ITcon

www.itcon.org - Journal of Information Technology in Construction - ISSN 1874-4753

A BIM-BASED WEB SERVICE FRAMEWORK FOR GREEN BUILDING ENERGY SIMULATION AND CODE CHECKING

SUBMITTED: January 2014
REVISED: June 2014
PUBLISHED: July 2014
EDITOR: Peter Katranuschkov

Jack C.P. Cheng
Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology
PH (852) 2358-8186;
email: cejcheng@ust.hk

Moumita Das
Department of Civil and Environmental Engineering, The Hong Kong University of Science and Technology
PH (852) 68033893;
email: mdas@ust.hk

SUMMARY: *Green building design has been a major trend in the last decade which has largely affected the AEC industry. As of 2013, for example, there were over 13,000 green buildings certified with LEED (Leadership in Energy and Environment Design) in the United States alone. Building Information Modeling (BIM) technology and computer simulations are adopted largely for green building design. However, while information sharing and automated, collaborative design review are important for the design of green buildings, the current way of BIM-based green building design relies mainly on individual file transfer and does not support collaboration in the distributed environment of construction projects. On the other hand, as the Internet becomes ubiquitous, the web provides convenient and cost-efficient means for multi-location cross-organizational collaboration. Energy analysis and validation against standard building codes are two major processes in green building design evaluation. This paper presents a modular web service based framework which integrates the information necessary for green building design, automates the building design evaluation processes, and facilitates simple updates on the building model on a common but distributed platform. This framework is based on BIM data models like gbXML (Green Building XML) which contain information for green building design like geometry of the building, material, and sensor information from more than one source. The BIM data models act as a single source of building information for all processes. Building design evaluation and updating are iterative in green building design and require information and inputs dispersed among various project participants. Since our framework follows a distributed architecture and is easily accessible from the Internet, it makes the information required to facilitate the iterative process and its results conveniently available to a multi-participant environment. The paper also presents an example scenario demonstrating the developed framework.*

KEYWORDS: *Green Buildings, gbXML, Energy Analysis, Automated Code Checking, Web Services.*

REFERENCE: *Jack C.P. Cheng, Moumita Das (2014). A BIM-based web service framework for green building energy simulation and code checking, Journal of Information Technology in Construction (ITcon), Vol. 19, pg. 150-168, http://www.itcon.org/2014/8*

1. INTRODUCTION

The AEC (architecture, engineering, and construction) industry is rapidly moving towards sustainable building development. A recent study in the year 2012 by Turner Construction has revealed that 90% of 718 U.S. real estate owners, developers, and tenants support sustainable construction (IGEL, 2013). The 2013 McGraw Hill Construction global report states that 61% of the construction work is going to be sustainable construction or green construction globally by 2015 (MHC, 2013). The concept of green buildings is catching a lot of attention among the government and private builders in most countries around the world as people are becoming more responsible towards the environment and want a healthy living environment for themselves. This has resulted in the increase of green building certification standards like LEED (USGBC, 2012) in North America, BREEAM (BRE, 2008) in the United Kingdom , Green Star (GBCA, 2014) in Australia, and BEAM Plus (HKGBC, 2010) in Hong Kong. As of the year 2013, there were 13,000 construction and renovation projects certified by LEED in the United States alone (USGBC, 2012). Green buildings are structures that are built environmentally responsible, and are energy efficient throughout their life cycle. The five major elements of green building design are: (1) sustainable site design, (2) water conservation and quality, (3) energy use, (4) indoor environmental quality, and (5) conservation of materials and resources. Energy use is however the most significant, as the energy demand can be predicted through digital simulation during the design phase. Energy simulation assists in effective decision making during the design phase for selecting economic energy efficient energy sources and materials. There are many software programs available for building energy simulation like EnergyPlus (Crawley et al. 2001), Ecotect (Autodesk, 2005), IES Virtual Environment (IES, 2005), eQuest (eQuest, 2012) and RIUSKA (Granlund, 2014). However, none of them supports a mobile, distributed working environment.

Green building design is challenging, multi-disciplinary and multi-participant as sharing, retrieval, and updating of information related to green building processes are performed in a distributed manner. The three important parts in green building design are: (1) building performance simulation and engineering, (2) code checking, and (3) design review and updating. These three processes are highly interrelated. Energy simulation results can assist in making decisions in changing essential building characteristics like geometry, orientation and material during the design phase to achieve sustainable construction and lifecycle performance. Energy codes list a maximum allowable energy usage in spaces of different types. Percentage saving below the maximum allowable energy can earn the owner monetary incentives or points for green building certification. Design review and updating allow the inputs of human comments and judgment, based on their expertise, the energy simulation results and the energy codes. Therefore, an approach towards an integrated platform for design and review of green buildings is required.

In this paper, we present a building information modeling (BIM) based web service framework that provides an integrated platform for automatic building code checking and building energy simulation. Based on the simulation results, end users can make changes through this web service framework to building attributes like the geometry of the building (simple changes), material, and occupants for an energy efficient design. Seamless communication is supported by the framework through the advanced web service technology, with the freedom to the users of not buying the license of propriety software. gbXML (gbXML.org, 2010) is used in the developed framework because it is an open standard capable of storing building models along with the details needed for code checking and energy simulation. Other BIM data models like the Industry Foundation Classes, IFC (buildingSMART, 2014) could be used to implement the proposed framework, but gbXML was chosen as it provides more extensive coverage of the energy simulation domain in comparison to the current IFC 2x3 schema.

2. RESEARCH BACKGROUND

2.1 Building energy simulation and analysis

The concept of energy models has been around since the 1980's (Judkoff et al. 1982). Energy simulation tools can predict the future energy consumption of a building during the design stage. Based on this prediction, domain experts may suggest alternative sustainable and cost effective materials and equipment. If accurate prediction of energy usage is made by a simulation tool during the early design phase of the building, cost, time and labor can be saved in significant amounts. The results of these simulation tools may however vary depending upon the underlying calculation method used in the simulation tools (Ryan and Sanquist, 2011). DOE-2 (LBNL,

1994) is a core energy simulation engine that was released in 1994 for predicting energy use and cost for all types of buildings. The DOE-2 simulation program is a batch program which provides flexibility to the researchers and experts but also requires significant training time. Commercial simulation software like EnergyPlus, coupled with Design Builder as pre-processor, or eQuest and RIUSKA, which use the DOE-2 engine underneath, provide more user friendly interfaces and are therefore much more popular among researchers and the AEC industry. An interesting effort is the use of the Artificial Neural Network (ANN) technique, implemented to assess building performance simulation results against actual measurements (Yezioro et al. 2008). In that study, four building energy simulation tools namely Energy_10, Green Building Studio web tool, eQuest, and EnergyPlus were compared with the ANN predicted energy consumption. Out of the four simulation tools, EnergyPlus was found to have the most consistent simulation performance in terms of heating and cooling electricity consumption. The HESMOS project in Europe (Katranuschkov and Balder, 2013) developed an IFC based holistic design system for integrating BIM compliant CAD models with design and analysis tools like energy analysis tools and sensor networks, providing the necessary mapping algorithms, filter functions and visual representation techniques. The FIEMSER project developed a modular framework for Building Management Systems by leveraging the Service Oriented Architecture (FIEMSER, 2011). However, none of these efforts provides a platform for communication between distributed project partners. Also, most of the software applications use a proprietary file structure and do not provide direct compatibility with BIM models. Third party applications are available to convert the BIM models into such proprietary file formats, but only semi-automatically. Moreover, all popular energy simulation tools are standalone. Hence, they are not very suitable for information sharing in the multi-party environment of the construction industry.

2.2 Building Code Checking

The concept of automating code checking for buildings has been around for more than two decades (Dym et al. 1988). Han (1998) proposed an automated code checking system for accessibility codes for handicapped persons using the IFC standard. Applicability of decision tables and servlets was also explored to automate code checking by using existing building codes (Yang and Yu, 2004). The major developments in rule-based code checking have been investigated and reviewed by Eastman et al. (2009). In summary, this research highlights that there is a lack of an overall integrated system environment which would provide the functionalities like clash detection, spatial validation, and semi-automatic conversion of human written rules in programmable rules in an open form allowing the developers to define new rules and functions easily. Nawari (2012) proposed a framework for converting the building rules written in natural language into computable rule sets by implementing general-purpose and special-purpose ontologies along with logic theories in different levels of implementation. Susmita et al. (2013) developed a framework which accesses the parametric properties of Autodesk Revit families and extracts information in order to check the compliance of building energy codes with the Revit building model. AutoCodes (Fiatech, 2012) have demonstrated automated code checking by the use of computable building code rule sets for access and egress provisions for retail structures in the first phase of their in-progress project. Also, commercial software like Solibri Model Checker (Solibri, 2008) is available which provide platform for rule set development and execution. However, there are no standard frameworks available at present to easily deploy these code checking systems and software on an integrated BIM platform. This results in gaps in communication and information exchange in the multi-party construction industry.

2.3 Web-based building model sharing and updating

BIM transforms the way information flows in the AEC industry. Currently, BIM is mainly used by designers. However, the other actors like contractors and material specialists are also exploring the application of BIM in their domain. The construction industry in the United States loses $15.8 billion annually due to its fragmented nature, the lack of standardization, and inconsistent adoption of technology among the construction stakeholders (Gallaher et al. 2004). Therefore, there is a need for an integrated server based platform that would facilitate effective flow of information among the construction stakeholders. Singh et al. (2011) identified the ease of model access and usability as one of the key features of a server based BIM. In addition, construction stakeholders also need to quickly retrieve and generate customized information out of a BIM model. Thus, Karola et al. (2002) proposed a middleware solution, called BSPro COM-Server for making energy simulation tools interoperable with IFC models. Ahn et al. (2014) proposed a model for integrating BIM with the energy simulation by providing an interface for generating EnergyPlus input files from IFC models. The European ISES project (Christodoulou and Xanthos, 2014) proposes a conceptual framework of an ontology based BIM system for integrating BIM data with non-BIM data (for example, thermal load and local weather) and facilitating efficient information exchange and parallel energy analyses through a cloud-based service, and Nepal et al. (2012) developed a Revit API for extracting spatial data from a Revit model and expressing it into a GML application schema which provides a common XML syntax to represent the heterogeneous BIM information. In our web service framework, we implement standardized web services which can be invoked through the Internet easily. The web services are programmed to retrieve and generate customized information needed to serve the purposes of different construction stakeholders from a BIM model.

