 October 2013.
17. Smith, D.K.; Tardif, M. *Building Information Modeling: A Strategic Implementation Guide for Architects, Engineers, Constructors, and Real Estate Asset Managers*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
18. Azhar, S. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. *Leadersh. Manag. Eng.* **2011**, *11*, 241–252.
19. Howell, I.; Batcheler, B. Building information modeling two years later—Huge potential, some success and several limitations. *Laiserin Lett.* **2005**, *22*, 4.
20. Goedert, J.D.; Meadati, P. Integrating construction process documentation into building information modeling. *J. Constr. Eng. Manag.* **2008**, *134*, 509–516.

21. Bynum, P.; Issa, R.R.; Olbina, S. Building information modeling in support of sustainable design and construction. *J. Constr. Eng. Manag.* **2012**, *139*, 24–34.
22. Zhang, S.; Teizer, J.; Lee, J.K.; Eastman, C.M.; Venugopal, M. Building information modeling (BIM) and safety: Automatic safety checking of construction models and schedules. *Autom. Constr.* **2013**, *29*, 183–195.
23. Kim, C.; Son, H.; Kim, C. Automated construction progress measurement using a 4D building information model and 3D data. *Autom. Constr.* **2013**, *31*, 75–82.
24. Messner, J.; Anumba, C.; Dubler, C.; Goodman, S.; Kasprzak, C.; Kreider, R.; Leicht, R.; Saluja, C.; Zikic, N. BIM project execution planning guide and templates: version 2.0. Available online: http://bim.psu.edu/Resources/Project/BIM_PxP-V2.1/BIM_PxP_Guide_&_Templates_V2.1.zip (accessed on 15 July 2015).
25. Maile, T.; Fischer, M.; Bazjanac, V. Building energy performance simulation tools—A life-cycle and interoperable perspective. *Center Integr. Facil. Eng. (CIFE) Working Pap.* **2007**, *107*, 1–49.
26. Bazjanac, V. Impact of the US national building information model standard (NBIMS) on building energy performance simulation. In Proceedings of the Building Simulation 2007 Conference, Beijing, China, 3–6 September 2007.
27. Young, N.; Jones, S.; Bernstein, H.M.; Gudgel, J. *The Business Value of Bim—Getting Building Information Modeling to the Bottom Line*; McGraw-Hill Construction: Bedford, MA, USA, 2009.
28. Moon, H.J.; Choi, M.S.; Kim, S.K.; Ryu, S.H. Case studies for the evaluation of interoperability between a BIM based architectural model and building performance analysis programs. In Proceedings of 12th Conference of International Building Performance Simulation Association, Sydney, Australia, 14–16 November 2011.
29. Jiang, Y.; Yu, N.; Ming, J.; Lee, S.; DeGraw, J.; Yen, J.; Messner, J.I.; Wu, D. Automatic building information model query generation. *J. Inf. Technol. Constr. (ITCon)* **2015**, *20*, 518–535.
30. Yu, N.; Jiang, Y.; Luo, L.; Lee, S.; Jallow, A.; Wu, D.; Messner, J.I.; Leicht, R.M.; Yen, J. Integrating BIMserver and OpenStudio for energy efficient building. In Proceedings of the 2013 ASCE International Conference on Computing in Civil Engineering, Los Angeles, CA, USA, 23–25 June 2013.
31. Liu, F.; Jallow, A.K.; Anumba, C.J.; Wu, D. A framework for integrating change management with building information modeling. In Proceedings of the 15th International Conference on Computing in Civil and Building Engineering (ICCCBE 2014), ASCE, Orlando, FL, USA, June 23–25 2014.
32. Dubler, C.R.; Messner, J.; Anumba, C.J. Using lean theory to identify waste associated with information exchanges on a building project. In Proceedings of the Construction Research Congress/ASCE Conference, Banff, AL, Canada, 8–10 May 2010.
33. Rogers, S. BIM Implementation Strategies: Two Models or One? Available online: http://old.agc.org/galleries/conmark/Rogers2008 (accessed on 16 July 2015).
34. Yan, H.; Damian, P. Benefits and barriers of building information modelling. In Proceedings of the 12th International Conference on Computing in Civil and Building Engineering, Beijing, China, 16–18 October 2008.
35. Curbera, F.; Duftler, M.; Khalaf, R.; Nagy, W.; Mukhi, N.; Weerawarana, S. Unraveling the Web services web: An introduction to SOAP, WSDL, and UDDI. *IEEE Internet Comput.* **2002**, *6*, 86–93.
36. Hardt, D. The OAuth 2.0 Authorization Framework. Available online: http://tools.ietf.org/html/rfc6749.html (accessed on 17 July 2015).