3. PROPOSED METHODOLOGY AND CONCEPTUAL SYSTEM FRAMEWORK

FIG 1 presents the proposed conceptual framework for green building design in a collaborative environment. It performs three main functions related to green building design: (1) building energy simulation, (2) green building code compliance checking, and (3) simple updates on the BIM building model. The center of this framework is the BIM building model, which acts as a repository of the information needed to carry out the aforesaid functions. It stores information related to the building geometry, material information like the name of material and its u-value, lighting information, and results of energy simulation like monthly lighting demand. Standardized web services are the key functional units of this framework which perform two broad categories of functions: (1) parse the BIM data model to extract or update information and (2) execute engines for green building performance analyses like energy simulation or code compliance checking against the parameters extracted from the BIM model. The framework implements a Service Oriented Architecture (SOA), which facilitates software applications to be delivered and consumed as services (MacVittie, 2008). SOA comprises a set of key components called services (implemented most commonly as web service units) whose interface descriptions can be published, discovered, and invoked. The standardized access protocol, SOAP (W3C 2003) is used for facilitating communication and invocation of the web service units. The interface descriptions of the web services are published using the standardized web service description language WSDL (W3C 2001), which is an XML based language for describing the technical specification of web services. A client application trying to consume a particular web service unit could determine the available functions provided by reading its WSDL file. The web services are web enabled applications which execute predefined tasks themselves and also dynamically engage other web services to collectively provide dynamic functionalities. SOA incorporates flexibility, scalability, reuse, encapsulation, and better value for money for the users of the services.

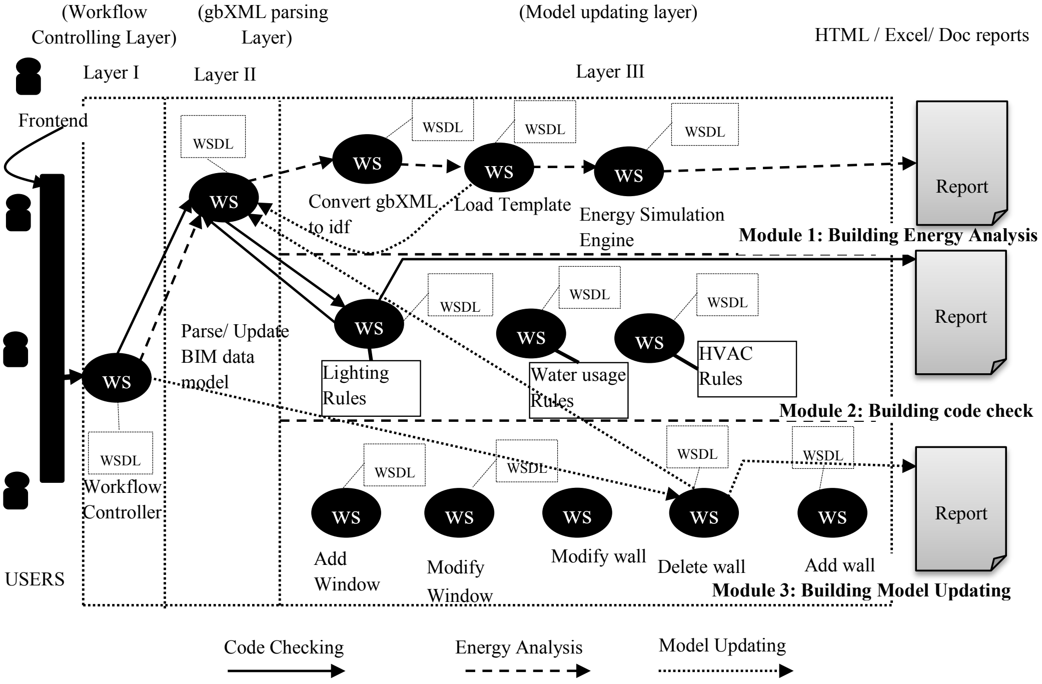


FIG 1: The conceptual web service framework for green building design.

As shown in FIG 1 our web service framework can be divided into three layers. Layer I comprises the workflow controller web service. The function of this web service is to route the control to the relevant web service in the next layers. The end user invokes the workflow controller web service in Layer I of the web service framework by using the binding information in its WSDL file. The flow controller web service takes input like project name or ID, domain name (for example, building code check) and sub-domain name (for example, water usage). Based on the user input, the control may be passed to Layer II or Layer III. Layer II contains the web service for parsing the underlying BIM model. XML parsers or APIs designed for a particular data model may be used for deploying this web service unit depending on the BIM data model. This web service performs two functions on the basis of the selection made by the end user: (1) parse and extract from BIM and (2) parse and update to BIM. For example, energy related information like u-value of materials, number of occupants, and window to wall ratio may be extracted from the BIM model for energy simulation. In cases where the BIM model has missing information, the web service unit prompts the end user to upload pre-defined templates containing information required for the energy simulation. The web service unit has pre-defined functions to read the user added information from the templates and update the BIM model through parser functions (for example using XML parsers to update a gbXML BIM model). The updated BIM model contains comprehensive information from more than one source and may be used later for other processes in green building design like code compliance checking. The third part of our web service framework is Layer III which contains web services for (1) energy simulation, (2) building code checking, and (3) model updating. The functions of these web services are described in detail in sections 3.1, 3.2 and 3.3, respectively. The web service framework outputs reports in formats like HTML and MS Excel, upon completion of any of the processes. These reports contain energy simulation results, building code checking results, and BIM model updates. The information required to run a web service unit is stored in data files (like gbXML file or weather file) in a web server hosted on the same machine as the web service unit. The data files are stored project wise in different folders named with the respective project name or ID. By hosting the documents on a web server, they can be uploaded and updated to the respective folders by authorized end users through a simple program written in a server-side scripting language like PHP (PHP Group, 1997). The files can also be viewed easily on web browsers using public URLs.

3.1 Module 1 - Building Energy Analysis

FIG 2 shows a flowchart describing the process of automatic energy analysis simulation conducted by our web service framework. The prototype web service framework performs the building energy simulation for four categories: (1) Lighting, (2) Equipment, (3) People, and (4) HVAC. The currently applied energy simulation engines are thereby directly or indirectly compatible to the BIM data models. For example, both IFC and gbXML BIM data models are compatible with the simulation software, EnergyPlus, through an internal process of conversion. At the beginning, the HVAC designers or the architects, who are experts in this area, add the values for lighting, equipment, people or HVAC to the initial CAD model. The energy simulation is then performed by the web service framework in seven steps. The first step is to parse the BIM building model and extract geometry and energy simulation related values. The BIM file is parsed by a customized parser software program embedded in the web service module in Layer II (see FIG 1). This web service extracts energy simulation parameters like energy consumption for lighting and HVAC installations and the number of people and stores in a data structure. In the second step, the user is prompted to load an information template containing information like building type and schedules. The building types supported by the prototype web service framework are “Office”, “Hotel”, and “Hospital”. In the third step, the parameters extracted from the gbXML file is checked for containing the parameters required for running an energy simulation process for each sub-domain like lighting and HVAC. If the simulation related parameters are not found, in the fourth step the end user is prompted to upload respective templates containing the missing values in a pre-defined structure. In the fifth step, extracted parameters are written to a standard input file validated against a schema or data dictionary compatible to the simulation engine. For example, EnergyPlus uses an IDD file (Data Dictionary file) that defines the conventions to be used to create its input file. The sixth step is then to upload the weather file of the region where the building is located. This is followed by the seventh step where the simulation engine is run. Finally in the seventh step, standard energy simulation reports are generated by the simulation engine and hosted on the same web server which hosts the web service unit. Additionally, the BIM model is updated with energy simulation related information, which was missing in the original BIM model data (as it was missing in the CAD model during design). Considering that our BIM model is a gbXML model, to explain how this is done let us take the example of lighting related information, “Watts per Zone Floor Area”, which has the following hierarchy: gbXML>Campus>Building>Space>LightPowerPerArea>powerPerAreaUnitEnum>WattPerSquareMeter. If this information was missing in the gbXML model, the end users are prompted to upload a file containing the relevant data which is either calculated through a function provided by the web service or manually provided. The gbXML model is updated at the relevant location in the hierarchy, thereby enriching it with information that can be used later for other processes like code checking and MEP design.

3.2 Module 2 - Building Code Checking

This module performs automated rule based code checking of the gbXML building model. The module contains a number of web service units which can be invoked through the frontend. Each of these web services executes a certain set of rules against the building model. For example, the user can invoke only the "lighting" web service unit when he wants to check if the light fittings in the building model are compliant with the guidelines for "Lighting" in the building energy code. The rules in the building energy codes which are written in natural languages are manually converted into machine computable sets of rules. In our web service framework these rules are implemented with a Java based rule engine called "Jess" (Friedman-Hill, 2009). However, other rule engines which are programmable and can be wrapped into web service units can also be used in the conceptual framework. FIG. 3 shows sample rules which are executed when the web service unit "Lighting" is invoked. Similarly, the user can perform the code compliance checking for other domains like "HVAC" and "Water Usage" by invoking other web service units in the same module (FIG 1).

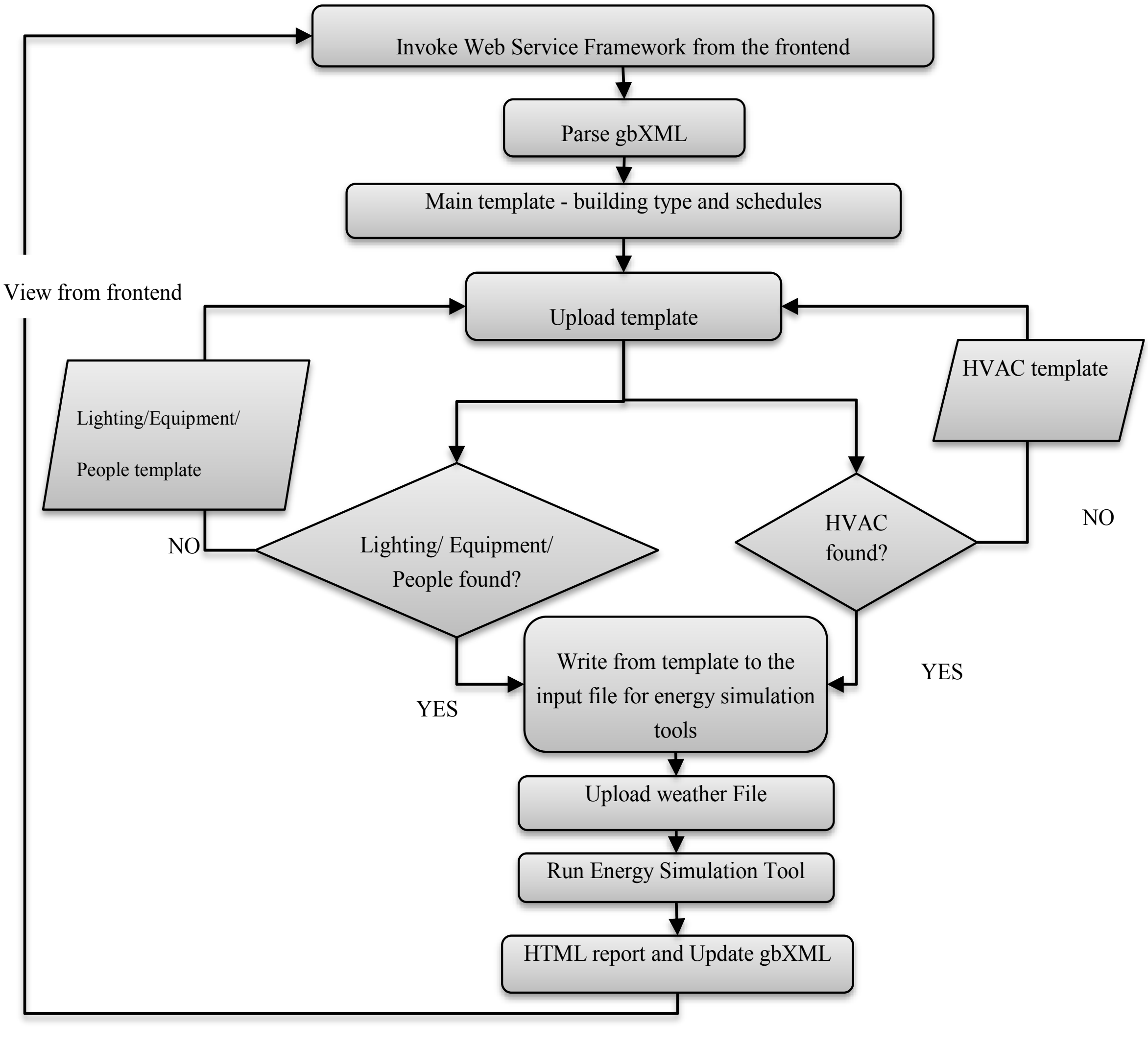
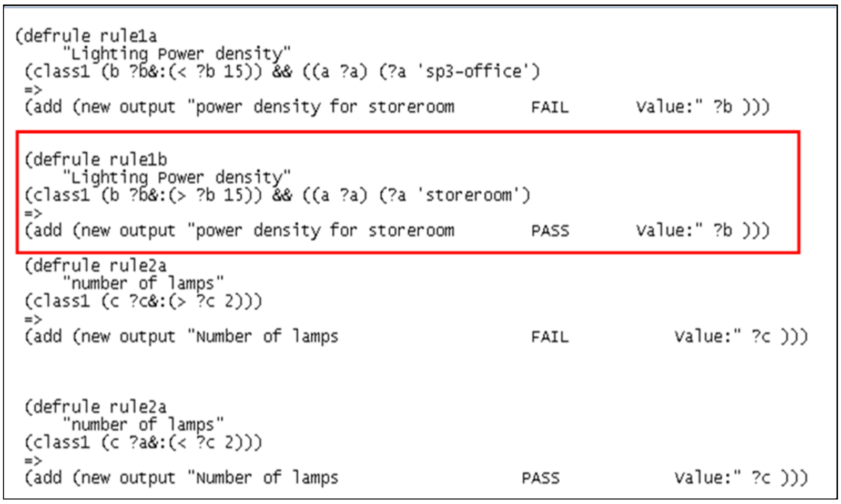

Invoke Web Service Framework from the frontend
Parse gbXML
Main template - building type and schedules
View from frontend
Upload template
Lighting/Equipment/
People template
HVAC template
NO
Lighting/ Equipment/
People found?
HVAC
found?
NO
Write from template to the
input file for energy simulation
tools
YES
YES
Upload weather File
Run Energy Simulation Tool
HTML report and Update gbXML

FIG 2: Flowchart describing the functioning of the web service framework for building energy analysis



```
(defrule rule1a
    "Lighting Power density"
 (class1 (b ?b&:(< ?b 15)) && ((a ?a) (?a 'sp3-office')
 =>
 (add (new output "power density for storeroom        FAIL       Value:" ?b )))

 (defrule rule1b
    "Lighting Power density"
 (class1 (b ?b&:(> ?b 15)) && ((a ?a) (?a 'storeroom')
 =>
 (add (new output "power density for storeroom        PASS       Value:" ?b )))
 (defrule rule2a
    "number of lamps"
 (class1 (c ?c&:(> ?c 2)))
 =>
 (add (new output "Number of lamps                    FAIL           Value:" ?c )))


 (defrule rule2a
    "number of lamps"
 (class1 (c ?a&:(< ?c 2)))
 =>
 (add (new output "Number of lamps                   PASS            Value:" ?c )))
```

FIG. 3: Jess rule file for lighting

3.3 Module 3 - Building Model Updating

The prototype web service framework can also manipulate the geometry of the BIM building model by invoking the web services in "Module 3" (FIG 1). The web services in this module can make simple updates like changing the location and thickness of a wall, and delete building elements like windows, and doors. The relevant web services are programmed using BIM data model parser features to perform functions like "Delete Wall" for deleting and "Modify Wall" for updating a building element in the BIM data model, respectively. These web service units are invoked depending upon the parameters passed to the web service framework by the user. For example, if the user passes the parameter "Delete Wall" from the frontend, then the workflow controller web service in Layer I will invoke the "Delete Wall" web service of the Layer III and delete the wall defined by the end user. The functions provided for geometry update by the web service framework is very basic and can deal only with simple geometrical changes to the gbXML data models which do not create any conflict with other building elements. However, in future, the proposed web service framework may be incorporated on top of a database with flexible schema, enabling more sophisticated geometry changes. Such an approach has been discussed by Cheng and Das (2013), who propose a cloud based framework for exchanging partial BIM information and a data model for storing BIM models on a cloud database.

4. IMPLEMENTATION OF A GBXML BASED WEB SERVICE FRAMEWORK

4.1 Green Building XML (gbXML)

Green Building XML is an open standard schema that was developed to facilitate the transfer of building properties stored in CAD and BIM to engineering analysis tools (gbXML.org, 2010). Development of Green Building XML was started by Green Building Studio in 1999. The first version was released in June 2000, which was followed by many revised versions. The latest version is gbXML 5.10 released in the year 2013. The gbXML standard uses XML which enables web based applications to interact automatically with each other, with no or minimal human intervention. A document encoded in XML is both machine and human readable. XML documents containing information can be exchanged over the Internet very easily and efficiently through web enabled applications. Due to these features, gbXML has become a standard industry schema for leading modelling tools like Autodesk Revit and Google SketchUp. Most of these BIM building modeling software provide import/export capabilities for gbXML. There are other popular industry standards for BIM like IFC, but none of them covers the building energy domain as extensively as gbXML does. The gbXML schema can accommodate building information like building geometry, schedules, weather, HVAC system, lighting, and thermal zone related data. FIG 4 and FIG 5 show that different spaces may be grouped together into a common thermal zone. FIG 4 shows graphically that space 2 and space 3 are grouped into thermal zone 1. FIG 6 (Left) shows the gbXML document for a simple building model. The element "Campus" defines the building location. One "Campus" may define one or more buildings under it using the child element "Building". The walls, roofs, and floors of the building are defined by “Surface” elements. The “Surface” element stores information about a surface such as an unique id, surface type, direction of the surface, openings in the surface, unique ids of space and other adjacent surfaces, and the geometry of the surface. The latter is stored in the child element “PlannerGeometry”.

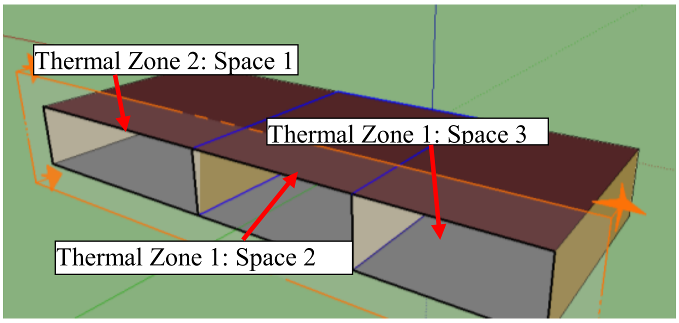
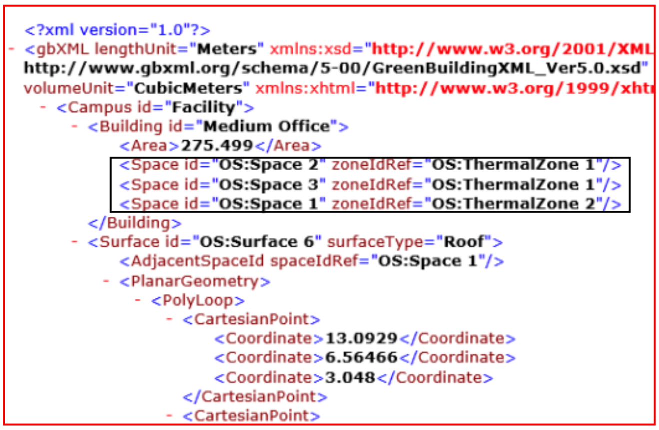
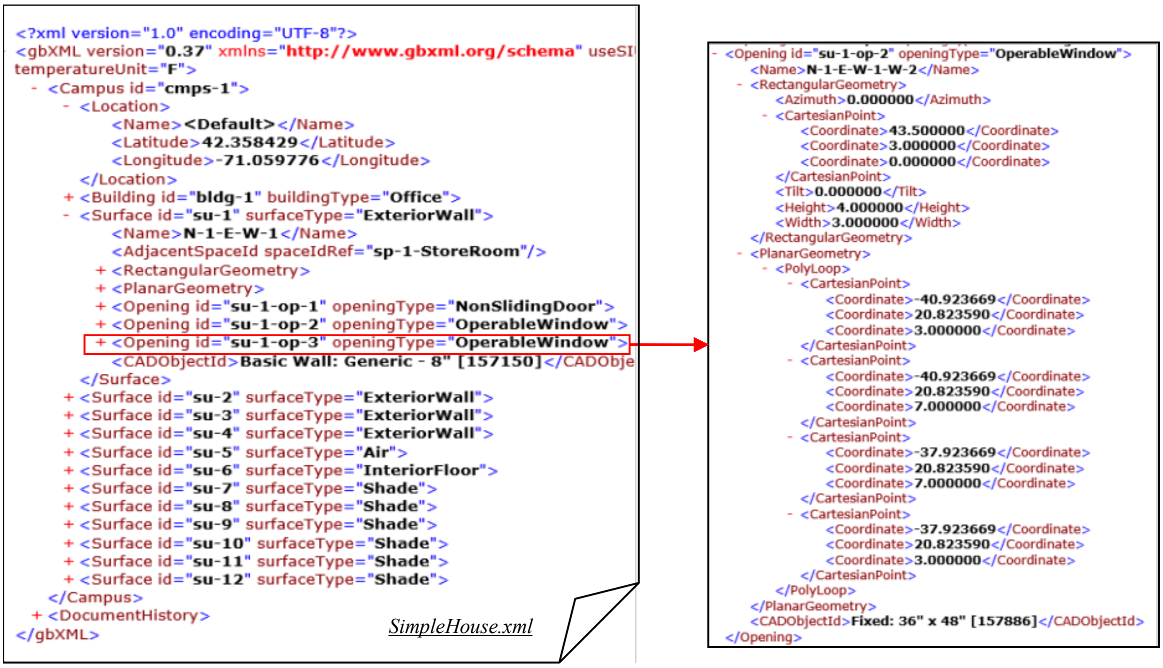