37. Is OAuth Stateless? Can it Work for REST? Available online: https://looselyconnected.wordpress.com/2010/11/12/is-oauth-stateless-can-it-work-for-rest/ (accessed on 18 July 2010).
38. Liu, D.T.; Xu, X.W. A review of web-based product data management systems. *Comput. Ind.* **2001**, *44*, 251–262.
39. Bilgic, T.; Rock, D. Product data management systems: State of the art and the future. In Proceedings of the 1997 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, paper No. DETC97/EIM-3720; Sacramento, CA, USA, 14–17 September 1997.
40. Murugesan, S. Understanding Web 2.0. *IT Prof.* **2007**, *9*, 34–41.
41. Van Berlo, L. BIMServer.org Release 1.3.0 Final. Available online: http://bimserver.org/2014/04/28/release-1-3-0-final/ (accessed on 20 July 2015).

42. Froese, T.; Fischer, M.; Grobler, F.; Ritzenthaler, J.; Yu, K.; Sutherland, S.; Staub, S.; Akinci, B.; Akbas, R.; Koo, B.; *et al.* Industry foundation classes for project management-a trial implementation. *Electr. J. Inf. Technol. Constr.* **1999**, *4*, 17–36.
43. Adachi, Y. Overview of partial model query language. In Proceedings of the International Society for Productivity Enhancement Concurrent Engineering Conference (ISPE CE), Madeira, Portugal, 26–30 July 2003.
44. Camposano, R.; Saunders, L.F.; Tabet, R.M. High-level synthesis from VHDL. *IEEE Des. Test Comput.* **1991**, *8*, 43–49.
45. Weise, M.; Katranuschkov, P.; Scherer, R.J. Generalised model subset definition schema. *CIB Rep.* **2003**, *284*, 440.
46. Mazairac, W.; Beetz, J. BIMQL—An open query language for building information models. *Adv. Eng. Inform.* **2013**, *27*, 444–456.
47. Daum, S.; Borrmann, A.; Langenhan, C.; Petzold, F. Automated generation of building fingerprints using a spatio-semantic query language for building information models. In *eWork and eBusiness in Architecture, Engineering and Construction: ECPPM 2014*; CRC Press: Vienna, Austria, 2014.
48. Gray, J.; Reuter, A. *Transaction Processing*; Morgan Kaufíann Publishers: San Francisco, CA, USA, 1993.
49. Allamaraju, S. *Restful Web Services Cookbook: Solutions for Improving Scalability and Simplicity*; O'Reilly Media, Inc.: Sebastopol, MA, USA, 2010.
50. DeGraw, J.W.; Macumber, D.; Bahnfleth, W.P. Generation of multizone airflow models from building energy models. *ASHRAE Trans.* **2014**, *120*, 1–8.
51. Emmerich, S.J.; Persily, A.K. US commercial building airtightness requirements and measurements. In Proceedings of the AIVC Conference, Brussels, Belgium, 12–13 October 2011.
52. yaml.org. YAML 1.2. Available online: http://www.yaml.org/ (accessed on 1 March 2016).
53. cURL project. Available online: https://curl.haxx.se/ (accessed on 1 March 2016).
54. SmartBear. Available online: http://swagger.io/ (accessed on 1 March 2016).
55. Davis, F.D., Jr. A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results. Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1986.
56. Adams, D.A.; Nelson, R.R.; Todd, P.A. Perceived usefulness, ease of use, and usage of information technology: A replication. *MIS Q.* **1992**, *16*, 227–247.
57. Chau, P.Y. An empirical assessment of a modified technology acceptance model. *J. Manag. Inf. Syst.* **1996**, *13*, 185–204.
58. Chau, P.Y. An empirical investigation on factors affecting the acceptance of CASE by systems developers. *Inf. Manag.* **1996**, *30*, 269–280.
59. Davis, F.D. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Q.* **1989**, *13*, 319–340.
60. Davis, F.D.; Bagozzi, R.P.; Warshaw, P.R. User acceptance of computer technology: A comparison of two theoretical models. *Manag. Sci.* **1989**, *35*, 982–1003.
61. Hu, P.J.; Chau, P.Y.; Sheng, O.R.L.; Tam, K.Y. Examining the technology acceptance model using physician acceptance of telemedicine technology. *J. Manag. Inf. Syst.* **1999**, *16*, 91–112.
62. Ajzen, I.; Fishbein, M. Attitude-behavior relations: A theoretical analysis and review of empirical research. *Psychol. Bull.* **1977**, *84*, 888.
63. Berlotti. BIMSie-API. Available online: https://github.com/BuildingSMART/BIMSie-API (accessed on 3 March 2015).
64. Berlotti. New Year's Resolutions. Available online: http://bimserver.org/2015/12/23/new-years-resolutions/ (accessed on 3 March 2015).