FIG 4: Simple Google SketchUp building model showing Space and Thermal zones



```
  <?xml version="1.0"?>
- <gbXML lengthUnit="Meters" xmlns:xsd="http://www.w3.org/2001/XML
  http://www.gbxml.org/schema/5-00/GreenBuildingXML_Ver5.0.xsd"
  volumeUnit="CubicMeters" xmlns:xhtml="http://www.w3.org/1999/xht
  - <Campus id="Facility">
    - <Building id="Medium Office">
        <Area>275.499</Area>
        <Space id="OS:Space 2" zoneIdRef="OS:ThermalZone 1"/>
        <Space id="OS:Space 3" zoneIdRef="OS:ThermalZone 1"/>
        <Space id="OS:Space 1" zoneIdRef="OS:ThermalZone 2"/>
      </Building>
    - <Surface id="OS:Surface 6" surfaceType="Roof">
        <AdjacentSpaceId spaceIdRef="OS:Space 1"/>
      - <PlanarGeometry>
        - <PolyLoop>
          - <CartesianPoint>
              <Coordinate>13.0929</Coordinate>
              <Coordinate>6.56466</Coordinate>
              <Coordinate>3.048</Coordinate>
            </CartesianPoint>
          - <CartesianPoint>
```

FIG 5: gbXML file showing the relation between Space, Zone and Surfaces



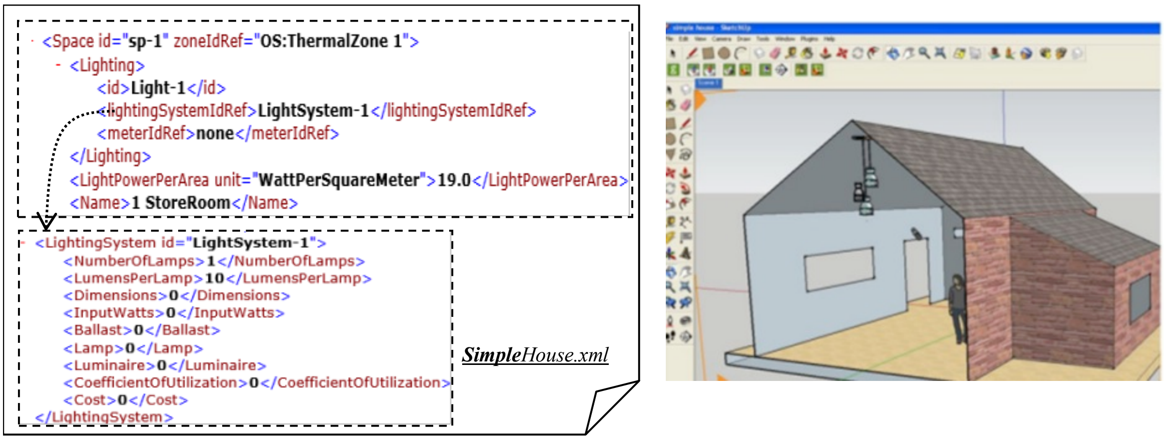
```
<?xml version="1.0" encoding="UTF-8"?>
<gbXML version="0.37" xmlns="http://www.gbxml.org/schema" useSI
temperatureUnit="F">
 - <Campus id="cmps-1">
   - <Location>
       <Name><Default></Name>
       <Latitude>42.358429</Latitude>
       <Longitude>-71.059776</Longitude>
     </Location>
   + <Building id="bldg-1" buildingType="Office">
   - <Surface id="su-1" surfaceType="ExteriorWall">
       <Name>N-1-E-W-1</Name>
       <AdjacentSpaceId spaceIdRef="sp-1-StoreRoom"/>
     + <RectangularGeometry>
     + <PlanarGeometry>
     + <Opening id="su-1-op-1" openingType="NonSlidingDoor">
     + <Opening id="su-1-op-2" openingType="OperableWindow">
     + <Opening id="su-1-op-3" openingType="OperableWindow">
       <CADObjectId>Basic Wall: Generic - 8" [157150]</CADObje
     </Surface>
   + <Surface id="su-2" surfaceType="ExteriorWall">
   + <Surface id="su-3" surfaceType="ExteriorWall">
   + <Surface id="su-4" surfaceType="ExteriorWall">
   + <Surface id="su-5" surfaceType="Air">
   + <Surface id="su-6" surfaceType="InteriorFloor">
   + <Surface id="su-7" surfaceType="Shade">
   + <Surface id="su-8" surfaceType="Shade">
   + <Surface id="su-9" surfaceType="Shade">
   + <Surface id="su-10" surfaceType="Shade">
   + <Surface id="su-11" surfaceType="Shade">
   + <Surface id="su-12" surfaceType="Shade">
   </Campus>
 + <DocumentHistory>
</gbXML>
```

SimpleHouse.xml

```
- <Opening id="su-1-op-2" openingType="OperableWindow">
    <Name>N-1-E-W-1-W-2</Name>
  - <RectangularGeometry>
      <Azimuth>0.000000</Azimuth>
    - <CartesianPoint>
        <Coordinate>43.500000</Coordinate>
        <Coordinate>3.000000</Coordinate>
        <Coordinate>0.000000</Coordinate>
      </CartesianPoint>
      <Tilt>0.000000</Tilt>
      <Height>4.000000</Height>
      <Width>3.000000</Width>
    </RectangularGeometry>
  - <PlanarGeometry>
    - <PolyLoop>
      - <CartesianPoint>
          <Coordinate>-40.923669</Coordinate>
          <Coordinate>20.823590</Coordinate>
          <Coordinate>3.000000</Coordinate>
        </CartesianPoint>
      - <CartesianPoint>
          <Coordinate>-40.923669</Coordinate>
          <Coordinate>20.823590</Coordinate>
          <Coordinate>7.000000</Coordinate>
        </CartesianPoint>
      - <CartesianPoint>
          <Coordinate>-37.923669</Coordinate>
          <Coordinate>20.823590</Coordinate>
          <Coordinate>7.000000</Coordinate>
        </CartesianPoint>
      - <CartesianPoint>
          <Coordinate>-37.923669</Coordinate>
          <Coordinate>20.823590</Coordinate>
          <Coordinate>3.000000</Coordinate>
        </CartesianPoint>
      </PolyLoop>
    </PlanarGeometry>
    <CADObjectId>Fixed: 36" x 48" [157886]</CADObjectId>
</Opening>
```

FIG 6: gbXML file of the shown simple building model (left), and element details of the gbXML file (right)

Any opening in the surface is represented by another child element of “Surface”, called “Opening”. The geometry of the opening is defined with reference to a local coordinate system in terms of coordinate points in the elements “CartesianPoint”. In addition to the geometry related data, gbXML has elements for storing energy related data. For example, gbXML stores lighting information in the element “Lighting” and “LightPowerPerArea” under the “Space” and “Zone” (thermal zone) it belongs to (FIG 7). Even though the gbXML schema covers many domains related to a building, some of the available software may require additional information which cannot be incorporated in the existing gbXML schema. In this case, the gbXML schema can be extended easily without understanding the whole schema. For example, Dong et al. (2007) extended the schema of gbXML to add the domain “lighting” to support lighting simulation to be run on simulation software called “Radiance”.



```
· <Space id="sp-1" zoneIdRef="OS:ThermalZone 1">
  - <Lighting>
       <id>Light-1</id>
       <lightingSystemIdRef>LightSystem-1</lightingSystemIdRef>
       <meterIdRef>none</meterIdRef>
    </Lighting>
    <LightPowerPerArea unit="WattPerSquareMeter">19.0</LightPowerPerArea>
    <Name>1 StoreRoom</Name>
- <LightingSystem id="LightSystem-1">
     <NumberOfLamps>1</NumberOfLamps>
     <LumensPerLamp>10</LumensPerLamp>
     <Dimensions>0</Dimensions>
     <InputWatts>0</InputWatts>
     <Ballast>0</Ballast>
     <Lamp>0</Lamp>
     <Luminaire>0</Luminaire>
     <CoefficientOfUtilization>0</CoefficientOfUtilization>
     <Cost>0</Cost>
  </LightingSystem>
```

FIG 7: gbXML schema showing information about lighting

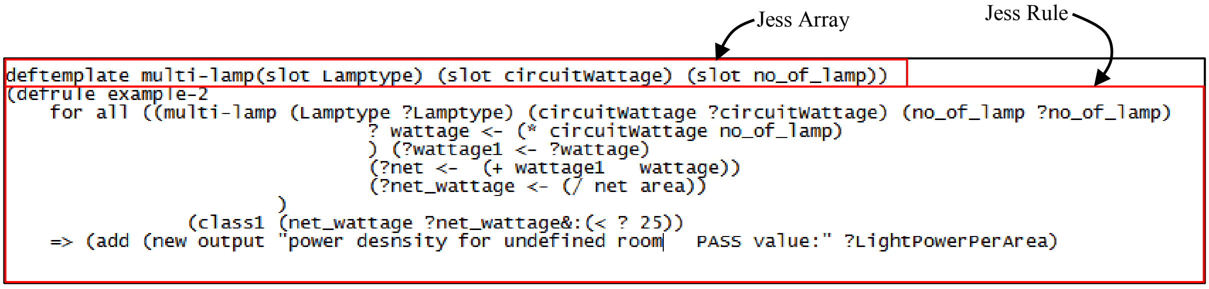
The building model is represented in gbXML. The building model is stored on a project basis. The architect or the designer has the access rights to upload a new building model. Upon successful login, a user is given access to the project folder that the user is working in. The gbXML file is used as single point source to perform building code checking, model updating, and building energy analysis. For the code checking and model updating, the gbXML file can be used directly. However, for energy analysis, gbXML has to be converted into the proprietary file representation compatible with the energy simulation engine. In our web service framework, the gbXML file is automatically converted to a file format called “.idf” which is compatible to the EnergyPlus simulation engine.

4.2 Energy simulation and analysis engine - EnergyPlus

EnergyPlus is a building performance simulation program for modeling heating, cooling, lighting, ventilation, and other energy flows (Crawley et al. 2008). The main goal of developing EnergyPlus was to provide an organized and modular energy simulation program that supports adding new features and links to other programs easily. EnergyPlus is popular in the construction industry due to its innovative simulation capabilities like multi-zone air flow, thermal comfort, natural ventilation, and photovoltaic systems. Also, due to its modular architecture and simple input and output data formats, it can be easily integrated with third-party software applications. In our web service framework, the EnergyPlus engine is packaged in a Java based web service which can be accessed through the Internet. The underlying calculation method of EnergyPlus uses a “Heat Balance Model”. The main assumptions of the heat balance model are that (1) each thermal zone can be modeled as well as stirred with uniform temperature throughout, and (2) the room surfaces have uniform surface temperature and one dimensional heat conduction throughout. For a successful simulation, EnergyPlus requires a minimum of two types of input files. The first input file is a text based file with an extension “.idf” containing information like schedule, construction type, HVAC elements, number of people, and building element geometries. The second input file is the weather file which has an extension of “.epw”. The weather file contains information like location, latitude, longitude, holidays, daylight saving and periodic weather data. EnergyPlus reports the building simulation results in multiple formats like “.csv”, “.html”, and “.txt”.

4.3 Code checking rule engine - Jess Rule Engine

As explained in Section 3.2, the building energy codes are converted into a computable set of rules which can be executed in Jess (Friedman-Hill, 2009). Jess is a Java-based rule engine and scripting environment developed by Sandia National Laboratories. The Jess engine is small, light and one of the fastest rule engines available. Since Jess is designed for Java, it is very easy to call the Jess engine from Java programs or vice versa. Multiple objects of the Jess engine can be created for a single program. Each individual Jess engine object has its own working memories and can call functions in separate threads. The implementation of the conceptual framework creates a Jess engine for every domain and performs parallel code checking. FIG 8 shows a Jess rule which calculates and checks the validity of the maximum lighting power density with that prescribed in standard energy codes. In this example, combination lighting is considered for a space. There might be several types of lamps in the combination lighting system. Four variables are required to calculate the lighting power density of the space, i.e. type of lamp, circuit wattage of each type of lamp, number of lamps of each type, and floor area of the space. As shown in FIG 8, the first line with keyword "*deftemplate*" declares a Jess Array with the three required variables related to lamps. In the second line, the Jess rule is defined with the keyword "defrule". Each Jess rule has two parts: (1) the antecedent and (2) the consequence (following the "=>" sign). In this rule, the circuit wattage of each type of lamp is multiplied by the number of lamps and summed up together. This sum is divided by the floor area to determine the lighting power density of the space. Then, using the class keyword, the lighting power density is checked for a maximum limit. If this condition is satisfied, "PASS" is printed. In this paper, we convert the green building codes written in natural language into Jess rules manually. By using these rules with Java, more extensive functions may be developed and illustrative reports may be generated.



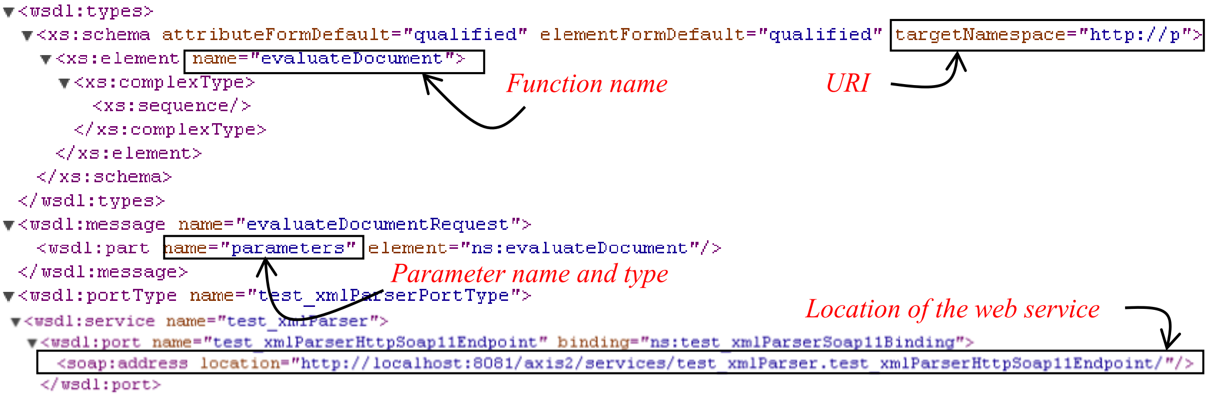
```
deftemplate multi-lamp(slot Lamptype) (slot circuitWattage) (slot no_of_lamp))
(defrule example-2
    for all ((multi-lamp (Lamptype ?Lamptype) (circuitWattage ?circuitWattage) (no_of_lamp ?no_of_lamp)
                                ? wattage <- (* circuitWattage no_of_lamp)
                                ) (?wattage1 <- ?wattage)
                                (?net <-  (+ wattage1   wattage))
                                (?net_wattage <- (/ net area))
                        )
                (class1 (net_wattage ?net_wattage&:(< ? 25))
    => (add (new output "power desnsity for undefined room   PASS value:" ?LightPowerPerArea)
```

FIG 8: Example Jess rule

4.4 Standardized web service technology – SOAP, WSDL, and BPEL

Web services are the key functional units of our web service framework. Web services are self-contained web enabled applications which have the ability to perform activities themselves and also invoke other web service units through the Internet (Christopher and Joel, 2003). A web service is a software tool identified by a URI (Uniform Resource Identifier), whose public interfaces and bindings are formally defined and described. The technical specifications of a web service are openly published and are accessible to all its target users. A web service may be discovered and invoked by using its technical specifications. Web services have been implemented in the construction industry for various purposes, such as information sharing among mobile construction workers (Aziz et al. 2006), supply chain collaboration (Cheng et al. 2010), measuring and monitoring carbon footprint in construction supply chains (Cheng, 2011), interoperability between web-based construction product catalogues (Kong et al. 2005), etc. Implementation of web services in a SOA has been explored for designing a seamlessly integrated platform using RFID/WSN technology for communicating and managing information from heterogeneous sources in construction supply chains (Shin et al. 2011).

Standardized web services use the Web Services Description Language (WSDL) to publish these technical specifications like operations hosted and the format of input/output related to a web service unit. Each of the web service units shown in FIG 1 have a unique WSDL file associated with it. FIG 9 shows a part of the WSDL associated with the web service unit which parses the gbXML file shown in FIG 5. The four parameters required to invoke a web service are (1) function name, (2) URI, (3) parameter name and type, and (4) location (address of the machine where the web service is hosted). With this information from the WSDL file, a web service can be invoked e.g. through a JSP (Java Server Pages) or a PHP frontend.



```
▼<wsdl:types>
  ▼<xs:schema attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://p">
    ▼<xs:element name="evaluateDocument">
      ▼<xs:complexType>
          <xs:sequence/>
        </xs:complexType>
      </xs:element>
    </xs:schema>
  </wsdl:types>
▼<wsdl:message name="evaluateDocumentRequest">
    <wsdl:part name="parameters" element="ns:evaluateDocument"/>
  </wsdl:message>
▼<wsdl:portType name="test_xmlParserPortType">
 ▼<wsdl:service name="test_xmlParser">
   ▼<wsdl:port name="test_xmlParserHttpSoap11Endpoint" binding="ns:test_xmlParserSoap11Binding">
      <soap:address location="http://localhost:8081/axis2/services/test_xmlParser.test_xmlParserHttpSoap11Endpoint/"/>
    </wsdl:port>
```

FIG 9: WSDL of the web service unit "gbXML parser"

Standardized web services use the open standard Simple Object Access Protocol (SOAP) which is an XML based standard to exchange information. SOAP is useful in distributed computing where big processes are modularized to implement security, reusability, and scalability. Each module which is a web application itself may reside on a different computer. A web application ("client") communicates by sending "requests" for data to another web application ("server") through the Internet. The communication is completed by sending a "reply" to the invoking web application. This communication of request and reply is facilitated through SOAP using XML and HTTP.

The standardized web services can invoke each other by using the information in the WSDL document by exchanging SOAP messages. Depending upon the requirements of the end user, these modular web components can be orchestrated to serve the functionality of a bigger application. The business logic of orchestrating these smaller web applications can be defined by using the XML-based Business Process Execution Language, BPEL. Many graphical user interfaces like the BPEL plugin for eclipse (Eclipse, 2004) are available to create the needed business logic through BPEL. Users can create links to the available web services (with the information from WSDL) through the Graphical User Interface for BPEL very easily.

5. EXAMPLE SCENARIO

The following section presents an example scenario demonstrating the functionalities of the proposed web service framework. FIG 10 shows how our web service framework is applicable to green building design of a small office building located in Hong Kong. In this example, we look at two high level requirements addressed by the owner:

1) Moving Wall1 to increase the area of the storeroom, and
2) Changing the material of the interior walls to a cheaper one.

In a multi-party environment where many actors like structural engineer, architect, and sustainable development specialist work together, it is very important to effectively communicate the information among the group. The requirements need to be validated across multiple domains, as a change made in one domain impacts the other. Relations between different domains in the initial phase of green building design will be explained in this section. As shown in FIG 10, the owner wants to increase the size of Room1 by moving Wall1. This change in the geometry of the building will instigate a new set of requirements for energy analysis and code checking of the modified spaces. Some examples of the new requirements are energy simulation for lighting power per area for Room1 and Room2 for the change in room area, and code compliance checking again to ensure that the new lighting is within the allowable limits of lighting power per unit area.

The consequent processes initiated due to the changes demanded by the owner are described in Table 1. In this scenario, many project stakeholders are working together in parallel and therefore need access to the most updated information model of the building. The gbXML model acts as the central repository of the most updated information for our web service framework. The end users can rely on it for running their processes like energy simulation and building code compliance checking on the most recent version of the building model.

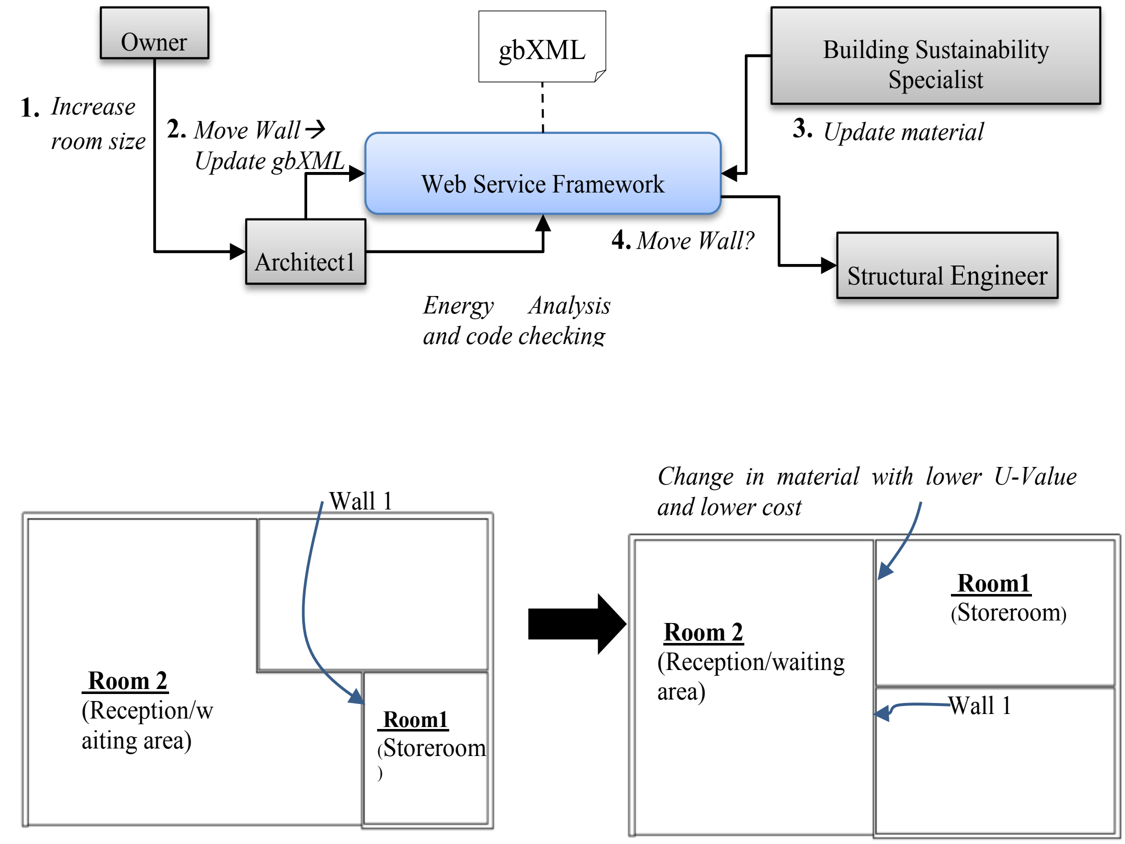


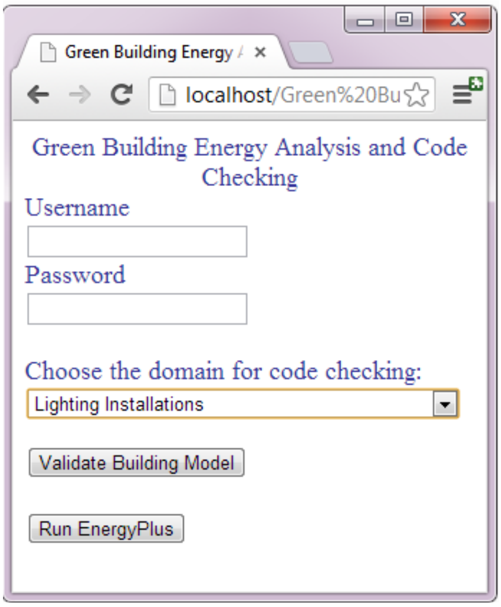
FIG 10: Example scenario demostrating the application of the web service framework in green building design

Table 1: Changes and impacts triggered in the example scenario

Changes	**Impacted processes and responsible project stakeholders**
1. Change in room area of Room1 and Room2	Energy analysis for the lighting power per area → Energy Code Checking (Architect)
2. Change in location of Wall1	Feasibility analysis of moving a brick wall on the floor and structural changes → Update BIM model (Structural Engineer)
3. Change in length of curtain wall	Less natural lighting in Room 2 → Energy Simulation → Code Checking (Architect)
4. Change in the material of the interior walls	Update BIM model to add new material (Building sustainability specialist)

In the following sections, we demonstrate the three key functions of the prototype web service framework namely *energy analysis*, *code compliance checking* and *model updating* for the scenario outlined in FIG 10. As mentioned above, communication is going on among the architect, the owner, the structural engineer and the sustainable material expert.

FIG 11 shows the frontend that is used by the users to access and use the functionalities provided by the prototype web service framework. This frontend first takes input information like the credentials of a user and the domain name on which the user intends to run an analysis process. In our example, we choose the domain "lighting installations". When clicked, the buttons "Validate Building Model" or "Run EnergyPlus" initiate the building code checking or energy simulation web service units respectively and report the results to the end users. Currently a simple prototype of the frontend is used for testing purposes. In future, the frontend may be updated to adapt to specific user roles based on the credentials.



Green Building Energy Analysis and Code Checking

Username

Password

Choose the domain for code checking: Lighting Installations

FIG 11: The frontend for invoking the web service framework

5.1 Update Building Model

A gbXML file stores the building information in XML format which can be easily parsed and manipulated automatically by a Java program. In this scenario, to move a wall from the building model in gbXML through the web service framework, the end user downloads the gbXML model and views it through an open source IFC/ gbXML viewer, FZKViewer (KIT, 2007). FIG 12 (left) shows the gbXML building model viewed using the FZKViewer interface. It allows the users to examine the breakdown structure of a building on element level. As shown in FIG 13 (left), a building element “wall” with id “N-1-E-W-1” was selected in the browser toolbar of the FZKViewer. FIG 12 (right) shows the graphical representation while FIG 13 (right) shows the geometrical properties of the same wall. The user inputs for wall ID “N-1-E-W-1” the new coordinates as shown in FIG 13 (right), and a parameter “modify Wall” through the frontend to the web service framework. The workflow controller web service (FIG 1) invokes the “Modify wall” web service unit based on the user input and replaces the old coordinates with the typed in new coordinates.

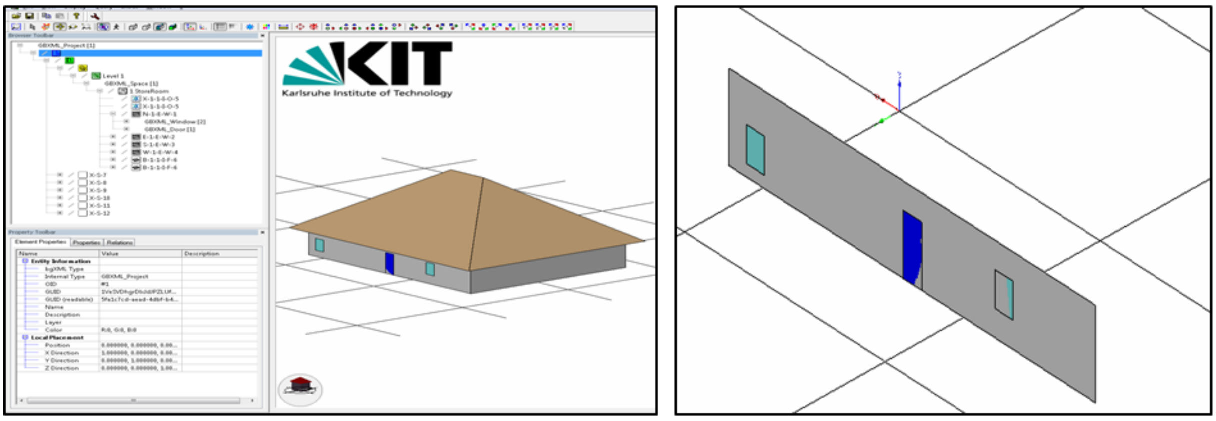
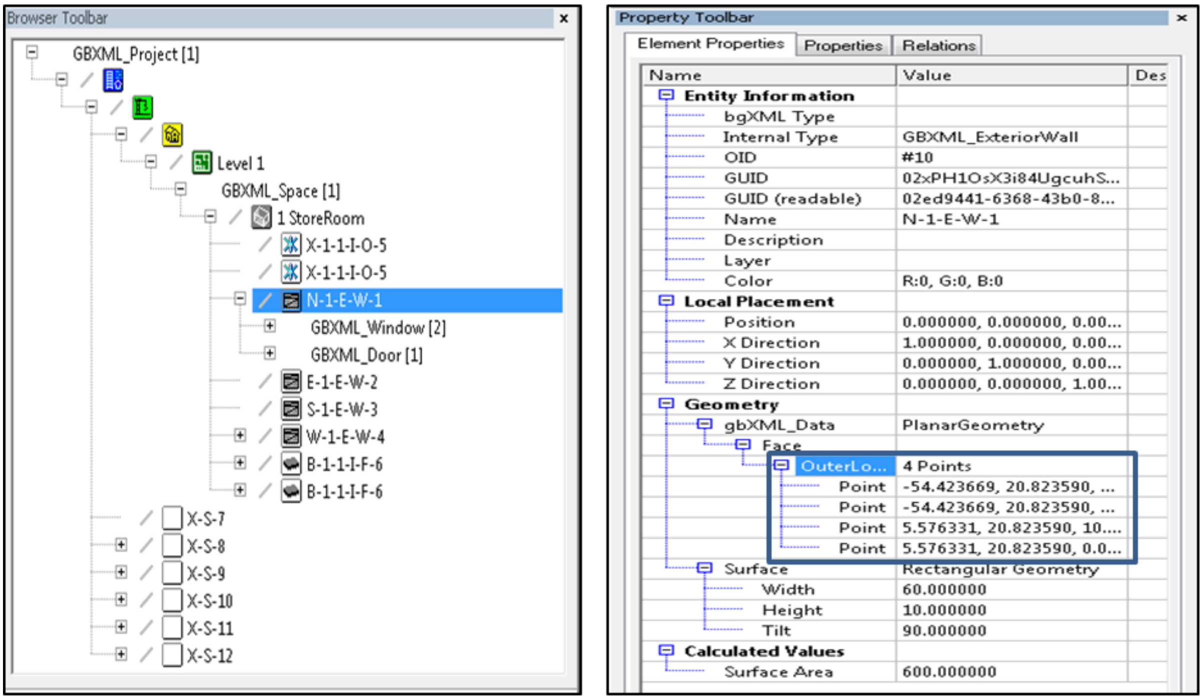


FIG 12: The interface of the FZKViewer for viewing gbXML models (left), and view in the graphical interface when a wall is selected from the gbXML tree (right)



Browser Toolbar

```
GBXML_Project [1]
  Level 1
    GBXML_Space [1]
      1 StoreRoom
        X-1-1-I-O-5
        X-1-1-I-O-5
        N-1-E-W-1
          GBXML_Window [2]
          GBXML_Door [1]
        E-1-E-W-2
        S-1-E-W-3
        W-1-E-W-4
        B-1-1-I-F-6
        B-1-1-I-F-6
  X-S-7
  X-S-8
  X-S-9
  X-S-10
  X-S-11
  X-S-12
```

Property Toolbar

Name	Value	Des
Entity Information		
bgXML Type		
Internal Type	GBXML_ExteriorWall	
OID	#10	
GUID	02xPH1OsX3i84UgcuhS...	
GUID (readable)	02ed9441-6368-43b0-8...	
Name	N-1-E-W-1	
Description		
Layer		
Color	R:0, G:0, B:0	
Local Placement		
Position	0.000000, 0.000000, 0.00...	
X Direction	1.000000, 0.000000, 0.00...	
Y Direction	0.000000, 1.000000, 0.00...	
Z Direction	0.000000, 0.000000, 1.00...	
Geometry		
gbXML_Data	PlanarGeometry	
Face		
OuterLo...	4 Points	
Point	-54.423669, 20.823590, ...	
Point	-54.423669, 20.823590, ...	
Point	5.576331, 20.823590, 10....	
Point	5.576331, 20.823590, 0.0...	
Surface	Rectangular Geometry	
Width	60.000000	
Height	10.000000	
Tilt	90.000000	
Calculated Values		
Surface Area	600.000000	

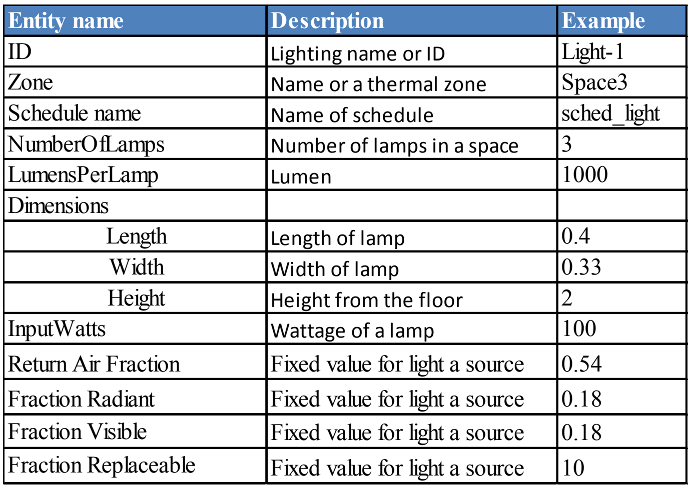
FIG 13: Browser toolbar of the FZKViewer showing the gbXML tree (left), and properties toolbar of the FZKViewer showing the properties of a selected element, wall N-1-E-W-1 (right)

5.2 Building Energy Analysis

As explained in Section 3.1, the EnergyPlus engine is used to conduct building energy simulation in our proposed framework. For Room 1 and Room 2 in the shown example (FIG 10), this means to first increase the room area, reduce daylight and change wall material. The energy simulation follows the procedure presented in FIG 2 which requires four types of files:

1) a gbXML building model,
2) a building template,
3) a lighting template,
4) a weather file.

In this example, a building template for a typical building and a weather file for the Hong Kong region were chosen. FIG 14 shows the contents of a “Lighting Template” uploaded by the end user if lighting information is not found in the gbXML model. The entity name is the same as the entity names for lighting present in the gbXML schema. The template contains entity names and values of the parameters required in performing energy analysis and code checking. For example, as shown in FIG 14, some entities from the template are unique ID for every light fitting, the associated zone (in this case, the whole building is being simulated as a single zone), and schedule for the lighting. The template also contains four additional factors which are “Return Air Fraction”, “Fraction Radiant”, “Fraction Visible”, and “Fraction Replaceable”. These factors are used for converting the electrical energy input for lighting that is ultimately converted to heat that contributes as zone loads. The web service framework performs the energy simulation in four steps. First, the gbXML file is converted into the EnergyPlus Input File (IDF). EnergyPlus provides a data dictionary file to generate or validate its input files. Hence, a mapping between the geometrical and interior lighting information for a basic gbXML house model with four walls and a roof was provided to handle the conversion between the gbXML data and the data dictionary using Java lists in the web service unit. Second, the contents of the lighting template (FIG 14) and its schedule are appended to the EnergyPlus input file (by validating it against the data dictionary). Third, the EnergyPlus engine located on a remote machine is run to do the energy analysis. Finally in the fourth step, building energy simulation report in “.html” format is output. The gbXML model is also updated with the results of lighting analysis.

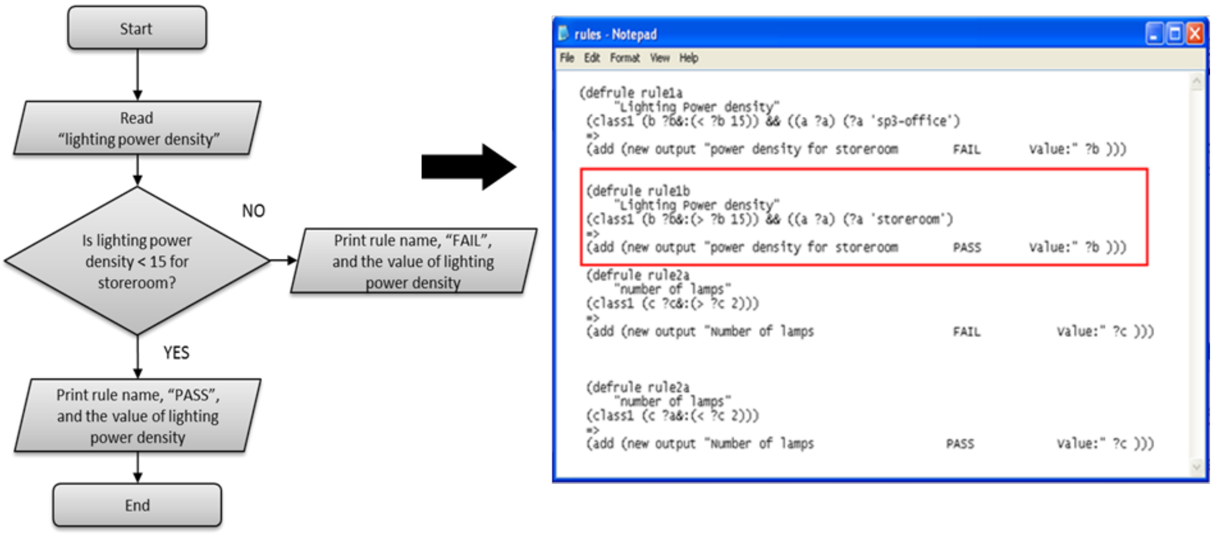


Entity name	Description	Example
ID	Lighting name or ID	Light-1
Zone	Name or a thermal zone	Space3
Schedule name	Name of schedule	sched_light
NumberOfLamps	Number of lamps in a space	3
LumensPerLamp	Lumen	1000
Dimensions		
Length	Length of lamp	0.4
Width	Width of lamp	0.33
Height	Height from the floor	2
InputWatts	Wattage of a lamp	100
Return Air Fraction	Fixed value for light a source	0.54
Fraction Radiant	Fixed value for light a source	0.18
Fraction Visible	Fixed value for light a source	0.18
Fraction Replaceable	Fixed value for light a source	10

FIG 14: Example of contents in the lighting template uploaded by the end users

5.3 Building Code Checking

In this example scenario, a gbXML building model of the type "storeroom" was checked against the lighting rules from the Building Energy Code of the Electrical & Mechanical Services Department (EMSD) (The Government of Hong Kong Special Administrative Region 2007). The EMSD Building Energy Code contains a set of five codes that covers lighting, air conditioning, electrical and lift & escalator installations. The EMSD building energy code for lighting was converted to Jess rules and executed by our framework. FIG 15 (left) shows the flowchart for checking the lighting power density of the lighting installation in a building. FIG 15 (right) shows the applied actual Jess rules from the EMSD building code. These rules check the maximum allowable lighting power density according to the EMSD rules (FIG 16 (left)). For example, the rule highlighted in FIG 15 (right) checks whether the lighting power density is lower than the allowable value, which is 15 Watt per square meter. If this condition is satisfied, the rule name is printed with the keyword PASS. Similarly, rules for other space types and other domains can be included. The rules have been kept modular and simple as most of the data processing is done through Java. Our web service framework executes the building codes against the gbXML model, which contains the geometry, installations and results from the lighting simulation. Hence, the gbXML model becomes an extensive source of information for code checking of the building for the lighting domain. The web service framework checks the gbXML building model for lighting power density for the modified floor plan and generates an ".html" report (FIG 16 (right)).


Start
Read "lighting power density"
Is lighting power density < 15 for storeroom?
NO
Print rule name, "FAIL", and the value of lighting power density
YES
Print rule name, "PASS", and the value of lighting power density
End
rules - Notepad
File Edit Format View Help

```
(defrule rule1a
     "Lighting Power density"
 (class1 (b ?b&:(< ?b 15)) && ((a ?a) (?a 'sp3-office')
 =>
 (add (new output "power density for storeroom        FAIL       value:" ?b )))

 (defrule rule1b
     "Lighting Power density"
 (class1 (b ?b&:(> ?b 15)) && ((a ?a) (?a 'storeroom')
 =>
 (add (new output "power density for storeroom        PASS       value:" ?b )))
 (defrule rule2a
     "number of lamps"
 (class1 (c ?c&:(> ?c 2)))
 =>
 (add (new output "Number of lamps                    FAIL           value:" ?c )))


 (defrule rule2a
     "number of lamps"
 (class1 (c ?a&:(< ?c 2)))
 =>
 (add (new output "Number of lamps                   PASS           value:" ?c )))
```

FIG 15: Flowchart describing the execution of a Jess rule for code checking against the Hong Kong EMSD lighting code (left), and Jess rules for the EMSD lighting code (right)

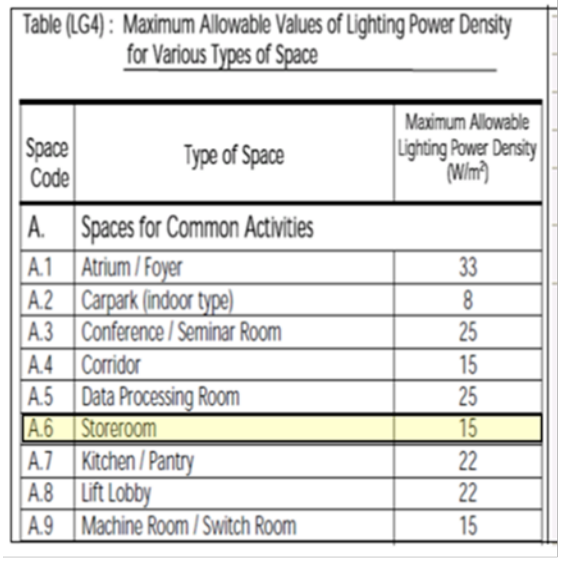
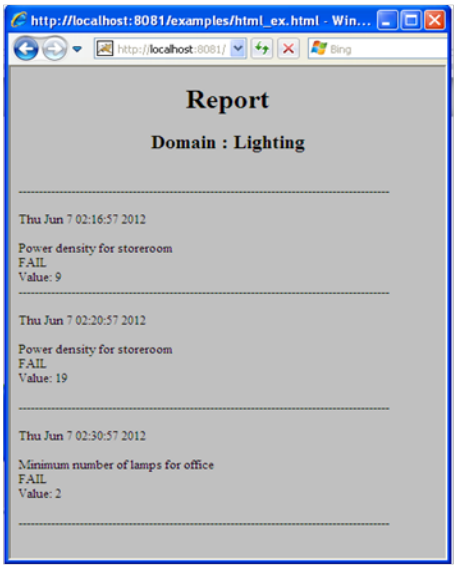


Table (LG4) : Maximum Allowable Values of Lighting Power Density for Various Types of Space

Space Code	Type of Space	Maximum Allowable Lighting Power Density (W/m²)
A.	Spaces for Common Activities	
A.1	Atrium / Foyer	33
A.2	Carpark (indoor type)	8
A.3	Conference / Seminar Room	25
A.4	Corridor	15
A.5	Data Processing Room	25
A.6	Storeroom	15
A.7	Kitchen / Pantry	22
A.8	Lift Lobby	22
A.9	Machine Room / Switch Room	15



Report

Domain : Lighting

Thu Jun 7 02:16:57 2012

Power density for storeroom
FAIL
Value: 9

Thu Jun 7 02:20:57 2012

Power density for storeroom
FAIL
Value: 19

Thu Jun 7 02:30:57 2012

Minimum number of lamps for office
FAIL
Value: 2

FIG 16: Lighting power density requirement from the EMSD building code (left), and HTML report generated by the Building Code Checking module (right)

6. CONCLUSION AND FUTURE WORK

This paper presented a BIM-based web service framework developed to perform three important functions related to the design of green buildings, i.e.:

1) Energy simulation,
2) Code checking,
3) Design and updating of the building model.

Green building design processes can achieve better solutions in terms of cost-effectiveness and energy-efficiency if an integrated approach is adopted to perform the functions related to the design of green buildings as described above. However, a green building design may involve many additional team members/stakeholders (geographically dispersed) as compared to the traditional design process, for example, a daylight specialist, a material consultant, and a natural ventilation expert. These team members perform different but interrelated operation. For example, decisions about choosing a greener alternative with cheaper materials highly depend on the energy simulation results. Therefore, an integrated platform for green design is necessary for exchanging information among the stakeholders efficiently and for making fast and effective decisions.

In the paper, we presented a BIM-based web service framework which performs green building processes like rule-based code checking, building energy simulation, and building model updating in a distributed manner. However, BIM models sometimes may miss the necessary data for the calculation of energy requirements and building code conformance of a building. Therefore, the prototype web service framework allows end users to upload files containing such information via web service units. The web service framework then updates the BIM model with the information uploaded by the end users, thereby creating an integrated BIM model that can be easily accessed and used through the web services in the developed framework.

BIM models in accordance to the gbXML schema are currently used in the prototyped web service framework presented in the paper because gbXML was designed specifically for the green building design domain. However, it is possible to use also other BIM data models with respective modification to the web services like changing the model parser web service with regard to the structure and type of the BIM data models used. Especially IFC can cover a wide scope of the AEC industry and is commonly used as the neutral file format of BIM models. However, IFC2x3 does not contain a sub-schema specifically dedicated to energy modeling and green building design. IFC2x4 which was released in 2013 covers the energy modeling and simulation domain, but it has gaps compared to gbXML for areas like weather, ventilation, and humidity. In spite of that, IFC remains the most comprehensive BIM data model to date. It covers the maximum number of AEC domains and can be used for life cycle information management of a building. Therefore in future, IFC may be extended and implemented in the proposed web service framework to provide a better approach to energy aware design and life cycle data management.

At last, although EnergyPlus is currently used in our prototype web service framework to perform energy simulation and generate standard energy simulation reports, the framework can integrate any other energy simulation engines that can be executed through the command line. Leveraging the Java based rule engine Jess, the framework also performs rule based code checking and can efficiently handle quantitative energy codes which can be expressed as mathematical expressions. However, qualitative areas like human comfort or those related to the surrounding built environment cannot be handled. Therefore, in future, semantic rule languages like the Semantic Web Rule Language (SWRL) may be explored to deploy qualitative codes into rule sets.

7. REFERENCES

Ahn, K.-U., Kim, Y.-J., Park, C.-S., Kim, I. & Lee, K. 2014. BIM interface for full vs. semi-automated building energy simulation. *Energy and Buildings*, 68, Part B, 671-678.

Autodesk 2005. *Ecotect* [Online]. Available: http://usa.autodesk.com/ecotect-analysis [Accessed 12 Oct. 2012].

Aziz, Z., Anumba, C. J., Ruikar, D., Carrillo, P. & Bouchlaghem, D. 2006. Intelligent wireless web services for construction - a review of the enabling technologies. *Automation in Construction,* 15**,** 113-123.

BRE (Building Research Establishment). 2008. *BRE Environmental Assessment Method* [Online]. Available: http://www.breeam.org [Accessed 10 July 2013].

buildingSMART 2014. *Industry Foundation Classes (IFC) data model* [Online].
Available: http://www.buildingsmart.org/standards/ifc [Accessed 17 Jan. 2014].

Cheng, J. C. 2011. A Web Service Framework for Measuring and Monitoring Environmental and Carbon Footprint in Construction Supply Chains. *Procedia Engineering,* 14**,** 141-147.

Cheng, J. C. P. & Das, M. 2013. A cloud computing approach to partial exchange of BIM models. *Proc. 30th CIB W78 International Conference*, 9-12 October 2013, Beijing, China.

Cheng, H.-C., Chen, M.-C. & Mao, C.-K.(2010). The evolutionary process and collaboration in supply chains. *Industrial Management & Data Systems*, 110 (3), 453-474.

Christodoulou, S. & Xanthos, S. 2014. *BIM extension for stochastic modelling* [Online].
Available: http://ises.eu-project.info/documents/ISES-D12.1.pdf [Accessed 18 May 2014].

Christopher, F. & Joel, F. 2003. What are Web services? *Journal of the ACM (JACM)* 46**,** 31.

Crawley, D. B., Hand, J. W., Kummert, M. l. & Griffith, B. T. 2008. Contrasting the capabilities of building energy performance simulation programs. *Building and Environment,* 43**,** 661-673.

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., Strand, R. K., Liesen, R. J., Fisher, D. E., Witte, M. J. & Glazer, J. 2001. EnergyPlus: creating a new-generation building energy simulation program. *Energy and Buildings*, 33, 319-331.

Dong, B., Lam, K. P., Huang, Y. C. & Dobbs, G. M. Year. A comparative study of the IFC and gbXML informational infrastructures for data exchange in computational design support environments. *Proc. Building Simulation Conference*, 3-6 September 2007, Beijing, China, 1530-1537.

Dym, C. L., Henchey, R. P., Delis, E. A. & Gonick, S. 1988. A knowledge-based system for automated architectural code checking. *Computer-Aided Design*, 20, 137-145.

Eastman, C., Lee, J.-M., Jeong, Y.-S. & Lee, J.-K. 2009. Automatic rule-based checking of building designs. *Automation in Construction*, 18, 1011-1033.

Eclipse 2004. *BPEL Designer Project* [Online]. Available: http://www.eclipse.org/bpel/ [Accessed 14 Aug. 2013].

eQuest 2012. *eQuest* [Online]. Available: http://doe2.com/equest [Accessed 12 October 2012].

Fiatech 2012. *AutoCodes: A Code-Check Revolution* [Online]. Available: http://www.fiatech.org/index.php/projects/active-projects/162-active-projects/projects-management/593-automated-code-plan-checking-tool-proof-of-concept [Accessed 8 Aug. 2013].

FIEMSER 2011. *FIEMSER System Architecture* [Online]. Available: http://www.fiemser.eu/ [Accessed 15 May 2014].

Friedman-Hill, E. 2009. *Jess - The rule engine for the Java platform*. Available: http://www.jessrules.com/jess/docs/

Gallaher, M. P., O'Connor, A. C., Dettbarn J. L. & Gilday L. T. 2004. *Cost Analysis of Inadequate Interoperability in the US Capital Facilities Industry*. Available: http://www.researchgate.net/publication/228583138_Cost_Analysis_of_Inadequate_Interoperability_in_the_US_Capital_Facilities_Industry [Accessed 13 July 2013].

GBCA (Green Building Council Australia). 2014. *greenstar* [Online]. Available: http://www.gbca.org.au/ [Accessed 12 August 2013].

gbXML.org 2010. *Green Building XML* [Online]. Available: http://www.gbXML.org [Accessed 17 July 2013].

Granlund 2014. *RIUSKA* [Online]. Available: http://www.granlund.fi/en/software/riuska/ [Accessed 6 May 2014].

Han, C. S. 1998. Client/server framework for on-line building code checking. *Journal of computing in civil engineering,* 12**,** 181-195.

HKGBC (Hong Kong Green Building Council). 2010. *Hong Kong Building Environmental Assessment Method* [Online]. Available: http://www.hkgbc.org.hk/eng/beamplus-main.aspx [Accessed 12 August 2013].

IES (Integrated Environment Solutions). 2005. *IES Virtual Environment* [Online]. Available: http://www.iesve.com [Accessed 3 october 2012].

Judkoff, R., Wortman, D. & Burch, J. 1982. *Empirical validation of building energy-analysis simulation programs: a status report*.

Karola, A., Lahtela, H., Hänninen, R., Hitchcock, R., Chen, Q., Dajka, S. & Hagström, K. 2002. BSPro COM-Server – interoperability between software tools using industrial foundation classes. *Energy and Buildings*, 34, 901-907.

Katranuschkov, P. & Balder, R. 2013. *Introduction to the HESMOS project* [Online]. Available: http://ises.eu-project.info/workshop-2013-10/01_Session_1_HESMOS-Introduction_to_the_HESMOS_project.pdf [Accessed 19 May 2014].

KIT (Karlsruhe Institute of Technology). 2007. *FZKViewer* [Online]. Available: http://iai-typo3.iai.fzk.de/www-extern/index.php?id=1931&L=1 [Accessed 16 Dec. 2013].

Kong, S. C. W., Li, H., Liang, Y., Hung, T., Anumba, C. & Chen, Z. 2005. Web services enhanced interoperable construction products catalogue. *Automation in Construction,* 14**,** 343-352.

LBNL (Lawrence Berkeley National Laboratory). 1994. *Overview of DOE-2.2* [Online]. Available: http://www.doe2.com/download/Docs/22_Oview.pdf [Accessed 17 Jan. 2014].

MacVittie, L. 2008. *SOA Infrastructure Reference Architecture: Defining the Key Elements of a Successful SOA Infrastructure Deployment (white paper)*.

MHC (McGraw Hill Construction) 2013. *World Green Building Trends.* Available: http://www.worldgbc.org/files/8613/6295/6420/World_Green_Building_Trends_SmartMarket_Report_2013.pdf [Accessed 17 Jan. 2014]

Nawari O. Nawari. 2012. Automated Code Checking in BIM Environment. *Proc. International Conference on Computing in Civil and Building Engineering*, Moscow.

Nepal, M. P., Staub-French, S., Pottinger, R. & Webster, A. 2012. Querying a building information model for construction-specific spatial information. *Advanced Engineering Informatics*, 26, 904-923.

Ryan, E. M. & Sanquist, T. F. 2011. Validation of building energy modeling tools under idealized and realistic conditions. *Energy and Buildings,* 47**,** 375-382.

Shin, T.-H., Chin, S., Yoon, S.-W. & Kwon, S.-W. 2011. A service-oriented integrated information framework for RFID/WSN-based intelligent construction supply chain management. *Automation in Construction,* 20**,** 706-715.

Singh, V., Gu, N. & Wang, X. 2011. A theoretical framework of a BIM-based multi-disciplinary collaboration platform. *Automation in Construction*, 20, 134-144.

Solibri 2008. *Solibri Model Checker* [Online]. Available: http://www.solibri.com/ [Accessed 8 Aug. 2013].

Susmita, S., Sawhney, A., Borrmann, A. & Ritter, F. 2013. Extracting Information from Building Information Models for Energy Code compliance of Building Envelope. *Proc. RICS COBRA Conference*, New Delhi, India.

The Government of Hong Kong Special Administrative Region 2007. Electrical and Mechanical Services Department. Available: http://www.emsd.gov.hk/emsd/e_download/pee/lightingcop_2007.pdf [Accessed 3 Oct. 2013].

PHP Group 1997. *PHP: Hypertext Preprocessor* [Online]. Available: http://php.net/ [Accessed 26 March 2014].

USGBC (U. S. Green Building Council). 2012. *LEED Projects & Case Studies Directory* [Online]. Available: http://www.usgbc.org/LEED/Project/CertifiedProjectList.aspx [Accessed 12 Oct. 2012].

W3C (World Wide Web Consortium). 2001. Web Services Description Language (WSDL) [Online]. Available: http://www.w3.org/TR/wsdl [Accessed 17 Jan. 2014].

W3C (World Wide Web Consortium). 2003. Simple Object Access Protocol (SOAP) [Online]. Available: http://www.w3.org/TR/soap/ [Accessed 17 Jan. 2014].

Yezioro, A., Dong, B. & Leite, F. 2008. An applied artificial intelligence approach towards assessing building performance simulation tools. *Energy and Buildings*, 40, 612-620.